itunes-iap Documentation

Release 2.6.0

Jeong YunWon

Jun 05, 2022

Contents:

itunes-iap is an easy way to verify iTunes In-App Purchase receipts. It also is a powerful and flexible interface for them too.

CHAPTER 1

Quickstart

Create request to create a request to itunes verify api.

```
>>> import itunesiap
>>> try:
>>>     response = itunesiap.verify(raw_data)  # base64-encoded data
>>> except itunesiap.exc.InvalidReceipt as e:
>>>     print('invalid receipt')
>>> print response.receipt.last_in_app.product_id
>>> # other values are also available as properties!
```

Practically useful attributes are: *product_id*, *original_transaction_id*, *quantity* and *unique_identifier*. See the full document in *itunesiap.receipt.InApp*.

For asyncio, replace *itunesiap.verify()* funciton to *itunesiap.aioverify()*. That's all.

```
>>> response = itunesiap.aioverify(raw_data)
```

1.1 itunesiap.verify()

Note that most of the use cases are covered by the *itunesiap.verify()* function.

itunesiap.**verify**(*receipt_data*, *password=None*, *exclude_old_transactions=False*, ***kwargs*)

Shortcut API for *itunesiap.request.Request*.

See also:

- Receipt_Validation_Programming_Guide.

Parameters

- **receipt_data** (*str*) – *itunesiap.request.Request* argument. An iTunes receipt data as Base64 encoded string.

- **password** (*str*) – *itunesiap.request.Request* argument. Optional. Only used for receipts that contain auto-renewable subscriptions. Your app's shared secret (a hexadecimal string).
- **exclude_old_transactions** (*bool*) – *itunesiap.request.Request* argument. Optional. Only used for iOS7 style app receipts that contain auto-renewable or non-renewing subscriptions. If value is true, response includes only the latest renewal transaction for any subscriptions.
- **env** (itunesiap.environment.Environment) – Set base environment value. See *itunesiap.environment* for detail.
- **timeout** (*float*) – *itunesiap.request.Request.verify()* argument. Keyword-only optional. The value is connection timeout of the verifying request. The default value is 30.0 when no *env* is given.
- **use_production** (*bool*) – *itunesiap.request.Request.verify()* argument. Keyword-only optional. The value is weather verifying in production server or not. The default value is bool True when no *env* is given.
- **use_sandbox** (*bool*) – *itunesiap.request.Request.verify()* argument. Keyword-only optional. The value is weather verifying in sandbox server or not. The default value is bool False when no *env* is given.
- **verify_ssl** (*bool*) – *itunesiap.request.Request.verify()* argument. Keyword-only optional. The value is weather enabling SSL verification or not. WARNING: DO NOT TURN IT OFF WITHOUT A PROPER REASON. IF YOU DON'T UNDERSTAND WHAT IT MEANS, NEVER SET IT YOURSELF.
- **proxy_url** (*str*) – Keyword-only optional. A proxy url to access the iTunes validation url.

Returns *itunesiap.receipt.Receipt* object if succeed.

Raises Otherwise raise a request exception in *itunesiap.exceptions*.

itunesiap.**aioverify**(*receipt_data*, *password=None*, *exclude_old_transactions=False*, ***kwargs*)

Shortcut API for *itunesiap.request.Request*.

Note that python3.4 support is only available at itunesiap==2.5.1

For params and returns, see *itunesiap.verify()*.

1.2 Apple in-review mode

In review mode, your actual users who use older versions want to verify in production server but the reviewers in Apple office want to verify in sandbox server.

Note: The default env is *production* mode which doesn't allow any sandbox verifications.

You can change the verifying mode by specifying *env*.

```
>>> # review mode
>>> itunesiap.verify(raw_data, env=itunesiap.env.review)
>>> # sandbox mode
>>> itunesiap.verify(raw_data, env=itunesiap.env.sandbox)
```

Also directly passing arguments are accepted:

```
>>> # review mode
>>> itunesiap.verify(raw_data, use_production=True, use_sandbox=True)
```

1.3 Password for shared secret

When you have shared secret for your app, the verifying process requires a shared secret password.

About the shared secret, See: In-App_Purchase_Configuration_Guide.

```
>>> try:
>>>     # Add password as a parameter
>>>     response = itunesiap.verify(raw_data, password=password)
>>> except itunesiap.exc.InvalidReceipt as e:
>>>     print('invalid receipt')
>>> in_app = response.receipt.last_in_app  # Get the latest receipt returned by Apple
```

CHAPTER 2

Request

itunesiap.request

class itunesiap.request.**Request**(*receipt_data*, *password=None*, *exclude_old_transactions=False*, ***kwargs*)

Validation request with raw receipt.

Use *verify* method to try verification and get Receipt or exception. For detail, see also the Apple document: https://developer.apple.com/library/content/releasenotes/General/ValidateAppStoreReceipt/Chapters/ValidateRemotely.html.

Parameters

- **receipt_data** (*str*) – An iTunes receipt data as Base64 encoded string.
- **password** (*str*) – Only used for receipts that contain auto-renewable subscriptions. Your app's shared secret (a hexadecimal string).
- **exclude_old_transactions** (*bool*) – Only used for iOS7 style app receipts that contain auto-renewable or non-renewing subscriptions. If value is true, response includes only the latest renewal transaction for any subscriptions.
- **proxy_url** – A proxy url to access the iTunes validation url. (It is an attribute of *verify()* but misplaced here)

aioverify(***options*)

Try to verify the given receipt with current environment.

Note that python3.4 support is only available at itunesiap==2.5.1

See also:

- Receipt_Validation_Programming_Guide.

Parameters

- **env** (itunesiap.environment.Environment) – Override the environment.
- **timeout** (*float*) – The value is connection timeout of the verifying request. The default value is 30.0 when no *env* is given.

- **use_production** (*bool*) – The value is weather verifying in production server or not. The default value is `bool` True when no *env* is given.
- **use_sandbox** (*bool*) – The value is weather verifying in sandbox server or not. The default value is `bool` False when no *env* is given.
- **verify_ssl** (*bool*) – The value will be ignored.

Returns *itunesiap.receipt.Receipt* object if succeed.

Raises Otherwise raise a request exception.

request_content
Instantly built request body for iTunes.

verify(***options*)
Try verification with current environment.

See also:

- Receipt_Validation_Programming_Guide.

Parameters

- **env** (`itunesiap.environment.Environment`) – Override the environment.
- **timeout** (*float*) – The value is connection timeout of the verifying request. The default value is 30.0 when no *env* is given.
- **use_production** (*bool*) – The value is weather verifying in production server or not. The default value is `bool` True when no *env* is given.
- **use_sandbox** (*bool*) – The value is weather verifying in sandbox server or not. The default value is `bool` False when no *env* is given.
- **verify_ssl** (*bool*) – The value is weather enabling SSL verification or not. WARNING: DO NOT TURN IT OFF WITHOUT A PROPER REASON. IF YOU DON'T UNDERSTAND WHAT IT MEANS, NEVER SET IT YOURSELF.

Returns *itunesiap.receipt.Receipt* object if succeed.

Raises Otherwise raise a request exception.

verify_from(*url*, *timeout=None*, *verify_ssl=True*)
The actual implemention of verification request.

verify() calls this method to try to verifying for each servers.

Parameters

- **url** (*str*) – iTunes verification API URL.
- **timeout** (*float*) – The value is connection timeout of the verifying request. The default value is 30.0 when no *env* is given.
- **verify_ssl** (*bool*) – SSL verification.

Returns *itunesiap.receipt.Receipt* object if succeed.

Raises Otherwise raise a request exception.

itunesiap.exceptions

exception itunesiap.exceptions.**ItunesServerNotAvailable**(**args*, ***kwargs*)
iTunes server is not available. No response.

exception itunesiap.exceptions.**ItunesServerNotReachable**(**args*, ***kwargs*)
iTunes server is not reachable - including connection timeout.

exception itunesiap.exceptions.**InvalidReceipt**(*response_data*)
A receipt was given by iTunes server but it has error.

CHAPTER 3

Receipt

itunesiap.receipt

A successful response returns a JSON object including receipts. To manipulate them in convinient way, *itunes-iap* wrapped it with *ObjectMapper*.

class itunesiap.receipt.**ObjectMapper**(*data*)

A pretty interface for decoded receipt object.

__DOCUMENTED_FIELDS__ and *__UNDOCUMENTED_FIELDS__* are managed lists of field names. They are regarded as safe values and guaranteed to be converted in python representation when needed. When a field exists in *__OPAQUE_FIELDS__*, its result will be redirected. The common type is `str`. When a field exists in *__FIELD_ADAPTERS__*, it will be converted to corresponding python data representation.

To access to the converted value, use a dictionary key as an attribute name. For example, the key *receipt* is accessible by:

```
>>> mapper.receipt  # return converted python object Receipt
>>> # == Receipt(mapper['receipt'])
```

To access to the raw JSON value, use a dictionary key as an attribute name but with the prefix _. For example, the key *receipt* is accessible by:

```
>>> mapper._receipt  # return converted python object Receipt
>>> # == mapper['receipt']
```

Parameters **data** (*dict*) – A JSON object.

Returns *ObjectMapper*

class itunesiap.receipt.**Response**(*data*)

The root response.

About the value of status:

- See https://developer.apple.com/library/ios/releasenotes/General/ValidateAppStoreReceipt/Chapters/ValidateRemotely.html#//apple_ref/doc/uid/TP40010573-CH104-SW1

```
__DOCUMENTED_FIELDS__ = frozenset({'latest_receipt', 'receipt', 'status', 'pending_ren
__FIELD_ADAPTERS__ = {'pending_renewal_info':  <bound method ObjectMapper.from_list of
__OPAQUE_FIELDS__ = frozenset({'latest_receipt'})
__UNDOCUMENTED_FIELDS__ = frozenset()
```

class itunesiap.receipt.**Receipt**(*data*)

The actual receipt.

The receipt may hold only one purchase directly in receipt object or may hold multiple purchases in *in_app* key. This object encapsulate it to list of *InApp* object in *in_app* property.

See https://developer.apple.com/library/content/releasenotes/General/ValidateAppStoreReceipt/Chapters/ReceiptFields.html

```
__DOCUMENTED_FIELDS__ = frozenset({'cancellation_date', 'in_app', 'auto_renew_product_
__FIELD_ADAPTERS__ = {'auto_renew_status':  <class 'int'>, 'cancellation_date':  <func
__OPAQUE_FIELDS__ = frozenset({'product_id', 'app_item_id', 'transaction_id', 'expires_
__UNDOCUMENTED_FIELDS__ = frozenset({'expiration_date_ms', 'expires_date_formatted', '
```

class itunesiap.receipt.**InApp**(*data*)

```
__DOCUMENTED_FIELDS__ = frozenset({'cancellation_date', 'product_id', 'cancellation_re
__FIELD_ADAPTERS__ = {'cancellation_date':  <function _rfc3339_to_datetime>, 'cancella
__OPAQUE_FIELDS__ = frozenset({'product_id', 'transaction_id', 'expires_date_formatted
__UNDOCUMENTED_FIELDS__ = frozenset({'cancellation_date_ms', 'expires_date_formatted',
```

CHAPTER 4

Environment

itunesiap.environment

Environment is designed to pass pre-defined policies in easy way. The major use cases are provided as pre-defined constants.

4.1 How to use environments

The default policy is *default* and it is the same to *production*. When your development lifecycle is proceed, you want to change it to *sandbox* or *review*.

The recommended way to use environments is passing the value to *itunesiap.verify()* function as keyword argument *env*.

```
>>> itunesiap.verify(receipt, env=itunesiap.env.production)
>>> itunesiap.verify(receipt, env=itunesiap.env.sandbox)
>>> itunesiap.verify(receipt, env=itunesiap.env.review)
```

4.2 Review mode

This is useful when your server is being used both for real users and Apple reviewers. Using review mode for a real service is possible, but be aware of: it is not 100% safe. Your testers can getting advantage of free IAP in production version. A rough solution what I suggest is:

```
>>> if client_version == review_version:
>>>     env = itunesiap.env.review
>>> else:
>>>     env = itunesiap.env.production
>>>
>>> itunesiap.verify(receipt, env=env)
```

4.3 Environment

class itunesiap.environment.**Environment**(***kwargs*)
Environment provides option preset for *Request*. *default* is default.

By passing an environment object to *itunesiap.verify()* or *itunesiap.request.Request.verify()* function, it replaces verifying policies.

clone(***kwargs*)
Clone the environment with additional parameter override

extract()
Extract options from *self* and merge to *kwargs* then return a new dictionary with the values.

override(***kwargs*)
Override options in kwargs to given object *self*.

itunesiap.environment.**default = <Environment use_production=True use_sandbox=False timeout=**
Use only production server with 30 seconds of timeout.

itunesiap.environment.**production = <Environment use_production=True use_sandbox=False timeo**
Use only production server with 30 seconds of timeout.

itunesiap.environment.**sandbox = <Environment use_production=False use_sandbox=True timeout=**
Use only sandbox server with 30 seconds of timeout.

itunesiap.environment.**review = <Environment use_production=True use_sandbox=True timeout=30**
Environment provides option preset for *Request*. *default* is default.

By passing an environment object to *itunesiap.verify()* or *itunesiap.request.Request.verify()* function, it replaces verifying policies.

itunesiap.environment.**unsafe = <Environment use_production=True use_sandbox=True timeout=No**
Environment provides option preset for *Request*. *default* is default.

By passing an environment object to *itunesiap.verify()* or *itunesiap.request.Request.verify()* function, it replaces verifying policies.

CHAPTER 5

itunes-iap v2

Python 2 & 3 compatible! Even with `asyncio` support!

- Source code: https://github.com/youknowone/itunes-iap
- Documentation: http://itunes-iap.readthedocs.io/
- Distribution: https://pypi.python.org/pypi/itunes-iap/

5.1 Quickstart

Create request to create a request to itunes verifying api.

```
>>> import itunesiap
>>> try:
>>>     response = itunesiap.verify(raw_data)  # base64-encoded data
>>> except itunesiap.exc.InvalidReceipt as e:
>>>     print('invalid receipt')
>>> print response.receipt.last_in_app.product_id  # other values are also available␣
↪as property!
```

The common attributes are: *product_id*, *original_transaction_id* and *quantity*.

See the full document in:

- *itunesiap.verify()*: The verifying function.
- *itunesiap.receipt.Receipt*: The receipt object.

5.2 asyncio

```
>>> import itunesiap
>>> response = await itunesiap.aioverify(raw_data)  # verify -> aioverify
```

The other parts are the same.

See the full document in:

- *itunesiap.aioverify()*: The verifying function.

5.3 Installation

PyPI is the recommended way.

```
$ pip install itunes-iap
```

To browse versions and tarballs, visit: https://pypi.python.org/pypi/itunes-iap/

5.4 Apple in-review mode

In review mode, your actual users who use older versions want to verify in production server but the reviewers in Apple office want to verify in sandbox server.

Note: The default env is *production* mode which doesn't allow any sandbox verifications.

You can change the verifying mode by specifying *env*.

```
>>> # review mode
>>> itunesiap.verify(raw_data, env=itunesiap.env.review)
```

5.5 Note for v1 users

There was breaking changes between v1 and v2 APIs.

- Specify version *0.6.6* for latest v1 API when you don't need new APIs.
- Or use *import itunesiap.legacy as itunesiap* instead of *import itunesiap*. (*from itunesiap import xxx* to *from itunesiap.legacy import xxx*)

5.6 Contributors

See https://github.com/youknowone/itunes-iap/graphs/contributors

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

i

Index

Symbols

A

C

D

E

I

O

P

R

S

U

V