trustpaylib Documentation

Release 0.0.1

Michal Kuffa

July 24, 2014

Contents

TrustPay payment solution integration helpers.

CHAPTER 1

Install

```
$ pip install trustpaylib
```

CHAPTER 2

Usage

2.1 Link approach

Create environment, payment request and generate signed link.

```
>>> import trustpaylib
>>>
>>> env = trustpaylib.build_environment(
...     aid="1234567890",
...     secret_key="abcd1234",
...     # api_url=trustpaylib.TRUSTCARD_API_URL,
... )
>>> pay_request = trustpaylib.build_pay_request(
...     AMT="123.45",
...     CUR="EUR",
...     REF="1234567890",
... )
>>> trustpay_client = trustpaylib.TrustPay(env)
>>> trustpay_client.build_link(pay_request)
'https://ib.trustpay.eu/mapi/paymentservice.aspx?AID=9876543210&REF=1234567890&AMT=123.45&SIG=DF174E6
```

2.2 Form approach

First merge payment request with environment variables, validate it and sign. *trustpaylib.TrustPay.finalize_request* returns prepared payment request. As form action use *trustpay_client.environment.api_url*.

```
>>> pay_request = trustpay_client.finalize_request(pay_request)
>>> trustpay_client.initial_data(pay_request)
{'AID': '9876543210', 'REF': u'1234567890', 'AMT': u'123.45', 'SIG': 'DF174E635DABBFF7897A82822521DD
>>> trustpay_client.environment.api_url
'https://ib.trustpay.eu/mapi/paymentservice.aspx'
```

2.3 Redirects handling

There's not much you can (or should) do with redirect.

Note: Official TrustPay documentation: DO NOT PERFORM ANY ACTION ON THIS REDIRECT. Data is not

signed and therefore cannot beconsidered as a verified payment result, such as the signed results sent to Notification URL or NotificationEmail.

But at least you can check the result code.

```
>>> # data received as get params to redirect
... redirect_data = {"REF": "1234567890", "RES": 3, "PID": "1212321"}
>>> redirect = trustpaylib.build_redirect(**redirect_data)
>>>
>>> trustpay_client.get_result_desc_from_redirect(redirect).short
'Authorized'
>>> trustpay_client.get_result_desc_from_redirect(redirect).long
'Payment was successfully authorized. Another notification (with result code 0 - success) will be ser
```

2.4 Notifications handling

For received notification, first check signature.

```
>>> notification_data = {
...     "AID": "1234567890",
...     "TYP": "CRDT",
...     "AMT": "123.45",
...     "CUR": "EUR",
...     "REF": "9876543210",
...     "RES": "0",
...     "TID": "11111",
...     "OID": "1122334455",
...     "TSS": "Y",
...     "SIG": (
...         "97C92D7A0C0AD99CE5DE55C3597D5ADA"
...         "0D423991E2D01938BC0F684244814A37"
...     ),
... }
>>> notification = trustpaylib.build_notification(**notification_data)
>>> assert trustpay_client.check_notification_signature(notification)
```

Then check result code.

```
>>> trustpay_client.get_result_desc_from_notification(notification).short
'Success'
>>> trustpay_client.get_result_desc_from_notification(notification).long
'Payment was successfully processed.'
```

CHAPTER 3

Api documentation

3.1 trustpaylib

TrustPay payment solution constants and utils.

class trustpaylib.**TrustPay**(*environment*)

CURRENCIES
Supportted currencies (trustpaylib.CURRENCIES)

LANGUAGES
Supportted languages (trustpaylib.LANGUAGES)

COUNTRIES
Supportted countries (trustpaylib.COUNTRIES)

RESULT_CODES
Known result codes of redirects and notifications (trustpaylib.RESULT_CODES)

RESULT_CODES_DESC
Short and long description for result codes of redirects and notifications (trustpaylib.RESULT_CODES_DESC)

build_link(*pay_request*, *sign=True*, *validate=True*, *merge_env=True*)
Finalizes raw payment request and generates redirect link.

Args: pay_request (trustpaylib.TrustPayRequest):

sign (bool): If *False*, don't sign pay request.

validate (bool): If *False*, don't validate pay request.

merge_env (bool): If *False*, don't merge pay request with env.

Returns: string: Redirect link.

check_notification_signature(*notification*)
Check if notification is signed with environment's secret key.

Args: notification (trustpaylib.TrustPayNotification)

Returns: bool

classmethod create_signature_msg(*pay_request*)
Concatenate set of payment request attributes and creates message to be hashed.

Args: pay_request (trustpaylib.TrustPayRequest):

Returns: string: Signature message.

finalize_request (*pay_request*, *sign=True*, *validate=True*, *merge_env=True*)
Raw payment request is merged with environment, signed and validated.

Args: pay_request (trustpaylib.TrustPayRequest):

sign (bool): If *False*, don't sign pay request.

validate (bool): If *False*, don't validate pay request.

merge_env (bool): If *False*, don't merge pay request with env.

Returns: New trustpaylib.TrustPayRequest prepared for building link or creating form.

classmethod get_result_desc (*rc*)
Returns description of result code.

Args:

rc (int|string): Result code from redirect or notification.

Returns: Named tuple with *short* and *long* attributes for short, long description. (trustpaylib.RESULT_CODES_DESC)

```
>>> TrustPay.get_result_desc(1001).short
'Invalid request'
>>> TrustPay.get_result_desc(1001).long
'Data sent is not properly formatted.'
```

merge_env_with_request (*pay_request*)
Merge specific attributes of environment with payment request.

Args:

pay_request (trustpaylib.TrustPayRequest): Payment request to merge.

Returns: New trustpaylib.TrustPayRequest instance with attributes merged with those in environment if not already set on *pay_request*.

pay_request_signature (*pay_request*)
Use environments secret key to generate hash to sign pay request.

Args:

pay_request (trustpaylib.TrustPayRequest): Payment request already prepared for signing.

Returns: Hash.

sign_request (*pay_request*)
Sign payment request.

Args:

pay_request (trustpaylib.TrustPayRequest): Payment request already prepared for signing.

Returns: New trustpaylib.TrustPayRequest instance with *SIG* attribute set to generated signature.

classmethod validate_request (*pay_request*)
Validate payment request.

Check if all attributes for signed/non-signed payment request are present. Check if amount has at max two decimal places.

On validation errors, raises `ValueError`.

Args: pay_request (`trustpaylib.TrustPayRequest`):

Returns: Given *pay_request*.

Raises: ValueError

class `trustpaylib.`**`TrustPayRequest`**
TrustPayRequest(AID, AMT, CUR, REF, URL, RURL, CURL, EURL, NURL, SIG, LNG, CNT, DSC, EMA)

class `trustpaylib.`**`TrustPayRedirect`**
TrustPayRedirect(REF, RES, PID)

class `trustpaylib.`**`TrustPayNotification`**
TrustPayNotification(AID, TYP, AMT, CUR, REF, RES, TID, OID, TSS, SIG)

3.2 Constants

`trustpaylib.`**`TEST_API_URL`** **= 'https://test.trustpay.eu/mapi/paymentservice.aspx'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`trustpaylib.`**`API_URL`** **= 'https://ib.trustpay.eu/mapi/paymentservice.aspx'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`trustpaylib.`**`TRUSTCARD_API_URL`** **= 'https://ib.trustpay.eu/mapi/cardpayments.aspx'**
str(object='') -> string

Return a nice string representation of the object. If the argument is a string, the return value is the same object.

`trustpaylib.`**`CURRENCIES`** **= TrustPayCurrencies(CZK='CZK', EUR='EUR', GBP='GBP', HUF='HUF', PLN='PLN', US**
TrustPayCurrencies(CZK, EUR, GBP, HUF, PLN, USD, RON, BGN, HRK, LTL, TRY)

`trustpaylib.`**`LANGUAGES`** **= TrustPayLanguages(bg='bg', bs='bs', cs='cs', de='de', en='en', es='es', et='et', hr='hr', hu='hu**
TrustPayLanguages(bg, bs, cs, de, en, es, et, hr, hu, it, lt, lv, pl, ro, ru, sk, sl, sr, uk)

`trustpaylib.`**`COUNTRIES`** **= TrustPayCountries(CZ='CZ', HU='HU', PL='PL', SK='SK', EE='EE', BG='BG', RO='RO', H**
TrustPayCountries(CZ, HU, PL, SK, EE, BG, RO, HR, LV, LT, SI, TR, FI)

`trustpaylib.`**`COUNTRIES_VERBOSE`** **= TrustPayCountriesVerbose(CZ='Czech Republic', HU='Hungary', PL='Poland', SK**
TrustPayCountriesVerbose(CZ, HU, PL, SK, EE, BG, RO, HR, LV, LT, SI, TR, FI)

`trustpaylib.`**`RESULT_CODES`** **= TrustPayResultCodes(SUCCESS='0', PENDING='1', ANNOUNCED='2', AUTHORIZED**
TrustPayResultCodes(SUCCESS, PENDING, ANNOUNCED, AUTHORIZED, PROCESSING, AUTHORIZED_ONLY, INVALID_REQUEST, UNKNOWN_ACCOUNT, MERCHANT_ACCOUNT_DISABLED, INVALID_SIGN, USER_CANCEL, INVALID_AUTHENTICATION, DISPOSABLE_BALANCE, SERVICE_NOT_ALLOWED, PAYSAFECARD_TIMEOUT, GENERAL_ERROR, UNSUPPORTED_CURRENCY_CONVERSION)

`trustpaylib.`**`RESULT_CODES_DESC`** **= {'1100': TrustPayResultCodesDesc(short='General Error', long='Internal error has**
dict() -> new empty dictionary dict(mapping) -> new dictionary initialized from a mapping object's

(key, value) pairs

dict(iterable) -> new dictionary initialized as if via: d = {} for k, v in iterable:

d[k] = v

dict(kwargs) -> new dictionary initialized with the name=value pairs** in the keyword argument list. For example: dict(one=1, two=2)

CHAPTER 4

Indices and tables

- *genindex*
- *modindex*
- *search*

t