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 6](#_Toc460920100)

[1.1 Intended readership 6](#_Toc460920101)

[1.2 In this guide 6](#_Toc460920102)

[1.3 References 6](#_Toc460920103)

[1.4 Acronyms and Abbreviations 6](#_Toc460920104)

[2 Introduction 7](#_Toc460920105)

[2.1 ASG High level overview 7](#_Toc460920106)

[2.2 EED details 7](#_Toc460920107)

[3 Client Interactions with EED and ASG 8](#_Toc460920108)

[3.1 Interaction between EED and ASG: 8](#_Toc460920109)

[3.2 Interaction between Client and EED 8](#_Toc460920110)

[3.3 Interaction between Client and ASG 9](#_Toc460920111)

[4 Code Examples 10](#_Toc460920112)

[4.1 Device Login 10](#_Toc460920113)

[4.1.1 Device Login Request 10](#_Toc460920114)

[4.1.2 Device Login Response 10](#_Toc460920115)

[4.2 Source Directory 11](#_Toc460920116)

[4.2.1 Source Directory Request 11](#_Toc460920117)

[4.2.2 Source Directory Response 12](#_Toc460920118)

[4.3 Client Login Message 13](#_Toc460920119)

[4.3.1 Client Login Request 13](#_Toc460920120)

[4.3.2 Client Login Response 14](#_Toc460920121)

[4.4 Auxiliary Service Content 15](#_Toc460920122)

[4.4.1 Timeseries/History (TS) 15](#_Toc460920123)

[4.5 Auxiliary Service batch messages 18](#_Toc460920124)

[4.5.1 Request 18](#_Toc460920125)

[4.5.2 Response 19](#_Toc460920126)

[4.6 Logout / Close Login stream 20](#_Toc460920127)

[4.6.1 Close Message 20](#_Toc460920128)

[4.6.2 Acknowledgement Message 21](#_Toc460920129)

[5 APPENDIX 1 Locations of ASG and Connectivity Diagram With Clients 22](#_Toc460920130)

[6 APPENDIX 2 Service Name and ID Published By ASG 23](#_Toc460920131)

[7 APPENDIX 3 Example XML of Messages Communicated with ASG 24](#_Toc460920132)

[7.1 Device Login Request 24](#_Toc460920133)

[7.2 Device Login response 24](#_Toc460920134)

[7.2.1 Success (Refresh Message) 24](#_Toc460920135)

[7.2.2 Failure (Status Message) 24](#_Toc460920136)

[7.3 Source Directory Request 24](#_Toc460920137)

[7.4 Source Directory response 25](#_Toc460920138)

[7.4.1 Success (Refresh Message) 25](#_Toc460920139)

[7.4.2 Failure (Status Message) 26](#_Toc460920140)

[7.5 Client Login Request 26](#_Toc460920141)

[7.5.1 With Username/password 26](#_Toc460920142)

[7.5.2 With SSO token 26](#_Toc460920143)

[7.6 Client Login Response 27](#_Toc460920144)

[7.6.1 Successful 27](#_Toc460920145)

[7.6.2 Failure 27](#_Toc460920146)

[7.7 Private stream close 27](#_Toc460920147)

[7.8 Acknowledgement Message 27](#_Toc460920148)

[7.9 Timeseries/History (TS) Request 27](#_Toc460920149)

[7.10 Timeseries/History (TS) Response 28](#_Toc460920150)

[7.10.1 Success 28](#_Toc460920151)

[7.10.2 Failure Response 30](#_Toc460920152)

[7.11 Elektron Analytics (EA) Request 30](#_Toc460920153)

[7.11.1 Streaming 30](#_Toc460920154)

[7.11.2 Snapshot 30](#_Toc460920155)

[7.12 Elektron Analytics (EA) Response 31](#_Toc460920156)

[7.12.1 Status 31](#_Toc460920157)

[7.12.2 Refresh 31](#_Toc460920158)

[7.12.3 Update 31](#_Toc460920159)

[7.13 Auxiliary Service batch message 32](#_Toc460920160)

[7.13.1 Request 32](#_Toc460920161)

[7.13.2 Response 32](#_Toc460920162)

[7.14 Auxiliary Service Sub Stream Close Message 33](#_Toc460920163)

[8 APPENDIX 4 Time Series possible fid values 34](#_Toc460920164)

[8.1 Time Frame Possible Values (Fid 12794) 34](#_Toc460920165)

[8.2 CORAX Request (FID 12886) 34](#_Toc460920166)

[9 APPENDIX 5 Auxiliary Service response fid description 35](#_Toc460920167)

[9.1 Time Series 35](#_Toc460920168)

[9.2 Elektron Analytics 35](#_Toc460920169)

[10 APPENDIX 6 Flow Charts 37](#_Toc460920170)

[10.1 Device Login (Client to Eed) 37](#_Toc460920171)

[10.2 Source Directory 38](#_Toc460920172)

[10.3 Client Login 39](#_Toc460920173)

[10.4 Timeseries 40](#_Toc460920174)

[10.5 EA Snapshop Request 41](#_Toc460920175)

[10.6 EA Streaming Request 42](#_Toc460920176)

[10.7 Batch Request 43](#_Toc460920177)

[11 APPENDIX 7 Auxiliary Service content supported by ASG 44](#_Toc460920178)

[12 APPENDIX 8 Code example 45](#_Toc460920179)

[12.1 Device Login 45](#_Toc460920180)

[12.1.1 Encoding Device Login Request 45](#_Toc460920181)

[12.2 Source Directory 48](#_Toc460920182)

[12.2.1 Decoding Source Directory Response 48](#_Toc460920183)

[12.3 Client Login 50](#_Toc460920184)

[12.4 Private Stream Close and Ack 54](#_Toc460920185)

[12.5 Auxiliary Service Content 56](#_Toc460920186)

[12.5.1 TS 56](#_Toc460920187)

[12.5.1 EA 60](#_Toc460920188)

[12.6 Auxiliary Service batch messages 65](#_Toc460920189)

[13 APPENDIX 8 Elektron Analytics (EA) 70](#_Toc460920190)

[13.1.1 Elektron Analytics (EA) 70](#_Toc460920191)

[13.2 Auxiliary Service Sub Stream Close Message 73](#_Toc460920192)

[13.3 Auxiliary Service Sub Stream Close Message (code example) 73](#_Toc460920193)

# 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: ')

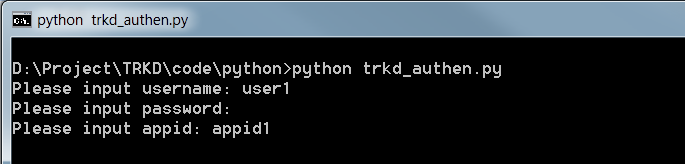


Figure-1: The trkd\_authen receives username, password and appid via commandline

### 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

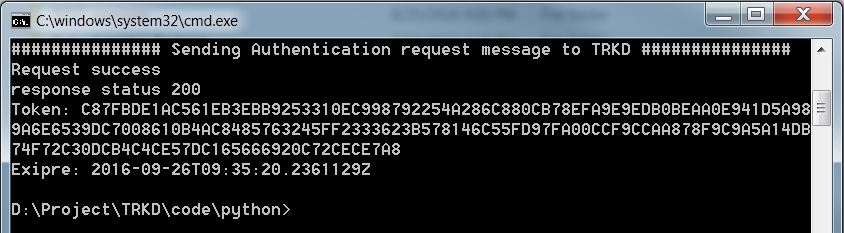


Figure-2: The Authentication result from TRKD Service Token endpoint

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

|  |  |  |
| --- | --- | --- |
| © 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 |