TRKD PROGRAMing example

USING python

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Date | version | Summary of changes |
| Wasin Waeosri | 26 Sept 2016 | V0.001 | Initial draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Review History

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Reviewer Name | Doc Version Reviewed | Date Sent for review | Date Review Filed | Approved/Rejected  (with Reasons) |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. It is the author’s responsibility to send this document out for review, making entries into columns 1-3
2. It is the reviewers’ responsibility to complete columns 4-5 and return the reviewed document to the author

Contents

[1 About this document 4](#_Toc462671460)

[1.1 Intended readership 4](#_Toc462671461)

[1.2 In this guide 4](#_Toc462671462)

[1.3 References 4](#_Toc462671463)

[1.4 Acronyms and Abbreviations 4](#_Toc462671464)

[2 Introduction 5](#_Toc462671465)

[2.1 What is TRKD 5](#_Toc462671466)

[2.2 About this document 5](#_Toc462671467)

[2.3 prerequisite 5](#_Toc462671468)

[3 TRKD REST API implementing process 6](#_Toc462671469)

[3.1 Authentication 6](#_Toc462671470)

[3.2 Subscribe data 6](#_Toc462671471)

[3.3 Data format 6](#_Toc462671472)

[4 trkd\_authen.py 7](#_Toc462671473)

[4.1 TRKD Service token detail 7](#_Toc462671474)

[4.1.1 TRKD Service Token URL and Header 7](#_Toc462671475)

[4.1.2 TRKD Service TOKEN Request Message 7](#_Toc462671476)

[4.2 Implementation details 8](#_Toc462671477)

[4.2.1 Import the required libraries 8](#_Toc462671478)

[4.2.2 Get the Username, Password and Application ID 8](#_Toc462671479)

[4.2.3 Create a request message 8](#_Toc462671480)

[4.2.4 Send a HTTP request and handle incoming HTTP response 8](#_Toc462671481)

# About this document

## Intended readership

This guide is aimed at developers who have a background in using Python and are interested in using Thomson Reuters Knowledge Direct API (TRKD) through the REST service.

## In this guide

This is a programmer’s examples to using TRKD API via the REST service. It goes through basic knowledge to use the TRKD API REST service with the Python.

## References

|  |  |
| --- | --- |
| **Reference** | **link** |
| RD. 1 – TRKD Request Builder web site | <https://www.trkd.thomsonreuters.com/SupportSite/login.aspx?ReturnUrl=%2fSupportSite%2fRequestBuilder%2frequestbuilder.aspx> |

## Acronyms and Abbreviations

|  |  |
| --- | --- |
| **ACRONYM** | **DEFINITION** |
| TRKD | Thomson Reuters Knowledge Direct API |
| TS | Timeseries |

# Introduction

## What is TRKD

The Thomson Reuters Knowledge Direct (TRKD) API integrates into your website, trading platform, company intranet/extranet, advisory portal and mobile applications to provide up-to-date financial market data, news and analytics and powerful investment tools.

TRKD offers a wide range of Thomson Reuters' information and services delivered in a request-response scenario via web services using today's industry standard protocols (SOAP/XML and REST/JSON). Connectivity can be via HTTP and HTTPS, over the Internet or Delivery Direct.Level 2. All data are snapshot (non-streaming) data.

## About this document

This document gives you examples of how to implement the Python application to consume TR information from the TRKD API via the REST service. This document is a part of the [TRKD\_REST\_Python\_Example](https://git.sami.int.thomsonreuters.com/wasin.waeosri/TRKD_REST_Python_Example) project in TR Sami Git repository. This document contains only examples description for the Authentication and Quote services. For the other examples, please see more details in the project web site (<https://git.sami.int.thomsonreuters.com/wasin.waeosri/TRKD_REST_Python_Example>)

## prerequisite

The following software and compiler are required for this document

* Python 2 SDK
* Python’s requests library (<http://docs.python-requests.org/en/master>)

You can install the requests library with the bundled Python’s pip tool as following steps

1. Export <Python\_folder>\Scripts to your OS PATH environment
2. Call the pip command to install requests

$> pip install requests

1. If you are behind proxy, set the proxy first

$> export https\_proxy="http://<proxy.server>:<port>"

$> pip install requests

# TRKD REST API implementing process

The consumer application requires the following steps to consume data from TRKD API

1. Authentication to get an authen token
2. Request data with the required input information and authen token

## Authentication

The consumer application needs to authenticate with TRKD service Token with the following information

* Username
* Application ID (AppID)
* Password

Once the application is permissioned success, the application gets the following information from TRKD

* Token: an encrypted, expiring string that securely identifies the service user (aka service Token)
* Expiration: Token expires after a configurable time period. The default expiration is 90 minutes.

## Subscribe data

After the application get the service Token, the application can subscribes data to the TRKD service. The application needs to set the Application ID and service Token with the request message header to identify the permissioned.

* X-Trkd-Auth-ApplicationID: Application ID
* X-Trkd-Auth-Token: service Token

## Data format

All request and response messages of the TRKD REST Services are in the JSON format. We will use the Python json library to encode the request message via the HTTP Post request. For the response message, we will use the JSON builtin encoder of the requests library to decode incoming JSON data.

# trkd\_authen.py

TRKD API uses an encrypted, expiring token to authenticate each request. To create this token, your application submits credentials to the Token Management service. The Token Management service authenticates the credentials and returns the token.

This section describes how to implement the trkd\_authen.py script that performs authentication with the TRDK API with the inputted username, password and appid via the command line.

## TRKD Service token detail

### TRKD Service Token URL and Header

The URL enponint for the TRKD Service Token is following:

<https://api.trkd.thomsonreuters.com/api/TokenManagement/TokenManagement.svc/REST/Anonymous/TokenManagement_1/CreateServiceToken_1>

Header:

* Content-type = application/json;charset=utf-8

Method:

* Post

### TRKD Service TOKEN Request Message

The CreateServiceToken\_1 operation requires the following information to perform authentication

* ApplicationID
* Username
* Password

The request message structure is following

{

“CreateServiceToken\_Request\_1”:{

“ApplicationID”: *<application id>*,

“Username”: *<username>*,

“Password”: *<password>*

}

}

The example of the response message is shown below

{

"CreateServiceToken\_Response\_1": {

"Expiration": "*2016-09-26T09:42:54.4335265Z*",

"Token": "*674E12E4EF35F181602672D5529D98379D4B42216057C7FF…*"

}

}

## Implementation details

### Import the required libraries

Firstly, we create a file named “trkd\_authen.py” in the working directory. Then we import all required libraries at the top of the source code

import os

import sys

import requests

import json

import getpass

### Get the Username, Password and Application ID

Then we add the code to receive inputted username, password and appid from the console as shown in figure 1

##Get username, password and applicationid

username = raw\_input('Please input username: ')

##use getpass.getpass to hide user inputted password

password = getpass.getpass(prompt='Please input password: ')

appid = raw\_input('Please input appid: ')

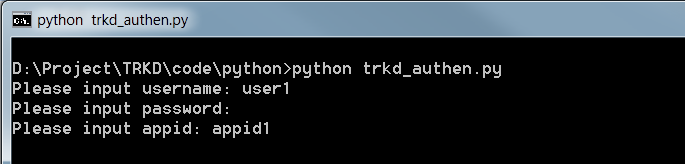


Figure-1: The trkd\_authen receives username, password and appid via commandline

### Create a request message

The next step is creating the request message (authenMsg), the url (authenURL) and headers variables.

##create authentication request URL, message and header

authenMsg = {'CreateServiceToken\_Request\_1': { **'ApplicationID'**:appid, **'Username'**:username,**'Password'**:password }}

authenURL = 'https://api.trkd.thomsonreuters.com/api/TokenManagement/TokenManagement.svc/REST/Anonymous/TokenManagement\_1/CreateServiceToken\_1'

headers = {'content-type': 'application/json;charset=utf-8'}

### Send a HTTP request and handle incoming HTTP response

This final step is creating a HTTP Post request to the TRKD Service Token endpoint that specified in the authenURL variable with the authenMsg request message. We use the requests.post function of the requests library to handle this operation.

##send request

result = requests.post(authenURL, data = json.dumps(authenMsg), headers=headers)

Then we check if the status code is 200 (OK), prints out the Token and Expiration. If the return status code is 500, prints out the whole return JSON message.

##send request

result = requests.post(authenURL, data = json.dumps(authenMsg), headers=headers)

if result.status\_code == 200:

print 'Request success'

print 'response status %s'%(result.status\_code)

##get Token

token = result.json()['CreateServiceToken\_Response\_1']['Token']

print 'Token: %s'%(token)

##get expiration

expire = result.json()['CreateServiceToken\_Response\_1']['Expiration']

print 'Exipre: %s'%(expire)

elif result.status\_code == 500:

print 'Request fail'

print 'response status %s'%(result.status\_code)

print 'Error: %s'%(result.json())

The result is shown in the Figure-2 below

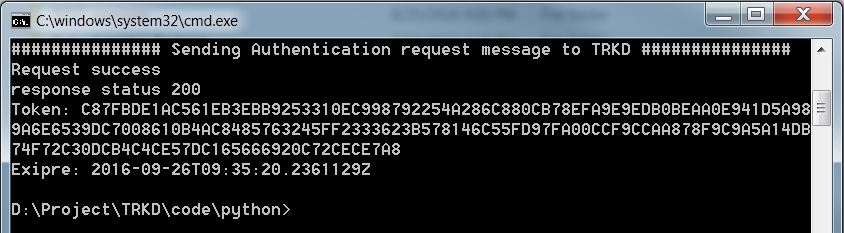


Figure-2: The Authentication result from TRKD Service Token endpoint

Please note that the application must keeps the application id and this service Token for futher operations.

# Trkd\_quote.py

The Quotes service (TRKD RetriveItem3 endpoint) enables you to retrieve a snap quote or a quote chain. TRKD API delivers snap prices only; you may not deliver streaming prices using TRKD API. By default TRKD provides delayed stock prices. However, if your organization has an agreement within an exchange to provide its prices in real-time, then real-time snap prices can be delivered by TRKD for that exchange.

The Quotes service requires the Application ID and Service Token information in the request message’s header.

Note: Up to 75 instruments and 25 fields may be included in each request.

This section describes how to implement the trkd\_quote.py script that request for quote data with the TRDK API with the inputted username, password, appid, RIC and fields via the command line.

## TRKD QUOTE detail

### TRKD RetriveItem3 URL and Header

The URL enponint for the TRKD Service Token is following:

<https://api.trkd.thomsonreuters.com/api/Quotes/Quotes.svc/REST/Quotes_1/RetrieveItem_3>

Header:

* Content-type = application/json;charset=utf-8
* X-Trkd-Auth-ApplicationID = Application ID
* X-Trkd-Auth-Token = Service Token

Method:

* Post

### TRKD RetriveItem3 Request Message

The RetrieveItem\_3 operation requires the item name information. If the fields information is not specify, the request will be all fields request message. The application can request multiple fields data by using “:” string as a separator for each field (example “BID:ASK:CF\_LAST”).

An example of request message structure is following

{

“RetrieveItem\_Request\_3”:{

“TrimResponse”: false,

“ItemRequest”: {

[{

“RequestKey”: [{“Name”: *<Item Name>*, “NameType”: “RIC”}],

“Fields”: *<Field Names, separate each field with ‘:’>*,

“Scope”: “List”,

“ProvideChainLinks”: true

}]

}

}

The example of the response message is shown below

{

"RetrieveItem\_Request\_3": {

"TrimResponse": false,

"ItemRequest": [{

"Fields": "*CF\_LAST:CF\_HIGH:CF\_LOW:CF\_BID:CF\_ASK:CF\_YIELD*",

"RequestKey": [{

"Name": "*MSFT.O*",

"NameType": "RIC"

}],

"Scope": "List",

"ProvideChainLinks": true

}]

}

}

## Implementation details

### Import the required libraries

Firstly, we create a file named “trkd\_quote.py” in the working directory. Then we import all required libraries at the top of the source code

import os

import sys

import requests

import json

import getpass

### Create the method to send the HTTP request to network

The script needs to send multiple HTTP request for authenting and requesting the quote data, so we will separate the HTTP request code to the dedicate function instead.

We create the function named “doSendRequest” which receives three parameters: the URL, request message string and header. This function return the HTTP response from the URL to the caller via the result parameter.

def doSendRequest(url, requestMsg, headers):

result = None

return result

Next, we implement the HTTP operations that send the request message to the inputted URL and passes the incoming response to the caller via the result variable.

def doSendRequest(url, requestMsg, headers):

result = None

try:

##send request

result = requests.post(url, data=json.dumps(requestMsg), headers=headers)

if result.status\_code == 500:

print 'Request fail'

print 'response status %s' % result.status\_code

print 'Error: %s' % result.json()

sys.exit(1)

except requests.exceptions.RequestException, e:

print 'Exception!!!'

print e

sys.exit(1)

return result

### Get the Username, Password and Application ID

Then we add the code to receive inputted username, password and appid from the console as shown in figure 2

## ---------------------------- Main App --------------------------- ##

##Get username, password and applicationid

username = raw\_input('Please input username: ')

##use getpass.getpass to hide user inputted password

password = getpass.getpass(prompt='Please input password: ')

appid = raw\_input('Please input appid: ')

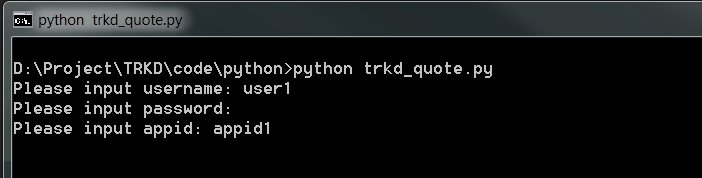


Figure-2: The trkd\_quote receives username, password and appid via commandline

### Manage authentication request

We start this step by sending the inputted username, password and appid to the new CreateAuthorization function which handle the authentication process. Then we print out the Service Token from TRKD in the console.

token = CreateAuthorization(username,password,appid)

print 'Token = %s'%(token)

Next we create the CreateAuthorization function that receives the inputted username, password and appid from the previous step to create the authentication message.

## Perform authentication

def CreateAuthorization(username, password, appid):

token = None

##create authentication request URL, message and header

authenMsg = {'CreateServiceToken\_Request\_1': { **'ApplicationID'**:appid,

**'Username'**:username,**'Password'**:password }}

authenURL = 'https://api.trkd.thomsonreuters.com/api/TokenManagement/TokenManagement.svc/REST/Anonymous/TokenManagement\_1/CreateServiceToken\_1'

headers = {'content-type': 'application/json;charset=utf-8'}

Then we send newly created request message, URL and header to the doSendRequest function. Once the code receives the response data, we send the incoming Service Token back to the caller.

## Perform authentication

def CreateAuthorization(username, password, appid):

...

print '############### Sending Authentication request message to TRKD ###############'

authenResult = doSendRequest(authenURL, authenMsg, headers)

if authenResult is not None and authenResult.status\_code == 200:

print 'Authen success'

print 'response status %s'%(authenResult.status\_code)

##get Token

token = authenResult.json()['CreateServiceToken\_Response\_1']['Token']

return token

### Send the Quote Request to TRKD

Once we get the Service Token, we will check if the Token is null or not (authentication fail). If the Token is not null, we pass the Service Token and application id to the function named “RetreiveQuotes”

## if authentiacation success, continue subscribing Quote

if token is not None:

RetrieveQuotes(token,appid)

Then we create the RetrieveQuotes function that receives the Service Token and Application ID to create the Quote request message and sends it to the TRKD RetrieveItem\_3 endpoint. Then we print out the incoming Quote response to the console. Please note that we will show how to create the Quote Request Message in the next step.

## Perform Quote request

def RetrieveQuotes(token, appid):

quoteRequestMsg = None

#.. Create Quote Request Message

quoteURL = 'https://api.trkd.thomsonreuters.com/api/Quotes/Quotes.svc/REST/Quotes\_1/RetrieveItem\_3'

headers = {'content-type': 'application/json;charset=utf-8' ,'X-Trkd-Auth-ApplicationID': appid, 'X-Trkd-Auth-Token' : token}

print '############### Sending Quote request message to TRKD ###############'

quoteResult = doSendRequest(quoteURL, quoteRequestMsg,headers)

if quoteResult is not None and quoteResult.status\_code == 200:

print 'Quote response message: '

print quoteResult.json()

### Create the Quote Request Message

The last step is to create the Quote Request Message in JSON format that supports both all fields and specific fields request. Firstly, we receive the user input for the following information via the command line

* Item name (Symbol)
* All field (yes or no)

We also define the fieldName variable witht the default fields for the fields subscription.

## Perform Quote request

def RetrieveQuotes(token, appid):

ricName = raw\_input('Please input Symbol: ')

fieldFiltering = raw\_input('Subscribe all Field? (Yes|No)')

quoteRequestMsg = None

fieldsName = 'CF\_LAST:CF\_HIGH:CF\_LOW:CF\_BID:CF\_ASK::CF\_LOTSIZE:CF\_DATE:CF\_TIME:CF\_TICK:CF\_NETCHNG:CF\_EXCHNG:CF\_VOLUME '

# ... Step 5.2.5 code

The above code will promt for the user input as shown in figure 3 below.

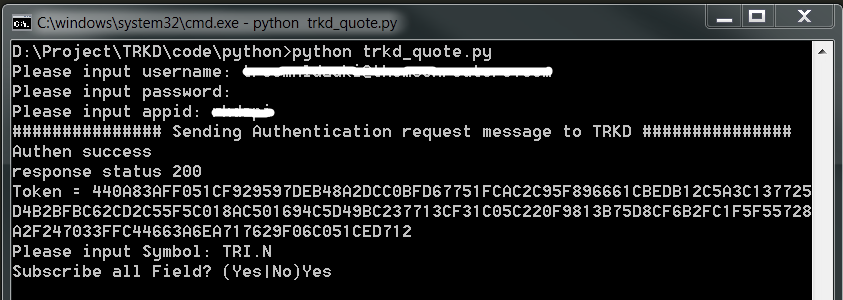


Figure-3: The trkd\_quote receives user input for the item name and all field questions.

We will create the request message for the all field subscription first (fieldFiltering variable = “Yes”).

if fieldFiltering == 'Yes':

## Request all Fields

quoteRequestMsg = \

{'RetrieveItem\_Request\_3': {'TrimResponse': False,

'ItemRequest': [{'RequestKey': [{'Name': *ricName*, 'NameType': *'RIC'*}], 'Scope': 'All',

'ProvideChainLinks': True}]}}

# ... Step 5.2.5 code

The above code will create a quote request message for the all fields subscription which the application will send it to the TRKD RetrieveItem\_Request\_3 endpoint as the code in step 5.2.5. The result is shown belown.

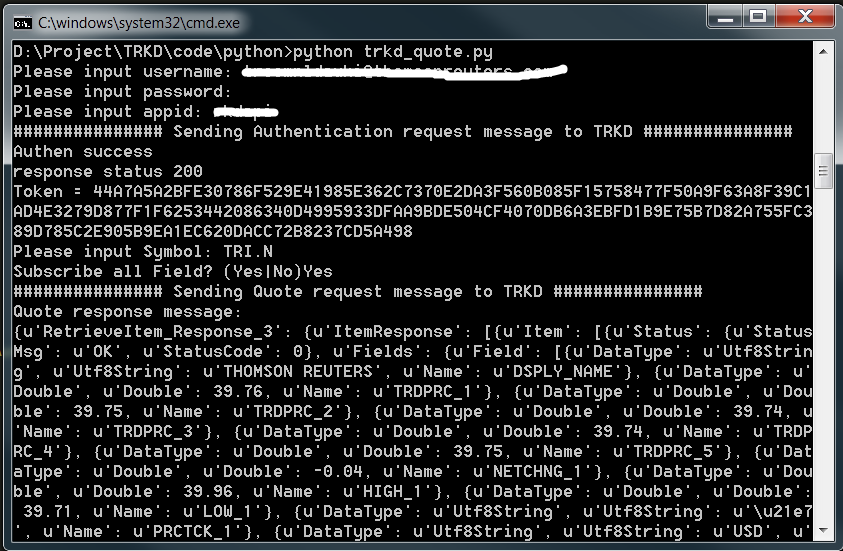


Figure-4: The trkd\_quote result for the all fields subscription.

Next we create the request message for the specific fields subscription case. We will let the user input their interested fields via the command line as well.

if fieldFiltering == 'Yes':

## Request all Fields

elif fieldFiltering == 'No':

## Request specific Fields

fieldsName = raw\_input('Input interested Field Name in the following format (BID:ASK:TRDPRC\_1)')

quoteRequestMsg = \

{'RetrieveItem\_Request\_3': {'TrimResponse': False,

'ItemRequest': [{

'RequestKey': [{'Name': *ricName*, 'NameType': 'RIC'}],

'Fields': fieldsName,

'Scope': 'List',

'ProvideChainLinks': True

}]}}

# ... Step 5.2.5 code

The above code will ask the user to input the interested fields,

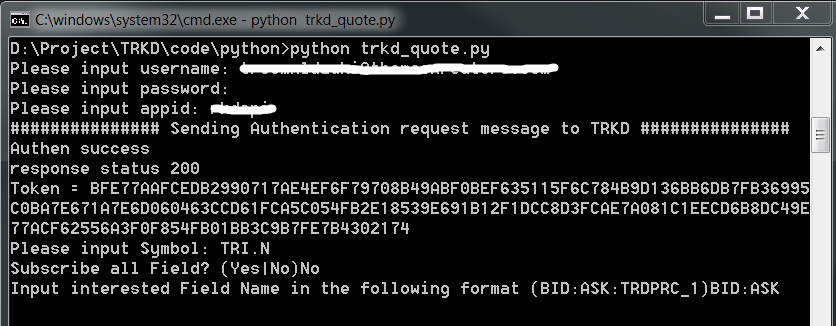


Figure-5: The trkd\_quote receives user input for the interested field names

Then the application creates the request message witht the ‘Fields’ property and send it to the TRKD RetrieveItem\_Request\_3 endpoint as step 5.2.5 code. The result is shown below.

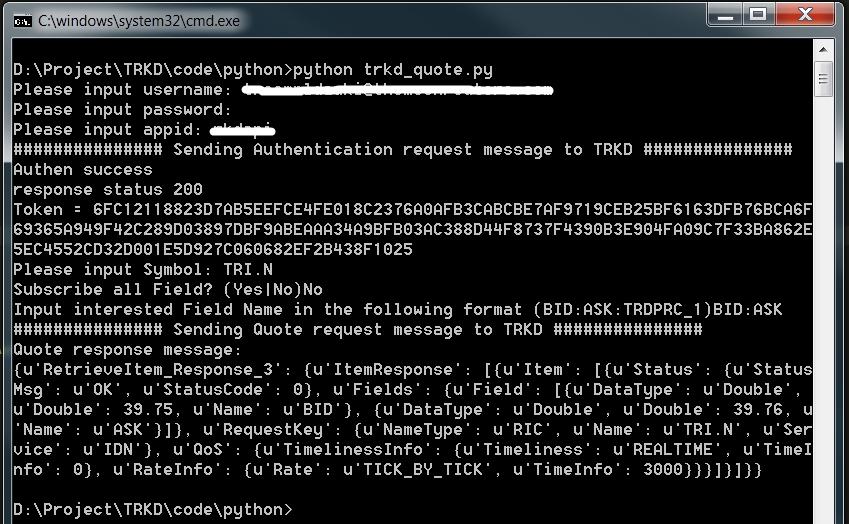


Figure-6: The trkd\_quote result for the specific fields subscription.

|  |  |  |
| --- | --- | --- |
| © 2016 Thomson Reuters. All rights reserved.  Republication or redistribution of Thomson Reuters content, including by framing or similar means, is prohibited without the prior written consent of Thomson Reuters. 'Thomson Reuters' and the Thomson Reuters logo are registered trademarks and trademarks of Thomson Reuters and its affiliated companies. |  | For more information  Send us a sales enquiry at  thomsonreuters.com/about/contact\_us  Read more about our products at  thomsonreuters.com/products\_services  Find out how to contact your local office  thomsonreuters.com/about/locations |