Qi API Client Documentation

Release 3.1.42

Oct 02, 2023

CONTENTS

CHAPTER

ONE

OVERVIEW

1.1 Product Overview

Quant Insight provide macro-economic analytics data via a client-server model, with analytics data computed and stored in the cloud, and accessed on-demand by our customers. Available through both a browser-based web application and a RESTful API service, we support both manual and programmatic access to our data.

1.2 Scope

This document provides guidance for the usage of the Qi client API package; a python package intended to facilitate access to and interrogation of the Qi API.

The package provides an easy to use, object-oriented API which supports both synchronous and asynchronous access.

CHAPTER

TWO

QUICK START GUIDE

Our REST API allows you and your applications to access near-real-time quantitative macro analytics that can be readily integrated with *your* systems. We cater to a range of use-cases, with our data helping to guide understanding of asset macro exposures and market regime, and providing quantitative identification of valuation opportunities.

This python package is automatically generated by the Swagger Codegen project:

- API version: v3.1.42
- Package version: 3.1.42
- Build package: io.swagger.codegen.languages.PythonClientCodegen

2.1 Requirements.

Python 2.7 and 3.4+

2.2 Installation

To install the API package, please first contact your Qi sales/support contact to retrieve an API download token.

Our API package is hosted by cloudsmith, and can be installed using the `pip` package manager by running the following command, ensuring that you replace `YOUR_DOWNLOAD_TOKEN` with the provided download token:

```
pip install \
  --extra-index-url=https://dl.cloudsmith.io/YOUR_DOWNLOAD_TOKEN/quant-insight/python/python/
↪index/ \
  qi-client
```

Note that this token faciltitates download and installation of the python API client package only, and that a separately issued *Qi API Key* is required to access Quant Insight data.

Once the client package has been installed, it can be imported as follows:

```
import qi_client
```

2.3 Upgrade

The package can be upgraded at a later time using the following command, again ensuring that you replace `YOUR_DOWNLOAD_TOKEN` with the provided download token:

```
pip install \
  --upgrade \
  --extra-index-url=https://dl.cloudsmith.io/YOUR_DOWNLOAD_TOKEN/quant-insight/python/python/
↪index/ \
  qi-client
```

In the event of update failure, please contact Qi support. We operate with expiration of cloudsmith tokens, and your previous download token may have expired. In this event, a new download token will be issued.

2.4 Getting Started

Please follow the steps outlined in *Installation*, and then run the following code example:

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorisation: Qi API Key.
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))
model = 'AAPL' # str | Numeric ID or name of the model to retrieve.
term = 'short term' # str | Which model term to retrieve. (optional)

try:
  # Get the specified model definition.
  api_response = api_instance.get_model(model, term=term)
  pprint(api_response)

except ApiException as e:
  # The ApiException class provides web API focused features, including instance
  # attributes for response status code and reason phrase, which can be leveraged
  # for your desired error handling approach.
  print(f"ApiException when calling DefaultApi: get_model: {e}")
  if e.status == 404:
    print("Not found... attempting to continue.")
    pass
  else:
    raise(e)

except Exception as e:
  print(f"Exception when calling DefaultApi: get_model: {e}")
  raise(e)
```

2.5 List of Available Methods

Class	Method	Operation
DefaultApi	check_excel_permission	Check whether user has permission to use excel add-in
DefaultApi	does_bucket_exist	Check for the existence of a bucket/driver group.
DefaultApi	does_model_exist	Check for the existence of a model.
DefaultApi	get_a2a_mapping	Get asset to asset mapping for a given watchlist id.
DefaultApi	get_bucket	Get bucket/driver group details.
DefaultApi	get_bucket_drivers	Get drivers for a given bucket/driver group.
DefaultApi	get_bucket_groups	Get bucket groups for a model.
DefaultApi	get_buckets	Get all available buckets/driver groups.
DefaultApi	get_buckets_by_asset_class_and_tags	Get a filtered list of buckets.
DefaultApi	get_buckets_by_models	Get a filtered list of buckets by models.
DefaultApi	get_driver	Get driver details.
DefaultApi	get_driver_zscore_histories	Get driver zscore histories within a certain bucket for a model.
DefaultApi	get_driver_zscore_moves	Get driver zscore moves for a model.
DefaultApi	get_drivers	Get driver details for a list of drivers.
DefaultApi	get_drivers_covariance_matrix	Get drivers covariance matrix for a specified model.
DefaultApi	get_instrument	Get serialized instrument definition.
DefaultApi	get_instruments	Get a list of all instruments.
DefaultApi	get_model	Get the specified model definition.
DefaultApi	get_model_data_csv_file	Get model data summary export file in csv format
DefaultApi	get_model_factor_attribution	Get factor attribution for a model.
DefaultApi	get_model_history	Get model definition history.
DefaultApi	get_model_momentum	Get momentum data for a model's underlying asset.
DefaultApi	get_model_momentum_summary	Get momentum summary data for a model's underlying asset.
DefaultApi	get_model_sensitivities	Get sensitivities for a model.
DefaultApi	get_model_sensitivities_aggregated	Get model sensitivities and factor attributions
DefaultApi	get_model_sensitivities_paginated_one_day	Get paginated models sensitivity data for one day
DefaultApi	get_model_shared_with	Get model group/public visibility.
DefaultApi	get_model_signal_summary	Get valuation summary data for a model's underlying asset.
DefaultApi	get_model_summary	Get condensed summary information for a model.
DefaultApi	get_model_timeseries	Get model timeseries data.
DefaultApi	get_model_valuation_summary	Get valuation summary data for a model's underlying asset.
DefaultApi	get_models	Get list of all defined models.
DefaultApi	get_models_paginated	Get paginated list of all defined models.
DefaultApi	get_models_with_pagination	Get list of all defined models with pagination.
DefaultApi	get_taa_asset_groups	Get list of all taa groups.
DefaultApi	get_taa_timeseries	Get taa timeseries data
DefaultApi	get_tags	Get list of all defined tags.
DefaultApi	get_user_watchlists	List any watchlists available for a given user.
DefaultApi	get_vol_indicator	Get vol indicator.
DefaultApi	get_vol_indicator_timeseries	Get vol indicator timeseries.
DefaultApi	list_model_notification_rules	List any existing notifications for the given user/model.
DefaultApi	list_user_notification_rules	List all existing notifications for the given user.
DefaultApi	list_watchlist_notification_rules	List any existing notifications for the given user/watchlist.
DefaultApi	send_a2a_mapping_request	Get asset to asset mapping for a given watchlist id.

**All methods marked with an asterisk exist for use with custom models and custom driver groups, which are not relevant for many use-cases. They are included for completeness, though if you wish to know more, please contact your Qi sales/support agent.*

2.6 Documentation For Authorization

2.6.1 Qi API Key

- Type: API key
- API key parameter name: X-API-KEY
- Location: HTTP header

CHAPTER

THREE

API REFERENCE

3.1 check_excel_permission

check_excel_permission()

Check whether user has permission to use excel add-in

3.1.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

try:
    api_instance.check_excel_permission()
except ApiException as e:
    print(f"Exception when calling DefaultApi:check_excel_permission: {e}")
```

3.1.2 Parameters

This endpoint does not need any parameter.

3.1.3 Return type

void (empty response body)

3.1.4 Authorization

Qi API Key

3.2 does_bucket_exist

InlineResponse200 does_bucket_exist(bucket)

Check for the existence of a bucket/driver group.

3.2.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Name of the bucket/driver group whose existence is to be queried.
# Datatype: str
bucket = 'bucket_example'

try:
    api_response = api_instance.does_bucket_exist(bucket)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:does_bucket_exist: {e}")
```

3.2.2 Parameters

Name	Description
bucket [str]	Name of the bucket/driver group whose existence is to be queried.

3.2.3 Return type

InlineResponse200

3.2.4 Authorization

Qi API Key

3.3 does_model_exist

InlineResponse200 does_model_exist(model)

Check for the existence of a model.

3.3.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Name of the model whose existence is to be queried.
# Datatype: str
model = 'model_example'

try:
    api_response = api_instance.does_model_exist(model)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:does_model_exist: {e}")
```

3.3.2 Parameters

Name	Description
model [str]	Name of the model whose existence is to be queried.

3.3.3 Return type

InlineResponse200

3.3.4 Authorization

Qi API Key

3.4 get_a2a_mapping

object get_a2a_mapping(watchlist_id, user_id=user_id)

Get asset to asset mapping for a given watchlist id.

3.4.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model for which proxy data is to be retrieved.
# Datatype: str
watchlist_id = 'watchlist_id_example'

# User ID. (optional)
# Datatype: str
user_id = 'user_id_example'

try:
    api_response = api_instance.get_a2a_mapping(watchlist_id, user_id=user_id)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_a2a_mapping: {e}")
```

3.4.2 Parameters

Name	Description
watchlist_id [str]	Numeric ID or name of the model for which proxy data is to be retrieved.
user_id [str]	User ID.

3.4.3 Return type

object

3.4.4 Authorization

Qi API Key

3.5 get_bucket

Bucket get_bucket(bucket, numeric_id=numeric_id)

Get bucket/driver group details.

3.5.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the bucket to retrieve.
# Datatype: str
bucket = 'bucket_example'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false

try:
    api_response = api_instance.get_bucket(bucket, numeric_id=numeric_id)
    pprint(api_response)
```

(continues on next page)

(continued from previous page)

```
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_bucket: {e}")
```

3.5.2 Parameters

Name	Description
bucket [str]	Numeric ID or name of the bucket to retrieve.
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.5.3 Return type

Bucket

3.5.4 Authorization

Qi API Key

3.6 get_bucket_drivers

list[Driver] get_bucket_drivers(bucket, numeric_id=numeric_id, asset_class=asset_class, tags=tags, model_type=model_type, include_inactive=include_inactive, collapsed=collapsed)

Get drivers for a given bucket/driver group.

3.6.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the bucket to examine.
# Datatype: str
bucket = 'bucket_example'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
```

(continues on next page)

(continued from previous page)

```
# Datatype: bool
numeric_id = false

# Return bucket drivers that are only associated with this asset class. (optional)
# Datatype: str
asset_class = 'asset_class_example'

# Comma delimited string containing tags with which to filter results. (optional)
# Datatype: str
tags = 'tags_example'

# Model type - QI, Custom, Both. (optional)
# Datatype: str
model_type = 'model_type_example'

# If true, results will include buckets which are not used in any active models. (optional)
# Datatype: bool
include_inactive = true

# If true, buckets containing grouped drivers may be returned as one, and buckets with yearly
↪ranges will come back in Bucket::Period format. (optional)
# Datatype: bool
collapsed = true

try:
    api_response = api_instance.get_bucket_drivers(bucket, numeric_id=numeric_id, asset_
↪class=asset_class, tags=tags, model_type=model_type, include_inactive=include_inactive,
↪collapsed=collapsed)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_bucket_drivers: {e}")
```

3.6.2 Parameters

Name	Description
bucket [str]	Numeric ID or name of the bucket to examine.
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.
asset_class [str]	Return bucket drivers that are only associated with this asset class.
tags [str]	Comma delimited string containing tags with which to filter results.
model_type [str]	Model type - QI, Custom, Both.
include_inactive [bool]	If true, results will include buckets which are not used in any active models.
collapsed [bool]	If true, buckets containing grouped drivers may be returned as one, and buckets with yearly ranges will come back in Bucket::Period format.

3.6.3 Return type

list[Driver]

3.6.4 Authorization

Qi API Key

3.7 get_bucket_groups

list[BucketGroup] get_bucket_groups(model)

Get bucket groups for a model.

3.7.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Name of the model to get bucket groups from
# Datatype: str
model = 'model_example'

try:
    api_response = api_instance.get_bucket_groups(model)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_bucket_groups: {e}")
```

3.7.2 Parameters

Name	Description
model [str]	Name of the model to get bucket groups from

3.7.3 Return type

list[BucketGroup]

3.7.4 Authorization

Qi API Key

3.8 get_buckets

> list[Bucket] get_buckets(asset_class=asset_class, include_inactive=include_inactive, org_names=org_names)

Get all available buckets/driver groups.

3.8.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Return buckets that are only associated with this asset_class (optional)
# Datatype: str
asset_class = 'asset_class_example'

# If true, results will include buckets which are not used in any active models. (optional)
# Datatype: bool
include_inactive = true

# Comma delimited list of org names to filter results with. (optional)
# Datatype: str
org_names = 'org_names_example'

try:
    api_response = api_instance.get_buckets(asset_class=asset_class, include_inactive=include_
↪inactive, org_names=org_names)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_buckets: {e}")
```

3.8.2 Parameters

Name	Description
asset_class [str]	Return buckets that are only associated with this asset_class
include_inactive [bool]	If true, results will include buckets which are not used in any active models.
org_names [str]	Comma delimited list of org names to filter results with.

3.8.3 Return type

list[Bucket]

3.8.4 Authorization

Qi API Key

3.9 get_buckets_by_asset_class_and_tags

list[Bucket] get_buckets_by_asset_class_and_tags(tags, model_type, asset_class=asset_class, tags_filter=tags_filter, include_inactive=include_inactive, org_names=org_names)

Get a filtered list of buckets.

3.9.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Comma delimited string containing tags with which to filter results. Results must contain
↪*all* tags specified.
# Datatype: str
tags = 'tags_example'

# Model type - QI, Custom, Both.
# Datatype: str
model_type = 'model_type_example'

# Asset class by which to filter. (optional)
```

(continues on next page)

(continued from previous page)

```
# Datatype: str
asset_class = 'asset_class_example'

# Filter results either by \"all\" or \"any\" tags specified. (optional)
# Datatype: str
tags_filter = 'tags_filter_example'

# If true, results will include buckets which are not used in any active models. (optional)
# Datatype: bool
include_inactive = true

# Comma delimited list of org names to filter results with. (optional)
# Datatype: str
org_names = 'org_names_example'

try:
    api_response = api_instance.get_buckets_by_asset_class_and_tags(tags, model_type, asset_
↪class=asset_class, tags_filter=tags_filter, include_inactive=include_inactive, org_names=org_
↪names)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_buckets_by_asset_class_and_tags: {e}")
```

3.9.2 Parameters

Name	Description
tags [str]	Comma delimited string containing tags with which to filter results. Results must contain *all* tags specified.
model_type [str]	Model type - QI, Custom, Both.
asset_class [str]	Asset class by which to filter.
tags_filter [str]	Filter results either by "all" or "any" tags specified.
include_inactive [bool]	If true, results will include buckets which are not used in any active models.
org_names [str]	Comma delimited list of org names to filter results with.

3.9.3 Return type

list[Bucket]

3.9.4 Authorization

Qi API Key

3.10 get_buckets_by_models

list[Bucket] get_buckets_by_models(models, include_inactive=include_inactive, org_names=org_names)

Get a filtered list of buckets by models.

3.10.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# List of models.
# Datatype: list[str]
models = [qi_client.list[str]()]

# If true, results will include buckets which are not used in any active models. (optional)
# Datatype: bool
include_inactive = true

# Comma delimited list of org names to filter results with. (optional)
# Datatype: str
org_names = 'org_names_example'

try:
    api_response = api_instance.get_buckets_by_models(models, include_inactive=include_inactive,
↪org_names=org_names)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_buckets_by_models: {e}")
```

3.10.2 Parameters

Name	Description
models [list[str]]	List of models.
include_inactive [bool]	If true, results will include buckets which are not used in any active models.
org_names [str]	Comma delimited list of org names to filter results with.

3.10.3 Return type

list[Bucket]

3.10.4 Authorization

Qi API Key

3.11 get_driver

Driver get_driver(driver, numeric_id=numeric_id)

Get driver details.

3.11.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the driver to examine
# Datatype: str
driver = 'driver_example'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false

try:
    api_response = api_instance.get_driver(driver, numeric_id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_driver: {e}")
```

3.11.2 Parameters

Name	Description
driver [str]	Numeric ID or name of the driver to examine
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.11.3 Return type

Driver

3.11.4 Authorization

Qi API Key

3.12 get_driver_zscore_histories

list[DriverZscoreHistory] get_driver_zscore_histories(model, bucket)

Get driver zscore histories within a certain bucket for a model.

3.12.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Name of the model to get the driver zscore histories from
# Datatype: str
model = 'model_example'

# Name of the bucket to get the drivers from
# Datatype: str
bucket = 'bucket_example'

try:
    api_response = api_instance.get_driver_zscore_histories(model, bucket)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_driver_zscore_histories: {e}")
```

3.12.2 Parameters

Name	Description
model [str]	Name of the model to get the driver zscore histories from
bucket [str]	Name of the bucket to get the drivers from

3.12.3 Return type

list[DriverZscoreHistory]

3.12.4 Authorization

Qi API Key

3.13 get_driver_zscore_moves

list[DriverZscoreMoves] get_driver_zscore_moves(model)

Get driver zscore moves for a model.

3.13.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Name of the model to get the driver zscores from
# Datatype: str
model = 'model_example'

try:
    api_response = api_instance.get_driver_zscore_moves(model)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_driver_zscore_moves: {e}")
```

3.13.2 Parameters

Name	Description
model [str]	Name of the model to get the driver zscores from

3.13.3 Return type

list[DriverZscoreMoves]

3.13.4 Authorization

Qi API Key

3.14 get_drivers

list[Driver] get_drivers(include_inactive=include_inactive, pattern=pattern)

Get driver details for a list of drivers.

3.14.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# If true, results will include drivers which are not used in any active models. (optional)
# Datatype: bool
include_inactive = true

# Optional pattern against which drivers will be filtered. (Uses driver long name). (optional)
# Datatype: str
pattern = 'pattern_example'

try:
    api_response = api_instance.get_drivers(include_inactive=include_inactive, pattern=pattern)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_drivers: {e}")
```

3.14.2 Parameters

Name	Description
include_inactive [bool]	If true, results will include drivers which are not used in any active models.
pattern [str]	Optional pattern against which drivers will be filtered. (Uses driver long name).

3.14.3 Return type

list[Driver]

3.14.4 Authorization

Qi API Key

3.15 get_drivers_covariance_matrix

> object get_drivers_covariance_matrix(model, date_from=date_from, date_to=date_to, term=term)

Get drivers covariance matrix for a specified model.

3.15.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Model name of the model drivers to retrieve
# Datatype: str
model = 'model_example'

# Date in YYYY-MM-DD format from which to build new values. (optional)
# Datatype: date
date_from = '2013-10-20'

# Date in YYYY-MM-DD format which to build new values until. (optional)
# Datatype: date
date_to = '2013-10-20'
```

(continues on next page)

(continued from previous page)

```
# Which model term to retrieve. (optional)
# Datatype: str
term = 'term_example'

try:
    api_response = api_instance.get_drivers_covariance_matrix(model, date_from=date_from, date_
↪to=date_to, term=term)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_drivers_covariance_matrix: {e}")
```

3.15.2 Parameters

Name	Description
model [str]	Model name of the model drivers to retrieve
date_from [date]	Date in YYYY-MM-DD format from which to build new values.
date_to [date]	Date in YYYY-MM-DD format which to build new values until.
term [str]	Which model term to retrieve.

3.15.3 Return type

object

3.15.4 Authorization

Qi API Key

3.16 get_instrument

Instrument get_instrument(instrument, numeric_id=numeric_id)

Get serialized instrument definition.

3.16.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
```

(continues on next page)

(continued from previous page)

```
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the instrument to retrieve.
# Datatype: str
instrument = 'instrument_example'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false

try:
    api_response = api_instance.get_instrument(instrument, numeric_id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_instrument: {e}")
```

3.16.2 Parameters

Name	Description
instrument [str]	Numeric ID or name of the instrument to retrieve.
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.16.3 Return type

Instrument

3.16.4 Authorization

Qi API Key

3.17 get_instruments

list[Instrument] get_instruments()

Get a list of all instruments.

3.17.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

try:
    api_response = api_instance.get_instruments()
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_instruments: {e}")
```

3.17.2 Parameters

This endpoint does not need any parameter.

3.17.3 Return type

list[Instrument]

3.17.4 Authorization

Qi API Key

3.18 get_model

Model get_model(model, term=term, version=version, numeric_id=numeric_id)

Get the specified model definition.

3.18.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model to retrieve.
# Datatype: str
model = 'model_example'

# Which model term to retrieve. (optional)
# Datatype: str
term = 'term_example'

# Version of the model to be retrieved. (optional)
# Datatype: int
version = 56

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false

try:
    api_response = api_instance.get_model(model, term=term, version=version, numeric_id=numeric_
↪id)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model: {e}")
```

3.18.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model to retrieve.
term [str]	Which model term to retrieve.
version [int]	Version of the model to be retrieved.
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.18.3 Return type

Model

3.18.4 Authorization

Qi API Key

3.19 get_model_data_csv_file

object get_model_data_csv_file(model, watchlist=watchlist, date_from=date_from, date_to=date_to)

Get model data summary export file in csv format

3.19.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Comma delimited list of models.
# Datatype: str
model = 'model_example'

# Name of a watchlist for the list of models. (optional)
# Datatype: str
watchlist = 'watchlist_example'

# Start of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
date_from = '2013-10-20'

# End of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
date_to = '2013-10-20'

try:
    api_response = api_instance.get_model_data_csv_file(model, watchlist=watchlist, date_
↪from=date_from, date_to=date_to)
    pprint(api_response)
```

(continues on next page)

(continued from previous page)

```
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_data_csv_file: {e}")
```

3.19.2 Parameters

Name	Description
model [str]	Comma delimited list of models.
watchlist [str]	Name of a watchlist for the list of models.
date_from [date]	Start of data date range, in YYYY-MM-DD format.
date_to [date]	End of data date range, in YYYY-MM-DD format.

3.19.3 Return type

object

3.19.4 Authorization

Qi API Key

3.20 get_model_factor_attribution

object get_model_factor_attribution(model, date_from=date_from, date_to=date_to, term=term, numeric_id=numeric_id, bucket_factors=bucket_factors)

Get factor attribution for a model.

3.20.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model for which sensitivities data is to be retrieved.
# Datatype: str
model = 'model_example'
```

(continues on next page)

(continued from previous page)

```
# Start of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
date_from = '2013-10-20'

# End of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
date_to = '2013-10-20'

# Which model term to retrieve. (optional)
# Datatype: str
term = 'term_example'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false

# If set to true, will return the bucket attribution. (optional) (default to true)
# Datatype: bool
bucket_factors = true

try:
    api_response = api_instance.get_model_factor_attribution(model, date_from=date_from, date_
↪to=date_to, term=term, numeric_id=numeric_id, bucket_factors=bucket_factors)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_factor_attribution: {e}")
```

3.20.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model for which sensitivities data is to be retrieved.
date_from [date]	Start of data date range, in YYYY-MM-DD format.
date_to [date]	End of data date range, in YYYY-MM-DD format.
term [str]	Which model term to retrieve.
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.
bucket_factors [bool]	If set to true, will return the bucket attribution.

3.20.3 Return type

object

3.20.4 Authorization

Qi API Key

3.21 get_model_history

list[ModelRevision] get_model_history(model, term=term, numeric_id=numeric_id)

Get model definition history.

3.21.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model for which history data is to be retrieved.
# Datatype: str
model = 'model_example'

# Which model term to retrieve. (optional)
# Datatype: str
term = 'term_example'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false

try:
    api_response = api_instance.get_model_history(model, term=term, numeric_id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_history: {e}")
```

3.21.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model for which history data is to be retrieved.
term [str]	Which model term to retrieve.
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.21.3 Return type

list[ModelRevision]

3.21.4 Authorization

Qi API Key

3.22 get_model_momentum

object get_model_momentum(model, term, date_from=date_from, date_to=date_to, numeric_id=numeric_id)

Get momentum data for a model's underlying asset.

3.22.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model to retrieve timeseries data for.
# Datatype: str
model = 'model_example'

# Momentum z-score term, i.e. 'short term' or 'long term'
# Datatype: date
term = '2013-10-20'

# Start of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
```

(continues on next page)

(continued from previous page)

```
date_from = '2013-10-20'

# End of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
date_to = '2013-10-20'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false

try:
    api_response = api_instance.get_model_momentum(model, term, date_from=date_from, date_
↪to=date_to, numeric_id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_momentum: {e}")
```

3.22.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model to retrieve timeseries data for.
term [date]	Momentum z-score term, i.e. 'short term' or 'long term'
date_from [date]	Start of data date range, in YYYY-MM-DD format.
date_to [date]	End of data date range, in YYYY-MM-DD format.
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.22.3 Return type

object

3.22.4 Authorization

Qi API Key

3.23 get_model_momentum_summary

object get_model_momentum_summary(model, _date=_date, numeric_id=numeric_id)

Get momentum summary data for a model's underlying asset.

3.23.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model to retrieve timeseries data for.
# Datatype: str
model = 'model_example'

# Date for summary, in YYYY-MM-DD format. (optional)
# Datatype: date
_date = '2013-10-20'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false

try:
    api_response = api_instance.get_model_momentum_summary(model, _date=_date, numeric_
↪id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_momentum_summary: {e}")
```

3.23.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model to retrieve timeseries data for.
_date [date]	Date for summary, in YYYY-MM-DD format.
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.23.3 Return type

object

3.23.4 Authorization

Qi API Key

3.24 get_model_sensitivities

object get_model_sensitivities(model, date_from=date_from, date_to=date_to, term=term, numeric_id=numeric_id, compact=compact, version=version)

Get sensitivities for a model.

3.24.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model for which sensitivities data is to be retrieved.
# Datatype: str
model = 'model_example'

# Start of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
date_from = '2013-10-20'

# End of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
date_to = '2013-10-20'

# Which model term to retrieve. (optional)
# Datatype: str
term = 'term_example'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false
```

(continues on next page)

(continued from previous page)

```
# If set to true, will return a smaller data format containing the results. (optional) (default
↪to false)
# Datatype: bool
compact = false

# Version of the model for which data is to be retrieved. (optional)
# Datatype: int
version = 56

try:
    api_response = api_instance.get_model_sensitivities(model, date_from=date_from, date_to=date_
↪to, term=term, numeric_id=numeric_id, compact=compact, version=version)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_sensitivities: {e}")
```

3.24.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model for which sensitivities data is to be retrieved.
date_from [date]	Start of data date range, in YYYY-MM-DD format.
date_to [date]	End of data date range, in YYYY-MM-DD format.
term [str]	Which model term to retrieve.
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.
compact [bool]	If set to true, will return a smaller data format containing the results.
version [int]	Version of the model for which data is to be retrieved.

3.24.3 Return type

object

3.24.4 Authorization

Qi API Key

3.25 get_model_sensitivities_aggregated

> list[object] get_model_sensitivities_aggregated(models, date_from=date_from, date_to=date_to, term=term, bucket_factors=bucket_factors)

Get model sensitivities and factor attributions

3.25.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# List of models.
# Datatype: list[str]
models = [qi_client.list[str]()]

# Start of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
date_from = '2013-10-20'

# End of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
date_to = '2013-10-20'

# Which model term to retrieve. (optional)
# Datatype: str
term = 'term_example'

# If set to true, will return the bucket attribution. (optional) (default to true)
# Datatype: bool
bucket_factors = true

try:
    api_response = api_instance.get_model_sensitivities_aggregated(models, date_from=date_from,
↪date_to=date_to, term=term, bucket_factors=bucket_factors)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_sensitivities_aggregated: {e}")
```

3.25.2 Parameters

Name	Description
models [list[str]]	List of models.
date_from [date]	Start of data date range, in YYYY-MM-DD format.
date_to [date]	End of data date range, in YYYY-MM-DD format.
term [str]	Which model term to retrieve.
bucket_factors [bool]	If set to true, will return the bucket attribution.

3.25.3 Return type

list[object]

3.25.4 Authorization

Qi API Key

3.26 get_model_sensitivities_paginated_one_day

object get_model_sensitivities_paginated_one_day(models=models, target_date=target_date, tags=tags, asset_classes=asset_classes, term=term, model_count=model_count, exclusive_start_key=exclusive_start_key)

Get paginated models sensitivity data for one day

3.26.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Comma delimited string containing models with which to filter results. (optional)
# Datatype: str
models = 'models_example'

# date of data required YYYY-MM-DD format. (optional)
# Datatype: date
target_date = '2013-10-20'

# Comma delimited string containing tags with which to filter results. Results must contain
↪*all* tags specified. (optional)
# Datatype: str
tags = 'tags_example'

# Comma delimited list of asset classes with which to filter results. Results must contain *any*
↪asset class specified. (optional)
# Datatype: str
asset_classes = 'asset_classes_example'

# Parameter of the models. (optional)
```

(continues on next page)

(continued from previous page)

```
# Datatype: str
term = 'term_example'

# Maximum number of models for which data is returned (page count). (optional)
# Datatype: int
model_count = 56

# Key to use to denote beginning of page. (optional)
# Datatype: str
exclusive_start_key = 'exclusive_start_key_example'

try:
    api_response = api_instance.get_model_sensitivities_paginated_one_day(models=models, target_
↪date=target_date, tags=tags, asset_classes=asset_classes, term=term, model_count=model_count,
↪exclusive_start_key=exclusive_start_key)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_sensitivities_paginated_one_day: {e}")
```

3.26.2 Parameters

Name	Description
models [str]	Comma delimited string containing models with which to filter results.
target_date [date]	date of data required YYYY-MM-DD format.
tags [str]	Comma delimited string containing tags with which to filter results. Results must contain *all* tags specified.
asset_classes [str]	Comma delimited list of asset classes with which to filter results. Results must contain *any* asset class specified.
term [str]	Parameter of the models.
model_count [int]	Maximum number of models for which data is returned (page count).
exclusive_start_key [str]	Key to use to denote beginning of page.

3.26.3 Return type

object

3.26.4 Authorization

Qi API Key

3.27 get_model_shared_with

Share get_model_shared_with(model, numeric_id=numeric_id, term=term)

Get model group/public visibility.

3.27.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Name of the model.
# Datatype: str
model = 'model_example'

# (DEPRECATED v0.9.24) If set to true, will consider identifier as a numeric ID - not as a name.
↪(optional) (default to false)
# Datatype: bool
numeric_id = false

# (DEPRECATED v0.9.24) Which model term to retrieve. (optional)
# Datatype: str
term = 'term_example'

try:
    api_response = api_instance.get_model_shared_with(model, numeric_id=numeric_id, term=term)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_shared_with: {e}")
```

3.27.2 Parameters

Name	Description
model [str]	Name of the model.
numeric_id [bool]	(DEPRECATED v0.9.24) If set to true, will consider identifier as a numeric ID - not as a name.
term [str]	(DEPRECATED v0.9.24) Which model term to retrieve.

3.27.3 Return type

Share

3.27.4 Authorization

Qi API Key

3.28 get_model_signal_summary

object get_model_signal_summary(model, _date=_date, term=term, numeric_id=numeric_id)

Get valuation summary data for a model's underlying asset.

3.28.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model to retrieve timeseries data for.
# Datatype: str
model = 'model_example'

# Date for summary, in YYYY-MM-DD format. (optional)
# Datatype: date
_date = '2013-10-20'

# Model term variant, i.e. 'short term' or 'long term' (optional)
# Datatype: date
term = '2013-10-20'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false

try:
    api_response = api_instance.get_model_signal_summary(model, _date=_date, term=term, numeric_
↪id=numeric_id)
    pprint(api_response)
```

(continues on next page)

(continued from previous page)

```
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_signal_summary: {e}")
```

3.28.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model to retrieve timeseries data for.
_date [date]	Date for summary, in YYYY-MM-DD format.
term [date]	Model term variant, i.e. 'short term' or 'long term'
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.28.3 Return type

object

3.28.4 Authorization

Qi API Key

3.29 get_model_summary

ModelSummaryCard get_model_summary(model, term=term, include_condensed=include_condensed, include_timeseries=include_timeseries, include_sensitivity=include_sensitivity, numeric_id=numeric_id)

Get condensed summary information for a model.

3.29.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model for which summary data is to be retrieved.
```

(continues on next page)

(continued from previous page)

```
# Datatype: str
model = 'model_example'

# Which model term to retrieve. (optional)
# Datatype: str
term = 'term_example'

# If set to true, result will be populated with condensed model summary information. (optional)
# Datatype: bool
include_condensed = true

# If set to true, result will be populated with model timeseries information. (optional)
# Datatype: bool
include_timeseries = true

# If set to true, result will be populated with model sensitivity information. (optional)
# Datatype: bool
include_sensitivity = true

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false

try:
    api_response = api_instance.get_model_summary(model, term=term, include_condensed=include_
↪condensed, include_timeseries=include_timeseries, include_sensitivity=include_sensitivity,
↪numeric_id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_summary: {e}")
```

3.29.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model for which summary data is to be retrieved.
term [str]	Which model term to retrieve.
include_condensed [bool]	If set to true, result will be populated with condensed model summary information.
include_timeseries [bool]	If set to true, result will be populated with model timeseries information.
include_sensitivity [bool]	If set to true, result will be populated with model sensitivity information.
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.29.3 Return type

ModelSummaryCard

3.29.4 Authorization

Qi API Key

3.30 get_model_timeseries

list[RegressionEntry] get_model_timeseries(model, date_from=date_from, date_to=date_to, term=term, numeric_id=numeric_id, version=version)

Get model timeseries data.

3.30.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model to retrieve timeseries data for.
# Datatype: str
model = 'model_example'

# Start of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
date_from = '2013-10-20'

# End of data date range, in YYYY-MM-DD format. (optional)
# Datatype: date
date_to = '2013-10-20'

# Which model term to retrieve (optional)
# Datatype: str
term = 'term_example'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false
```

(continues on next page)

(continued from previous page)

```
# Which version of the model to retrieve data for. (optional)
# Datatype: int
version = 56

try:
    api_response = api_instance.get_model_timeseries(model, date_from=date_from, date_to=date_to,
↪ term=term, numeric_id=numeric_id, version=version)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_timeseries: {e}")
```

3.30.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model to retrieve timeseries data for.
date_from [date]	Start of data date range, in YYYY-MM-DD format.
date_to [date]	End of data date range, in YYYY-MM-DD format.
term [str]	Which model term to retrieve
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.
version [int]	Which version of the model to retrieve data for.

3.30.3 Return type

list[RegressionEntry]

3.30.4 Authorization

Qi API Key

3.31 get_model_valuation_summary

object get_model_valuation_summary(model, _date=_date, term=term, numeric_id=numeric_id)

Get valuation summary data for a model's underlying asset.

3.31.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model to retrieve timeseries data for.
# Datatype: str
model = 'model_example'

# Date for summary, in YYYY-MM-DD format. (optional)
# Datatype: date
_date = '2013-10-20'

# Model term variant, i.e. 'short term' or 'long term' (optional)
# Datatype: date
term = '2013-10-20'

# If set to true, will consider identifier as a numeric ID - not as a name. (optional) (default
↪to false)
# Datatype: bool
numeric_id = false

try:
    api_response = api_instance.get_model_valuation_summary(model, _date=_date, term=term,
↪numeric_id=numeric_id)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_model_valuation_summary: {e}")
```

3.31.2 Parameters

Name	Description
model [str]	Numeric ID or name of the model to retrieve timeseries data for.
_date [date]	Date for summary, in YYYY-MM-DD format.
term [date]	Model term variant, i.e. 'short term' or 'long term'
numeric_id [bool]	If set to true, will consider identifier as a numeric ID - not as a name.

3.31.3 Return type

object

3.31.4 Authorization

Qi API Key

3.32 get_models

list[ModelSummaryLite] get_models(tags=tags, asset_classes=asset_classes, security_names=security_names)

Get list of all defined models.

3.32.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Comma delimited string containing tags with which to filter results. Results must contain
↪*all* tags specified. (optional)
# Datatype: str
tags = 'tags_example'

# Comma delimited list of asset classes with which to filter results. Results must contain *all*
↪asset classes specified. (optional)
# Datatype: str
asset_classes = 'asset_classes_example'

# Comma delimited list of all security_names (human readable instrument name) with which to
↪filter results. Results must contain *all* security_names specified. (optional)
# Datatype: str
security_names = 'security_names_example'

try:
    api_response = api_instance.get_models(tags=tags, asset_classes=asset_classes, security_
↪names=security_names)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_models: {e}")
```

3.32.2 Parameters

Name	Description
tags [str]	Comma delimited string containing tags with which to filter results. Results must contain *all* tags specified.
asset_classes [str]	Comma delimited list of asset classes with which to filter results. Results must contain *all* asset classes specified.
security_names [str]	Comma delimited list of all security_names (human readable instrument name) with which to filter results. Results must contain *all* security_names specified.

3.32.3 Return type

list[ModelSummaryLite]

3.32.4 Authorization

Qi API Key

3.33 get_models_paginated

> ModelSummaryPaginated get_models_paginated(tags=tags, tags_filter=tags_filter, asset_classes=asset_classes, security_names=security_names, models=models, drivers=drivers, fvg_min=fvg_min, fvg_max=fvg_max, fvg_type=fvg_type, term=term, confidence_min=confidence_min, confidence_max=confidence_max, sigma_min=sigma_min, sigma_max=sigma_max, sigma_type=sigma_type, offset=offset, limit=limit, orderby=orderby, ordertype=ordertype, status=status, type=type, collapsed=collapsed, group_filter=group_filter)

Get paginated list of all defined models.

3.33.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# List of tags with which to filter results. (optional)
# Datatype: list[str]
```

(continues on next page)

(continued from previous page)

```
tags = [qi_client.list[str]()]

# Filter results either by \"all\" or \"any\" tags specified. (optional)
# Datatype: str
tags_filter = 'tags_filter_example'

# List of asset classes with which to filter results. Results must contain *all* asset classes
↪specified. (optional)
# Datatype: list[str]
asset_classes = [qi_client.list[str]()]

# List of all security_names (human readable instrument name) which which to filter results.
↪Results must contain *all* security_names specified. (optional)
# Datatype: list[str]
security_names = [qi_client.list[str]()]

# List of models to include. Results may contain *any* of the models specified. (optional)
# Datatype: list[Models]
models = [qi_client.Models()]

# Up to 3 drivers may be provided here. This will limit the result exclusively to models which
↪have any of these 3 drivers in their definition. (optional)
# Datatype: list[str]
drivers = [qi_client.list[str]()]

# Valuation Gap minimum value. (optional)
# Datatype: int
fvg_min = 56

# Valuation Gap maximum value. (optional)
# Datatype: int
fvg_max = 56

# Type of filter to apply using provided fvg_min & fvg_max. \"inside\" means between min & max. \
↪"outside\" means ranges outside of that between min & max, within the overall min and max of
↪Valuation (-3 to 3). This must be provided to enable fvg_min and fvg_max. (optional)
# Datatype: str
fvg_type = 'fvg_type_example'

# Term (or Timeframe) of model. (optional)
# Datatype: str
term = 'term_example'

# Confidence minimum value. (optional)
# Datatype: int
confidence_min = 56

# Confidence maximum value. (optional)
# Datatype: int
confidence_max = 56

# Sigma minimum value. (optional)
# Datatype: int
sigma_min = 56

# Sigma maximum value. (optional)
```

(continues on next page)

(continued from previous page)

```
# Datatype: int
sigma_max = 56

# Type of filter to apply using provided sigma_min & sigma_max. \"inside\" means between min &
↪max. \"outside\" means ranges outside of that between min & max, within the overall min and
↪max of Valuation (-3 to 3). This must be provided to enable sigma_min and sigma_max. (optional)
# Datatype: str
sigma_type = 'sigma_type_example'

# Pagination value to state the offset from start of results. (optional)
# Datatype: int
offset = 56

# Pagination value to state the size of a given page of results. (optional)
# Datatype: int
limit = 56

# String with which to order results by. String must be the name of one of the results fields.
↪Column \"confidence_delta_1m\" can also be ordered by ordertype \"magnitude\" (Please see
↪ordertype description). (optional)
# Datatype: str
orderby = 'orderby_example'

# State ascending or descending (Please Note Option \"magnitude\" is also available for orderby
↪column \"confidence_delta_1m\"). (optional)
# Datatype: str
ordertype = 'ordertype_example'

# Type of filter to output models that match the specified state.  Available states are ACTIVE,
↪DEACTIVATED, BUILDING and PENDING. (optional)
# Datatype: str
status = 'status_example'

# Type of filter to output based on permission type QI, CUSTOM, etc. (optional)
# Datatype: str
type = 'type_example'

# If true, buckets with similar drivers inside may be returned as one, and buckets with yearly
↪ranges will come back in Bucket::5y format. (optional)
# Datatype: bool
collapsed = true

# Filter by groups.  Provide a list of groups and type of filter to apply. (optional)
# Datatype: ModelsPaginatedGroupFilter
group_filter = qi_client.ModelsPaginatedGroupFilter()

try:
    api_response = api_instance.get_models_paginated(tags=tags, tags_filter=tags_filter, asset_
↪classes=asset_classes, security_names=security_names, models=models, drivers=drivers, fvg_
↪min=fvg_min, fvg_max=fvg_max, fvg_type=fvg_type, term=term, confidence_min=confidence_min,
↪confidence_max=confidence_max, sigma_min=sigma_min, sigma_max=sigma_max, sigma_type=sigma_type,
↪ offset=offset, limit=limit, orderby=orderby, ordertype=ordertype, status=status, type=type,
↪collapsed=collapsed, group_filter=group_filter)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_models_paginated: {e}")
```

3.33.2 Parameters

Name	Description
tags [list[str]]	List of tags with which to filter results.
tags_filter [str]	Filter results either by "all" or "any" tags specified.
asset_classes [list[str]]	List of asset classes with which to filter results. Results must contain *all* asset classes specified.
security_names [list[str]]	List of all security_names (human readable instrument name) which which to filter results. Results must contain *all* security_names specified.
models [list[Models]]	List of models to include. Results may contain *any* of the models specified.
drivers [list[str]]	Up to 3 drivers may be provided here. This will limit the result exclusively to models which have any of these 3 drivers in their definition.
fvg_min [int]	Valuation Gap minimum value.
fvg_max [int]	Valuation Gap maximum value.
fvg_type [str]	Type of filter to apply using provided fvg_min & fvg_max. "inside" means between min & max. "outside" means ranges outside of that between min & max, within the overall min and max of Valuation (-3 to 3). This must be provided to enable fvg_min and fvg_max.
term [str]	Term (or Timeframe) of model.
confidence_min [int]	Confidence minimum value.
confidence_max [int]	Confidence maximum value.
sigma_min [int]	Sigma minimum value.
sigma_max [int]	Sigma maximum value.
sigma_type [str]	Type of filter to apply using provided sigma_min & sigma_max. "inside" means between min & max. "outside" means ranges outside of that between min & max, within the overall min and max of Valuation (-3 to 3). This must be provided to enable sigma_min and sigma_max.
offset [int]	Pagination value to state the offset from start of results.
limit [int]	Pagination value to state the size of a given page of results.
orderby [str]	String with which to order results by. String must be the name of one of the results fields. Column "confidence_delta_1m" can also be ordered by ordertype "magnitude" (Please see ordertype description).
ordertype [str]	State ascending or descending (Please Note Option "magnitude" is also available for orderby column "confidence_delta_1m").
status [str]	Type of filter to output models that match the specified state. Available states are ACTIVE, DEACTIVATED, BUILDING and PENDING.
type [str]	Type of filter to output based on permission type QI, CUSTOM, etc.
collapsed [bool]	If true, buckets with similar drivers inside may be returned as one, and buckets with yearly ranges will come back in Bucket::5y format.
group_filter [ModelsPaginatedGroupFilter]	Filter by groups. Provide a list of groups and type of filter to apply.

3.33.3 Return type

ModelSummaryPaginated

3.33.4 Authorization

Qi API Key

3.34 get_models_with_pagination

> object get_models_with_pagination(tags=tags, asset_classes=asset_classes, security_names=security_names, term=term, exclusive_start_key=exclusive_start_key, count=count)

Get list of all defined models with pagination.

3.34.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Comma delimited string containing tags with which to filter results. Results must contain
↪*all* tags specified. (optional)
# Datatype: str
tags = 'tags_example'

# Comma delimited list of asset classes with which to filter results. Results must contain *all*
↪asset classes specified. (optional)
# Datatype: str
asset_classes = 'asset_classes_example'

# Comma delimited list of all security_names (human readable instrument name) with which to
↪filter results. Results must contain *all* security_names specified. (optional)
# Datatype: str
security_names = 'security_names_example'

# Which model term to retrieve. (optional)
# Datatype: str
term = 'term_example'
```

(continues on next page)

(continued from previous page)

```
# Key to the next page, which is the model ID of the last item in the page (optional)
# Datatype: int
exclusive_start_key = 56

# Number of models to retrieve per page - the default is 10k and 5k for shot term or long term
↪(optional)
# Datatype: str
count = 'count_example'

try:
    api_response = api_instance.get_models_with_pagination(tags=tags, asset_classes=asset_
↪classes, security_names=security_names, term=term, exclusive_start_key=exclusive_start_key,
↪count=count)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_models_with_pagination: {e}")
```

3.34.2 Parameters

Name	Description
tags [str]	Comma delimited string containing tags with which to filter results. Results must contain *all* tags specified.
asset_classes [str]	Comma delimited list of asset classes with which to filter results. Results must contain *all* asset classes specified.
security_names [str]	Comma delimited list of all security_names (human readable instrument name) with which to filter results. Results must contain *all* security_names specified.
term [str]	Which model term to retrieve.
exclusive_start_key [int]	Key to the next page, which is the model ID of the last item in the page
count [str]	Number of models to retrieve per page - the default is 10k and 5k for shot term or long term

3.34.3 Return type

object

3.34.4 Authorization

Qi API Key

3.35 get_taa_asset_groups

list[TaaAssetGroups] get_taa_asset_groups()

Get list of all taa groups.

3.35.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

try:
    api_response = api_instance.get_taa_asset_groups()
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_taa_asset_groups: {e}")
```

3.35.2 Parameters

This endpoint does not need any parameter.

3.35.3 Return type

list[TaaAssetGroups]

3.35.4 Authorization

Qi API Key

3.36 get_taa_timeseries

TaaTimeseries get_taa_timeseries(asset_group)

Get taa timeseries data

3.36.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Name of the asset group for the taa
# Datatype: str
asset_group = 'asset_group_example'

try:
    api_response = api_instance.get_taa_timeseries(asset_group)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_taa_timeseries: {e}")
```

3.36.2 Parameters

Name	Description
asset_group [str]	Name of the asset group for the taa

3.36.3 Return type

TaaTimeseries

3.36.4 Authorization

Qi API Key

3.37 get_tags

list[Tag] get_tags(asset_classes=asset_classes)

Get list of all defined tags.

3.37.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Comma delimited list of asset classes with which to filter results. Results must contain *all*
↪asset classes specified. (optional)
# Datatype: str
asset_classes = 'asset_classes_example'

try:
    api_response = api_instance.get_tags(asset_classes=asset_classes)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_tags: {e}")
```

3.37.2 Parameters

Name	Description
asset_classes [str]	Comma delimited list of asset classes with which to filter results. Results must contain *all* asset classes specified.

3.37.3 Return type

list[Tag]

3.37.4 Authorization

Qi API Key

3.38 get_user_watchlists

object get_user_watchlists(user_id=user_id)

List any watchlists available for a given user.

3.38.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# User ID. (optional)
# Datatype: str
user_id = 'user_id_example'

try:
    api_response = api_instance.get_user_watchlists(user_id=user_id)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_user_watchlists: {e}")
```

3.38.2 Parameters

Name	Description
user_id [str]	User ID.

3.38.3 Return type

object

3.38.4 Authorization

Qi API Key

3.39 get_vol_indicator

VolIndicator get_vol_indicator()

Get vol indicator.

3.39.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

try:
    api_response = api_instance.get_vol_indicator()
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_vol_indicator: {e}")
```

3.39.2 Parameters

This endpoint does not need any parameter.

3.39.3 Return type

VolIndicator

3.39.4 Authorization

Qi API Key

3.40 get_vol_indicator_timeseries

list[VolIndicatorTimeseries] get_vol_indicator_timeseries()

Get vol indicator timeseries.

3.40.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

try:
    api_response = api_instance.get_vol_indicator_timeseries()
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:get_vol_indicator_timeseries: {e}")
```

3.40.2 Parameters

This endpoint does not need any parameter.

3.40.3 Return type

list[VolIndicatorTimeseries]

3.40.4 Authorization

Qi API Key

3.41 list_model_notification_rules

object list_model_notification_rules(model, user_id=user_id, user_email=user_email)

List any existing notifications for the given user/model.

3.41.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Name of model.
# Datatype: str
model = 'model_example'

# User ID override (Cognito username). Functionality restricted to Qi admins. (optional)
# Datatype: str
user_id = 'user_id_example'

# User e-mail override. Functionality restricted to Qi admins. (optional)
# Datatype: str
user_email = 'user_email_example'

try:
    api_response = api_instance.list_model_notification_rules(model, user_id=user_id, user_
↪email=user_email)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:list_model_notification_rules: {e}")
```

3.41.2 Parameters

Name	Description
model [str]	Name of model.
user_id [str]	User ID override (Cognito username). Functionality restricted to Qi admins.
user_email [str]	User e-mail override. Functionality restricted to Qi admins.

3.41.3 Return type

object

3.41.4 Authorization

Qi API Key

3.42 list_user_notification_rules

object list_user_notification_rules(user_id=user_id, user_email=user_email)

List all existing notifications for the given user.

3.42.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# User ID override (Cognito username). Functionality restricted to Qi admins. (optional)
# Datatype: str
user_id = 'user_id_example'

# User e-mail override. Functionality restricted to Qi admins. (optional)
# Datatype: str
user_email = 'user_email_example'

try:
    api_response = api_instance.list_user_notification_rules(user_id=user_id, user_email=user_
↪email)
```

(continues on next page)

(continued from previous page)

```
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:list_user_notification_rules: {e}")
```

3.42.2 Parameters

Name	Description
user_id [str]	User ID override (Cognito username). Functionality restricted to Qi admins.
user_email [str]	User e-mail override. Functionality restricted to Qi admins.

3.42.3 Return type

object

3.42.4 Authorization

Qi API Key

3.43 list_watchlist_notification_rules

object list_watchlist_notification_rules(watchlist, user_id=user_id, user_email=user_email, shared=shared)

List any existing notifications for the given user/watchlist.

3.43.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Watchlist ID.
# Datatype: str
watchlist = 'watchlist_example'
```

(continues on next page)

(continued from previous page)

```
# User ID override (Cognito username). Functionality restricted to Qi admins. (optional)
# Datatype: str
user_id = 'user_id_example'

# User e-mail override. Functionality restricted to Qi admins. (optional)
# Datatype: str
user_email = 'user_email_example'

# Flag for shared watchlist. (optional)
# Datatype: bool
shared = true

try:
    api_response = api_instance.list_watchlist_notification_rules(watchlist, user_id=user_id,
↪user_email=user_email, shared=shared)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:list_watchlist_notification_rules: {e}")
```

3.43.2 Parameters

Name	Description
watchlist [str]	Watchlist ID.
user_id [str]	User ID override (Cognito username). Functionality restricted to Qi admins.
user_email [str]	User e-mail override. Functionality restricted to Qi admins.
shared [bool]	Flag for shared watchlist.

3.43.3 Return type

object

3.43.4 Authorization

Qi API Key

3.44 send_a2a_mapping_request

object send_a2a_mapping_request(watchlist_id)

Get asset to asset mapping for a given watchlist id.

3.44.1 Examples

```
import qi_client
from qi_client.rest import ApiException
from pprint import pprint

# Configure API key authorization: Qi API Key
configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

# Uncomment to use proxy - please refer to Connectivity Guidance doc for more details.
# configuration.proxy = 'http://corporateproxy.business.com:8080'

# Instantiate API class.
api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))

# Numeric ID or name of the model for which proxy data is to be retrieved.
# Datatype: str
watchlist_id = 'watchlist_id_example'

try:
    api_response = api_instance.send_a2a_mapping_request(watchlist_id)
    pprint(api_response)
except ApiException as e:
    print(f"Exception when calling DefaultApi:send_a2a_mapping_request: {e}")
```

3.44.2 Parameters

Name	Description
watchlist_id [str]	Numeric ID or name of the model for which proxy data is to be retrieved.

3.44.3 Return type

object

3.44.4 Authorization

Qi API Key

Model List

3.45 Bucket

The model Bucket has the following properties:

Name	Description
drivers [list[Driver]]	List of drivers within this bucket
group [str]	The group which owns this bucket
id [int]	System-internal numeric identifier for this bucket / driver group.
name [str]	The name of this bucket
public [bool]	Whether the bucket is it be publically available or not.
state [str]	The state of whether this Bucket belongs to an ACTIVE or INACTIVE model
type [str]	Type of model based on its permissions for the querying user.

3.46 Driver

The model Driver has the following properties:

Name	Description
buckets [list[str]]	
deactivated [datetime]	
expression [str]	
id [int]	
name [str]	
short_name [str]	
status [str]	
timeseries_ety1 [TimeseriesEntity]	
timeseries_ety2 [object]	
timeseries_ety3 [object]	
timeseries_ety4 [object]	

3.47 Instrument

The model Instrument has the following properties:

Name	Description
asset_class [str]	
id [int]	
identifiers [object]	
is_future [bool]	
name [str]	
source [str]	
status [str]	

3.48 Model

The model Model has the following properties:

Name	Description
asset_class [str]	This is the name of the asset class to which this model belongs (e.g. Equity, FX etc).
calc_complete [datetime]	Last model calculation complete time [UTC].
created [datetime]	This is the date when this model was created.
definition [Combo]	This is the combination of instruments and arithmetic that gives this model its values.
drivers [list[object]]	This is the list of drivers which are expected to influence this model.
group [str]	The name of the group which this Model belongs to.
id [int]	System-internal numeric identifier for this Qi Model. This will change on update so please don't use this as an external identifier.
model_parameter [str]	Term for this parameter (e.g. 'short term' or 'long term').
name [str]	A unique name for this model. Any lookups for this model should be made according to this name.
notes [str]	This is a field which may be populated in the model to give a better description of its purpose, function and constitution.
public [bool]	The public viewing flag for this model.
security_name [str]	Human readable model name.
status [str]	This is the current state of the Model.
tags [list[str]]	List of tags associated with this model to assist with filtering.
type [str]	Type of model based on its permissions for the querying user.
version [int]	This is a numeric value for the current model revision.

CHAPTER

FOUR

CODE EXAMPLES

This section provides code examples that demonstrate common scenarios using Qi's Module for Python. Note that each example assumes that code similar to the following is at the top of each script:

```
import qi_client
import pandas
from datetime import datetime

configuration = qi_client.Configuration()
configuration.api_key['X-API-KEY'] = 'YOUR_API_KEY'

api_instance = qi_client.DefaultApi(qi_client.ApiClient(configuration))
```

The examples also make use of several standard, publicly available APIs. These APIs will need to be installed prior to running any of the examples. They may be installed using pip:

```
$ pip install matplotlib pandas
```

4.1 Historical R-Squared Data

Define a function to get a specified Model's R-Squared data for a given date range and model term.

```
def get_rsq(model, start, end, term):
    # Note that this may be  more than 1 year of data, so need to split requests
    year_start = int(start[:4])
    year_end = int(end[:4])
    time_series = []
    for year in range(year_start, year_end + 1):
        query_start = start
        if year != year_start:
            date_from = '%d-01-01' % year
        else:
            date_from = start
        if year != year_end:
            date_to = '%d-12-31' % year
        else:
            date_to = end

        print("Gathering data for %s from %s to %s..." % (model,
                                                          date_from,
                                                          date_to))
        time_series += api_instance.get_model_timeseries(
            model,
            date_from=date_from,
            date_to=date_to,
            term=term)

    rsq = [data.rsquare for data in time_series]
    dates = [data._date for data in time_series]

    df = pandas.DataFrame({'Dates': dates, 'Rsq': rsq})
    df.set_index('Dates', inplace=True)

    return df
```

Having defined a function, we can use it to graph the R-Squared timeseries for a specified model. In this example we plot:

SPX Index LT model

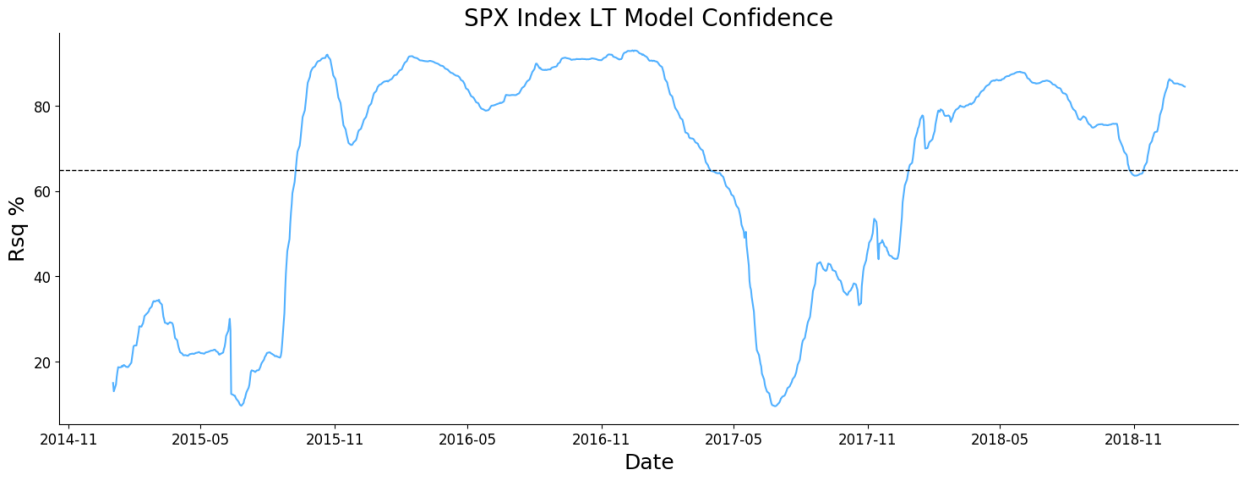
```
def example_historical_rsq():
    import matplotlib.pyplot as plt

    df = get_rsq('SPX',
                 '2015-01-01',
                 '2019-01-10',
                 'Long Term')

    fig = plt.figure(figsize=(18, 6))

    ax = plt.subplot(111)
    fig.patch.set_facecolor('#FFFFFF')

    plt.plot(df.index.values, df['Rsq'], color='#53B2FF')
```

(continues on next page)

(continued from previous page)

```
    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)

    plt.ylabel("Rsq %", fontsize=18)
    plt.xlabel('Date', fontsize=18)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)

    plt.title('SPX Index LT Model Confidence', fontsize=20)

    ax.axhline(65, lw=1, linestyle='--', color='k')

    plt.show()

    fig.savefig('Qi_API_Rsq_Graph_Example_(SPX_Index)',
                bbox_inches="tight",
                facecolor=fig.get_facecolor())
```

R-Squared can be interpreted as a measure of model confidence. From the end of 2015 to the middle of 2017 macro had strong explanatory power for US equities. The sharp fall in R-Squared in the middle of 2017 suggests other non-macro factors became more significant; these could potentially include momentum, sentiment, positioning, geopolitics etc. More recently the power of macro has re-asserted itself.



4.2 Historical Factor Sensitivities

Define a function to get a specified Model's factor sensitivity data for a given date range, factor and model term.

```
def get_sensitivities(model, factors, start, end, term):
    # Note that this may be  more than 1 year of data, so need to split requests
    year_start = int(start[:4])
    year_end = int(end[:4])
    time_series_sensitivities = {}
    for year in range(year_start, year_end + 1):
        query_start = start
        if year != year_start:
            date_from = '%d-01-01' % year
        else:
            date_from = start
        if year != year_end:
            date_to = '%d-12-31' % year
        else:
            date_to = end

        print("Gathering data for %s from %s to %s..." % (model,
                                                          date_from,
                                                          date_to))
        time_series_sensitivities.update(
            api_instance.get_model_sensitivities(
                model,
                date_from=date_from,
                date_to=date_to,
                term=term
            )
        )

    dates = []
    factor_results = {}
    # Intialize factor lists for results
    for factor in factors:
        factor_results[factor] = []

    # Iterate over each time series result
    for date in sorted(time_series_sensitivities.keys()):
        # Use the real date for the index
        dates.append(datetime.strptime(date, '%Y-%m-%d'))
        # This will be the sensitivities for this particular day
        daily_sensitivities = time_series_sensitivities[date]
        # Only consider factors requested
        for factor in factors:
            factor_result = [sensitivity['sensitivity']
                             for sensitivity in daily_sensitivities
                             if sensitivity['driver_short_name'] == factor]
            if len(factor_result) > 0:
                factor_results[factor].append(factor_result[0])
            else:
                # Pyplot will ignore nan values rather than shift datapoint left
                factor_results[factor].append(float('nan'))

    # Add date range for X axis first
```

(continues on next page)

(continued from previous page)

```
    dataframe_columns = {
        'Dates': dates
    }

    # Add factor specific columns
    for factor in factors:
        dataframe_columns[factor] = factor_results[factor]

    # Initialize dataframe
    df = pandas.DataFrame(dataframe_columns)
    df.set_index('Dates', inplace=True)

    return df
```

Having defined a function, we can use it to graph a comparison across a number of model factor sensitivities. In this example we plot:

USDJPY LT Model - Interest Rate Differential Sensitivities

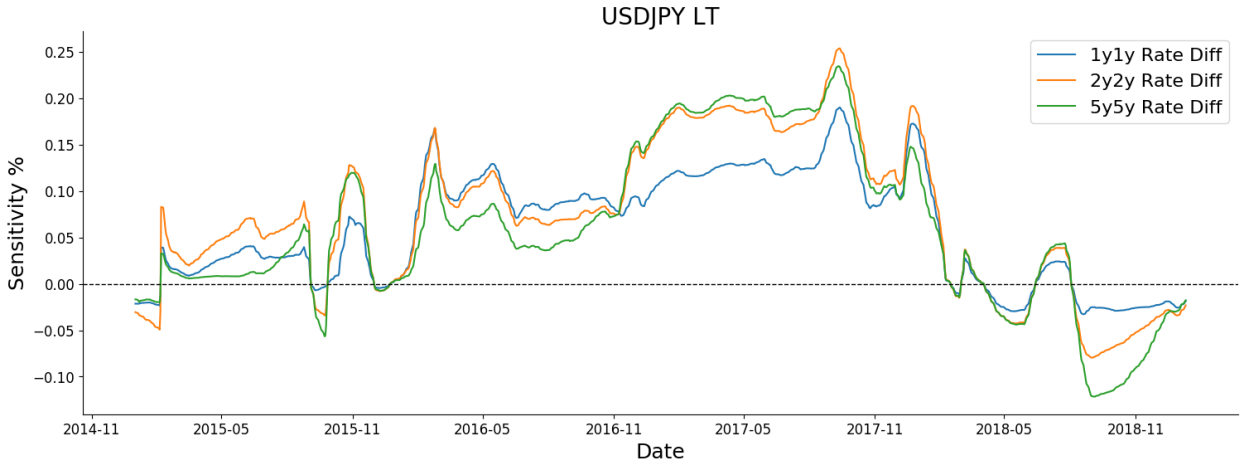
```
def example_historical_factor_sensitivities():
    import matplotlib.pyplot as plt

    df = get_sensitivities('USDJPY',
                           [
                               '1y1y Rate Diff.',
                               '2y2y Rate Diff.',
                               '5y5y Rate Diff.',
                           ],
                           '2015-01-01',
                           '2019-01-10',
                           'Long Term')

    fig = plt.figure(figsize=(18, 6))

    ax = plt.subplot(111)
    fig.patch.set_facecolor('#FFFFFF')

    plt.plot(df.index.values, df['1y1y Rate Diff.'], label='1y1y Rate Diff')
    plt.plot(df.index.values, df['2y2y Rate Diff.'], label='2y2y Rate Diff')
    plt.plot(df.index.values, df['5y5y Rate Diff.'], label='5y5y Rate Diff')

    plt.legend(loc='upper right', prop={'size': 16})

    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)

    plt.ylabel("Sensitivity %", fontsize=18)
    plt.xlabel('Date', fontsize=18)
    plt.xticks(fontsize=12)
    plt.yticks(fontsize=12)

    plt.title('USDJPY LT', fontsize=20)

    ax.axhline(0, lw=1, linestyle='--', color='k')

    plt.show()
```

(continues on next page)

(continued from previous page)

```
    fig.savefig('Qi_API Factor_Sensitivity_Graph_Example_(USDJPY)',
                bbox_inches="tight",
                facecolor=fig.get_facecolor())
```

Tracking USDJPY's sensitivity to interest rate differentials across different tenors. Notice that for most of 2017, FX was highly sensitive to changes in interest rates. Since the start of 2018 however, this relationship has deteriorated, and spot FX is currently indifferent to interest rate differentials.



4.3 Bucket Driver Sensitivities

Define a function to get a specified Model's bucket sensitivity data for a given date range and model term.

```
def get_bucket_drivers(model, date, term):
    sensitivity = api_instance.get_model_sensitivities(
                        model=model,
                        date_from=date,
                        date_to=date,
                        term=term
                    )

    df_sensitivities = pandas.DataFrame()
    for data in sensitivity[date]:
        if data['bucket_name'] in df_sensitivities.columns:
            df_sensitivities[str(data['bucket_name'])][0] = (
                df_sensitivities[str(data['bucket_name'])][0] +
                [data['sensitivity']]
            )
        else:
            df_sensitivities[str(data['bucket_name'])] = [data['sensitivity']]

    # Column heading
    df_sensitivities.index = ['Sensitivity']

    return df_sensitivities
```

Having defined a function, we can use it to graph the bucket driver sensitivities for a given model. In this example we plot:

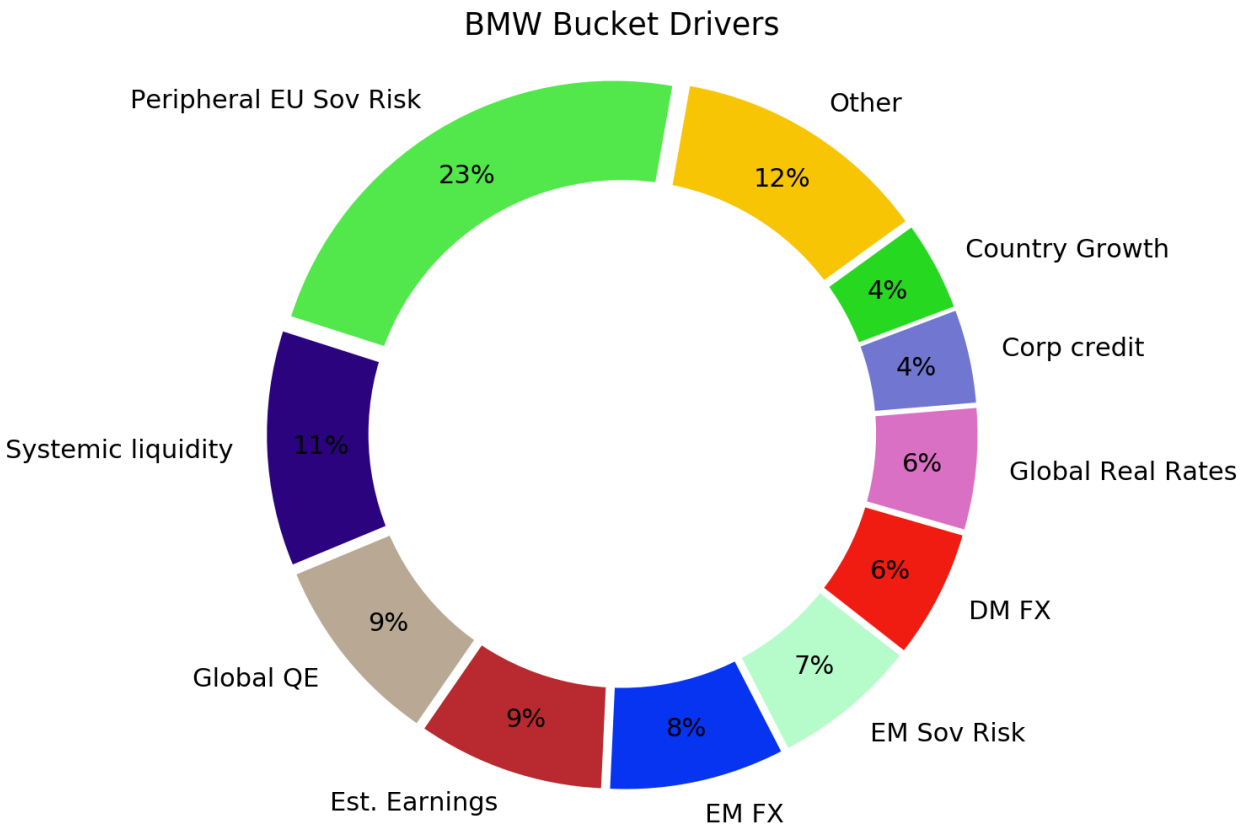
BMW LT model - bucket drivers

```
def example_bucket_driver_sensitivities():
    bucket_drivers = get_bucket_drivers('BMW',
                                        '2019-01-14',
                                        'Long Term')

    # Create Pie Data
    other = abs(bucket_drivers).transpose().nsmallest(5, 'Sensitivity').sum()
    df_other = pandas.DataFrame(other)
    df_other = df_other.rename(index={'Sensitivity': 'Other'},
                               columns={0: 'Sensitivity'})

    pie_bucket_drivers = abs(bucket_drivers).transpose().nlargest(10,
                                                                  'Sensitivity')
    pie_bucket_drivers = pie_bucket_drivers.append(df_other)

    # Create Colours
    import random

    number_of_colors = 11

    color = ["#"+''.join([random.choice('0123456789ABCDEF') for j in range(6)])
                 for i in range(number_of_colors)]

    df_color = pandas.DataFrame({'Color': color})
    df_color.index = pie_bucket_drivers.index.tolist()
```

(continues on next page)

(continued from previous page)

```
# Set up the plotting
import matplotlib as mpl
import matplotlib.pyplot as plt
mpl.rcParams['font.size'] = 22

fig = plt.figure(figsize=(12, 12))

ax = plt.subplot(111)

# Data to plot
labels = pie_bucket_drivers.index.tolist()
sizes = [100*float(x) for x in pie_bucket_drivers['Sensitivity']]
colors = df_color.loc[pie_bucket_drivers.index.tolist()]['Color'].tolist()

# Plot
plt.pie(sizes,
        labels=labels,
        colors=colors,
        autopct='%1.f%%',
        pctdistance=0.84,
        shadow=False,
        explode=(
            0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05, 0.05
        ),
        startangle=80)

centre_circle = plt.Circle((0, 0), 0.75, fc='white')
fig = plt.gcf()
fig.gca().add_artist(centre_circle)

plt.axis('equal')
plt.title('BMW Bucket Drivers')
plt.show()
fig.savefig('Qi_API_Bucket_Drivers_Pie_Chart_Example_(BMW).png',
            bbox_inches="tight",
            facecolor=fig.get_facecolor())
```


BMW Bucket Drivers
Peripheral EU Sov Risk
23%
Other
12%
Country Growth
4%
Corp credit
4%
Global Real Rates
6%
DM FX
6%
EM Sov Risk
7%
EM FX
8%
Est. Earnings
9%
Global QE
9%
Systemic liquidity
11%

4.4 Valuation Gaps

Define functions to get specified Model data for a given date and find the largest valuation gaps across a given set of assets.

```
def get_vals(model, start, end):
    year_start = int(start[:4])
    year_end = int(end[:4])
    time_series = []
    for year in range(year_start, year_end + 1):
        query_start = start
        if year != year_start:
            date_from = '%d-01-01' % year
        else:
            date_from = start
        if year != year_end:
            date_to = '%d-12-31' % year
        else:
            date_to = end

        print("Gathering data for %s from %s to %s..." % (model,
                                                          date_from,
                                                          date_to))
        time_series += api_instance.get_model_timeseries(
            model,
            date_from=date_from,
            date_to=date_to)

    Rsq = [data.rsquare for data in time_series]
    FVG = [data.sigma for data in time_series]
    dates = [data._date for data in time_series]

    df = pandas.DataFrame({'Dates': dates, 'Rsq': Rsq, 'FVG': FVG})
    df.set_index('Dates', inplace=True)

    return df
```

```
def get_top_valuation_gaps(date, models):
    FVG_s, ID, Names, Rsq_s = ([] for i in range(4))
    for asset in models:
        try:
            if float(get_vals(asset, date, date)['Rsq']) > 65:
                FVG_s.append(float(get_vals(asset, date, date)['FVG']))
                ID.append(asset)
                Names.append(
                    api_instance.get_model(
                        model=asset
                    ).name
                )
                Rsq_s.append(float(get_vals(asset, date, date)['Rsq']))
        # Skip if data not available
        except TypeError:
            continue

    df_FVG = pandas.DataFrame({'Name': Names, 'FVG': FVG_s, 'Rsq': Rsq_s})
    df_FVG.index = ID
```

(continues on next page)

(continued from previous page)

```
    top_gaps = pandas.concat([
        df_FVG.nsmallest(5, 'FVG'), df_FVG.nlargest(5, 'FVG')
    ])
    MAX = []
    MIN = []

    for asset in top_gaps['Name']:
        MAX.append(max(get_vals(asset, '2015-01-02', date)['FVG']))
        MIN.append(min(get_vals(asset, '2015-01-02', date)['FVG']))

    top_gaps['Max'] = MAX
    top_gaps['Min'] = MIN

    return top_gaps
```

Having defined these functions, we can use them to graph these top valuation gaps for a given date. In this example we plot:

Top Valuation Gaps

```
def example_valuation_gaps():

    stoxx_600 = list(set([
        model.name for model in api_instance.get_models(tags='STOXX Europe 600')
    ]))

    df = get_top_valuation_gaps('2019-01-14', sorted(stoxx_600))

    import matplotlib
    import matplotlib.pyplot as plt
    matplotlib.rcParams.update({'errorbar.capsize': 15})

    fig = plt.figure(figsize=(10, 7))

    ax = plt.subplot()

    plt.yticks(fontsize=12)
    plt.xticks(fontsize=12)

    ax.axhline(0, color='k', lw=1)
    fig.patch.set_facecolor('#FFFFFF')

    ax.errorbar(df['Name'],
                df['FVG'],
                [df['FVG'] - df['Min'], df['Max'] - df['FVG']],
                fmt='o',
                ms='20',
                color='#C3423F',
                ecolor='#53B2FF',
                lw=10,
                capthick=8,
                solid_capstyle='round',
                solid_joinstyle='round')

    ax.spines["top"].set_visible(False)
    ax.spines["right"].set_visible(False)
```

(continues on next page)

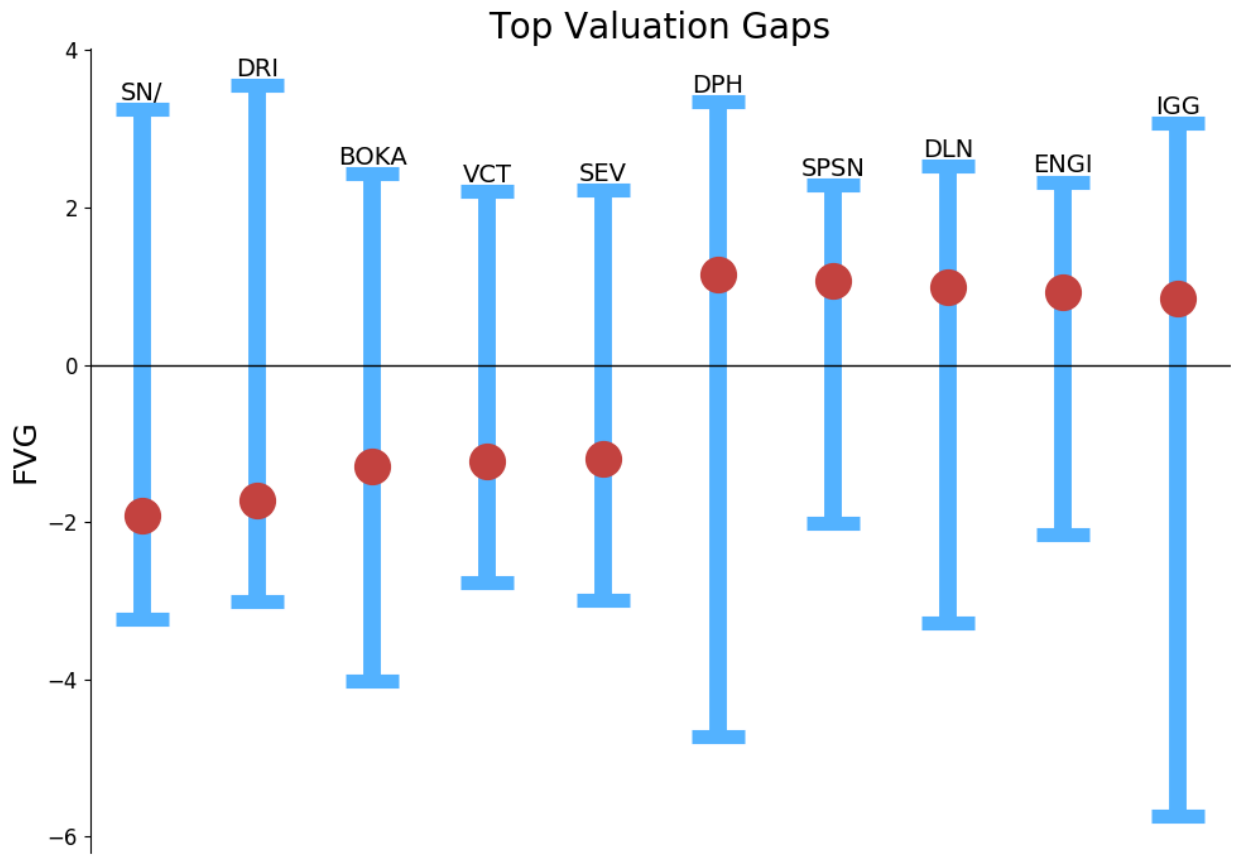
(continued from previous page)

```
    ax.spines["bottom"].set_visible(False)
    ax.tick_params(axis='x', bottom=False, labelbottom=False)

    plt.ylabel('FVG', fontsize=18)
    plt.title('Top Valuation Gaps', fontsize=20)

    for x in range(0, len(df['Name'][:10])):
        ax.annotate(df['Name'][x], (df['Name'][x], df['Max'][x]+0.12),
                    ha='center',
                    color='k',
                    fontsize=14)

    plt.tight_layout()

    df.to_csv(path_or_buf='Qi_API_Top_10_Valuation_Gaps.csv',
              float_format='%.5f',
              index_label='Model Name',
              columns=[
                  'Min',
                  'Max',
                  'FVG',
              ])

    fig.savefig('Qi_API_Top_10_Valuation_Gaps.png',
                bbox_inches="tight",
                facecolor=fig.get_facecolor())
    plt.show()
```

Displaying the results in a table and candle chart:

Table 1: Valuation Gap Summary

Model Name	Min	Max	FVG
SN/	-3.23550	3.24739	-1.91751
DRI	-3.00432	3.55386	-1.72134
BOKA	-4.02201	2.42945	-1.29859
VCT	-2.77558	2.20676	-1.23452
SEV	-3.00034	2.22265	-1.19572
DPH	-4.73037	3.34520	1.15388
SPSN	-2.01844	2.28753	1.06253
DLN	-3.28653	2.52369	0.98228
ENGI	-2.15225	2.32524	0.91743
IGG	-5.73275	3.06874	0.84491


Top Valuation Gaps
FVG
4
2
0
−2
−4
−6
SN/
DRI
BOKA
VCT
SEV
DPH
SPSN
DLN
ENGI
IGG

CHAPTER

FIVE

GLOSSARY

5.1 Macro Factors

Below is a glossary for all Qi macro factors. In the first column, under 'Macro Factors', when there is a discrepancy between the Factor name and how it's labelled in the API, the appropriate label is shown in quotation marks

Table 1: Glossary for Qi Macro Factors

Macro Factors	Description	Definition	Interpretation
US High Yield 'US HY'	US High Yield Credit Spreads	An index based on a basket of 100 US single-name high yield Credit Default Swaps (CDS).	The CDS spread identifies concerns over default risk; it is the price the market is willing to pay for insurance against corporates defaulting. It can be thought of as a fear indicator. Spreads rise due to credit stress as the market pays more for credit protection. Up = higher credit risk in US High Yield corporates. Typically equities have a negative sensitivity to High Yield; i.e. a 1 SD move higher in CDS spreads would be negative for equity markets as they fear rising corporate defaults.
Itraxx Japan	Japanese Credit Spreads	An index based on a basket of 40 equally-weighted CDS on investment grade Japanese corporate credit.	Up = higher credit risk in Japan. Negative relationship with equities. Domestic equities suffer on the fear of rising corporate defaults. Note international equities benefited from tight spreads during the BoJ QE as it prompted capital flight from Japanese investors seeking higher yield.
Itraxx Crossover 'Itraxx Xover'	EUR High Yield Credit Spreads	An index of 75 equally weighted CDS on the most liquid sub investment grade European corporate entities.	Up = higher credit risk in Europe. Again, the typical relationship is wider CDS are negative for European equities. Greater demand for insurance against defaults is bad news for EU stocks.

continues on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
FinSub Credit	EUR Financial Credit	An index of 30 equally weighted CDS on investment grade European financial entities.	Up = higher credit risk in EU banks. As above, greater demand for insurance against banks potentially defaulting would typically be negative for European financials and the broader equity market.
Brent	Crude Oil (Brent)	Energy commodity	Assets can have either a positive or negative relationship. Yields and energy stocks would expect to be positive: higher crude fuels higher inflation fears and helps the bottom line of energy stocks. Conversely, for other equities or even countries which are net importers, higher crude prices may be seen as a tax on consumers, businesses, sovereigns.
Western Texas Intermediate 'WTI'	Crude Oil (WTI)	Energy commodity	Assets can have either a positive or negative relationship. Yields and energy stocks would expect to be positive: higher crude fuels higher inflation fears and helps the bottom line of energy stocks. Conversely, for other equities or even countries which are net importers, higher crude prices may be seen as a tax on consumers, businesses, sovereigns.
Natural Gas	Natural Gas	Energy commodity	Assets can have either a positive or negative relationship. Yields and energy stocks would expect to be positive: higher crude fuels higher inflation fears and helps the bottom line of energy stocks. Conversely, for other equities or even countries which are net importers, higher crude prices may be seen as a tax on consumers, businesses, sovereigns.

continues on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
EUR Swaption Rates Nvol 1y5y 'EUR 1y5y Rate Nvol'	ECB Quantitative Tightening Expectations	A swaption (swap option) is the option to enter into an interest rate swap. In exchange for option premium, the buyer gains the right but not the obligation to enter into a specified swap agreement on a specified future date. Here we talk about options on 5yr interest rate swaps 1yr forward in USD, EUR, JPY & GBP.	During Quantitative Easing, Central Bank buying of government, mortgage, corporate bonds suppressed vol across all asset classes but especially in the intermediate part of the government yield curve where the bulk of purchases took place. By keeping yields and vol low, CBs encouraged portfolios to allocate towards more risky financial products. Hence, low rate vol (ongoing QE) is typically seen as positive for risky assets. An aggressive QT approach (e.g. taper tantrum) has the potential to upset markets, with risky assets particularly vulnerable.
USD Swaption Rates Nvol 1y5y 'USD 1y5y Rate Nvol'	Fed Quantitative Tightening Expectations	A swaption (swap option) is the option to enter into an interest rate swap. In exchange for option premium, the buyer gains the right but not the obligation to enter into a specified swap agreement on a specified future date. Here we talk about options on 5yr interest rate swaps 1yr forward in USD, EUR, JPY & GBP.	During Quantitative Easing, Central Bank buying of government, mortgage, corporate bonds suppressed vol across all asset classes but especially in the intermediate part of the government yield curve where the bulk of purchases took place. By keeping yields and vol low, CBs encouraged portfolios to allocate towards more risky financial products. Hence, low rate vol (ongoing QE) is typically seen as positive for risky assets. An aggressive QT approach (e.g. taper tantrum) has the potential to upset markets, with risky assets particularly vulnerable.

continues on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
JPY Swaption Rates Nvol 1y5y 'JPY 1y5y Rate Nvol'	BoJ Quantitative Tightening Expectations	A swaption (swap option) is the option to enter into an interest rate swap. In exchange for option premium, the buyer gains the right but not the obligation to enter into a specified swap agreement on a specified future date. Here we talk about options on 5yr interest rate swaps 1yr forward in USD, EUR, JPY & GBP.	During Quantitative Easing, Central Bank buying of government, mortgage, corporate bonds suppressed vol across all asset classes but especially in the intermediate part of the government yield curve where the bulk of purchases took place. By keeping yields and vol low, CBs encouraged portfolios to allocate towards more risky financial products. Hence, low rate vol (ongoing QE) is typically seen as positive for risky assets. An aggressive QT approach (e.g. taper tantrum) has the potential to upset markets, with risky assets particularly vulnerable.
GBP Swaption Rates Nvol 1y5y 'GBP 1y5y Rate Nvol'	BoE Quantitative Tightening Expectations	A swaption (swap option) is the option to enter into an interest rate swap. In exchange for option premium, the buyer gains the right but not the obligation to enter into a specified swap agreement on a specified future date. Here we talk about options on 5yr interest rate swaps 1yr forward in USD, EUR, JPY & GBP.	During Quantitative Easing, Central Bank buying of government, mortgage, corporate bonds suppressed vol across all asset classes but especially in the intermediate part of the government yield curve where the bulk of purchases took place. By keeping yields and vol low, CBs encouraged portfolios to allocate towards more risky financial products. Hence, low rate vol (ongoing QE) is typically seen as positive for risky assets. An aggressive QT approach (e.g. taper tantrum) has the potential to upset markets, with risky assets particularly vulnerable.
CRB Food	Food prices	Commodity Research Bureau index of foodstuffs and components	More often than not, soft commodity prices are viewed as inflation proxies. A positive relationship implies the asset performs during a reflationary environment.
Wheat	Wheat	Wheat	More often that not, soft commodity prices are viewed as inflation proxies. A positive relationship implies the asset performs during a reflationary environment.
Copper	Copper	Industrial metal - futures contract	Can be 'spurious' but the typical pattern is higher industrial metal prices are positively associated with risky assets and bond yields, as they speak to a growing global economy.

continues on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
Iron Ore	Iron Ore	Industrial metal - futures contract	Can be 'spurious' but the typical pattern is higher industrial metal prices are positively associated with risky assets and bond yields, as they speak to a growing global economy.
CRB Rind	Industrial Metals	Commodity Research Bureau Raw industrials Index	Can be 'spurious' but the typical pattern is higher industrial metal prices are positively associated with risky assets and bond yields, as they speak to a growing global economy.
Gold Silver Ratio	Gold Silver Ratio	Gold to Silver Ratio - futures contracts	Gold is widely perceived as a safe haven proxy. Higher gold prices relative to silver prices suggest uncertainty / fear is higher and there's a "flight-to-quality" dynamic at work.
VIX	VIX	A measure of the implied volatility of S&P 500 index options. Often referred to as the fear index or the fear gauge, it represents one measure of the market's expectation of stock market volatility over the next 30-day period.	Up = higher equity risk in US, more fear
VXEEM	VIX EEM	CBOE Emerging Markets ETF Volatility Index	Up = higher equity risk in EM, more fear
VDAX New	VDAX	Deutsche Boerse volatility index	Up = higher equity risk in EU, more fear
China GDP	Chinese GDP	China Current Quarter tracking GDP forecast (QoQ %) from Nowcast	Up = higher growth
Euro GDP	European GDP	Europe Current Quarter tracking GDP forecast (QoQ %) from Nowcast	Up = higher growth
US GDP	US GDP	US Current Quarter tracking GDP forecast (QoQ %) from Nowcast	Up = higher growth
Baltic Dry	Baltic Freight	An index used as a proxy for dry shipping stocks, often seen as a bellwether for industrial activity	Up = higher activity (implied stronger global growth)
EM CDS	EM sovereign risk	Generic 5y CDS of the sovereign(s) . The cost to insure against a country defaulting.	Up = higher default risk. US federal shutdowns, hard Brexit, fears about Chinese WMPs would all be examples of negative credit events that might cause the demand for protection to rise. Risky assets typically have a negative relationship, i.e. want sovereign risk to remain low.

continues on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
China 5y CDS	China sovereign risk	Generic 5y CDS of the country. The cost to insure against a country defaulting.	Up = higher default risk. US federal shutdowns, hard Brexit, fears about Chinese WMPs would all be examples of negative credit events that might cause the demand for protection to rise. Risky assets typically have a negative relationship, i.e. want sovereign risk to remain low.
Greece 10y ASW 'Greek Sov. Confidence'	Greek Sovereign Risk	EUR 10y Swap rate minus Greece 10y Generic Bond Yield	Positive relationship means the financial asset performs when GGB yields stay comparatively low, i.e. there is less sovereign stress.
Italy 5y ASW 'Italian Sov. Confidence'	Italian Sovereign Risk	EUR 5y Swap rate minus Italy 5y Generic Bond Yield	In periods of increased political stress (like in the aftermath of the M5S / Lega Nord coalition winning the May 2018 election & in their spat with Brussels over the Italian budget) BTP yields rise relative to swaps. Any model that has a positive relationship at that time implies Italian politics needs to be benign for that asset to perform.
Spain 5y ASW 'Spain Sov. Confidence'	Spanish Sovereign Risk	EUR 5y Swap rate minus Spain 5y Generic Bond Yield	Negative relationship means the financial asset performs when SPGB yields spike versus swaps, i.e. there is increased sovereign stress.
Inflation expectations 2y EU '2y Infl. Expec.'	2y Inflation Expectations	2y inflation expectation for the country as measured by Zero Coupon inflation swaps	Up = higher inflation. Typically positive for risky assets (although not always) and negative for Fixed Income instruments.
Inflation expectations 5y EU '5y Infl. Expec.'	5y Inflation Expectations	5y inflation expectation for the country as measured by Zero Coupon inflation swaps	Up = higher inflation. Typically positive for risky assets (although not always) and negative for Fixed Income instruments.
Inflation expectations 10y EU '10y Infl. Expec.'	10y Inflation Expectations	10y inflation expectation for the country as measured by Zero Coupon inflation swaps	Up = higher inflation. Typically positive for risky assets (although not always) and negative for Fixed Income instruments.

continues on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
EUR 1y CCY Basis Swap 'EUR 1y Basis Swap'	USD liquidity (EUR)	The cost for a European bank to fund themselves in USD	Up = lower risk in EUR currency. In general, cross currency basis is a measure of any Dollar shortage in financial markets. The more negative the basis becomes, the more severe the shortage. It is the additional hedging cost added to the interest differential of the two currencies. The most important drivers of the cross-currency basis spreads appear to be short and medium term EU financial sector credit risk and, to a slightly lesser extent, the equivalent US indicators. Calendar distortions like the year-end turn can impact but, in general, periods of more negative basis reflect worries about the EuroZone: peripheral stress, the sovereign-bank feedback loop etc.
JPY 1y CCY Basis Swap 'Jpy 1y Basis Swap'	USD liquidity (JPY)	The cost for a Japanese bank to fund themselves in USD	Up = lower risk in JPY currency. Given negative interest rates, Japanese investors typically will look to pick up yield overseas. Japan has huge savings pot that is looking for yield. USDJPY is a 'risk on' / 'risk off' metric. During periods of 'risk on' they will fund themselves using cheap currency which is Yen; so buy USD/sell JPY, and in 'risk off' it is usually sell USD/buy JPY.
5s30s Swapcurve '5s30s Swap Ccy1' '5s30s Swap Ccy2' 'Country 5s30s Swap'	Forward growth expectations	The spread between the 5y yield and the 30y yield of the relevant currency	Interest Rate curve proxy for medium term growth expectations. A bullish growth scenario implies higher 30yr yields versus 5yrs. Conversely, a recession would typically see the curve flatten / invert.
Country Economic Data 'Economic Data Ccy1' 'Economic Data Ccy2'	Economic Data	Country's Current Quarter GDP forecast (QoQ %) from Now-Cast. If it is not available, this factor represents the Surprise Index of that country (if NA then proxy: EM Surprise Index)	Up = stronger economic growth

continues on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
Country TWI	Currency - Trade Weighted index	An index showing the value of a country's currency in relation to the currencies of a group of countries with which it trades. In the index, each country's currency is given an importance in relation to the amount of trade it does.	Up = Stronger Currency
Dollar Index 'DXY'	DXY Dollar Index	The U.S. Dollar Index (USDX, DXY) is an index of the value of the dollar (USD) in comparison to selected other currencies. Since the dollar index reflects the value of a basket of currencies relative to the dollar, it gives a more representative picture of the dollar's strength or weakness than a single currency pair like EUR/USD.	Up = USD higher vs DM FX.
Asia Dollar Index 'ADXY'	Asian EM FX	JPMorgan Asia Currency Index (ADXY), the first U.S. dollar tradable index of emerging Asian currencies. The ADXY is a spot index of emerging Asia's most actively traded currency pairs valued against the U.S. Dollar.	Higher means weaker Dollar, i.e. Up = USD lower vs Asia FX
EUR 10y Real Rate	EUR 10y Real Rate	EUR 10y Generic Nominal Yield minus 10y Expected Inflation	Reflect the real cost of capital so are often seen as the best single indicator for overall financial conditions in an economy. Can have a positive or negative relationship with equities. It was negative during large parts of the Great Financial Crisis - risky assets needed the ECB to keep monetary policy easy. A positive relationship suggests equities are sufficiently confident in a self-sustaining cyclical upswing that higher rates / tighter financial conditions reflect a strong economy which is good for earnings.

continues on next page

Table 1 – continued from previous page

Macro Factors	Description	Definition	Interpretation
USD 10y Real Rate	USD 10y Real Rate	US 10y Generic Nominal Yield minus 10y Expected Inflation	Reflect the real cost of capital so are often seen as the best single indicator for overall financial conditions in an economy. Can have a positive or negative relationship with equities. It was negative during large parts of the Great Financial Crisis - risky assets needed the Fed to keep monetary policy easy. A positive relationship suggests equities are sufficiently confident in a self-sustaining cyclical upswing that higher rates / tighter financial conditions reflect a strong economy which is good for earnings/margins.
JPY 10y Real Rate	JPY 10y Real Rate	JPY 10y Generic Nominal Yield minus 10y Expected Inflation	Reflect the real cost of capital so are often seen as the best single indicator for overall financial conditions in an economy. Can have a positive or negative relationship with equities. It was negative during large parts of the Great Financial Crisis - risky assets needed the BoJ to keep monetary policy easy. A positive relationship suggests equities are sufficiently confident in a self-sustaining cyclical upswing that higher rates / tighter financial conditions reflect a strong economy which is good for earnings.
Index 1y Forward Earnings (eg EU) 'Index 1y Fwd Earnings'	1yr Forward Earnings	Market expectations for future earnings.	Up = better earnings ahead

5.2 Confidence

(Also known as R Squared or RSq) is a gauge of how sensitive the asset is to macroeconomic forces. Typically values above 65% are considered to show strong explanatory power or 'confidence'.

5.3 Qi Model Value

Represents the macro warranted fair value of the asset based on the Qi methodology.

5.4 Valuation Gap (VG) or Fair Valuation Gap (FVG)

The difference between the actual price of an asset and the Qi Model Value. This is quoted both as an absolute (can be percentage, USD or basis points depending on the instrument in question) and in standard deviation terms.

5.5 Rich

Represented by a positive 'FVG' and occurs when the market value of an asset is overvalued relative to the Qi macro warranted fair value.

5.6 Cheap

Represented by a negative 'FVG' and occurs when the market value of an asset is undervalued relative to the Qi macro warranted fair value.

5.7 Sensitivity

The impact on the price of the asset in %(basis points for rates) for a 1 standard deviation move up in the macro driver. This tells us the influence a driver has on the price of an asset, allowing us to rank drivers and also to identify the directional impact.

5.8 Historical Sensitivity

A historical time series of how the asset's sensitivity to a macro factor evolves.

5.9 Standard Deviation

Measures the dispersion of a set of data from its mean. It measures the absolute variability of a distribution; the higher the dispersion or variability, the greater is the standard deviation and greater will be the magnitude of the deviation of the value from their mean.

5.10 Driver

The individual macro factor we use as an explanatory variable in the Qi methodology e.g. US GDP

5.11 Driver Group

(Previously known as a bucket) The aggregation of related individual macro drivers into segments. Driver Group: Global Growth comprises US GDP, EU GDP and China GDP.

5.12 Driver Attribution

A breakdown of how much the movement in a driver has contributed to changes in the price of the underlying asset.

5.13 Custom Driver

A driver selected outside of the standard Qi driver set.

5.14 Model Type

Identifies whether the underlying explanatory variables have been curated by 'Qi' or whether they are a customised set, in which case they are 'Custom' models.

5.15 Custom Model

An asset modelled using a custom set of explanatory macro drivers.

5.16 Timeframe

The period over which the model is run can either be ST or LT.

- ST - Refers to a Short term model which is defined as 83 day lookback period.
- LT - Refers to a Long Term model and is defined as 250 day rolling lookback period.

5.17 Z-Score

The number of standard deviations from the mean a data point is.

5.18 Macro Regime

The broad macroeconomic dynamics impacting the asset. Within Qi, the Regime is defined as the set of most important macro drivers.

5.19 Macro Exposure

The overall level of sensitivity to macro drivers that the asset or portfolio is exhibiting.

5.20 Asset Class

The broad grouping of assets by similarities in market behaviour , laws and regulations.

5.21 Ticker

The corresponding bloomberg reference code for a given asset.