

---

# MyMagento

*Release 2.1.1*

**Adam Korn**

Feb 20, 2023



# README

<b>1</b>	<b>About MyMagento</b>	<b>3</b>
1.1	Main Components . . . . .	3
1.2	Available Endpoints . . . . .	4
<b>2</b>	<b>Installation</b>	<b>5</b>
<b>3</b>	<b>QuickStart: Login with MyMagento</b>	<b>7</b>
3.1	Setting the Login Credentials . . . . .	7
3.2	Getting a Client . . . . .	7
3.2.1	Option 1: Initialize a Client Directly . . . . .	7
3.2.2	Option 2: Call <code>get_api()</code> . . . . .	8
3.3	Getting an <code>ACCESS_TOKEN</code> . . . . .	8
<b>4</b>	<b>Interacting with the API</b>	<b>9</b>
4.1	Performing a <code>search()</code> . . . . .	9
4.2	Search Results: The Model Classes . . . . .	10
4.3	Building Custom Search Queries . . . . .	11
4.4	Making Authorized Requests . . . . .	12
4.4.1	Example: Making a <code>get()</code> Request . . . . .	12
4.4.2	Example: Making a <code>post()</code> Request . . . . .	12
<b>5</b>	<b>The <code>magento</code> Package</b>	<b>15</b>
5.1	The <code>clients</code> module . . . . .	15
5.2	The <code>search</code> module . . . . .	21
5.2.1	Available Endpoints . . . . .	21
5.3	The <code>exceptions</code> module . . . . .	34
5.4	The <code>utils</code> module . . . . .	35
<b>6</b>	<b>The <code>magento.models</code> subpackage</b>	<b>41</b>
6.1	The <code>model</code> module . . . . .	41
6.2	The <code>product</code> module . . . . .	45
6.3	The <code>category</code> module . . . . .	53
6.4	The <code>order</code> module . . . . .	55
6.5	The <code>invoice</code> module . . . . .	58
<b>7</b>	<b>Get a Magento 2 REST API Token With MyMagento</b>	<b>61</b>
7.1	Setting the Login Credentials . . . . .	61
7.2	Getting a Client . . . . .	61
7.3	Setting Environment Variables . . . . .	62
<b>8</b>	<b>Add discount on each product based on product price</b>	<b>63</b>

8.1 Solution Using MyMagento . . . . .	63
<b>9 Changelog</b>	<b>65</b>
9.1 v2.1.0 . . . . .	65
<b>10 Indices and tables</b>	<b>67</b>
<b>Python Module Index</b>	<b>69</b>
<b>Index</b>	<b>71</b>



A Python package that wraps and extends the Magento 2 REST API

[Explore the docs »](#)



## ABOUT MYMAGENTO

---

### What's MyMagento?

MyMagento is a highly interconnected package that wraps and extends the Magento 2 REST API, providing a more intuitive and user-friendly interface to access and update your store.

---

### MyMagento simplifies interaction with the Magento 2 REST API

If you've worked with the Magento 2 API, you'll know that not all endpoints are created equally.

MyMagento aims to streamline your workflow by simplifying a variety of commonly needed API operations.

## 1.1 Main Components

---

### The Client

- Handles all API interactions
  - Supports multiple store views
  - Provides access to all other package components
- 

### The SearchQuery and Subclasses

- `execute()` a predefined or custom search query on any endpoint
  - Simplified creation of basic and advanced [searches using REST endpoints](#)
- 

### The Model Subclasses

- Wrap all API responses in the package
  - Provide additional endpoint-specific methods to retrieve and update data
-

## 1.2 Available Endpoints

MyMagento is compatible with every API endpoint

Endpoints are wrapped with a `Model` and `SearchQuery` subclass as follows:

Endpoint	Client Shortcut	SearchQuery Subclass	Model Subclass
orders	<code>Client.orders</code>	<code>OrderSearch</code>	<code>Order</code>
orders/items	<code>Client.order_items</code>	<code>OrderItemSearch</code>	<code>OrderItem</code>
invoices	<code>Client.invoices</code>	<code>InvoiceSearch</code>	<code>Invoice</code>
products	<code>Client.products</code>	<code>ProductSearch</code>	<code>Product</code>
products/attributes	<code>Client.product_attributes</code>	<code>ProductAttributeSearch</code>	<code>ProductAttribute</code>
categories	<code>Client.categories</code>	<code>CategorySearch</code>	<code>Category</code>
endpoint	<code>Client.search("endpoint")</code>	<code>SearchQuery</code>	<code>APIResponse</code>

...



## INSTALLATION

---

### Installing MyMagento

To install using pip:

```
pip install my-magento
```

Please note that MyMagento requires Python  $\geq$  3.10

---

...



## QUICKSTART: LOGIN WITH MYMAGENTO

MyMagento uses the `Client` class to handle all interactions with the API.

---

**Tip:** See *Get a Magento 2 REST API Token With MyMagento* for full details on generating an access token

---

### 3.1 Setting the Login Credentials

To generate an `ACCESS_TOKEN` you'll need to `authenticate()` your `USER_CREDENTIALS`.

Creating a `Client` requires a domain, username, and password at minimum.

```
>> domain = 'website.com'
>> username = 'username'
>> password = 'password'
```

If you're using a local installation of Magento you'll need to set `local=True`. Your domain should look like this:

```
>> domain = '127.0.0.1/path/to/magento'
```

...

### 3.2 Getting a Client

#### 3.2.1 Option 1: Initialize a Client Directly

```
from magento import Client

>>> api = Client(domain, username, password, **kwargs)
```

### 3.2.2 Option 2: Call `get_api()`

```
import magento

>>> api = magento.get_api(**kwargs)
```

`get_api()` takes the same keyword arguments as the `Client`, but if the domain, username, or password are missing, it will attempt to use the following environment variables:

```
import os

os.environ['MAGENTO_DOMAIN'] = domain
os.environ['MAGENTO_USERNAME'] = username
os.environ['MAGENTO_PASSWORD'] = password
```

...

## 3.3 Getting an ACCESS\_TOKEN

Unless you specify `login=False`, the `Client` will automatically call `authenticate()` once initialized:

```
>>> api.authenticate()

[ MyMagento | website_username ]: Authenticating username on website.com...
[ MyMagento | website_username ]: Logged in to username
```

## INTERACTING WITH THE API

Did you *Get a Magento 2 REST API Token With MyMagento?* Then let's start using the API!

### 4.1 Performing a search()

The `Client.search()` method lets you `execute()` a query on any API endpoint. It creates a `SearchQuery` for the endpoint, allowing you to retrieve data about

- An individual item (ex. `by_id()`)
- A list of items (ex. `by_list()`)
- Any search criteria you desire (see *Building Custom Search Queries*)

---

#### From the Docs...

`Client.search(endpoint)`

Initializes and returns a `SearchQuery` corresponding to the specified endpoint

---

**Note:** Several endpoints have predefined `SearchQuery` and `Model` subclasses

If a subclass hasn't been defined for the endpoint yet, a general `SearchQuery` will be returned, which wraps the `result` with `APIResponse`

---

#### Parameters

**endpoint** (`str`) – a valid Magento API search endpoint

#### Return type

`SearchQuery`

---

### Example: search() an endpoint by\_id()

```
# Query the "invoices" endpoint (also: api.invoices)
>>> api.search("invoices").by_id(1)

<Magento Invoice: "#000000001"> for <Magento Order: "#000000001" placed on 2022-11-01_
↪03:27:33>
```

### Example: search() an endpoint by\_list()

```
# Retrieve invoices from a list of invoice ids
>>> ids = list(range(1,101))
>>> api.invoices.by_list("entity_id", ids)

[<Magento Invoice: "#000000001"> for <Magento Order: "#000000001" placed on 2022-11-01_
↪03:27:33>, ...]
```

...

## 4.2 Search Results: The Model Classes

The result of any `SearchQuery` will be parsed and wrapped by a `Model` class in the `magento.models` subpackage, making the API response data easier to work with.

These classes also provide endpoint-specific methods to search for related items and update store data.

### Example: Retrieving every Order containing a Product

Let's retrieve a `Product` using `by_sku()`

```
>>> product = api.products.by_sku("24-MB01")
```

We can search for orders containing this product as follows:

```
# Using the Product itself
>>> product.get_orders()

[<Magento Order: "#000000003" placed on 2022-12-21 08:09:33>, ... ]

# Using an OrderSearch
>>> api.orders.by_product(product)
>>> api.orders.by_product_id(product.id)
>>> api.orders.by_sku(product.sku)

[<Magento Order: "#000000003" placed on 2022-12-21 08:09:33>, ... ]
```

### Example: Retrieving some items related to a Category

```
# Get Category data
>>> category = api.categories.by_name("Watches")
>>> category.get_products()
>>> category.get_invoices()

[<Magento Product: 24-MG04>, <Magento Product: 24-MG01>, <Magento Product: 24-MG03>, ... ]
[<Magento Invoice: "#000000004"> for <Magento Order: "#000000004" placed on 2022-11-14_
↳03:27:33>, ... ]
```

### Example: Updating the Thumbnail MediaEntry of a Product

```
# Update product thumbnail label on specific store view
>>> product.thumbnail.set_alt_text("bonjour", scope="FR")
>>> print(product.thumbnail)

<MediaEntry 3417 for <Magento Product: 24-MB01>: bonjour>
```

...

**Tip:** If you have multiple store views, a `store_code` can be specified when retrieving/updating data

- The `Client.scope` is used by default - simply change it to switch store views
- Passing the scope keyword argument to `Client.url_for()`, `Model.refresh()`, and some Model update methods will temporarily override the Client scope

...

## 4.3 Building Custom Search Queries

In addition to the predefined methods, you can also build your own queries

- Simply `add_criteria()`, `restrict_fields()`, and `execute()` the search
- The `since()` and `until()` methods allow you to further filter your query by date

### Example

```
# Retrieve orders over $50 placed since the start of 2023
>>> api.orders.add_criteria(
...     field="grand_total",
...     value="50",
...     condition="gt"
... ).since("2023-01-01").execute()

[<Magento Order: "#000000012" placed on 2023-01-02 05:19:55>, <Magento Order: "#000000013"
↳placed on 2023-01-05 09:24:13>]
```

...

## 4.4 Making Authorized Requests

The `Client` can be used to generate the `url_for()` any API endpoint, including a store `scope`.

You can use this URL to make an authorized `get()`, `post()`, `put()`, or `delete()` request.

### 4.4.1 Example: Making a `get()` Request

```
# Request the data for credit memo with id 7
>>> url = api.url_for('creditmemo/7')
>>> response = api.get(url)
>>> print(response.json())

{'adjustment': 1.5, 'adjustment_negative': 0, 'adjustment_positive': 1.5, 'base_adjustment': 1.5, ... }
```

---

**Note:** A `search()` is simpler than making `get()` requests, as the result will be wrapped by `APIResponse` or other `Model`

```
# Retrieve credit memo with id 7 using a search
>>> memo = api.search("creditmemo").by_id(7)
>>> print(memo.data)
>>> print(memo)

{'adjustment': 1.5, 'adjustment_negative': 0, 'adjustment_positive': 1.5, 'base_adjustment': 1.5, ... }
<magento.models.model.APIResponse object at 0x000001BA42FD0FD1>
```

### 4.4.2 Example: Making a `post()` Request

```
# Add a comment to credit memo with id 7
>>> url = api.url_for("creditmemo/7/comments")
>>> payload = {
    "entity": {
        "comment": "this is a comment",
        "is_customer_notified": 0,
        "is_visible_on_front": 0,
        "parent_id": 20
    }
}
>>> response = api.post(url, payload)
```

---

**Tip:** The `Model.data_endpoint()` will usually be close to the url to `post()` to



```
# The same as above, but using a search
>>> memo = api.search("creditmemo").by_id(7)
>>> url = memo.data_endpoint() + '/comments'
>>> response = api.post(url, payload)
```



## THE MAGENTO PACKAGE

The magento package provides various tools to help simplify interaction with the Magento 2 API.

`magento.get_api(**kwargs)`

Initialize a `Client` using credentials stored in environment variables

Any valid `Client` kwargs can be used in addition to and/or instead of environment variables

**Usage:**

```
import magento

api = magento.get_api()
```

**Parameters**

`kwargs` – any valid kwargs for `Client`

**Raises**

`ValueError` – if login credentials are missing

**Return type**

`Client`

...

### 5.1 The clients module

```
class magento.clients.Client(domain, username, password, scope="", local=False, user_agent=None,
                             token=None, log_level='INFO', login=True, **kwargs)
```

Bases: `object`

The class that handles all interaction with the API

```
__init__(domain, username, password, scope="", local=False, user_agent=None, token=None,
         log_level='INFO', login=True, **kwargs)
```

Initialize a `Client`

---

**Important!**

The Magento account you use to log in must be assigned a **User Role** that has the appropriate API resources included in its **Resource Access** settings

This can be verified in Magento Admin by going to:

System -> Permissions -> User Roles -> {Role} -> Role Resources -> Resource Access

and ensuring that Sales, Catalog, Customers, and any other desired resources are included

---

### Parameters

- **domain** (*str*) – domain name of the Magento store (ex. domain.com or 127.0.0.1/magento24)
- **username** (*str*) – username of the Magento Admin account
- **password** (*str*) – password of the Magento Admin account
- **scope** (*Optional[str]*) – the store view scope to `search()` and make requests on
- **local** (*bool*) – whether the Magento store is hosted locally
- **user\_agent** (*Optional[str]*) – the user agent to use in requests
- **token** (*Optional[str]*) – an existing access token
- **log\_level** (*str*) – the logging level for logging to stdout
- **login** (*bool*) – if True, calls `authenticate()` upon initialization
- **kwargs** – see below

...

### Extra Keyword Arguments

- **log\_file** (*str*) – log file to use for the client's `logger`
- **log\_requests** (*bool*) - if True, the logs from `requests` will be added to the client's `log_file`

**BASE\_URL:** *str*

The base API URL

**USER\_CREDENTIALS:** *Dict[str, str]*

The user credentials

**ACCESS\_TOKEN:** *str*

The API access token

**domain:** *str*

The Magento store domain

**scope:** *str*

The store view code to request/update data on

**user\_agent:** *str*

The user agent to use in requests

**logger:** *MagentoLogger*

The `MagentoLogger` for the domain/username combination

**store:** *Store*

An initialized `Store` object

**classmethod** `new()`

Prompts for input to log in to the Magento API

**Return type**

`Client`

**classmethod** `load(pickle_bytes)`

Initialize a `Client` using a pickle bytestring from `to_pickle()`

**Return type**

`Client`

**classmethod** `from_json(json_str)`

Initialize a `Client` from a JSON string of settings

**Return type**

`Client`

**classmethod** `from_dict(d)`

Initialize a `Client` from a dictionary of settings

**Return type**

`Client`

**url\_for(endpoint, scope=None)**

Returns the appropriate request url for the given API endpoint and store scope

**Example**

```
# Generate the url for credit memo with id 7
>> api=Client("domain.com", "user", "password")
>> api.url_for('creditmemo/7')
"https://domain.com/rest/V1/creditmemo/7"

# Generate the same url on the "en" store view
>> api.url_for('creditmemo/7', scope='en')
"https://domain.com/rest/en/V1/creditmemo/7"
```

**Parameters**

- **endpoint** (`str`) – the API endpoint
- **scope** (`Optional[str]`) – the scope to generate the url for; uses the `Client.scope` if not provided

**Return type**

`str`

**search(endpoint)**

Initializes and returns a `SearchQuery` corresponding to the specified endpoint

**Note:** Several endpoints have predefined `SearchQuery` and `Model` subclasses

If a subclass hasn't been defined for the endpoint yet, a general `SearchQuery` will be returned, which wraps the `result` with `APIResponse`

**Parameters**

**endpoint** (*str*) – a valid Magento API search endpoint

**Return type**

*SearchQuery*

**property orders:** **OrderSearch**

Initializes an *OrderSearch*

**property order\_items:** **OrderItemSearch**

Initializes an *OrderItemSearch*

**property invoices:** **InvoiceSearch**

Initializes an *InvoiceSearch*

**property categories:** **CategorySearch**

Initializes a *CategorySearch*

**property products:** **ProductSearch**

Initializes a *ProductSearch*

**property product\_attributes:** **ProductAttributeSearch**

Initializes a *ProductAttributeSearch*

**get**(*url*)

Sends an authorized GET request

**Parameters**

**url** (*str*) – the URL to make the request on

**Return type**

*Response*

**post**(*url, payload*)

Sends an authorized POST request

**Parameters**

- **url** (*str*) – the URL to make the request on
- **payload** (*dict*) – the JSON payload for the request

**Return type**

*Response*

**put**(*url, payload*)

Sends an authorized PUT request

**Parameters**

- **url** (*str*) – the URL to make the request on
- **payload** (*dict*) – the JSON payload for the request

**Return type**

*Response*

**delete**(*url*)

Sends an authorized DELETE request

**Parameters**

**url** (*str*) – the URL to make the request on

**Return type***Response***authenticate()**Authenticates the `USER_CREDENTIALS` and retrieves an access token**Return type***bool***validate()**Validates the `token` by sending an authorized request to a standard API endpoint**Raises**`AuthenticationError` if the token is invalid**Return type***bool***request(*method*, *url*, *payload=None*)**

Sends an authorized API request. Used for all internal requests

---

**Tip:** Use `get()`, `post()`, `put()` or `delete()` instead

---

**Parameters**

- **method** (*str*) – the request method
- **url** (*str*) – the url to send the request to
- **payload** (*Optional[dict]*) – the JSON payload for the request (if the method is POST or PUT)

**Return type***Response***get\_logger(*log\_file=None*, *stdout\_level='INFO'*, *log\_requests=True*)**Retrieve a `MagentoLogger` for the current username/domain combination. Log files are `DEBUG`.**Parameters**

- **log\_file** (*Optional[str]*) – the file to log to
- **stdout\_level** (*str*) – the logging level for stdout logging
- **log\_requests** (*bool*) – if `True`, adds the `FileHandler` to the `connectionpool` logger

**Return type**`MagentoLogger`**property headers: dict**

Authorization headers for API requests

Automatically generates a `token` if needed**property token: str**Returns or generates an `ACCES_TOKEN`**to\_pickle(*validate=False*)**Serializes the `Client` to a pickle bytestring

**Parameters**

**validate** (*bool*) – if True, validates the `token/USER_CREDENTIALS` before serializing

**Return type**

*bytes*

`to_json(validate=False)`

Serializes the Client to a JSON string

**Parameters**

**validate** (*bool*) – if True, validates the `token/USER_CREDENTIALS` before serializing

**Return type**

*str*

`to_dict(validate=False)`

Serializes the Client to a dictionary

**Parameters**

**validate** (*bool*) – if True, validates the `token/USER_CREDENTIALS` before serializing

**Return type**

*Dict[str, str]*

`view_config()`

Prints the Client configuration settings

**class** `magento.clients.Store(client)`

Bases: `object`

Class containing store configurations and cached attribute lists

`__init__(client)`

Initialize a Store object

**Parameters**

**client** (`Client`) – an initialized `Client` object

**property** `is_single_store`: `bool`

Whether the store has a single store view (default) or multiple store views

**property** `active`: `APIResponse`

Returns the store config corresponding to the current `scope` of the `Client`

**property** `configs`: `Optional[Union[APIResponse, List[APIResponse]]]`

Returns a list of all store configurations

**property** `views`: `Optional[Union[APIResponse, List[APIResponse]]]`

Returns a list of all store views

**property** `all_product_attributes`: `List[ProductAttribute]`

A cached list of all product attributes

**property** `store_view_product_attributes`: `List[ProductAttribute]`

A cached list of all product attributes with the Store View scope

**property** `website_product_attributes`: `List[ProductAttribute]`

A cached list of all product attributes with the Web Site scope



**property global\_product\_attributes:** `List[ProductAttribute]`

A cached list of all product attributes with the Global scope

**property website\_attribute\_codes:** `List[str]`

The attribute codes of the `website_product_attributes`

**filter\_website\_attrs(*attribute\_data*)**

Filters a product attribute dict and returns a new one that contains only the website scope attributes

Website scoped attributes must be updated on the admin by making a second request on the all scope

- This method is called by `update_attributes()` and `update_custom_attributes()` to see if the second request is needed

---

### Example

The price attribute is Website scope and the meta\_title attribute is Store View scope

```
>> attribute_data = {'price': 12, 'meta_title': 'My Product'}
>> store.filter_website_attrs(attribute_data)
{'price': 12}
```

---

#### Parameters

**attribute\_data** (*dict*) – a dict of product attributes

#### Return type

*dict*

#### refresh()

Clears all cached properties

#### Return type

*bool*

## 5.2 The search module

### 5.2.1 Available Endpoints

The following endpoints are currently wrapped with a `Model` and `SearchQuery` subclass

Endpoint	Client Attribute	SearchQuery Subclass	Model Subclass
orders	<code>Client.orders</code>	<code>OrderSearch</code>	<code>Order</code>
orders/items	<code>Client.order_items</code>	<code>OrderItemSearch</code>	<code>OrderItem</code>
invoices	<code>Client.invoices</code>	<code>InvoiceSearch</code>	<code>Invoice</code>
products	<code>Client.products</code>	<code>ProductSearch</code>	<code>Product</code>
products/ attributes	<code>Client.product_attributes</code>	<code>ProductAttributeSearch</code>	<code>ProductAttribute</code>
categories	<code>Client.categories</code>	<code>CategorySearch</code>	<code>Category</code>

**class** magento.search.**SearchQuery**(*endpoint, client, model=<class 'magento.models.model.APIResponse'>*)

Bases: `object`

Queries any endpoint that invokes the `searchCriteria` interface. Parent of all endpoint-specific search classes

---

**Tip:** See <https://developer.adobe.com/commerce/webapi/rest/use-rest/performing-searches/> for official docs

---

**\_\_init\_\_**(*endpoint, client, model=<class 'magento.models.model.APIResponse'>*)

Initialize a `SearchQuery` object

**Parameters**

- **endpoint** (`str`) – the base search API endpoint (for example, `orders`)
- **client** (`Client`) – an initialized `Client` object
- **model** (`Type[Model]`) – the `Model` to parse the response data with; uses `APIResponse` if not specified

**client**

The `Client` to send the search request with

**endpoint**

The endpoint being queried

**Model**

The *magento.models* subpackage class to wrap the response with

**query**

The current url for the search request

**fields**

Restricted fields, from `restrict_fields()`

**add\_criteria**(*field, value, condition='eq', \*\*kwargs*)

Add criteria to the search query

**Parameters**

- **field** – the API response field to search by
- **value** – the value of the field to compare
- **condition** – the comparison condition
- **kwargs** – additional search option arguments (group and filter)

**Returns**

the calling `SearchQuery` object

**Return type**

`Self`

---

**Keyword Argument Options: Condition**

The `condition` argument specifies the condition used to evaluate the attribute value

- "eq" (default): matches items for which field=value
- "gt": matches items for which field>value
- "lt": matches items for which field<value
- "gteq": matches items for which field>=value
- "lteq": matches items for which field<=value
- "in": matches items for which field in value.split(",")
  - Tip: for in, use `by_list()` if not building a complex query

---

### Example

```
# Search for Orders created in 2023
>>> orders = api.orders.add_criteria(
...     field="created_at",
...     value="2023-01-01",
...     condition='gteq'
... ).execute()
```

---



---

### Keyword Argument Options: Using Filter Groups

group - filter group number

filter - filter number (within the specified filter group)

---

#### Using Filter Groups

Filter groups are filter criteria in the form of { field: value }

Group 0 Filter 0 -> Filter 0 Group 0 Filter 0 + Group 0 Filter 1 -> Filter 0 OR  
Filter 1 Group 0 Filter 0 + Group 1 Filter 0 -> Filter 0 AND Filter 0

#### `restrict_fields(fields)`

Constrain the API response data to only contain the specified fields

##### Parameters

**fields** (Iterable[str]) – an iterable or comma separated string of fields to include in the response

##### Returns

the calling SearchQuery object

##### Return type

Self

#### `execute()`

Sends the search request using the current `scope` of the `client`

---

**Tip:** Change the `Client.scope` to retrieve `result` data from different store `views`

---

##### Returns

the search query `result`

**Return type***Optional[Union[Model, List[Model]]]***by\_id**(*item\_id*)

Retrieve data for an individual item by its id

---

**Note:** The id field used is different depending on the endpoint being queried

- Most endpoints use an `entity_id` or `id`
- The `orders/items` endpoint uses `item_id`
- The `products` endpoint uses `product_id`, but can also be queried `by_sku()`

The `IDENTIFIER` attribute of each `Model` contains the appropriate field

---

**Parameters**`item_id` (*Union[int, str]*) – id of the item to retrieve**Return type***Optional[Model]***by\_list**(*field, values*)

Search for multiple items using an iterable or comma-separated string of field values

---

**Examples**Retrieve `Product` with ids from 1 to 10:

```
# Values can be a list/tuple/iterable
>> api.products.by_list('entity_id', range(1,11))
```

Search for `Order` that are processing, pending, or completed:

```
# Values can be a comma-separated string
>> api.orders.by_list('status', 'processing,pending,completed')
```

---

**Parameters**

- **field** (*str*) – the API response field to search for matches in
- **values** (*Iterable*) – an iterable or comma separated string of values

**Return type***Optional[Model, List[Model]]***since**(*sinceDate=None*)Retrieve items for which `created_at` `>=` `sinceDate`**Example:**

```
# Retrieve products created in 2023
>> api.products.since('2023-01-01').execute()
```

---

**Tip:** Calling with no arguments retrieves all items

```
# Retrieve all products
>> api.products.since().execute()
```

---

**Parameters**

**sinceDate** (*str*) – the date for response data to start from

**Returns**

the calling *SearchQuery*

**Return type**

Self

**until**(*toDate*)

Retrieve items for which `created_at <= toDate`

**Parameters**

**toDate** (*str*) – the date for response data to end at (inclusive)

**Returns**

the calling *SearchQuery*

**Return type**

Self

**property result:** `Optional[Union[Model, List[Model]]]`

The result of the search query, wrapped by the *Model* corresponding to the endpoint

**Returns**

the API response as either an individual or list of *Model* objects

**validate\_result**()

Parses the response and returns the actual result data, regardless of search approach

**Return type**

`Optional[Union[Dict, List[Dict]]]`

**parse**(*data*)

Parses the API response with the corresponding *Model* object

**Parameters**

**data** (*dict*) – API response data of a single item

**Return type**

*Model*

**reset**()

Resets the query and result, allowing the object to be reused

**property result\_count:** `int`

Number of items that matched the search criteria

**property result\_type:** `Type`

The type of the result

property `last_group`: `int`

The most recent filter group on the query

**Returns**

the most recent filter group, or -1 if no criteria has been added

**class** `magento.search.OrderSearch`(*client*)

Bases: `SearchQuery`

`SearchQuery` subclass for the orders endpoint

`__init__`(*client*)

Initialize an `OrderSearch`

**Parameters**

`client` (`Client`) – an initialized `Client` object

`by_number`(*order\_number*)

Retrieve an `Order` by number

**Parameters**

`order_number` (`Union[int, str]`) – the order number (`increment_id`)

**Return type**

`Optional[Order]`

`by_product`(*product*)

Search for all `Order` s of a `Product`

**Parameters**

`product` (`Product`) – the `Product` to search for in orders

**Return type**

`Optional[Union[Order, List[Order]]]`

`by_sku`(*sku*)

Search for `Order` by product sku

---

**Note:** Like `OrderItemSearch.by_sku()`, the sku will need to be an exact match to the sku of a simple product, including a custom option if applicable

- Use `by_product()` or `by_product_id()` to find orders containing any of the `option_skus` and/or all `children` of a configurable product
- 

**Parameters**

`sku` (`str`) – the exact product sku to search for in orders

**Return type**

`Optional[Union[Order, List[Order]]]`

`by_product_id`(*product\_id*)

Search for `Order` s by `product_id`

**Parameters**

`product_id` (`Union[int, str]`) – the id (`product_id`) of the product to search for in orders

**Return type***Optional[Union[Order, List[Order]]]***by\_category\_id**(*category\_id*, *search\_subcategories=False*)Search for `Order` s by `category_id`**Parameters**

- **category\_id** (`Union[int, str]`) – id of the category to search for in orders
- **search\_subcategories** (`bool`) – if `True`, also searches for orders from `all_subcategories`

**Returns**any `Order` containing a `Product` in the corresponding `Category`**Return type***Optional[Union[Order, List[Order]]]***by\_category**(*category*, *search\_subcategories=False*)Search for `Order` s that contain any of the category's `products`**Parameters**

- **category** (`Category`) – the `Category` to use in the search
- **search\_subcategories** (`bool`) – if `True`, also searches for orders from `all_subcategories`

**Returns**any `Order` that contains a product in the provided category**Return type***Optional[Union[Order, List[Order]]]***by\_skulist**(*skulist*)Search for `Order` s using a list or comma separated string of product SKUs**Parameters****skulist** (`Union[str, Iterable[str]]`) – an iterable or comma separated string of product SKUs**Return type***Optional[Union[Order, List[Order]]]***from\_items**(*items*)Retrieve unique `Order` objects from `OrderItem` entries using a single request**Parameters****items** (`Optional[OrderItem | List[OrderItem]]`) – an individual/list of order items**Return type***Optional[Order, List[Order]]***class** `magento.search.OrderItemSearch`(*client*)Bases: `SearchQuery``SearchQuery` subclass for the `orders/items` endpoint

`__init__(client)`

Initialize an `OrderItemSearch`

**Parameters**

`client` (`Client`) – an initialized `Client` object

**property result:** `Optional[Union[OrderItem, List[OrderItem]]]`

The result of the search query, wrapped by the `Model` corresponding to the endpoint

**Returns**

the API response as either an individual or list of `Model` objects

`parse(data)`

Overrides `SearchQuery.parse()` to fully hydrate `OrderItem` objects

Extra validation is required for `OrderItems`, as duplicated and/or incomplete data is returned when the child of a configurable product is searched `by_sku()` or `by_product()`

**Parameters**

`data` – API response data

**Return type**

`Optional[OrderItem]`

`by_product(product)`

Search for `OrderItem` entries by `Product`

---

**Note:** This will match `OrderItems` that contain

- Any of the child products of a configurable product
  - Any of the `option_skus` of a product with custom options
- 

**Parameters**

`product` (`Product`) – the `Product` to search for in order items

**Return type**

`Optional[Union[OrderItem, List[OrderItem]]]`

`by_sku(sku)`

Search for `OrderItem` entries by product sku.

---

**The SKU must be an exact match to the OrderItem SKU**

`OrderItems` always use the SKU of a simple product, including any custom options. This means that:

- Searching the SKU of a configurable product returns nothing
- If a product has custom options, the search will only find `OrderItems` that contain the specific option sku (or base sku) that's provided

To search for `OrderItems` containing all children of a configurable product and/or all possible `option_skus`, use `by_product()` or `by_product_id()`

---

**Parameters**

`sku` (`str`) – the exact product sku to search for in order items



**Return type***Optional[Union[OrderItem, List[OrderItem]]]***by\_product\_id**(*product\_id*)Search for `OrderItem` entries by product id.**Parameters****product\_id** (`Union[int, str]`) – the id (`product_id`) of the `Product` to search for in order items**Return type***Optional[Union[OrderItem, List[OrderItem]]]***by\_category\_id**(*category\_id*, *search\_subcategories=False*)Search for `OrderItem` entries by `category_id`**Parameters**

- **category\_id** (`Union[int, str]`) – id of the `Category` to search for in order items
- **search\_subcategories** (`bool`) – if `True`, also searches for order items from `all_subcategories`

**Returns**any `OrderItem` containing a `Product` in the corresponding `Category`**Return type***Optional[Union[OrderItem, List[OrderItem]]]***by\_category**(*category*, *search\_subcategories=False*)Search for `OrderItem` entries that contain any of the category's products**Parameters**

- **category** (`Category`) – the `Category` to use in the search
- **search\_subcategories** (`bool`) – if `True`, also searches for order items from `all_subcategories`

**Return type***Optional[Union[OrderItem, List[OrderItem]]]***by\_skulist**(*skulist*)Search for `:class:`~.OrderItem`s` using a list or comma-separated string of product SKUs**Parameters****skulist** (`Union[str, Iterable[str]]`) – an iterable or comma separated string of product SKUs**Return type***Optional[Union[OrderItem, List[OrderItem]]]***class** `magento.search.InvoiceSearch`(*client*)Bases: `SearchQuery``SearchQuery` subclass for the invoices endpoint**\_\_init\_\_**(*client*)Initialize an `InvoiceSearch`

**Parameters**

**client** (*Client*) – an initialized *Client* object

**by\_number**(*invoice\_number*)

Retrieve an *Invoice* by number

**Parameters**

**invoice\_number** (*Union[int, str]*) – the invoice number (*increment\_id*)

**Return type**

*Optional[Invoice]*

**by\_order\_number**(*order\_number*)

Retrieve an *Invoice* by order number

**Parameters**

**order\_number** (*Union[int, str]*) – the order number (*increment\_id*)

**Return type**

*Optional[Invoice]*

**by\_order**(*order*)

Retrieve the *Invoice* for an *Order*

**Parameters**

**order** (*Order*) – the *Order* object to retrieve an invoice for

**Return type**

*Optional[Invoice]*

**by\_order\_id**(*order\_id*)

Retrieve an *Invoice* by *order\_id*

**Parameters**

**order\_id** (*Union[int, str]*) – the *order\_id* of the order to retrieve an invoice for

**Return type**

*Optional[Invoice]*

**by\_product**(*product*)

Search for all *Invoice* s of a *Product*

**Parameters**

**product** (*Product*) – the *Product* to search for in invoices

**Return type**

*Optional[Union[Invoice, List[Invoice]]]*

**by\_sku**(*sku*)

Search for *Invoice* s by product sku

---

**Note:** Like *OrderItemSearch.by\_sku()*, the sku will need to be an exact match to the sku of a simple product, including a custom option if applicable

- Use *by\_product()* or *by\_product\_id()* to find orders containing any of the *option\_skus* and/or all *children* of a configurable product

---

**Parameters**

**sku** (*str*) – the exact product sku to search for in invoices

**Return type***Optional[Union[Invoice, List[Invoice]]]***by\_product\_id**(*product\_id*)Search for *Invoice* s by *product\_id***Parameters****product\_id** (*Union[int, str]*) – the id (*product\_id*) of the product to search for in invoices**Return type***Optional[Union[Invoice, List[Invoice]]]***by\_category\_id**(*category\_id*, *search\_subcategories=False*)Search for *Invoice* s by *category\_id***Parameters**

- **category\_id** (*Union[int, str]*) – id of the category to search for in orders
- **search\_subcategories** (*bool*) – if *True*, also searches for orders from *all\_subcategories*

**Returns**any *Invoice* containing a *Product* in the corresponding *Category***Return type***Optional[Union[Invoice, List[Invoice]]]***by\_category**(*category*, *search\_subcategories=False*)Search for *Invoice* s that contain any of the category's products**Parameters**

- **category** (*Category*) – the *Category* to use in the search
- **search\_subcategories** (*bool*) – if *True*, also searches for orders from *all\_subcategories*

**Returns**any *Invoice* that contains a product in the provided category**Return type***Optional[Union[Invoice, List[Invoice]]]***by\_skulist**(*skulist*)Search for *Invoice* s using a list or comma separated string of product SKUs**Parameters****skulist** (*Union[str, Iterable[str]]*) – an iterable or comma separated string of product SKUs**Return type***Optional[Union[Invoice, List[Invoice]]]***from\_order\_items**(*items*)Retrieve unique *Invoice* objects from *OrderItem* entries using a single request

---

**Tip:** Since there is no *invoices/items* endpoint, to search for invoices we must first do an *OrderItemSearch*, then retrieve the *order\_ids* and search *by\_order\_id()*

---

**Parameters**

**items** (Optional[OrderItem | List[OrderItem]]) – an individual/list of order items

**Return type**

Optional[Invoice, List[Invoice]]

**class** magento.search.ProductSearch(client)

Bases: SearchQuery

SearchQuery subclass for the products endpoint

**\_\_init\_\_**(client)

Initialize a ProductSearch

**Parameters**

**client** (Client) – an initialized Client object

**property attributes:** ProductAttributeSearch

Alternate way to access the SearchQuery for ProductAttribute data

**by\_id**(item\_id)

Retrieve a Product by product\_id

---

**Note:** Response data from the products endpoint only has an id field, but all other endpoints that return data about products will use product\_id

---

**Parameters**

**item\_id** (Union[int, str]) – the id (product\_id) of the product

**Return type**

Optional[Product]

**by\_sku**(sku)

Retrieve a Product by sku

**Parameters**

**sku** – the product sku

**Return type**

Optional[Product]

**by\_skulist**(skulist)

Search for :class:`~Product`'s using a list or comma separated string of SKUs

**Parameters**

**skulist** (Union[str, Iterable[str]]) – an iterable or comma separated string of SKUs

**Return type**

Optional[Union[Product, List[Product]]]

**by\_category**(category, search\_subcategories=False)

Search for Product s in a Category

**Parameters**

- **category** (Category) – the Category to retrieve products from

- **search\_subcategories** (bool) – if True, also retrieves products from `all_subcategories`

**Return type***Optional[Union[Product, List[Product]]]***by\_category\_id**(*category\_id*, *search\_subcategories=False*)Search for `Product`s by `category_id`**Parameters**

- **category\_id** (Union[int, str]) – the id of the `Category` to retrieve products from
- **search\_subcategories** (bool) – if True, also retrieves products from `all_subcategories`

**Return type***Optional[Union[Product, List[Product]]]***get\_stock**(*sku*)Retrieve the `stock` of a product by `sku`**Parameters****sku** – the product `sku`**Return type***Optional[int]***class** `magento.search.ProductAttributeSearch`(*client*)Bases: `SearchQuery``SearchQuery` subclass for the `products/attributes` endpoint**\_\_init\_\_**(*client*)Initialize a `ProductAttributeSearch`**Parameters****client** (`Client`) – an initialized `Client` object**get\_all**()Retrieve a list of all `:class:`~.ProductAttribute`s`**Return type***Optional[List[ProductAttribute]]***by\_code**(*attribute\_code*)Retrieve a `ProductAttribute` by its attribute code**Parameters****attribute\_code** (str) – the code of the `ProductAttribute`**Return type***Optional[ProductAttribute]***get\_types**()Retrieve a list of all available `ProductAttribute` types**Return type***Optional[List[APIResponse]]*

**class** magento.search.**CategorySearch**(*client*)

Bases: [SearchQuery](#)

[SearchQuery](#) subclass for the categories endpoint

**\_\_init\_\_**(*client*)

Initialize a [CategorySearch](#)

**Parameters**

**client** ([Client](#)) – an initialized [Client](#) object

**get\_root**()

Retrieve the top level/default [Category](#) (every other category is a subcategory)

**Return type**

[Category](#)

**get\_all**()

Retrieve a list of all categories

**Return type**

[List](#)[[Category](#)]

**by\_name**(*name*, *exact=True*)

Search for a [Category](#) by name

**Parameters**

- **name** ([str](#)) – the category name to search for
- **exact** ([bool](#)) – whether the name should be an exact match

**Return type**

[Optional](#)[[Union](#)[[Category](#), [List](#)[[Category](#)]]]

## 5.3 The exceptions module

**exception** magento.exceptions.**MagentoError**(*client*, *msg=None*, *response=None*)

Bases: [Exception](#)

Base exception class for error responses returned by the Magento API

**Variables**

**DEFAULT\_MSG** – default exception message to use if a message isn't provided

**DEFAULT\_MSG** = 'An error occurred while processing the request.'

**\_\_init\_\_**(*client*, *msg=None*, *response=None*)

Log and raise a [MagentoError](#)

**Parameters**

- **client** ([Client](#)) – an initialized [Client](#) object
- **msg** ([Optional](#)[[str](#)]) – optional exception message; prepended to the error message of the response
- **response** ([Optional](#)[[Response](#)]) – optional response to [parse\(\)](#) an error message from

**static parse**(*response*)

Parses the error message from the response

**Parameters**

**response** (Union[Response, Dict]) – a bad response returned by the Magento API

**Raises**

TypeError if response is not a Response or Dict

**Return type**

*str*

**exception** magento.exceptions.AuthenticationError(*client, msg=None, response=None*)

Bases: MagentoError

Exception class for errors when trying to `authenticate()` a Client

DEFAULT\_MSG = 'Failed to authenticate credentials.'

**\_\_init\_\_**(*client, msg=None, response=None*)

Log and raise a MagentoError

**Parameters**

- **client** (Client) – an initialized Client object
- **msg** (Optional[str]) – optional exception message; prepended to the error message of the response
- **response** (Optional[Response]) – optional response to parse() an error message from

## 5.4 The utils module

**class** magento.utils.ItemManager(*items=None*)

Bases: object

**\_\_init\_\_**(*items=None*)

**add**(*item*)

**get\_attrs**(*attr*)

**sum\_attrs**(*attr*)

**class** magento.utils.LoggerUtils

Bases: object

Utility class that simplifies access to logger handler info

**static get\_handler\_names**(*logger*)

Get all handler names

**Return type**

List[str]

**static** `get_stream_handlers(logger)`

Get all the StreamHandlers of the current logger (NOTE: StreamHandler subclasses excluded)

**Return type**

*List[Handler]*

**static** `get_file_handlers(logger)`

Get all the FileHandlers of the current logger

**Return type**

*List[FileHandler]*

**static** `get_log_files(logger)`

Get the log file paths from all FileHandlers of a logger

**Return type**

*List[str]*

**static** `get_handler_by_log_file(logger, log_file)`

Returns the FileHandler logging to the specified file, given it exists

**Return type**

*Union[FileHandler, List[FileHandler]]*

**static** `clear_handlers(logger)`

**Return type**

*bool*

**static** `clear_stream_handlers(logger)`

Removes all StreamHandlers from a logger

**Return type**

*bool*

**static** `clear_file_handlers(logger)`

Removes all FileHandlers from a logger

**Return type**

*bool*

**static** `map_handlers_by_name(logger)`

Map the handlers of a logger first by type, and then by their name

FileHandlers are mapped to both their handlers and log file, while StreamHandlers are just mapped to the handler Handlers without a name will be skipped, because look at the method name (:

**class** `magento.utils.MagentoLogger(name, log_file=None, stdout_level='INFO', log_requests=True)`

Bases: `object`

Logging class used within the package

**Variables**

- **PREFIX** – hardcoded prefix to use in log messages
- **PACKAGE\_LOG\_NAME** – the default name for the package logger
- **CLIENT\_LOG\_NAME** – the default format for the client logger name



- **LOG\_MESSAGE** – the default format for the message component of log messages. (Use `magento.logger.LOG_MESSAGE` for easy access)
- **FORMATTER** – the default logging format
- **HANDLER\_NAME** – the default format for the names of handlers created by this package

`PREFIX = 'MyMagento'`

`PACKAGE_LOG_NAME = 'my-magento'`

`CLIENT_LOG_NAME = '{domain}_{username}'`

`HANDLER_NAME = 'MyMagento_{name}_{stdout_level}'`

`LOG_MESSAGE = '|[ MyMagento | {name} ]|: {message}'`

`FORMATTER = <logging.Formatter object>`

`__init__(name, log_file=None, stdout_level='INFO', log_requests=True)`

Initialize the logger

Each Client object corresponds to a unique username/domain combination, which is used to attach it to its associated MagentoLogger and log file, allowing all activity across all endpoints to be tracked. A package logger exists as well, which logs all activity from the package. All log files have their log level set to DEBUG

#### Parameters

- **name** (*str*) – logger name
- **log\_file** (*Optional[str]*) – log file name; default is `{name}.log`
- **stdout\_level** (*Union[int, str]*) – logging level for stdout logger; default is “INFO” (which is also `logging.INFO` and 10)
- **log\_requests** (*bool*) – set to True to add logging from the requests package logger

`setup_logger(stdout_level='INFO', log_requests=True)`

Configures a logger and assigns it to the `logger` attribute.

#### Parameters

- **stdout\_level** (*Union[int, str]*) – logging level to use for logging to console
- **log\_requests** (*bool*) – set to True to add logs from the requests package (ie. API call logging)

#### Return type

*bool*

`format_msg(msg)`

Formats the `LOG_MESSAGE` using the specified message

#### Return type

*str*

`debug(msg)`

Formats the `LOG_MESSAGE` with the specified message, then logs it with `Logger.debug()`

`info(msg)`

Formats the `LOG_MESSAGE` with the specified message, then logs it with `Logger.info()`

**error**(*msg*)

Formats the `LOG_MESSAGE` with the specified message, then logs it with `Logger.error()`

**warning**(*msg*)

Formats the `LOG_MESSAGE` with the specified message, then logs it with `Logger.warning()`

**critical**(*msg*)

Formats the `LOG_MESSAGE` with the specified message, then logs it with `Logger.critical()`

**property handlers**

**property handler\_names**

**property handler\_map**

**property file\_handlers**

**property stream\_handlers**

**property log\_files**

**property log\_path**

**static get\_magento\_handlers**(*logger*)

**static clear\_magento\_handlers**(*logger, handler\_type, clear\_pkg=False*)

Clear all handlers from a logger that were created by `MagentoLogger`

**Parameters**

- **logger** (`Logger`) – any logger
- **handler\_type** (`Union[Type[FileHandler], Type[StreamHandler]]`) – the logging handler type to check for and remove
- **clear\_pkg** (`bool`) – if `True`, will delete the package handler for writing to `my-magento.log` (Default is `False`)

**static clear\_magento\_file\_handlers**(*logger, clear\_pkg=False*)

**static clear\_magento\_stdout\_handlers**(*logger, clear\_pkg=False*)

**static owns\_handler**(*handler*)

Checks if a handler is a `Stream/FileHandler` from this package or not

**static get\_package\_handler**()

Returns the `FileHandler` object that writes to the `magento.log` file

**Return type**

*FileHandler*

**static add\_request\_logging**(*handler*)

Adds the specified handler to the requests package logger, allowing for easier debugging of API calls

`magento.utils.get_agents()`

Scrapes a list of user agents. Returns a default list if the scrape fails.

**Return type**

*list*

`magento.utils.get_agent(index=0)`

Returns a single user agent string from the specified index of the AGENTS list

**Return type**

*str*

`magento.utils.get_package_file_handler()`

...

The `magento.models` Subpackage

- The `model` module
- The `product` module
- The `category` module
- The `order` module
- The `invoice` module



## THE MAGENTO.MODELS SUBPACKAGE

The `magento.models` subpackage contains all of the `Model` API response wrapper classes.

### 6.1 The `model` module

**class** `magento.models.model.Model`(*data*, *client*, *endpoint*, *private\_keys=True*)

Bases: `ABC`

The abstract base class of all API response wrapper classes

#### Overview

- A `Model` wraps the response data from an API endpoint
- Several endpoints have subclasses with additional methods to retrieve/update data
- All other endpoints are wrapped using a general `APIResponse`
- The endpoint's corresponding `SearchQuery` can be accessed via `query_endpoint()`

**DOCUMENTATION:** `str = None`

Link to the Official Magento 2 API documentation for the endpoint wrapped by the `Model`

**IDENTIFIER:** `str = None`

The API response field that the endpoint's `uid` comes from

**\_\_init\_\_**(*data*, *client*, *endpoint*, *private\_keys=True*)

Initialize a `Model` object from an API response and the endpoint that it came from

...

---

**Tip:** The `endpoint` is used to:

- Generate the `url_for()` any requests made by subclass-specific methods
  - Match the `Model` to its corresponding `SearchQuery` object
  - Determine how to `parse()` new `Model` objects from API responses
- 

...

**Parameters**

- **data** (`dict`) – the JSON from an API response to use as source data
- **client** (`Client`) – an initialized `Client`
- **endpoint** (`str`) – the API endpoint that the `Model` wraps
- **private\_keys** (`bool`) – if `True`, sets the keys in the `excluded_keys` as private attributes (prefixed with `__`) instead of fully excluding them

**set\_attrs**(`data`, `private_keys=True`)

Initializes object attributes using the JSON from an API response as the data source

Called at the time of object initialization, but can also be used to update the source data and reinitialize the attributes without creating a new object

#### Parameters

- **data** (`dict`) – the API response JSON to use as the object source data
- **private\_keys** (`bool`) – if set to `True`, will set the `excluded_keys` as private attributes (prefixed with `__`) instead of fully excluding them

---

#### Private Keys Clarification

Let's say that "status" is in the `excluded_keys`

- No matter what, the status attribute will not be set on the `Model`
  - If `private_keys==True`, the `__status` attribute will be set (using the status data)
  - If `private_keys==False`, the data from status is completely excluded
- 

**abstract property excluded\_keys:** `List[str]`

API response keys that shouldn't be set as object attributes by `set_attrs()`

#### Returns

list of API response keys that shouldn't be set as attributes

**property uid:** `Union[str, int]`

Unique item identifier; used in the url of the `data_endpoint()`

**data\_endpoint**(`scope=None`)

Endpoint to use when requesting/updating the item's data

#### Parameters

**scope** (`Optional[str]`) – the scope to generate the `url_for()`

#### Return type

`str`

**query\_endpoint**()

Initializes and returns the `SearchQuery` object corresponding to the `Model`'s endpoint

#### Returns

a `SearchQuery` or subclass, depending on the endpoint

#### Return type

`SearchQuery`

**parse**(`response`)

Uses the instance's corresponding `SearchQuery` to parse an API response

**Parameters**

**response** (*dict*) – API response dict to use as source data

**Returns**

a *Model* with the same endpoint as the calling instance

**Return type**

*Model*

**refresh**(*scope=None*)

Updates object attributes in place using current data from the `data_endpoint()`

---

**Hint:** `refresh()` can be used to switch the scope of the source data without creating a new object or changing the `Client.scope`

---

**Example**

```
# Get product data on 'default' scope
>>> product = client.products.by_sku('sku42')
# Get fresh product data from different scope
>>> product.refresh('all')

Refreshed <Magento Product: sku42> on scope all
```

---

**Parameters**

**scope** (*Optional[str]*) – the scope to send the request on; uses the `Client.scope` if not provided

**Return type**

*bool*

**static unpack\_attributes**(*attributes, key='attribute\_code'*)

Unpacks a list of attribute dictionaries into a single dictionary

**Example**

```
>> custom_attrs = [{'attribute_code': 'attr', 'value': 'val'}, {'attribute_code':
↳ 'will_to_live', 'value': '0'}]
>> print(Model.unpack_attributes(custom_attrs))

{'attr': 'val', 'will_to_live': '0'}
```

---

**Parameters**

- **attributes** (*List[dict]*) – a list of custom attribute dictionaries
- **key** (*str*) – the key used in the attribute dictionary (ex. `attribute_code` or `label`)

**Returns**

a single dictionary of all custom attributes formatted as `{"attr": "val"}`

**Return type***dict***static pack\_attributes**(*attribute\_data*, *key*='attribute\_code')

Packs a dictionary containing attributes into a list of attribute dictionaries

---

**Example**

```
>> attribute_data = {'special_price': 12, 'meta_title': 'My Product'}
>> print(Model.pack_attributes(attribute_data))
>> print(Model.pack_attributes(attribute_data, key='label'))

[{'attribute_code': 'special_price', 'value': 12}, {'attribute_code': 'meta_title',
↪ 'value': 'My Product'}]
[{'label': 'special_price', 'value': 12}, {'label': 'meta_title', 'value': 'My_
↪ Product'}]
```

---

**Parameters**

- **attribute\_data** (*dict*) – a dictionary containing attribute data
- **key** (*str*) – the key to use when packing the attributes (ex. attribute\_code or label)

**Returns**

a list of dictionaries formatted as {key : "attr", "value": "value"}

**Return type***List[dict]***static encode**(*string*)URL-encode with `urllib.parse`; used for requests that could contain special characters**Parameters****string** (*str*) – the string to URL-encode**Return type***str***property cached: List[str]**Names of properties that are wrapped with `functools.cached_property()`**clear**(*\*keys*)Deletes the provided keys from the object's `__dict__`

To clear all cached properties:

```
>> self.clear(*self.cached)
```

**Parameters****keys** (*str*) – name of the object attribute(s) to delete**get\_scope\_name**(*scope*)

Returns the appropriate scope name to use for logging messages



**Return type***str*

...

**class** magento.models.model.**APIResponse**(*data, client, endpoint*)Bases: `Model`**IDENTIFIER:** `str = 'entity_id'`The API response field that the endpoint's `uid` comes from**\_\_init\_\_**(*data, client, endpoint*)A generic `Model` subclassWraps API responses when there isn't a `Model` subclass defined for the endpoint**Parameters**

- **data** (`dict`) – the API response from an API endpoint
- **client** (`Client`) – an initialized `Client` object
- **endpoint** (`str`) – the endpoint that the API response came from

**property** `excluded_keys:` `List[str]`API response keys that shouldn't be set as object attributes by `set_attrs()`**Returns**

list of API response keys that shouldn't be set as attributes

**property** `uid:` `Optional[int]`

Unique item identifier

---

**Note:** Since the `APIResponse` can wrap any endpoint, the response is checked for commonly used id fields (`entity_id` and `id`)

If the endpoint doesn't use those fields, `None` will be returned**data\_endpoint**(*scope=None*)

Endpoint to use when requesting/updating the item's data

**Parameters****scope** (`Optional[str]`) – the scope to generate the `url_for()`**Return type**`Optional[str]`

## 6.2 The product module

**class** magento.models.product.**Product**(*data, client*)Bases: `Model`

Wrapper for the products endpoint

`STATUS_ENABLED = 1`

`STATUS_DISABLED = 2`

`VISIBILITY_NOT_VISIBLE = 1`

`VISIBILITY_CATALOG = 2`

`VISIBILITY_SEARCH = 3`

`VISIBILITY_BOTH = 4`

**DOCUMENTATION:** `str = 'https://adobe-commerce.redoc.ly/2.3.7-admin/tag/products/'`

Link to the Official Magento 2 API documentation for the endpoint wrapped by the Model

**IDENTIFIER:** `str = 'sku'`

The API response field that the endpoint's `uid` comes from

`__init__(data, client)`

Initialize a Product object using an API response from the products endpoint

**Parameters**

- `data` (`dict`) – the API response from the products endpoint
- `client` (`Client`) – an initialized `Client` object

**property** `excluded_keys`

API response keys that shouldn't be set as object attributes by `set_attrs()`

**Returns**

list of API response keys that shouldn't be set as attributes

`update_stock(qty)`

Updates the stock quantity

**Parameters**

- `qty` (`int`) – the new stock quantity

**Return type**

`bool`

`update_status(status)`

Update the product status

**Parameters**

- `status` (`int`) – either 1 (for `STATUS_ENABLED`) or 2 (for `STATUS_DISABLED`)

**Return type**

`bool`

`update_price(price)`

Update the product price

**Parameters**

- `price` (`Union[int, float]`) – the new price

**Return type**

`bool`

**update\_special\_price**(*price*)

Update the product special price

**Parameters**

**price** (`Union[float, int]`) – the new special price

**Return type**

*bool*

**update\_name**(*name*, *scope=None*)

Update the product name

**Parameters**

- **name** (`str`) – the new name to use
- **scope** (`Optional[str]`) – the scope to send the request on; will use the `Client.scope` if not provided

**Return type**

*bool*

**update\_description**(*description*, *scope=None*)

Update the product description

**Parameters**

- **description** (`str`) – the new HTML description to use
- **scope** (`Optional[str]`) – the scope to send the request on; will use the `Client.scope` if not provided

**Return type**

*bool*

**update\_metadata**(*metadata*, *scope=None*)

Update the product metadata

**Parameters**

- **metadata** (`dict`) – the new `meta_title`, `meta_keyword` and/or `meta_description` to use
- **scope** (`Optional[str]`) – the scope to send the request on; will use the `Client.scope` if not provided

**Return type**

*bool*

**update\_attributes**(*attribute\_data*, *scope=None*)

Update top level product attributes with scoping taken into account

---

**Note:** Product attributes can have a Global, Store View or Website scope

**Global Attributes**

Values are updated on all store views and the admin

**Website Attributes**

Values are updated on all store views

**Store View Attributes**

Values are updated on the store view specified in the request scope

A second request will be made to update Store View and Website attributes on the admin, depending on how many [Store views](#) you have:

- **1 View:** admin values are updated for all attributes, regardless of scope
- **2+ Views:** admin values are updated only for [website\\_product\\_attributes](#)

**Parameters**

- **attribute\_data** ([dict](#)) – a dictionary of product attributes to update
- **scope** ([Optional\[str\]](#)) – the scope to send the request on; will use the [Client.scope](#) if not provided

**Return type**

[bool](#)

`update_custom_attributes(attribute_data, scope=None)`

Update custom attributes with scoping taken into account

See [update\\_attributes\(\)](#) for details

---

**Important**

This method only supports updating **custom attributes**

---

**Parameters**

- **attribute\_data** ([dict](#)) – a dictionary of custom attributes to update
- **scope** ([Optional\[str\]](#)) – the scope to send the request on; will use the [Client.scope](#) if not provided

**Return type**

[bool](#)

`get_orders()`

Searches for orders that contain the product

If the product is configurable, returns orders containing any of its child products

**Returns**

orders that contain the product, as an individual or list of [Order](#) objects

**Return type**

[Optional\[Order | List\[Order\]\]](#)

`get_order_items()`

Searches for order items that contain the product

If the product is configurable, returns order items containing any of its child products

**Returns**

order items that contain the product, as an individual or list of [OrderItem](#) objects

**Return type**

[Optional\[OrderItem | List\[OrderItem\]\]](#)

**get\_invoices()**

Searches for invoices that contain the product

If the product is configurable, returns invoices containing any of its child products

**Returns**

invoices that contain the product, as an individual or list of `Invoice` objects

**Return type**

`Optional[Invoice | List[Invoice]]`

**delete()**

Deletes the product

---

**Hint:** If you delete a product by accident, the `Product` object's `data` attribute will still contain the raw data, which can be used to recover it.

Alternatively, don't delete it by accident.

---

**Return type**

`bool`

**get\_children(*refresh=False, scope=None*)**

Retrieve the child simple products of a configurable product

**Parameters**

- **refresh** (`bool`) – if True, calls `refresh()` on the child products to retrieve full data
- **scope** (`Optional[str]`) – the scope to refresh the children on (when `refresh=True`)

**Return type**

`List[Product]`

**property children: List[Product]**

If the `Product` is a configurable product, returns a list of its child products

**property categories: Optional[Category | List[Category]]**

Categories the product is in, returned as a list of `Category` objects

**property media\_gallery\_entries: List[MediaEntry]**

The product's media gallery entries, returned as a list of `MediaEntry` objects

**property thumbnail: MediaEntry**

The `MediaEntry` corresponding to the product's thumbnail

**property thumbnail\_link: str**

Link of the product's `thumbnail` image

**get\_media\_by\_id(*entry\_id*)**

Access a `MediaEntry` of the product by id

**Parameters**

- **entry\_id** (`int`) – the id of the media gallery entry

**Return type**

`MediaEntry`

**property** `encoded_sku`: `str`

URL-encoded SKU, which is used in request endpoints

**property** `option_skus`: `List[str]`

The full SKUs for the product's customizable options, if they exist

---

**Hint:** When a product with customizable options is ordered, these SKUs are used by the API when retrieving and searching for `Order` and `OrderItem` data

---

**property** `stock`: `int`

Current stock quantity

**property** `stock_item`: `dict`

Stock data from the `StockItem` Interface

**property** `stock_item_id`: `int`

Item id of the `StockItem`, used to `update_stock()`

**property** `description`: `str`

Product description (as HTML)

**property** `special_price`: `float`

The current special (sale) price

**class** `magento.models.product.MediaEntry`(*product*, *entry*)

Bases: `Model`

Wraps a media gallery entry of a `Product`

`MEDIA_TYPES` = ['base', 'small', 'thumbnail', 'swatch']

`DOCUMENTATION`: `str` =

`'https://adobe-commerce.redoc.ly/2.3.7-admin/tag/productsskumediaentryId'`

Link to the Official Magento 2 API documentation for the endpoint wrapped by the `Model`

`IDENTIFIER`: `str` = 'id'

The API response field that the endpoint's `uid` comes from

`__init__`(*product*, *entry*)

Initialize a `MediaEntry` object for a `Product`

**Parameters**

- **product** (`Product`) – the `Product` that the gallery entry is associated with
- **entry** (`dict`) – the json response data to use as the source data

`query_endpoint`()

No search endpoint exists for media gallery entries

**property** `excluded_keys`: `List[str]`

API response keys that shouldn't be set as object attributes by `set_attrs()`

**Returns**

list of API response keys that shouldn't be set as attributes

**property is\_enabled**

**property is\_thumbnail**

**property link**

Permalink to the image

**disable**(*scope=None*)

Disables the MediaEntry on the given scope

**Parameters**

**scope** (*Optional[str]*) – the scope to send the request on; will use the `Client.scope` if not provided

**Return type**

*bool*

**enable**(*scope=None*)

Enables the MediaEntry on the given scope

**Parameters**

**scope** (*Optional[str]*) – the scope to send the request on; will use the `Client.scope` if not provided

**Return type**

*bool*

**add\_media\_type**(*media\_type, scope=None*)

Add a media type to the MediaEntry on the given scope

**Caution:** If the media type is already assigned to a different entry, it will be removed

**Parameters**

- **media\_type** (*str*) – one of the `MEDIA_TYPES`
- **scope** (*Optional[str]*) – the scope to send the request on; will use the `Client.scope` if not provided

**Return type**

*bool*

**remove\_media\_type**(*media\_type, scope=None*)

Remove a media type from the MediaEntry on the given scope

**Parameters**

- **media\_type** (*str*) – one of the `MEDIA_TYPES`
- **scope** (*Optional[str]*) – the scope to send the request on; will use the `Client.scope` if not provided

**Return type**

*bool*

**set\_media\_types**(*types, scope=None*)

Set media types for the MediaEntry on the given scope

**Parameters**

- **types** (*list*) – a list containing all `MEDIA_TYPES` to assign
- **scope** (*Optional[str]*) – the scope to send the request on; will use the `Client.scope` if not provided

**Return type**

*bool*

**set\_position**(*position, scope=None*)

Set the position of the MediaEntry on the given scope

**Parameters**

- **position** (*int*) – the position to change to
- **scope** (*Optional[str]*) – the scope to send the request on; will use the `Client.scope` if not provided

**Return type**

*bool*

**set\_alt\_text**(*text, scope=None*)

Set the alt text (label) of the MediaEntry on the given scope

**Parameters**

- **text** (*str*) – the alt text to use
- **scope** (*Optional[str]*) – the scope to send the request on; will use the `Client.scope` if not provided

**Return type**

*bool*

**update**(*scope=None*)

Uses the data dict to update the media entry

---

**Note:** Some updates alter the data of other entries; if the update is successful, the associated `Product` will be refreshed on the same scope to keep the data consistent

---

---

**Tip:** If there's only 1 store view, the admin will also be updated

---

**Parameters**

**scope** (*Optional[str]*) – the scope to send the request on; will use the `Client.scope` if not provided

**Return type**

*bool*

**class** `magento.models.product.ProductAttribute`(*data, client*)

Bases: `Model`

Wrapper for the products/attributes endpoint



**DOCUMENTATION:** `str =`

`'https://adobe-commerce.redoc.ly/2.3.7-admin/tag/productsattributes/'`

Link to the Official Magento 2 API documentation for the endpoint wrapped by the Model

**IDENTIFIER:** `str = 'attribute_code'`

The API response field that the endpoint's `uid` comes from

`__init__(data, client)`

Initialize a ProductAttribute object using an API response from the products/attributes endpoint

**Parameters**

- `data (dict)` – the API response from the products/attributes endpoint
- `client (Client)` – an initialized `Client` object

**property** `excluded_keys: List[str]`

API response keys that shouldn't be set as object attributes by `set_attrs()`

**Returns**

list of API response keys that shouldn't be set as attributes

**property options**

## 6.3 The category module

**class** `magento.models.category.Category(data, client)`

Bases: `Model`

Wrapper for the categories endpoint

**DOCUMENTATION:** `str = 'https://adobe-commerce.redoc.ly/2.3.7-admin/tag/categories'`

Link to the Official Magento 2 API documentation for the endpoint wrapped by the Model

**IDENTIFIER:** `str = 'id'`

The API response field that the endpoint's `uid` comes from

`__init__(data, client)`

Initialize a Category object using an API response from the categories endpoint

**Parameters**

`data (dict)` – raw API response

**property** `excluded_keys`

API response keys that shouldn't be set as object attributes by `set_attrs()`

**Returns**

list of API response keys that shouldn't be set as attributes

**property** `custom_attributes: Dict[str, str]`

**property** `subcategories: List[Category]`

The child categories, returned as a list of `Category` objects

---

**Note:** Only the direct child categories are returned. For a list of all descendants, use `all_subcategories`

---

**property subcategory\_ids: List[int]**

The `category_ids` of the `subcategories`

**property subcategory\_names: List[str]**

The names of the category's `subcategories`

**property all\_subcategories: Optional[List[Category]]**

Recursively retrieves all descendants of the category

**property all\_subcategory\_ids: List[int]**

The `category_ids` of `all_subcategories`

**property products: List[Product]**

The `Product` s in the category

Alias for `get_products()`

**property product\_ids: List[int]**

The `product_ids` of the category's `products`

**property skus: List[str]**

The `skus` of the category's `products`

**property all\_products: List[Product]**

The `Product` s in the category and in `all_subcategories`

Alias for `get_products()` with `search_subcategories=True`

**property all\_product\_ids: Set[int]**

The `product_ids` of the products in the category and in `all_subcategories`

**property all\_skus: Set[str]**

The `skus` of the products in the category and in `all_subcategories`

**get\_products(*search\_subcategories=False*)**

Retrieves the category's products

**Parameters**

**search\_subcategories** (bool) – if True, also retrieves products from `all_subcategories`

**Return type**

*Optional[Product | List[Product]]*

**get\_orders(*search\_subcategories=False*)**

Retrieve any `Order` that contains one of the category's `products`

**Parameters**

**search\_subcategories** (bool) – if True, also searches for orders from `all_subcategories`

**Return type**

*Optional[Order | List[Order]]*

**get\_order\_items(*search\_subcategories=False*)**

Retrieve any `OrderItem` that contains one of the category's `products`

**Parameters**

**search\_subcategories** (bool) – if True, also searches for order items from `all_subcategories`

**Return type***Optional[OrderItem | List[OrderItem]]***get\_invoices**(*search\_subcategories=False*)Retrieve any *Invoice* that contains one of the category's *products***Parameters****search\_subcategories** (*bool*) – if *True*, also searches for invoices from *all\_subcategories***Return type***Optional[Invoice | List[Invoice]]*

## 6.4 The order module

**class** `magento.models.order.Order`(*data, client*)Bases: `Model`

Wrapper for the orders endpoint

**DOCUMENTATION:** `str = 'https://adobe-commerce.redoc.ly/2.3.7-admin/tag/orders'`Link to the Official Magento 2 API documentation for the endpoint wrapped by the `Model`**IDENTIFIER:** `str = 'entity_id'`The API response field that the endpoint's `uid` comes from`__init__`(*data, client*)Initialize an `Order` object using an API response from the orders endpoint**Parameters**

- **data** (`dict`) – API response from the orders endpoint
- **client** (`Client`) – an initialized `Client` object

**property** `excluded_keys`: `List[str]`API response keys that shouldn't be set as object attributes by `set_attrs()`**Returns**

list of API response keys that shouldn't be set as attributes

**property** `id`: `int`Alias for `entity_id`**property** `number`: `str`Alias for `increment_id`**property** `items`: `List[OrderItem]`The ordered items, returned as a list of `OrderItem` objects

---

**Note:** When a configurable `Product` is ordered, the API returns data for both the configurable and simple product

- The `OrderItem` is initialized using the configurable product data, since the simple product data is incomplete
- The `product` and `product_id` will still match the simple product though

If both entries are needed, the unparsed response is in the data dict

---

**property item\_ids:** `List[int]`

The `item_ids` of the ordered `items`

**property products:** `List[Product]`

The ordered `items`, returned as their corresponding `Product` objects

**get\_invoice()**

Retrieve the `Invoice` of the Order

**Return type**

`Invoice`

**property shipping\_address:** `dict`

Shipping details, from `extension_attributes.shipping_assignments`

**property bill\_to:** `dict`

Condensed version of the `billing_address` dict

**property bill\_to\_address:** `str`

The billing address, parsed into a single string

**property ship\_to:** `dict`

Condensed version of the `shipping_address` dict

**property ship\_to\_address:** `str`

The shipping address, parsed into a single string

**property payment:** `dict`

Payment data

**property net\_tax:** `float`

Final tax amount, with refunds and cancellations taken into account

**property net\_total:** `float`

Final Order value, with refunds and cancellations taken into account

**property item\_refunds:** `float`

Total amount refunded for items; excludes shipping and adjustment refunds/fees

**property total\_qty\_invoiced:** `int`

Total number of units invoiced

**property total\_qty\_shipped:** `int`

Total number of units shipped

**property total\_qty\_refunded:** `int`

Total number of units refunded

**property total\_qty\_canceled:** `int`

Total number of units canceled

**property total\_qty\_outstanding:** `int`

Total number of units that haven't been shipped/fulfilled yet

**property net\_qty\_ordered:** `int`

Total number of units ordered, after accounting for refunds and cancellations

**class** magento.models.order.**OrderItem**(item, client=None, order=None)

Bases: [Model](#)

Wrapper for the order/items endpoint

**DOCUMENTATION:** **str** = 'https://adobe-commerce.redoc.ly/2.3.7-admin/tag/ordersitems'

Link to the Official Magento 2 API documentation for the endpoint wrapped by the Model

**IDENTIFIER:** **str** = 'item\_id'

The API response field that the endpoint's **uid** comes from

**\_\_init\_\_**(item, client=None, order=None)

Initialize an OrderItem using an API response from the orders/items endpoint

---

**Note:** Initialization requires either a [Client](#) or [Order](#) object

---

#### Parameters

- **item** (**dict**) – API response from the orders/items endpoint
- **order** (Optional[[Order](#)]) – the [Order](#) that this is an item of
- **client** (Optional[[Client](#)]) – the [Client](#) to use (if not initializing with an [Order](#))

#### Raises

**ValueError** – if both the order and client aren't provided

**property excluded\_keys:** [List\[str\]](#)

API response keys that shouldn't be set as object attributes by [set\\_attrs\(\)](#)

#### Returns

list of API response keys that shouldn't be set as attributes

**property order:** [Order](#)

The corresponding [Order](#)

**property product:** [Product](#)

The item's corresponding [Product](#)

---

**Note: If the ordered item:**

- Is a configurable product - the child simple product is returned
  - Has custom options - the base product is returned
- 

**property product\_id:** **int**

Id of the corresponding simple [Product](#)

**property extension\_attributes:** **dict**

**property qty\_outstanding:** **int**

Number of units that haven't been shipped/fulfilled yet

**property net\_qty\_ordered:** **int**

Number of units ordered, after accounting for refunds and cancellations

**property net\_tax: float**

Tax amount after accounting for refunds and cancellations

**property net\_total: float**

Row total (incl. tax) after accounting for refunds and cancellations

**property net\_refund: float**

Refund amount after accounting for tax and discount refunds

**property total\_canceled: float**

Cancelled amount; note that partial cancellation is not possible

## 6.5 The invoice module

**class** magento.models.invoice.**Invoice**(data, client)

Bases: `Model`

Wrapper for the invoices endpoint

**DOCUMENTATION:** `str` = '<https://adobe-commerce.redoc.ly/2.3.7-admin/tag/invoices>'

Link to the Official Magento 2 API documentation for the endpoint wrapped by the Model

**IDENTIFIER:** `str` = 'entity\_id'

The API response field that the endpoint's `uid` comes from

**\_\_init\_\_**(data, client)

Initialize an Invoice object using an API response from the invoices endpoint

### Parameters

- **data** (`dict`) – API response from the invoices endpoint
- **client** (`Client`) – an initialized `Client` object

**property excluded\_keys:** `List[str]`

API response keys that shouldn't be set as object attributes by `set_attrs()`

### Returns

list of API response keys that shouldn't be set as attributes

**property id:** `int`

Alias for `entity_id`

**property number:** `str`

Alias for `increment_id`

**property order:** `Order`

The corresponding `Order`

**property items:** `List[InvoiceItem]`

The invoiced items, returned as a list of `InvoiceItem` objects

**class** magento.models.invoice.**InvoiceItem**(item, invoice)

Bases: `Model`

Wraps an item entry of an `Invoice`

**DOCUMENTATION:** `str = 'https://adobe-commerce.redoc.ly/2.3.7-admin/tag/invoicesid'`

Link to the Official Magento 2 API documentation for the endpoint wrapped by the Model

**IDENTIFIER:** `str = 'entity_id'`

The API response field that the endpoint's `uid` comes from

`__init__(item, invoice)`

Initialize an InvoiceItem of an Invoice

#### Parameters

- `item` (`dict`) – API response to use as source data
- `invoice` (`Invoice`) – the `Invoice` that this is an item of

`data_endpoint(scope=None)`

No data endpoint exists for invoice items

`query_endpoint()`

No search endpoint exists for invoice items

`property excluded_keys: List[str]`

API response keys that shouldn't be set as object attributes by `set_attrs()`

#### Returns

list of API response keys that shouldn't be set as attributes

`property order: Order`

The `Order` this item is from

`property order_item: OrderItem`

The item's corresponding `OrderItem`

`property product: Product`

The item's corresponding simple `Product`

`property product_id: int`

Id of the corresponding simple `Product`





## GET A MAGENTO 2 REST API TOKEN WITH MYMAGENTO

Let's generate a Magento 2 REST API token using the [MyMagento](#) REST API wrapper package.

### 7.1 Setting the Login Credentials

Use your Magento 2 login credentials to generate an `ACCESS_TOKEN`

```
domain = 'domain.com'  
username = 'username'  
password = 'password'
```

If you're using a local installation of Magento, your domain should look like this:

```
domain = '127.0.0.1/path/to/magento'
```

### 7.2 Getting a Client

Now that that's set that up, let's start using the API.

MyMagento uses the `Client` class to handle all interactions with the API

```
import magento  
  
api = magento.get_api(domain=domain, username=username, password=password, local=True)
```

```
2023-02-15 01:09:05 INFO    |[ MyMagento | 127_adam ]|: Authenticating adam on 127.0.0.1/  
↪magento24...  
2023-02-15 01:09:06 INFO    |[ MyMagento | 127_adam ]|: Logged in to adam
```

## 7.3 Setting Environment Variables

To log in faster in the future, you can set the following environment variables:

```
import os

os.environ['MAGENTO_DOMAIN'] = domain
os.environ['MAGENTO_USERNAME'] = username
os.environ['MAGENTO_PASSWORD'] = password
```

The `Client` can now be initialized as follows

```
import magento

api = magento.get_api(local=True)
```

```
2023-02-15 01:09:28 INFO    |[ MyMagento | 127_adam ]|: Authenticating adam on 127.0.0.1/
↪magento24...
2023-02-15 01:09:30 INFO    |[ MyMagento | 127_adam ]|: Logged in to adam
```

## ADD DISCOUNT ON EACH PRODUCT BASED ON PRODUCT PRICE

I saw this question on [Magento StackExchange](#):

---

### Question

I have 100 products in my store and each product has different price. I want to add discount on each product based on specific price. For example, If a product is added into the cart and it has 100usd price then i want to apply 10% discount on it and if a product is added into the cart and it has 110usd price then i want to apply 11% discount on it and so on. I hope you understand what I want to achieve. in simple words, discount on each product based on product price. Thanks

---

### 8.1 Solution Using MyMagento

First, you'll want to *Get a Magento 2 REST API Token With MyMagento*

```
import magento

>>> api = magento.get_api()
```

Let's say we have the skus of the 100 products in an array.

We can use a `ProductSearch` retrieve these products as `Product` objects using `by_skulist()`

```
>>> skus = [f"test_sku{n}" for n in range(1, 101)]
>>> products = api.products.by_skulist(skus)
```

To retrieve the `Product` objects using a field other than sku, like `product_id`, we can use `by_list()`:

```
>>> product_ids = list(range(1,101))
>>> products = api.products.by_list(
...     field="entity_id", # To search by product_id
...     values=product_ids
... )
```

Once we have our list of `Product` objects, we can calculate the discount based on their price, then update the `special_price` (discount price) using the `Product.update_special_price()` method

```
>>> for product in products:
...     if product.price < 100:
...         continue
```

(continues on next page)

(continued from previous page)

```
...  
...     discount = product.price / 1000  
...     price = product.price * (1 - discount)  
...     product.update_special_price(round(price, 2))
```

We loop through our list of products and

1. Skip the products if the price is not \$100 or more
2. Calculate the discount percentage by dividing the price by 1000
3. Calculate the new, discounted product price by multiplying the current price by 1 - discount
4. Use the new price we calculated to update the special\_price of the product

## CHANGELOG

### 9.1 v2.1.0

- Added `get_api()` to login using credentials stored in environment variables
  - The environment variables `MAGENTO_USERNAME`, `MAGENTO_PASSWORD`, `MAGENTO_DOMAIN` will be used if the domain, username or password kwargs are missing

```
import magento

>>> magento.get_api()

2023-02-08 03:34:20 INFO    |[ MyMagento | 127_user ]|: Authenticating user on 127.0.0.1/
↳path/to/magento
2023-02-08 03:34:23 INFO    |[ MyMagento | 127_user ]|: Logged in to user
<magento.clients.Client object at 0x000001CA83E1A200>
```

...

- Added local kwarg to `Client` to support locally hosted Magento stores and test environments
  - By default, `local=False`

```
from magento import Client

>>> api = Client("127.0.0.1/path/to/magento", "username", "password", local=True)
```

...

- Add `since()` and `until()` method to `SearchQuery` classes, which search the `created_at` field
  - They can be chained together and also with `add_criteria()`

---

#### Example

```
# Retrieve orders from the first 7 days of 2023
>>> api.orders.since("2023-01-01").until("2023-01-07").execute()

[<Magento Order: #000000012 placed on 2023-01-02 05:19:55>, ]
```

---

#### Example

```
# Retrieve orders over $50 placed since 2022
>>> api.orders.add_criteria(
...     field="grand_total",
...     value="50",
...     condition="gteq"
... ).since("2022-01-01").execute()

[<Magento Order: #000000003 placed on 2022-12-21 08:09:33>, ...]
```

...

- Changed `add_criteria()` to auto-increment the filter group by default if no group is specified (ie. AND condition)

```
# Retrieving products that are over $10 AND in the category with id 15
#
# Before v2.1.0
>>> api.products.add_criteria('category_id', '15').add_criteria('price', '10', 'gteq', group=1)

# v2.1.0+
>>> api.products.add_criteria('category_id', '15').add_criteria('price', '10', 'gteq')
```

...

- Changed the `Client.BASE_URL` to not include "www." at the start (see #8)
- Added unit tests for `url_for()`
- Added Jupyter notebook examples

## INDICES AND TABLES

- genindex
- modindex
- *Full Table of Contents*





## PYTHON MODULE INDEX

### m

- `magento.clients`, 15
- `magento.exceptions`, 34
- `magento.models.category`, 53
- `magento.models.invoice`, 58
- `magento.models.model`, 41
- `magento.models.order`, 55
- `magento.models.product`, 50
- `magento.search`, 21
- `magento.utils`, 35



## Symbols

\_\_init\_\_() (*magento.clients.Client* method), 15  
 \_\_init\_\_() (*magento.clients.Store* method), 20  
 \_\_init\_\_() (*magento.exceptions.AuthenticationError* method), 35  
 \_\_init\_\_() (*magento.exceptions.MagentoError* method), 34  
 \_\_init\_\_() (*magento.models.category.Category* method), 53  
 \_\_init\_\_() (*magento.models.invoice.Invoice* method), 58  
 \_\_init\_\_() (*magento.models.invoice.InvoiceItem* method), 59  
 \_\_init\_\_() (*magento.models.model.APIResponse* method), 45  
 \_\_init\_\_() (*magento.models.model.Model* method), 41  
 \_\_init\_\_() (*magento.models.order.Order* method), 55  
 \_\_init\_\_() (*magento.models.order.OrderItem* method), 57  
 \_\_init\_\_() (*magento.models.product.MediaEntry* method), 50  
 \_\_init\_\_() (*magento.models.product.Product* method), 46  
 \_\_init\_\_() (*magento.models.product.ProductAttribute* method), 53  
 \_\_init\_\_() (*magento.search.CategorySearch* method), 34  
 \_\_init\_\_() (*magento.search.InvoiceSearch* method), 29  
 \_\_init\_\_() (*magento.search.OrderItemSearch* method), 27  
 \_\_init\_\_() (*magento.search.OrderSearch* method), 26  
 \_\_init\_\_() (*magento.search.ProductAttributeSearch* method), 33  
 \_\_init\_\_() (*magento.search.ProductSearch* method), 32  
 \_\_init\_\_() (*magento.search.SearchQuery* method), 22  
 \_\_init\_\_() (*magento.utils.ItemManager* method), 35  
 \_\_init\_\_() (*magento.utils.MagentoLogger* method),

37

## A

ACCESS\_TOKEN (*magento.clients.Client* attribute), 16  
 active (*magento.clients.Store* property), 20  
 add() (*magento.utils.ItemManager* method), 35  
 add\_criteria() (*magento.search.SearchQuery* method), 22  
 add\_media\_type() (*magento.models.product.MediaEntry* method), 51  
 add\_request\_logging() (*magento.utils.MagentoLogger* static method), 38  
 all\_product\_attributes (*magento.clients.Store* property), 20  
 all\_product\_ids (*magento.models.category.Category* property), 54  
 all\_products (*magento.models.category.Category* property), 54  
 all\_skus (*magento.models.category.Category* property), 54  
 all\_subcategories (*magento.models.category.Category* property), 54  
 all\_subcategory\_ids (*magento.models.category.Category* property), 54  
 APIResponse (class in *magento.models.model*), 45  
 attributes (*magento.search.ProductSearch* property), 32  
 authenticate() (*magento.clients.Client* method), 19  
 AuthenticationError, 35

## B

BASE\_URL (*magento.clients.Client* attribute), 16  
 bill\_to (*magento.models.order.Order* property), 56  
 bill\_to\_address (*magento.models.order.Order* property), 56  
 by\_category() (*magento.search.InvoiceSearch* method), 31

- by\_category() (*magento.search.OrderItemSearch method*), 29
- by\_category() (*magento.search.OrderSearch method*), 27
- by\_category() (*magento.search.ProductSearch method*), 32
- by\_category\_id() (*magento.search.InvoiceSearch method*), 31
- by\_category\_id() (*magento.search.OrderItemSearch method*), 29
- by\_category\_id() (*magento.search.OrderSearch method*), 27
- by\_category\_id() (*magento.search.ProductSearch method*), 33
- by\_code() (*magento.search.ProductAttributeSearch method*), 33
- by\_id() (*magento.search.ProductSearch method*), 32
- by\_id() (*magento.search.SearchQuery method*), 24
- by\_list() (*magento.search.SearchQuery method*), 24
- by\_name() (*magento.search.CategorySearch method*), 34
- by\_number() (*magento.search.InvoiceSearch method*), 30
- by\_number() (*magento.search.OrderSearch method*), 26
- by\_order() (*magento.search.InvoiceSearch method*), 30
- by\_order\_id() (*magento.search.InvoiceSearch method*), 30
- by\_order\_number() (*magento.search.InvoiceSearch method*), 30
- by\_product() (*magento.search.InvoiceSearch method*), 30
- by\_product() (*magento.search.OrderItemSearch method*), 28
- by\_product() (*magento.search.OrderSearch method*), 26
- by\_product\_id() (*magento.search.InvoiceSearch method*), 31
- by\_product\_id() (*magento.search.OrderItemSearch method*), 29
- by\_product\_id() (*magento.search.OrderSearch method*), 26
- by\_sku() (*magento.search.InvoiceSearch method*), 30
- by\_sku() (*magento.search.OrderItemSearch method*), 28
- by\_sku() (*magento.search.OrderSearch method*), 26
- by\_sku() (*magento.search.ProductSearch method*), 32
- by\_skulist() (*magento.search.InvoiceSearch method*), 31
- by\_skulist() (*magento.search.OrderItemSearch method*), 29
- by\_skulist() (*magento.search.OrderSearch method*), 27
- by\_skulist() (*magento.search.ProductSearch method*), 32
- ## C
- cached (*magento.models.model.Model property*), 44
- categories (*magento.clients.Client property*), 18
- categories (*magento.models.product.Product property*), 49
- Category (*class in magento.models.category*), 53
- CategorySearch (*class in magento.search*), 33
- children (*magento.models.product.Product property*), 49
- clear() (*magento.models.model.Model method*), 44
- clear\_file\_handlers() (*magento.utils.LoggerUtils static method*), 36
- clear\_handlers() (*magento.utils.LoggerUtils static method*), 36
- clear\_magento\_file\_handlers() (*magento.utils.MagentoLogger static method*), 38
- clear\_magento\_handlers() (*magento.utils.MagentoLogger static method*), 38
- clear\_magento\_stdout\_handlers() (*magento.utils.MagentoLogger static method*), 38
- clear\_stream\_handlers() (*magento.utils.LoggerUtils static method*), 36
- Client (*class in magento.clients*), 15
- client (*magento.search.SearchQuery attribute*), 22
- CLIENT\_LOG\_NAME (*magento.utils.MagentoLogger attribute*), 37
- configs (*magento.clients.Store property*), 20
- critical() (*magento.utils.MagentoLogger method*), 38
- custom\_attributes (*magento.models.category.Category property*), 53
- ## D
- data\_endpoint() (*magento.models.invoice.InvoiceItem method*), 59
- data\_endpoint() (*magento.models.model.APIResponse method*), 45
- data\_endpoint() (*magento.models.model.Model method*), 42
- debug() (*magento.utils.MagentoLogger method*), 37
- DEFAULT\_MSG (*magento.exceptions.AuthenticationError attribute*), 35
- DEFAULT\_MSG (*magento.exceptions.MagentoError attribute*), 34
- delete() (*magento.clients.Client method*), 18

- delete() (*magento.models.product.Product* method), 49
- description (*magento.models.product.Product* property), 50
- disable() (*magento.models.product.MediaEntry* method), 51
- DOCUMENTATION (*magento.models.category.Category* attribute), 53
- DOCUMENTATION (*magento.models.invoice.Invoice* attribute), 58
- DOCUMENTATION (*magento.models.invoice.InvoiceItem* attribute), 59
- DOCUMENTATION (*magento.models.model.Model* attribute), 41
- DOCUMENTATION (*magento.models.order.Order* attribute), 55
- DOCUMENTATION (*magento.models.order.OrderItem* attribute), 57
- DOCUMENTATION (*magento.models.product.MediaEntry* attribute), 50
- DOCUMENTATION (*magento.models.product.Product* attribute), 46
- DOCUMENTATION (*magento.models.product.ProductAttribute* attribute), 52
- domain (*magento.clients.Client* attribute), 16
- ## E
- enable() (*magento.models.product.MediaEntry* method), 51
- encode() (*magento.models.model.Model* static method), 44
- encoded\_sku (*magento.models.product.Product* property), 49
- endpoint (*magento.search.SearchQuery* attribute), 22
- error() (*magento.utils.MagentaLogger* method), 37
- excluded\_keys (*magento.models.category.Category* property), 53
- excluded\_keys (*magento.models.invoice.Invoice* property), 58
- excluded\_keys (*magento.models.invoice.InvoiceItem* property), 59
- excluded\_keys (*magento.models.model.APIResponse* property), 45
- excluded\_keys (*magento.models.model.Model* property), 42
- excluded\_keys (*magento.models.order.Order* property), 55
- excluded\_keys (*magento.models.order.OrderItem* property), 57
- excluded\_keys (*magento.models.product.MediaEntry* property), 50
- excluded\_keys (*magento.models.product.Product* property), 46
- excluded\_keys (*magento.models.product.ProductAttribute* property), 53
- execute() (*magento.search.SearchQuery* method), 23
- extension\_attributes (*magento.models.order.OrderItem* property), 57
- ## F
- fields (*magento.search.SearchQuery* attribute), 22
- file\_handlers (*magento.utils.MagentaLogger* property), 38
- filter\_website\_attrs() (*magento.clients.Store* method), 21
- format\_msg() (*magento.utils.MagentaLogger* method), 37
- FORMATTER (*magento.utils.MagentaLogger* attribute), 37
- from\_dict() (*magento.clients.Client* class method), 17
- from\_items() (*magento.search.OrderSearch* method), 27
- from\_json() (*magento.clients.Client* class method), 17
- from\_order\_items() (*magento.search.InvoiceSearch* method), 31
- ## G
- get() (*magento.clients.Client* method), 18
- get\_agent() (in module *magento.utils*), 39
- get\_agents() (in module *magento.utils*), 38
- get\_all() (*magento.search.CategorySearch* method), 34
- get\_all() (*magento.search.ProductAttributeSearch* method), 33
- get\_api() (in module *magento*), 15
- get\_attrs() (*magento.utils.ItemManager* method), 35
- get\_children() (*magento.models.product.Product* method), 49
- get\_file\_handlers() (*magento.utils.LoggerUtils* static method), 36
- get\_handler\_by\_log\_file() (*magento.utils.LoggerUtils* static method), 36
- get\_handler\_names() (*magento.utils.LoggerUtils* static method), 35
- get\_invoice() (*magento.models.order.Order* method), 56
- get\_invoices() (*magento.models.category.Category* method), 55
- get\_invoices() (*magento.models.product.Product* method), 48
- get\_log\_files() (*magento.utils.LoggerUtils* static method), 36
- get\_logger() (*magento.clients.Client* method), 19

- get\_magento\_handlers() (*magento.utils.MagentaLogger* static method), 38  
 get\_media\_by\_id() (*magento.models.product.Product* method), 49  
 get\_order\_items() (*magento.models.category.Category* method), 54  
 get\_order\_items() (*magento.models.product.Product* method), 48  
 get\_orders() (*magento.models.category.Category* method), 54  
 get\_orders() (*magento.models.product.Product* method), 48  
 get\_package\_file\_handler() (in module *magento.utils*), 39  
 get\_package\_handler() (*magento.utils.MagentaLogger* static method), 38  
 get\_products() (*magento.models.category.Category* method), 54  
 get\_root() (*magento.search.CategorySearch* method), 34  
 get\_scope\_name() (*magento.models.model.Model* method), 44  
 get\_stock() (*magento.search.ProductSearch* method), 33  
 get\_stream\_handlers() (*magento.utils.LoggerUtils* static method), 35  
 get\_types() (*magento.search.ProductAttributeSearch* method), 33  
 global\_product\_attributes (*magento.clients.Store* property), 20
- ## H
- handler\_map (*magento.utils.MagentaLogger* property), 38  
 HANDLER\_NAME (*magento.utils.MagentaLogger* attribute), 37  
 handler\_names (*magento.utils.MagentaLogger* property), 38  
 handlers (*magento.utils.MagentaLogger* property), 38  
 headers (*magento.clients.Client* property), 19
- ## I
- id (*magento.models.invoice.Invoice* property), 58  
 id (*magento.models.order.Order* property), 55  
 IDENTIFIER (*magento.models.category.Category* attribute), 53  
 IDENTIFIER (*magento.models.invoice.Invoice* attribute), 58  
 IDENTIFIER (*magento.models.invoice.InvoiceItem* attribute), 59  
 IDENTIFIER (*magento.models.model.APIResponse* attribute), 45  
 IDENTIFIER (*magento.models.model.Model* attribute), 41  
 IDENTIFIER (*magento.models.order.Order* attribute), 55  
 IDENTIFIER (*magento.models.order.OrderItem* attribute), 57  
 IDENTIFIER (*magento.models.product.MediaEntry* attribute), 50  
 IDENTIFIER (*magento.models.product.Product* attribute), 46  
 IDENTIFIER (*magento.models.product.ProductAttribute* attribute), 53  
 info() (*magento.utils.MagentaLogger* method), 37  
 Invoice (class in *magento.models.invoice*), 58  
 InvoiceItem (class in *magento.models.invoice*), 58  
 invoices (*magento.clients.Client* property), 18  
 InvoiceSearch (class in *magento.search*), 29  
 is\_enabled (*magento.models.product.MediaEntry* property), 50  
 is\_single\_store (*magento.clients.Store* property), 20  
 is\_thumbnail (*magento.models.product.MediaEntry* property), 51  
 item\_ids (*magento.models.order.Order* property), 56  
 item\_refunds (*magento.models.order.Order* property), 56  
 ItemManager (class in *magento.utils*), 35  
 items (*magento.models.invoice.Invoice* property), 58  
 items (*magento.models.order.Order* property), 55
- ## L
- last\_group (*magento.search.SearchQuery* property), 25  
 link (*magento.models.product.MediaEntry* property), 51  
 load() (*magento.clients.Client* class method), 17  
 log\_files (*magento.utils.MagentaLogger* property), 38  
 LOG\_MESSAGE (*magento.utils.MagentaLogger* attribute), 37  
 log\_path (*magento.utils.MagentaLogger* property), 38  
 logger (*magento.clients.Client* attribute), 16  
 LoggerUtils (class in *magento.utils*), 35
- ## M
- magento.clients module, 15  
 magento.exceptions module, 34  
 magento.models.category module, 53  
 magento.models.invoice module, 58

- magento.models.model
    - module, 41
  - magento.models.order
    - module, 55
  - magento.models.product
    - module, 50
  - magento.search
    - module, 21
  - magento.utils
    - module, 35
  - MagentoError, 34
  - MagentoLogger (class in magento.utils), 36
  - map\_handlers\_by\_name() (magento.utils.LoggerUtils static method), 36
  - media\_gallery\_entries (magento.models.product.Product property), 49
  - MEDIA\_TYPES (magento.models.product.MediaEntry attribute), 50
  - MediaEntry (class in magento.models.product), 50
  - Model (class in magento.models.model), 41
  - Model (magento.search.SearchQuery attribute), 22
  - module
    - magento.clients, 15
    - magento.exceptions, 34
    - magento.models.category, 53
    - magento.models.invoice, 58
    - magento.models.model, 41
    - magento.models.order, 55
    - magento.models.product, 50
    - magento.search, 21
    - magento.utils, 35
- N**
- net\_qty\_ordered (magento.models.order.Order property), 56
  - net\_qty\_ordered (magento.models.order.OrderItem property), 57
  - net\_refund (magento.models.order.OrderItem property), 58
  - net\_tax (magento.models.order.Order property), 56
  - net\_tax (magento.models.order.OrderItem property), 57
  - net\_total (magento.models.order.Order property), 56
  - net\_total (magento.models.order.OrderItem property), 58
  - new() (magento.clients.Client class method), 16
  - number (magento.models.invoice.Invoice property), 58
  - number (magento.models.order.Order property), 55
- O**
- option\_skus (magento.models.product.Product property), 50
  - options (magento.models.product.ProductAttribute property), 53
  - Order (class in magento.models.order), 55
  - order (magento.models.invoice.Invoice property), 58
  - order (magento.models.invoice.InvoiceItem property), 59
  - order (magento.models.order.OrderItem property), 57
  - order\_item (magento.models.invoice.InvoiceItem property), 59
  - order\_items (magento.clients.Client property), 18
  - OrderItem (class in magento.models.order), 56
  - OrderItemSearch (class in magento.search), 27
  - orders (magento.clients.Client property), 18
  - OrderSearch (class in magento.search), 26
  - owns\_handler() (magento.utils MagentoLogger static method), 38
- P**
- pack\_attributes() (magento.models.model.Model static method), 44
  - PACKAGE\_LOG\_NAME (magento.utils MagentoLogger attribute), 37
  - parse() (magento.exceptions MagentoError static method), 34
  - parse() (magento.models.model.Model method), 42
  - parse() (magento.search.OrderItemSearch method), 28
  - parse() (magento.search.SearchQuery method), 25
  - payment (magento.models.order.Order property), 56
  - post() (magento.clients.Client method), 18
  - PREFIX (magento.utils MagentoLogger attribute), 37
  - Product (class in magento.models.product), 45
  - product (magento.models.invoice.InvoiceItem property), 59
  - product (magento.models.order.OrderItem property), 57
  - product\_attributes (magento.clients.Client property), 18
  - product\_id (magento.models.invoice.InvoiceItem property), 59
  - product\_id (magento.models.order.OrderItem property), 57
  - product\_ids (magento.models.category.Category property), 54
  - ProductAttribute (class in magento.models.product), 52
  - ProductAttributeSearch (class in magento.search), 33
  - products (magento.clients.Client property), 18
  - products (magento.models.category.Category property), 54
  - products (magento.models.order.Order property), 56
  - ProductSearch (class in magento.search), 32
  - put() (magento.clients.Client method), 18

## Q

qty\_outstanding (*magento.models.order.OrderItem* property), 57  
 query (*magento.search.SearchQuery* attribute), 22  
 query\_endpoint() (*magento.models.invoice.InvoiceItem* method), 59  
 query\_endpoint() (*magento.models.model.Model* method), 42  
 query\_endpoint() (*magento.models.product.MediaEntry* method), 50

## R

refresh() (*magento.clients.Store* method), 21  
 refresh() (*magento.models.model.Model* method), 43  
 remove\_media\_type() (*magento.models.product.MediaEntry* method), 51  
 request() (*magento.clients.Client* method), 19  
 reset() (*magento.search.SearchQuery* method), 25  
 restrict\_fields() (*magento.search.SearchQuery* method), 23  
 result (*magento.search.OrderItemSearch* property), 28  
 result (*magento.search.SearchQuery* property), 25  
 result\_count (*magento.search.SearchQuery* property), 25  
 result\_type (*magento.search.SearchQuery* property), 25

## S

scope (*magento.clients.Client* attribute), 16  
 search() (*magento.clients.Client* method), 17  
 SearchQuery (class in *magento.search*), 21  
 set\_alt\_text() (*magento.models.product.MediaEntry* method), 52  
 set\_attrs() (*magento.models.model.Model* method), 42  
 set\_media\_types() (*magento.models.product.MediaEntry* method), 51  
 set\_position() (*magento.models.product.MediaEntry* method), 52  
 setup\_logger() (*magento.utils.MagentaLogger* method), 37  
 ship\_to (*magento.models.order.Order* property), 56  
 ship\_to\_address (*magento.models.order.Order* property), 56  
 shipping\_address (*magento.models.order.Order* property), 56

since() (*magento.search.SearchQuery* method), 24  
 skus (*magento.models.category.Category* property), 54  
 special\_price (*magento.models.product.Product* property), 50  
 STATUS\_DISABLED (*magento.models.product.Product* attribute), 46  
 STATUS\_ENABLED (*magento.models.product.Product* attribute), 45  
 stock (*magento.models.product.Product* property), 50  
 stock\_item (*magento.models.product.Product* property), 50  
 stock\_item\_id (*magento.models.product.Product* property), 50  
 Store (class in *magento.clients*), 20  
 store (*magento.clients.Client* attribute), 16  
 store\_view\_product\_attributes (*magento.clients.Store* property), 20  
 stream\_handlers (*magento.utils.MagentaLogger* property), 38  
 subcategories (*magento.models.category.Category* property), 53  
 subcategory\_ids (*magento.models.category.Category* property), 53  
 subcategory\_names (*magento.models.category.Category* property), 54  
 sum\_attrs() (*magento.utils.ItemManager* method), 35

## T

thumbnail (*magento.models.product.Product* property), 49  
 thumbnail\_link (*magento.models.product.Product* property), 49  
 to\_dict() (*magento.clients.Client* method), 20  
 to\_json() (*magento.clients.Client* method), 20  
 to\_pickle() (*magento.clients.Client* method), 19  
 token (*magento.clients.Client* property), 19  
 total\_canceled (*magento.models.order.OrderItem* property), 58  
 total\_qty\_canceled (*magento.models.order.Order* property), 56  
 total\_qty\_invoiced (*magento.models.order.Order* property), 56  
 total\_qty\_outstanding (*magento.models.order.Order* property), 56  
 total\_qty\_refunded (*magento.models.order.Order* property), 56  
 total\_qty\_shipped (*magento.models.order.Order* property), 56

## U

uid (*magento.models.model.APIResponse* property),



45

`uid` (*magento.models.model.Model* property), 42

`unpack_attributes()` (*magento.models.model.Model* static method), 43

`until()` (*magento.search.SearchQuery* method), 25

`update()` (*magento.models.product.MediaEntry* method), 52

`update_attributes()` (*magento.models.product.Product* method), 47

`update_custom_attributes()` (*magento.models.product.Product* method), 48

`update_description()` (*magento.models.product.Product* method), 47

`update_metadata()` (*magento.models.product.Product* method), 47

`update_name()` (*magento.models.product.Product* method), 47

`update_price()` (*magento.models.product.Product* method), 46

`update_special_price()` (*magento.models.product.Product* method), 46

`update_status()` (*magento.models.product.Product* method), 46

`update_stock()` (*magento.models.product.Product* method), 46

`url_for()` (*magento.clients.Client* method), 17

`user_agent` (*magento.clients.Client* attribute), 16

`USER_CREDENTIALS` (*magento.clients.Client* attribute), 16

## V

`validate()` (*magento.clients.Client* method), 19

`validate_result()` (*magento.search.SearchQuery* method), 25

`view_config()` (*magento.clients.Client* method), 20

`views` (*magento.clients.Store* property), 20

`VISIBILITY_BOTH` (*magento.models.product.Product* attribute), 46

`VISIBILITY_CATALOG` (*magento.models.product.Product* attribute), 46

`VISIBILITY_NOT_VISIBLE` (*magento.models.product.Product* attribute), 46

`VISIBILITY_SEARCH` (*magento.models.product.Product* attribute), 46

## W

`warning()` (*magento.utils.MagentoLogger* method), 38