AltaPay Python SDK Documentation

Release 0.1

Coolshop.com

April 19, 2016

Contents

This is an unofficial Python SDK for AltaPay (formerly Pensio), https://altapay.com/. The SDK is maintained by Coolshop.com, https://www.coolshop.com/.

The AltaPay Python SDK attempts to consume the AltaPay API in a clean and Pythonic way, without getting in your way. As a result, you will often find yourself thinking that the API is similar, but has been changed ever so slightly to make it easier for you to use.

As a simple example, once you have your API credentials and a terminal, it is very easy to create a payment in the test environment and get a redirect URL to the payment page.

```
>>> from altapay import API, Payment
>>> api = API(mode='test', account='login', password='password')
>>> payment = Payment(api=api)
>>> payment.create('Test Terminal', 1234567, 13.95, 'EUR')
>>> payment.success
True
>>> print(payment.url)
'https://...'
```

CHAPTER 1

Guide

1.1 Introduction

The Python SDK for AltaPay attempts to make it possible to work with the AltaPay API in the most Pythonic way possible. As such, several things are slightly different, and will not match exactly to what is described in the documentation.

Noticeable differences are;

- The Python SDK is object based, with an object per resource (e.g. an *altapay.Payment* resource). Actions are performed on these objects in order to carry out API calls in the AltaPay API
- All values that you receive from responses are accessbile as attributes on the objects. Note that if an attributes contains a dictionary, the values of this dictionary of course needs to be accessed as it is - a dictionary. If you, for whatever reason, needs access to the underlying dictionary representation, it is available at the attribute `__data__`
- To make the naming scheme feel more Pythonic, names returned by AltaPay is mapped accordingly; e.g. `CamelCase` becomes `camel_case`
- In the AltaPay API, when you send parameters to calls, and these contains nested structures in the form of either arrays or hashed values, the PHP query parameter syntax is used. In the Python SDK, this has been changed to lists and dictionaries for easier use. For a concrete use case, see the payment creation examples

1.1.1 The API Object

All resources that expose AltaPay functionality requires an *altapay.API* object to be passed. The object is what authenticates you to the AltaPay API service, and is also what determines whether or not you should connect to the test service or the production service.

```
from altapay import API

# Create an API object that will connect to the test service
api = API(mode='test', account='account', password='password')

# If you instead want to create an object for production calls, simply
# change the mode
api = API(
    mode='production', account='account', password='password',
    shop_name='test-shop')
```

Optionally, the environment variables `ALTAPAY_ACCOUNT_NAME` and `ALTAPAY_ACCOUNT_PASSWORD` can be used instead of passing the account and password directly to *`altapay.API`*.

The `shop_name` parameter will be used to populate the AltaPay service URL, and is not required when running in test mode.

Making an instance of *`altapay.API`* will automatically attempt to do the login service call in the AltaPay API, which will verify your account and password. This is reccomended behaviour by the AltaPay service, and will only happen when the instance is created. If this is not the desired behaviour, an optional parameter `auto_login` can be set to `False` to disable the automatic login. If you do this, you should call *`altapay.API.login()`* yourself before you do any other calls.

1.2 Create Payment

Creating a new *`altapay.Payment`* resource will result in an URL that you should redirect your customer to.

It is in creating the payment you describe what payment options the customer should have, the total order amount, and other optional parameters. The parameters you use will depend on the terminal you wish to use.

1.2.1 Basic Payment

In the most basic form, a *`altapay.Payment`* object requires a terminal, order ID amount and currency. Creating such payment will look something like this:

```
from altapay import API, Payment

# Create an API object using your credentials
api = API(...)

# Create an empty payment object using the API
payment = Payment(api=api)

# Create a new payment using the AltaPay service.
payment.create('Terminal Name', 'OrderID1234', 13.95, 'EUR')

if payment.success:
    # Assuming a function redirect() which redirects the customer
    redirect(payment.url)
else:
    raise Exception('Payment not successful')
```

This payment is, of course, only very basic, and will only render the standard AltaPay payment form - but it will work. There are, of course, a lot of configuration options.

For a detailed view of these, see the AltaPay API documentation for `API/createPaymentRequest`, bearing in mind the conventions described in *Introduction*.

1.2.2 Complex Payment

Using only the basic payment information will not get you very far. In reality, you are going to want to customize it to your liking, for example using a custom callback form and more detailed order and customer information. Our recommendation would be to always include as much information as possible regarding the order and the customer, since you will be covered for more of the possible terminals this way.

This following example implements a lot of the different possibilities using the AltaPay service.

```
from altapay import API, Payment

api = API(...)
payment = Payment(api=api)

# We can pass all of the optional parameters as keyword arguments to
# the payment creation
params = {
    'config': {
        'callback_form': 'https://your-callback-form/form.html',
    },
    'transaction_info': [
        'ArbitraryInfo1',
        'ArbitraryInfo2'
    ],
    'customer_info': {
        'billing_postal': '9400',
        'billing_address': 'Address 12',
        'billing_firstname': 'First name',
        'billing_lastname': 'Last name',
        'email': 'foo@bar.com'
    },
    'orderLines': [
        {
            'description': 'Description of the order line',
            'itemId': '123456',
            'quantity': 1
        }
    ]
}

payment.create('Terminal name', 'OrderID1234', 13.95, 'EUR', **params)

if payment.success:
    redirect(payment.url)
```

The above example obviously is not complete; there are many more parameters which are described in the AltaPay API documentation. Remember: more data is better, and will result in more terminals working.

1.3 Callback Handling

To make it easy to parse callbacks received from AltaPay, the special *altapay.Callback* class can be used. Note that when you receive callbacks from AltaPay, it comes as an HTTP POST. Within this is a field called `xml`, and this is the response you should use for the *altapay.Callback* class.

Given a callback response in the variable `xml`, this is how a callback instance can be instantiated:

```
from altapay import Callback

xml = ''  # XML response here

callback = Callback.from_xml_callback(xml)

if callback.result == 'Success':
    for transaction in callback.transactions():
        print(transaction)
```

```
else:
    raise Exception('Callback not successful')
```

altapay.Callback.transactions() will contain a list of *altapay.Transaction* objects. Note that even if there is only one transaction in the callback, you will have a list of just one *altapay.Transaction*.

Using *altapay.Callback.transactions()* it is possible to filter based on a authentication type (this will depend on what you chose for the payment type). This can be useful if you have chosen to make a subscription and reservation in the same payment; in this case you can receive a callback with two transactions, and in some cases you might want to process a specific of them ahead of the other one.

This example extends the usecase described above, and filters out the subscription portion of the payment:

```
from altapay import Callback

xml = ''  # XML response here

callback = Callback.from_xml_callback(xml)

if callback.result == 'Success':
    transactions = callback.transactions(auth_type='subscription_payment')
    for transaction in transactions:
        # Will only show transactions of the authentication type
        # subscription_payment
        print(transaction)
else:
    raise Exception('Callback not successful')
```

1.4 Working with Transactions

When the customer completed their payment, you received a callback which contained a transaction ID. From this, you can load an *altapay.Transaction* object, which will be used to perform further backoffice functions, such as capture money on a reservation.

The transaction itself is thought of as a separate resource from the :py:class‘altapay.Payment‘, and can be found using the *altapay.Transaction.find()* call:

```
from altapay import Transaction

transaction = Transaction.find('TransactionID', api=api)
```

On the *altapay.Transaction* object, you will find all of the information described in the AltaPay API, listed under the `API/payments` call. The usual rules for naming applies.

1.4.1 Capturing a Transaction

Once you have an instance of *altapay.Transaction*, it is possible to perform actions on this instance. One of the most common one, is that of capturing it. In the simplest form, it is possible to capture the full amount on the transaction:

```
from altapay import Transaction

transaction = Transaction.find('TransactionID', api=api)
callback = transaction.capture()
```

```
if callback.result == 'Success':
    # Capture was successful
    pass
else:
    raise Exception('Not able to capture')
```

The response is an *altapay.Callback* object and will contain the full response returned by AltaPay.

Of course, this is often not the desired behaviour. You can provider further information to *altapay.Transaction.capture()* as described in the AltaPay API of `API/captureReservation`. For example, you can capture a partial amount in the following way, which also shows how to supply an order line.

```
from altapay import Transaction

transaction = Transaction.find('TransactionID', api=api)
respnse = transaction.capture(
    orderLines=[{
        'description': 'Blue Shirt',
        'itemId': '12345',
        'quantity': 1.0,
        'unitPrice': 19.95
    }],
    amount=19.95)
```

1.4.2 Charging a Subscription

Given a *altapay.Transaction* which is a subscription, it is possible to make a charge (effectively issuing a capture directly on the subscription):

```
from altapay import Transaction

transaction = Transaction.find('TransactionID', api=api)
callback = transaction.charge_subscription(amount=49.00)
```

Charging a subscription will return a Callback object that has a list of transactions; one representing the original *altapay.Transaction* you charged on, and a new one for the actual capture.

As always, see the AltaPay documentation for a list of possible arguments.

1.4.3 Reserving a Subscription

Reserving a transaction works much like *Charging a Subscription*. The only difference is of course in the name: the amount will create a reservation instead of directly charging the amount straight away.

```
from altapay import Transaction

transaction = Transaction.find('TransactionID', api=api)
callback = transaction.reserve_subscription_charge(amount=49.00)
```

Reserving an amount on a will return a Callback object that has a list of transactions; one representing the original *altapay.Transaction* you reserved on, and a new one for the actual reservation.

As always, see the AltaPay documentation for a list of possible arguments.

1.4.4 Releasing a Reservation

In some cases you may choose to not capture your reservation. If so, it's better to release the reservation you have. This is also a good practice in cases where you have a subscription setup on a transaction ID, but you do not have any need for it anymore (for example if your customer cancels their subscription).

Note that there are certain edge cases for calling this method, see the AltaPay documentation for `API/releaseReservation` for full details.

```
from altapay import Transaction

transaction = Transaction.find('TransactionID', api=api)
callback = transaction.release()

if callback.result != 'Success':
    raise Exception('Could not release the reservation')
```

1.4.5 Creating an Invoice Reservation

In some cases, you might want to create an invoice reservation without first creating a Payment object. This could be if you do not want to redirect your customer to the payment provider for validation. In the example of Klarna payments, if you provide the customer's personal identification number together with the normal payment parameters, you can complete the transaction without further confirmation.

Note that this might require approval by the invoice company you are using

As always, see the full list of possible arguments in the AltaPay documentation.

```
from altapay import Callback

parameters = {
    'terminal': 'AltaPay Test Terminal',
    'shop_orderid': 'asdf23',
    'amount': 20.0,
    'currency': 'EUR',
    'customer_info': {
        'billing_postal': '1234',
        'billing_address': 'Test Street',
        'email': 'foo@bar.com'
    },
    'personalIdentifyNumber': '123456-1234'
}

transaction = Callback.create_invoice_reservation(api=api, **parameters)
```

CHAPTER 2

API Documentation

2.1 API

class `altapay.`**`API`**`(`***kwargs*`)`

`download``(`*resource*, *parameters={}*, *headers={}*`)`
Downloads a resource. Acts as a custom HTTP GET. Not that it is considered the callers responsibility to actually flush/close the stream.

`get``(`*resource*, *parameters={}*, *headers={}*`)`
Perform a GET HTTP request on a resource.

Parameters

- **`resource`** – the resource to GET
- **`parameters`** – a dictionary of GET parameters for the resource
- **`headers`** – optional headers. If specified, these will override the default headers.

Returns A response from the AltaPay service as a `dict`.

Raises

UnauthorizedAccessError If the supplied credentials are not valid.

ResponseStatusError If the response code from AltaPay is not a subset of the allowed response codes.

`login``()`
Validates the account name and password against the AltaPay service. This method should always be called before attempting any other calls, and is automatically called once the `altapay.api.API` object is instantiated, unless explictly disabled.

Raises: UnauthorizedAccessError: if the supplied credentials are not valid.

`post``(`*resource*, *parameters={}*, *data={}*, *headers={}*`)`
Perform a POST HTTP requeste on a resource.

Parameters

- **`resource`** – the resource to POST to
- **`parameters`** – a dictionary of GET parameters for the resource
- **`data`** – a dictionary of POST parameters for the resource

- **headers** – optional headers. If specified, these will override the default headers.

Returns A response from the AltaPay service as a `dict`.

Raises

UnauthorizedAccessError If the supplied credentials are not valid.

ResponseStatusError If the response code from AltaPay is not a subset of the allowed response codes.

test_connection()
Tests the connection to the AltaPay service. This operation does not require valid API credentials, and as such can only be used to assert if AltaPay is responding.

Return type `True` if a valid response is returned, otherwise `False`.

2.2 Resource

class altapay.**Resource**(*version=None*, *header=None*, *body=None*, *api=None*)
Base class that maps an AltaPay response into a Python like representation.

classmethod create_from_response(*response*)
Instantiate a new *altapay.Resource* object from a response.

Parameters **response** – a response of type `dict`. The response most conform to the AltaPay response format, which means it must carry four keys called `@version`, `APIResponse`, `Header` and `Body`.

Return type a new version object of type *altapay.Resource*.

2.3 Payment

class altapay.**Payment**(*version=None*, *header=None*, *body=None*, *api=None*)
Bases: `altapay.resource.Resource`

create(*terminal*, *shop_orderid*, *amount*, *currency*, ***kwargs*)
Create a payment request.

Parameters

- **terminal** – name of the targeted AltaPay terminal
- **shop_orderid** – your order ID to be attached to the payment resource
- **amount** – order amount in floating point
- **currency** – currency for the payment resource
- ****kwargs** – used for remaining, optional, payment request parameters, see the AltaPay documentation for a full list. Note that you will need to use lists and dictionaries to map the URL structures from the AltaPay documentation into these kwargs.

Return type `True` if a payment was created, otherwise `False`.

2.4 Callback

class altapay.**Callback**(*version=None*, *header=None*, *body=None*, *api=None*)
Bases: altapay.resource.Resource

classmethod create_invoice_reservation(*terminal*, *shop_orderid*, *amount*, *currency*, *api*, ***kwargs*)
Create a new invoice without first creating a payment.

Return type *altapay.Transaction*

classmethod from_xml_callback(*callback*)
Instantiate a *altapay.Callback* object from an XML response.

Return type *altapay.Callback* instance.

transactions(***kwargs*)
List all of the transactions returned by the callback.

Parameters **auth_type** – the authentication type you wish to filter. Defaults to empty string, which means no filter will be made.

Return type List of *altapay.Transaction* objects.

2.5 Transaction

class altapay.**Transaction**(*version=None*, *header=None*, *body=None*, *api=None*)
Bases: altapay.resource.Resource

capture(***kwargs*)
Capture a reservation on a transaction.

Parameters ****kwargs** – used for optional capture parameters, see the AltaPay documentation for a full list. Note that you will need to use lists and dictionaries to map the URL structures from the AltaPay documentation into these kwargs.

Return type *altapay.Callback* object.

charge_subscription(***kwargs*)
This will charge a subscription using a capture. Can be called many times on a subscription.

If amount is not sent as an optinal parameter, the amount specified in the original setup of the subscription will be used.

Parameters ****kwargs** – used for optional charge subscription parameters, see the AltaPay documentation for a full list. Note that you will need to use lists and dictionaries to map the URL structures from the AltaPay documentation into these kwargs.

Return type *altapay.Callback* object.

classmethod find(*transaction_id*, *api*)
Find exactly one transaction by a transaction ID.

Parameters

- **transaction_id** – ID of the transaction in AltaPay
- **api** – An API object which will be used for AltaPay communication.

Return type *altapay.Transaction*

release()
This will release the reservation on the transaction. This is useful if you for whatever reason do not want to capture the payment.

Refer to the AltaPay documentation for edge cases surround this method.

Return type *altapay.Callback* object.

reserve_subscription_charge(***kwargs*)
This will create a reservation on a subscription. Can be called many times on a subscription.

If amount is not sent as an optinal parameter, the amount specified in the original setup of the subscription will be used.

Parameters ****kwargs** – used for optional reserve subscription parameters, see the AltaPay documentation for a full list. Note that you will need to use lists and dictionaries to map the URL structures from the AltaPay documentation into these kwargs.

Return type *altapay.Callback* object.

2.6 Funding

class altapay.**Funding**(*version=None*, *header=None*, *body=None*, *api=None*)
Bases: altapay.resource.Resource

A funding file that can be either viewed or downloaded.

content()
The funding file content as bytes.

download(*save_to*)
Download the CSV funding file.

Parameters **save_to** – the path to save the funding file to. The filename will match the name of the file at AltaPay, and will have the extension .csv attached.

Return type a string with the complete filepath to the CSV funding file

2.7 FundingList

class altapay.**FundingList**(*api*)
A list of funding files to paginate through.

fundings
altapay.Funding objects on the current page of funding files.

next_page()
Loads the next page of funding files.

Raises: altapay.exceptions.ResourceNotFoundError: if the next page is not available, typically meaning you have scrolled past the available pages.

number_of_pages
Returns the total amount of pages with fundings. Each page can hold 100 fundings.

2.8 Exceptions

class altapay.exceptions.**AltaPayException**
Generic exception class for the AltaPay SDK. All specific exceptions raised from the SDK will inherit from this exceptions.

class altapay.exceptions.**UnauthorizedAccessError**
Raised on unauthorized errors against the AltaPay service.

Corresponds to HTTP status code 401.

class altapay.exceptions.**ResponseStatusError**
The response carried out against the AltaPay service did not respond with the expected response status code.

2.9 Utilities

altapay.utils.**etree_to_dict**(*tree*)
Note: This is an internal API and may be changed without notice.

altapay.utils.**handle_xml_value**(*value*)
The AltaPay XML does not contain a scheme, and as such, guesswork has to be employed in order to produce decent values.

This function parses values of the decoded XML, and ensures both digits and boolean values.

Note: This is an internal API and may be changed without notice.

Parameters **value** – value to be parsed (can be a complex datatype)

Return type depends on the input argument

altapay.utils.**http_build_query**(*payload*)
Build a query string that matches the way PHP does it with `http_build_query`.

In output, this function loosely matches what PHP does in the function `http_build_query`. It handles complex types of both dict and list.

If `collections.OrderedDict` is used, the order of the keys will be preserved in the finalized query string.

Note: This is an internal API and may be changed without notice.

Parameters **payload** – the payload to convert to a query string. This has to be `dict` compatible, but can hold lists as values in the dictionary. Nested dictionaries can be used, and lists can hold dictionaries.

Return type `string` that can be used as a GET parameter for HTTP requests

altapay.utils.**to_pythonic_name**(*name*)
Create a Pythonic version of a string.

Note: This is an internal API and may be changed without notice.

Parameters **name** – string to build a Pythonic version of.

Return type string

altapay.utils.**to_pythonic_dict**(*dictionary*)
Note: This is an internal API and may be changed without notice.

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

a