dataZoa.com API Documentation

Published by Leading Market Technologies, Inc.

Version 1.7.0

Publication Date: January 11, 2017

Table of Contents

1 Introduction 3

1.1 About dataZoa 3

1.2 Scope of this document 3

2 Terms and Concepts 3

2.1 API 3

2.2 API Key 4

2.3 Metadata 4

2.4 Data Types 4

2.5 Series and Serieskey 4

2.5.1 Header 4

2.5.2 Header Functional Keywords 5

2.5.3 Descriptive Keywords 5

2.5.4 DataSection 5

2.5.5 Footer 6

2.5.6 Footer Functional Keywords 6

2.6 Grouplists 7

3 Functions 7

3.1 Utility calls 8

3.1.1 Func=Version 8

3.1.2 Func=Runstate 8

3.2 Series Manipulation 8

3.2.1 Func=SeriesExists 9

3.2.2 Func=SeriesGet 9

3.2.3 Func=SeriesCreate 9

3.2.4 Func=SeriesReplace 10

3.2.5 Func=SeriesFromURL 10

3.2.6 Func=SeriesList 11

3.2.7 Func=SeriesDelete 12

3.2.8 Func=SeriesIsPrivate 12

3.2.9 Func=SeriesIsFindable 13

3.3 Grouplist Manipulation 13

3.3.1 Func=GLList 13

3.3.2 Func=GLDelete 13

3.3.3 Func=GLFromSeries 14

3.3.4 Func=SeriesFromGL 14

4 Encodings 14

4.1 URLEncode before sending argument values 14

4.2 JSON decode responses 15

4.3 Miscellaneous escapes 15

5 API Usage Governors 15

5.1 Connection limit (simultaneous calls) 15

5.1 Rate limits 15

6 API Versioning 16

7 Standard Response Codes, applicable to all Calls 16

1 Introduction

1.1 About dataZoa

dataZoa.com is a website dedicated to the storage, manipulation, management and display of time-series data. dataZoa users maintain individual accounts that hold a mixture of private data, data from public sources, and data shared among other dataZoa users.

dataZoa supports a variety of methods to store and retrieve user data, including Drag-and-Drop among sites and programs, automated and scheduled retrieval, manual input and editing, and programmatic interfacing through an API (Application Programming Interface).

1.2 Scope of this document

This document is centered on the dataZoa API capabilities and methods, but includes some discussion of dataZoa formats and conventions. These discussions do not document all of dataZoa, but are intended to provide the understanding required to build meaningful applications.

This document assumes familiarity with commonly used Web programming concepts, in particular http POSTing and JSON encapsulation.

2 Terms and Concepts

2.1 API

The API provided for dataZoa is at heart a pure *data-exchange interface*. That is, this API does not require any libraries or other client-side code. The conventions and protocols used for the data interchange are common to typical Web applications; http(s) and JSON.

The target URL for all API calls is:

datazoa.com/api/apimain.asp

Thus a typical GET type transaction might look like from an ordinary browser might look like:

http://www.datazoa.com/api/apimain.asp/?apikey=nqgjhkiqeolhlhdpgbaigjppnapghaplppgnkbql&func=version

Important: API calls that write data to dataZoa must be issued *serially* and must not be issued in parallel. This is required to avoid race conditions and resource conflicts and to guarantee sequential allocation of Series Keys.

2.2 API Key

In order to use any functionality from the dataZoa API, your must supply your login credentials in the form of an *API Key*. This is a hashed alphanumeric key that is a unique function of your userid, password, and certain other information. You can obtain your API Key by logging into your dataZoa account, visiting the Profile page, and opening the "api key" link. Copy and paste the information at that link for use in your API calls.

Please remember, you must obtain an updated API Key whenever you change your password.

2.3 Metadata

Metadata is the dataZoa term for any attributes of data that are not dates or values of the actual time series. Some metadata, such as *frequency,* is functional; affecting the way data is stored and processed. Other metadata, such as *units*, may be purely descriptive.

2.4 Data Types

Because the dataZoa API employs JSON text transfers, strict binary data typing is not a concern, and the intrinsic text-to-binary conversion functions of most common languages work seamlessly.

The specific formats for *parameters sent* to dataZoa are shown in the call-specific documentation.

For *results returned* by dataZoa, users should anticipate that numeric values returned by the API can take on a variety of natural human-readable forms, such as: "*1", "1.0"," -0.1", "-.1" or "1.01E12."*

Similarly, date or date/time strings *returned* may variously include dates only or date/times, and will be shown in natural forms such as *"4/26/2014 or "4/26/2014 12:43:22 AM,"* with MM/DD/YYYY month/day ordering presumed.

2.5 Series and Serieskey

A dataZoa series is a single time series, comprising a Header, DataSection, and Footer. These three main sections are separated by one or more blank lines.

Series are identified by a unique key that is initially generated by dataZoa.com but can be used by the application builder to later replace a series. The series key is built from the account UserName and a generated serial number. A typical Serieskey might look like "UserAlice/00000321".

2.5.1 Header

Example:

Series title:	M1 Money Stock
Series key:	Citizens/00000556
Data-set title:	Macro Data
Units:	Dollars

Unit multiplier:	Billions
Frequency:	Weekly
Start date:	1/6/1975
End date:	11/28/2011
Release date:	12/15/2011
Publisher:	Leading Market Technologies
Source:	Board of Governors of the Federal Reserve System

2.5.2 Header Functional Keywords

2.5.2.1 “Series title:”

This is used for display purposes throughout dataZoa.

2.5.2.2 “Series key:”

See discussion above. Although it is part of the header as stored, it should always be supplied by dataZoa, never by the end user.

2.5.2.3 “Frequency”

If not supplied, dataZoa will infer the frequency of the data. If supplied, it must match one of the frequencies supported by dataZoa. As of this writing, these possible values are: Daily, Weekly, Monthly, Quarterly, Semiannual and Annual.

2.5.3 Descriptive Keywords

All keywords shown in the header example above that are not explicitly noted as Functional Keywords will be parsed and stored as meta-data, but do not have particular functional roles as of this writing.

2.5.4 DataSection

The DataSection begins following the first blank line after any Header section. If there is no Header, the DataSection can begin immediately.

A DataSection should have at least three lines of date value pairs.

A DataSection can have more than one value column for each row. When multiple values columns are present, one unique series is created for each values column, applying the date column to each. All rows should have the same number of tokens. A pair of double quotes can be used as a placeholder for any missing tokens.

The first row can be a set of column labels. If present, the column labels are incorporated into the series titles. Multi-word column labels should be enclosed in double quotes.

Example:

DATE	“My Values”
1/5/1981	407.4
1/12/1981	409.4
1/19/1981	413.2
1/26/1981	413.8
2/2/1981	410.8
2/9/1981	413.6
…	

2.5.5 Footer

The Footer begins following the first blank line after the DataSection, and can be free-form, except for certain functional keywords that will have significance if they start a line.

Example:

```
This is footer.
It is a great place for footnotes!

UPDATE_URL: http://research.stlouisfed.org/fred2/series/M1
UPDATE_FREQ: 1440
FAVICON_LOC: http://research.stlouisfed.org/favicon.ico
FAVICON_DEST: http://research.stlouisfed.org/fred2/series/M1?cid=121
```

2.5.6 Footer Functional Keywords

2.5.6.1 “UPDATE_URL:”

dataZoa will refresh data from this location.

2.5.6.2 “UPDATE_FREQ:”

dataZoa will refresh data this often – value in minutes; suggested, not guaranteed. As of this writing, the standard update cycle is once per day in the early morning US Eastern time zone.

2.5.6.3 “FAVICON_LOC:”

Location of a standard favicon to display with this data.

2.5.6.4 “FAVICON_DEST”

Location of click-through on the favicon.

2.6 Grouplists

A dataZoa *grouplist* is an ordered collection of series keys that are used in the dataZoa user interface to organize sets of data.

In the dataZoa API, grouplists are referred to with a "glname" argument. Grouplist names are fairly free form, except that the vertical bar ("|") character is not allowed.

There is a special grouplist name, "All of Latest Upload," that is re-created whenever a series or group of series is acquired from a URL, either via the dataZoa UI Drag&Drop to target or through the API SeriesFromURL or SeriesCreate calls. This list can be queried and saved to a new unique name to keep series grouped as they are acquired.

3 Functions

All calls are made to the API data interface URL, generally in the form:

http://www.datazoa.com/api/apimain.asp?apikey=xxxxxxxx&func=FUNC&ARGn...

where xxxxxxxx, FUNC and ARGs are specified as warranted.

Both the http and https protocols are supported.

The API will accommodate both GETs and POSTs as is convenient for the developer. GET arguments have precedence over POSTed arguments when there is conflict, and GETs are subject to the typical "few thousand character" limitations typically encountered in such interactions. dataZoa make no particular guarantees about data capacities of GET operations.

Responses are typically four JSON encapsulated fields; func (what you asked for), code (integer response), message (for humans), and payload (to digest). Please see the Encodings discussion for further details.

Example:

http://www.datazoa.com/api/apimain.asp?apikey=4/12053/e94040d81ba312888a810a8488165b94&func=version

Response when proper apikey is supplied:

```
{"func":"version","code":"0","message":"","payload":"1.1.1"}
```

Otherwise:

```
{"func":"version","code":"802","message":"keylogin failed","payload":""}
```

3.1 *Utility calls*

3.1.1 Func=Version

Purpose:
Determine current version of the dataZoa API

Arguments:
None.

Payload:
Version string.

Example:

```
Call: func=version

Response: {"func":"version","code":"0","message":"","payload":"1.1.1"}
```

3.1.2 Func=Runstate

Purpose:
Determine current runstate of the dataZoa API

Arguments:
None.

Payload:
Current runstate string.

Example:

```
Call: func=runstate

Response: {"func":"version","code":"0","message":"","payload":"NORMAL"}
```

Note: The other commonplace response is “READONLY” which is returned when the dataZoa site is in READONLY mode for maintenance.

3.2 *Series Manipulation*

These calls are used to manipulate series at dataZoa. Note that some calls take a single *serieskey* argument, while others accept a comma-delimited *serieskeylist*.

3.2.1 Func=SeriesExists

Purpose:

Determine whether a given series exists.

Arguments:

serieskeylist=*serieskey [,serieskey] …*

Payload:

Comma delimited list of serieskey:true or *serieskey:false*

Example:

```
Call: func=seriesexists&serieskey=UserAlice/00000213,UserAlice/00000214

Response: {"func":"seriesexists","code":"0","message":"","payload":"
UserAlice/00000213:true, UserAlice/00000214:true"}
```

3.2.2 Func=SeriesGet

Purpose:

Return a single series exactly as stored at dataZoa. Analogous to the dataZoa Drag-and-Drop drag source functionality.

Arguments:

serieskey= *serieskey*

startdate=*startdate* (optional; if present, return data on or after this date)

enddate=*enddate* (optional; if present, return data on or before this date)

maxtoload=*maxtoload* (optional; if present, along with at least one start or end date, return no more than this many observations)

payloadasjson=true (optional; default = false; if true payload will be series encoded in JSON format)

Payload:

Text of series as stored by dataZoa.

Notes:

Date formats should be of the form MM/DD/YYYY.

If startdate, enddate and maxtoload are all supplied, maxtoload is calculated from enddate.

3.2.3 Func=SeriesCreate

Purpose:

Create one or more series from properly formatted text. Analogous to the dataZoa “Add New Series” functionality.

Arguments:

raw=[properly formatted plain text data]
allowreplacement=true (optional; default = false)

Payload:
Comma delimited list of newly created serieskey(s)

Notes:
If "allowreplacement=true" is present, the raw data series are expected to have headers that include an accurate "SERIES_KEY:" entry so that when the series are created they will entirely replace those with corresponding keys. If a "SERIES_KEY:" is not present in the series header, a new key will be created. This is the equivalent of checking the "Allow Replace" checkbox in the dataZoa create/update data entry area,

3.2.4 Func=SeriesReplace

Purpose:
Replace a single series entirely at dataZoa. Analogous to the dataZoa edit functionality.

Arguments:
serieskey= *serieskey*
strict=false (optional; default = true)
raw=[properly formatted plain text data]

Payload:
None.

Notes:
If "strict=false" is present, the data format requirements for SeriesReplace are the same as those for SeriesCreate, as described earlier in this document.

If "strict=false" is not present, the requirements for the data format for SeriesReplce are more restrictive, in that the data format must be nearly identical to the format in which dataZoa returns data.

3.2.5 Func=SeriesFromURL

Purpose:
Create one or more series by acquisition from a 3rd party URL. Analogous to the dataZoa URL Drag&Drop "target" functionality.

Arguments:
url= *any appropriate URL*

Payload:
Comma delimited list of newly created serieskey(s)

Example:

```
Call: func=seriesfromurl&url= http://research.stlouisfed.org/fred2/series/STLFSI?cid=121
```

```
Response:
{"func":"seriesfromurl","code":"0","message":"OK","payload":"PublisherAlice/00000187"}
```

3.2.6 Func=SeriesList

Purpose:

Enumerate all series keys in the account.

Arguments:

serieskeylist=*serieskey[,serieskey] … | glname*	(optional; defaults = empty string)
pagesize=integer	(optional; default = 8000, min = 10, max = 8000)
pagenum=integer	(optional; default = 1)
titlefilt=string	(optional; default = empty string)
fields=field,field…	(optional; default = none other than serieskey)

Payload:

Comma delimited list of serieskeys

If additional fields are specified, payload is a comma delimited list of serieskey|fieldvalue|fieldvalue…

Example:

```
Call: func=serieslist&titlefilt=civilian&fields=date_modified,is_private

Response: {"func":"serieslist","code":"0","message":"OK","payload":"
PublisherAlice/00000190|4/25/2013 11:37:57 AM|true,PublisherAlice/00000130|4/25/2013
11:38:43 AM|true"}
```

Notes:

The "serieskeylist" argument controls the pool of which series to gather information for. This can be a single series key, a comma delimited series key list, a grouplist name or left unspecified implying the pool is all series in the account.

The “pagesize” argument controls the largest number of serieskeys that will be listed in one call. Setting a smaller pagesize may improve performance in some cases, but typically the default is fine.

The “pagenum” argument is used to specify which set of serieskeys is returned, in groups of pagesize. If you have more than the maximum pagesize number of series in your account, you will need to call serieslist in a loop, incrementing pagenum, until no more keys are returned.

The “titlefilt” argument is used to limit the series considered to only those with titles that contain the value given.

The “fields” argument can contain any of:

Field name	Comments
sorting_date	As used in the main dataZoa list display
date_added	
date_modified	Last manual or automatic update

is_private	true or false
is_findable	true or false
start_date	
end_date	
title	
updateurls_url	URL from which series is updated
frequency	
adjustment	
unit	
multiplier	
publisher	
publisher_url	
n_viewers	Number of sharees + followers
is_borrowed	Is being “followed” by this account
n_borrowers	Number of “followers”

IMPORTANT: Because some returned fields could contain the comma or vertical bar delimiters as part of their text, all fields have substitutions made as needed, which the caller must reverse upon receipt. The substitutions are:

“_comma_” for “,”
“_vertbar_” for “|”

3.2.7 Func=SeriesDelete

Purpose:
Delete series by key.

Arguments:
serieskeylist=*serieskey [,serieskey]…*

Payload:
Comma delimited list of *serieskey* for any series that were deleted.

3.2.8 Func=SeriesIsPrivate

Purpose:
To mark series as public or private, or to determine how currently marked

Arguments:
serieskeylist=*serieskey [,serieskey]…*
request=[get | settrue | setfalse]

Payload:

Comma delimited list of *serieskey:true* or *serieskey:false.* Note that any series not found by key will be reported as false.

3.2.9 Func=SeriesIsFindable

Purpose:

To mark series as findable or hidden, or to determine how currently marked

Arguments:

serieskeylist=*serieskey [,serieskey]* ...
request=[get | settrue | setfalse]

Payload:

Comma delimited list of *serieskey:true* or *serieskey:false.* Note that any series not found by key will be reported as false.

3.3 Grouplist Manipulation

These calls are used to manipulate Grouplists at dataZoa.

3.3.1 Func=GLList

Purpose:

Enumerate all grouplists in the account

Arguments:

None.

Payload:

Comma delimited list of grouplist names.

Note:

Commas in returned names will be encoded to %2C so that result set itself can be comma delimited.

3.3.2 Func=GLDelete

Purpose:

Delete a grouplist.

Arguments:

glname=*glname*

Payload:

None, but may return a descriptive message.

Note:

The descriptive message will always report the grouplist as dropped, whether or not it existed at the time. Use the function SeriesFromGL if you need to know whether a grouplist actually exists before or after GLDelete.

3.3.3 Func=GLFromSeries

Purpose:

Create a grouplist.

Arguments:

glname=*glname*
serieskeylist=*serieskey [,serieskey]…*

Payload:

None, but may return a descriptive message and a status code of 200 to indicate a problem condition.

3.3.4 Func=SeriesFromGL

Purpose:

Returns the serieskey(s) for each series contained within a given grouplist.

Arguments:

glname=*glname*

Payload:

Comma delimited list of *serieskeys.* Note that any series no longer available will be reported with “NOTFOUND:sidGUID” where the sidGUID is an internal dataZoa encoding of the key that is no longer available. .

.

4 Encodings

4.1 URLEncode before sending argument values

While not strictly needed in every case, it is good practice to URLEncode the *value* portion of any arg=value pairs being sent in an API request.

When POSTed data is read by the API receiver, a request header of

"Content-type","application/x-www-form-urlencoded" is presumed.

4.2 JSON decode responses

Generally speaking, there will be four standard JSON encapsulated values returned for any function call, as discussed earlier. Within those responses, however, further "un-escaping" may be required, as discussed below.

4.3 Miscellaneous escapes

After extracting the JSON payload of a response from the API, certain substitutions made in the payload will need to be translated back from an escaped form before the payload is back to its original representation.

Specifically, and in this order, perform these substitutions:

"&quot;" → double quote character
"&gt;" → ">"
"&lt;" → "<"
"\n" → system native newline
"\t" → system native TAB
"\\" → "\"

5 API Usage Governors

In order to ensure balanced throughput, API calls are subject to certain usage limitations, described here. The numeric limits given are per account. These limitations are upper bounds, and may be dynamically lowered during periods of high overall system loading.

Generally speaking, dataZoa handles the limitations transparently with usage governors, and API client programs themselves do not have to implement rate limits.

5.1 Connection limit (simultaneous calls)

It is often sensible to launch several API client processes in parallel to reduce wall clock time for a given task. Please note that no more than 50 API calls per account can be in process at any one time. When exceeded, the API call will fail with error code 901 (E_CONLIMIT). While the connection limit may be adjusted downward dynamically, it will never be adjusted below the value 5. Therefore applications can either limit themselves to 5 request threads or else try for more and be able to adjust for fewer.

5.1 Rate limits

There are limits to the rate at which API requests will be processed. No more than 50 requests per second or 500 requests per minute will be served. As a rate limit is approached, dataZoa will begin to apply small sleep periods to requests before processing them. Thus the calling application does not need to implement a rate governor. While there are error codes (e.g. E_HPSLIMIT) defined for excessive rates, they will never be returned in normal practice.

6 API Versioning

The dataZoa API uses a three part versioning scheme; Version.Revision.BuildNumber.

BuildNumber is incremented with each public release, and is zeroed when Revision is incremented. It will change for bug fixes or undocumented changes.

Revision is incremented when documented changes are made, such as new functions or arguments. It is zeroed when Version is incremented.

Version is incremented when entirely new families of capabilities are introduced.

7 Standard Response Codes, applicable to all Calls

0 = Success

901 = Too many simultaneous connections (E_CONLIMIT)
911 = Too many hits per second (E_HPSLIMIT)
912 = Too many hits per minute (E_HPMLIMIT)

800 = Malformed authentication
801 = Unable to authenticate – user not recognized
802 = Unable to authenticate – incorrect password

700 = Malformed request
701 = Unknown function
702 = Missing argument
703 = Timed out
704 = RUNSTATE too low
705 = Invalid argument