# Factiva Core

***Release 0.2.5***

**Dow Jones**

**Oct 05, 2022**

# OVERVIEW

This package contains fundamental objects shared by other Factiva packages implemented in Python consume services from the Dow Jones Developer Platform.

Check out the Dow Jones Developer Platofrm site for more information about the available services.

CHAPTER

ONE

# OVERVIEW

Factiva Core is a package that provides the fundamental elements to use multiple services across Dow Jones Developer Platform. This packages aims to ease routinary tasks like authentication or translating and taxonomies present in the Dow Jones data.

## 1.1 Authentication

The different Factiva APIs use multiple types of authentication mechanisms like OAuth and User Key. To simplify coding tasks around authentication, the following classes are available.

CHAPTER

TWO

# INSTALLATION

This package can be installed using PIP, the recommended pocedure is running:

```
pip install -u factiva-core
```

This will install and update the package to the latest version.

CHAPTER

THREE

# QUICKSTART

The easiest way to start using *factiva-core* is by creating an instance of a User Key object.

```
from factiva.core import UserKey
u = UserKey(key='abcd1234abcd1234abcd1234abcd1234', request_info=True)
u
```

After its execution, the *UserKey* instance will contain details about the account settings and limits.

CHAPTER

FOUR

# CORE.AUTH

Factiva authentication modules and classes

## 4.1 userKey class

Factiva Core Auth Module

**class** factiva.core.auth.**UserKey**(*key=None*, *stats=False*)

Bases: object

Class that represents an API user. This entity is identifiable by a User Key.

**Parameters**

- **key** (*str*) – String containing the 32-character long APi Key. If not provided, the constructor will try to obtain its value from the FACTIVA_USERKEY environment variable.
- **stats** (*bool, optional (Default: False)*) – Indicates if user data has to be pulled from the server. This operation fills account detail properties along with maximum, used and remaining values. It may take several seconds to complete.

### Examples

**Creating a new UserKey instance providing the key string explicitly and requesting to retrieve the latest account details:**

```
>>> u = UserKey('abcd1234abcd1234abcd1234abcd1234', stats=True)
>>> print(u)
    <class 'factiva.core.userkey.UserKey'>
    |-key = ****************************1234
    |-cloud_token = **Not Fetched**
    |-account_name = AccName1234
    |-account_type = account_with_contract_limits
    |-active_products = DNA
    |-max_allowed_concurrent_extractions = 5
    |-max_allowed_extracted_documents = 200,000
    |-max_allowed_extractions = 3
    |-currently_running_extractions = 0
    |-total_downloaded_bytes = 7,253,890
    |-total_extracted_documents = 2,515
    |-total_extractions = 1
    |-total_stream_instances = 4
```

(continues on next page)

(continued from previous page)

```
    |-total_stream_subscriptions = 1
    |-enabled_company_identifiers = [{'id': 4, 'name': 'isin'}, {'id': 3, 'name
↪': 'cusip'}, {'id': 1, 'name': 'sedol'}, {'id': 5, 'name': 'ticker_exchange'}]
    |-remaining_documents = 197,485
    |-remaining_extractions = 2
```

**Creating a new instance taking the key value from the environment varaible FACTIVA_USERKEY, and not requesting account statistics (default).**

```
>>> u = UserKey()
>>> print(u)
    <class 'factiva.core.userkey.UserKey'>
    |-key = ****************************1234
    |-cloud_token = **Not Fetched**
    |-account_name =
    |-account_type =
    |-active_products =
    |-max_allowed_concurrent_extractions = 0
    |-max_allowed_extracted_documents = 0
    |-max_allowed_extractions = 0
    |-currently_running_extractions = 0
    |-total_downloaded_bytes = 0
    |-total_extracted_documents = 0
    |-total_extractions = 0
    |-total_stream_instances = 0
    |-total_stream_subscriptions = 0
    |-enabled_company_identifiers = []
    |-remaining_documents = 0
    |-remaining_extractions = 0
```

**account_name = ''**

**account_type = ''**

**active_products = ''**

**cloud_token = {}**

**static create_user_key**(*key=None*, *stats=False*)

Determine the way to initialize an api key user according to the type of parameter provided.

**Parameters**

- **api_user** (*None, str,* UserKey) – Source to create a UserKey instance
- **stats** (*boolean, optional (Default: False)*) – Indicates if user data has to be pulled from the server

**Returns**

- When None is passed, UserKey instance using credentials from ENV variables
- When str is passed, UserKey instance using the provided parameter as credentials
- When UserKey is passed, it returns the same instance

**Return type**

UserKey instance accordingly

**Raises**
**RuntimeError** – When an UserKey instance cannot be created using the provided parameters:

**currently_running_extractions = 0**

**enabled_company_identifiers = []**

**property extractions_done**
Number of executed extractions

**get_cloud_token**() → bool
Request a cloud token to the API and saves its value in the cloud_token property

**Returns**
- *True if the operation was completed successfully. Returned value*
- *is assigned to the cloud_token property.*

**Raises**
- **ValueError** – When the credentials are not valid:
- **RuntimeError** – When API request returns unexpected error:

**get_extractions**(*updates=False*) → DataFrame
Request a list of the extractions of the account.

**Parameters**
**updates** (*bool*) – Flag that indicates whether the retrieved list should include (True) or not (False) Snapshot Update calls.

**Return type**
Dataframe containing the list of historical extractions for the account.

**Raises**
- **- ValueError when the API Key provided is not valid** –
- **- RuntimeError when the API returns an unexpected error** –

**get_stats**() → bool
Request the account details to the Factiva Account API Endpoint.

This operation can take several seconds to complete.

**Returns**
- *True if the operation was completed successfully. All returned values*
- *are assigned to the object's properties directly.*

### Examples

```
>>> u = UserKey('abcd1234abcd1234abcd1234abcd1234')
>>> print(u)
    <class 'factiva.core.userkey.UserKey'>
    |-key = ****************************1234
    |-cloud_token = **Not Fetched**
    |-account_name =
    |-account_type =
    |-active_products =
    |-max_allowed_concurrent_extractions = 0
    |-max_allowed_extracted_documents = 0
    |-max_allowed_extractions = 0
    |-currently_running_extractions = 0
    |-total_downloaded_bytes = 0
    |-total_extracted_documents = 0
    |-total_extractions = 0
    |-total_stream_instances = 0
    |-total_stream_subscriptions = 0
    |-enabled_company_identifiers = []
    |-remaining_documents = 0
    |-remaining_extractions = 0
>>> u.get_stats()
>>> print(u)
    <class 'factiva.core.userkey.UserKey'>
    |-key = ****************************1234
    |-cloud_token = **Not Fetched**
    |-account_name = AccName1234
    |-account_type = account_with_contract_limits
    |-active_products = DNA
    |-max_allowed_concurrent_extractions = 5
    |-max_allowed_extracted_documents = 200,000
    |-max_allowed_extractions = 3
    |-currently_running_extractions = 0
    |-total_downloaded_bytes = 7,253,890
    |-total_extracted_documents = 2,515
    |-total_extractions = 1
    |-total_stream_instances = 4
    |-total_stream_subscriptions = 1
    |-enabled_company_identifiers = [{'id': 4, 'name': 'isin'}, {'id': 3, 'name
↪': 'cusip'}, {'id': 1, 'name': 'sedol'}, {'id': 5, 'name': 'ticker_exchange'}]
    |-remaining_documents = 197,485
    |-remaining_extractions = 2
```

**get_streams**(*running=True*) → DataFrame

Obtain streams from a given user.

Function which returns the streams a given user with its respective key using the default stream url

**Parameters**

**running** (*bool*) – Flag that indicates whether the retrieved list should be restricted to only running streams (True) or also include cancelled and failed ones (False).

**Return type**

DataFrame -> DataFrame with the list of historical extractions

**Raises**

- **AttributeError:** – When is not possible to parse the data as json or dataframe
- **ValueError:** – When API key is not valid
- **RuntimeError:** – When API request returns unexpected error

**key = ''**

**max_allowed_concurrent_extractions = 0**

**max_allowed_extracted_documents = 0**

**max_allowed_extractions = 0**

**property remaining_documents**

Account remaining documents.

**property remaining_extractions**

Account remaining extractions.

**show_extractions**(*updates=False*)

Shows a list of the extractions of the account.

**Parameters**

**updates** (*bool*) – Flag that indicates whether the displayed list should include (True) or not (False) Snapshot Update calls.

**Return type**

Dataframe containing the list of historical extractions for the account.

**Raises**

- **- ValueError when the API Key provided is not valid** –
- **- RuntimeError when the API returns an unexpected error** –

**show_streams**(*running=True*)

Shows the list of streams for a given user.

This function runs the existing function get_streams and prints a user-friendly table with stream details.

**Parameters**

**running** (*bool*) – Flag that indicates whether the displayed list should be restricted to only running streams (True) or also include cancelled and failed ones (False).

**Return type**

None

**Raises**

- **AttributeError:** – When is not possible to parse the data as json or dataframe
- **ValueError:** – When API key is not valid
- **RuntimeError:** – When API request returns unexpected error

**property streams_running**

Number of currently running Streaming Instances

**total_downloaded_bytes = 0**

**total_extracted_documents = 0**

```
total_extractions = 0

total_stream_instances = 0

total_stream_subscriptions = 0
```

CHAPTER

FIVE

# CORE.DICTS

Factiva dictionaries with an optimised structure for visualisation.

Note: Dictionaries content not necessarily match exactly the current taxonomies.

## 5.1 Dictionaries

Define basic dictionaries of Hierarchies adn Taxonomies.

`factiva.core.dicts.`**`countries_list`**`()` → DataFrame

Read a list of official countries.

Reads a list of official countries with several additional fields that are helpful in data merges. All contries have the Factiva Code along with other identifiers.

**Returns**
**DataFrame**

**Return type**
A Pandas DataFrame

`factiva.core.dicts.`**`industries_hierarchy`**`()` → DataFrame

Read the Dow Jones Industry hierarchy CSV file.

Reads the Dow Jones Industry hierarchy CSV file and returns its content as a Pandas DataFrame. The root node has the fcode *indroot* and an empty parent.

**Returns**

**DataFrame** –

- **ind_fcode**
  [string] Industry Factiva Code
- **name**
  [string] Name of the Industry
- **parent**
  [string] Factiva Code of the parent Industry

**Return type**
A Pandas DataFrame with the columns:

`factiva.core.dicts.`**`regions_hierarchy`**`()` → DataFrame

Read the Dow Jones Regions hierarchy CSV file.

Reads the Dow Jones Regions hierarchy CSV file and returns its content as a Pandas DataFrame. The root node has the fcode *WORLD* and an empty parent.

Names containng an asterisk denote nodes not officially in the hierarchy, but that help balancing and reading the structure. Level balancing is quite useful specially for visualising data.

**Returns**

**DataFrame –**

- **name**
  [string] Name of the region node
- **reg_fcode**
  [string] Factiva Code of the region
- **parent**
  [string] Factiva Code of the parent region
- **level**
  [int] Level number of the node

**Return type**
A Pandas DataFrame with the columns:

CHAPTER

SIX

# CORE.CONST

Package constants.

## 6.1 Constants

# PYTHON MODULE INDEX

## f

# INDEX