Vingd API for Python Documentation

Release 0.1

Radomir Stevanovic, Vingd Inc.

May 15, 2014

Contents

Contents:

CHAPTER 1

Vingd

Vingd enables users to pay with money or with time. Money goes directly to publishers and time is monetized indirectly through interaction with brands, content creation, loyalty, bringing new users, etc. As a result Vingd dramatically increases monetization while keeping reach. Vingd's secret sauce are mathematical models that are adapting to each user in order to extract as much value as possible from their time.

We use vingds (think of it as "digital currency", points, or credits) to express the value ("price") of intangible goods (such as TV streams or newspaper articles), to reward users for their activity (time), or to authorize ("charge") them access to digital goods.

1.1 Vingd API for Python

Vingd API enables you to register Vingd objects you're selling, create Vingd purchase orders, verify and commit Vingd purchases. You can also reward users, either directly (in backend), or indirectly via Vingd vouchers. Detailed docs and demos are available.

1.2 Installation

To install the last stable release of Vingd API:

```
$ pip install vingd
```

Or, to install from GitHub source:

```
$ git clone https://github.com/vingd/vingd-api-python
$ cd vingd-api-python
$ make env && source env/bin/activate   (skip if already in virtualenv)
$ python setup.py install
```

1.3 Examples

Client initialization and account balance fetching:

```
from vingd import Vingd

VINGD_USERNAME = 'test@vingd.com'
VINGD_PASSWORD = '123'

# Initialize Vingd client.
v = Vingd(username=VINGD_USERNAME, password=VINGD_PASSWORD,
          endpoint=Vingd.URL_ENDPOINT_SANDBOX, frontend=Vingd.URL_FRONTEND_SANDBOX)
```

```
# Fetch user balance.
balance = v.get_user_balance()
```

1.3.1 Sell content

Wrap up Vingd order and redirect user to confirm his purchase at Vingd frontend:

```
# Selling details.
OBJECT_NAME = "My test object"
OBJECT_URL = "http://localhost:666/"
ORDER_PRICE = 200 # VINGD 2.00

# Register Vingd object (once per selling item).
oid = v.create_object(OBJECT_NAME, OBJECT_URL)

# Prepare Vingd order.
order = v.create_order(oid, ORDER_PRICE)

# Order ready, redirect user to confirm his purchase at Vingd frontend.
redirect_url = order['urls']['redirect']
```

As user confirms his purchase on Vingd frontend he is redirected back to object URL expanded with purchase verification parameters.

```
# User confirmed purchase on Vingd frontend and came back to http://localhost:666/?oid=<oid>&tid=
purchase = v.verify_purchase(oid, tid)

# Purchase successfully verified, serve purchased content to user.
# ... content serving ...

# Content is successfully served, commit Vingd transaction.
commit = v.commit_purchase(purchase['purchaseid'], purchase['transferid'])
```

1.3.2 Reward user with vingd

Reward user with vingd:

```
# Vingd hashed user id, as obtained in purchase procedure (previous example).
REWARD_HUID = purchase['huid']
REWARD_AMOUNT = 75 # VINGD 0.75
REWARD_DESCRIPTION = "Testing direct rewarding"

# Reward user.
reward = v.reward_user(REWARD_HUID, REWARD_AMOUNT, REWARD_DESCRIPTION)
```

1.3.3 Reward user with voucher

Redirect user to redeem his reward on vingd frontend:

```
VOUCHER_AMOUNT = 100; # 1.00 vingd
VOUCHER_EXPIRES = {'days': 14}

# Create vingd voucher.
voucher = v.create_voucher(amount=VOUCHER_AMOUNT, expires=VOUCHER_EXPIRES)

# Redirect user to use voucher on vingd frontend:
redirect_url = voucher['urls']['redirect']
```

For more examples, see example/test.py in source.

1.4 Documentation

Automatically generated documentation for latest stable version is available on: https://vingd-api-for-python.readthedocs.org/en/latest/.

1.5 Copyright and License

Vingd API is Copyright (c) 2012 Vingd, Inc and licensed under the MIT license. See the LICENSE file for full details.

Vingd API

2.1 Overview of key functions

2.1.1 Account related functions

get_user_profile()	FETCHES profile dictionary of the authenticated user.
get_user_balance	

2.1.2 Vingd authorization ("selling access")

create_object(name, url)	CREATES a single object in Vingd Object registry.
create_order(oid, price[, context, expires])	CREATES a single order for object oid, with price set to price and validity
verify_purchase(oid, tid)	VERIFIES token tid and returns token data associated with tid and bound to
commit_purchase(purchaseid, transferid)	DECLARES a purchase defined with purchaseid (bound to vingd transfer r

2.1.3 Vingd rewarding

create_voucher(amount[, expires, message, gid])	CREATES a new preallocated voucher with amount vingd cents reserv
revoke_vouchers([vid_encoded, uid_from, ...])	REVOKES/INVALIDATES a filtered list of vouchers.
reward_user(huid_to, amount[, description])	PERFORMS a single reward.

2.2 Interface

class `vingd.`**`Vingd`**(*key=None*, *secret=None*, *endpoint=None*, *frontend=None*, *username=None*, *password=None*)

`authorized_create_user`(*identities=None*, *primary=None*, *permissions=None*)

Creates Vingd user (profile & account), links it with the provided identities (to be verified later), and sets the delegate-user permissions (creator being the delegate). Returns Vingd user's *huid* (hashed user id).

Example:

```
vingd.authorized_create_user(
    identities={"facebook": "12312312", "mail": "user@example.com"},
    primary="facebook",
    permissions=["get.account.balance", "purchase.object"]
)
```

If *identities* and *primary* are unspecified, a “zombie” (“headless”) account is created (i.e. account with no identities associated, user-unreachable).

Return type `dict`

Returns `{'huid': <huid>}`

Raises GeneralException

Resource `id/objects/<oid>/purchases`

Access authorized users with ACL flag `user.create`

`authorized_get_account_balance`(*huid*)
FETCHES the account balance for the user defined with *huid*.

Return type `bigint`

Returns `<amount_in_cents>`

Raises GeneralException

Resource `fort/accounts/<huid>`

Access authorized users; delegate permission required for the requester to read user’s balance: `get.account.balance`

`authorized_purchase_object`(*oid*, *price*, *huid*)
Does delegated (pre-authorized) purchase of *oid* in the name of *huid*, at price *price* (vingd transferred from *huid* to consumer’s acc).

Raises GeneralException

Resource `objects/<oid>/purchases`

Access authorized users with ACL flag `purchase.object.authorize` + delegate permission required for the requester to charge the user: `purchase.object`

`commit_purchase`(*purchaseid*, *transferid*)
DECLARES a purchase defined with `purchaseid` (bound to vingd transfer referenced by `transferid`) as finished, with user being granted the access to the service or goods.

If seller fails to commit the purchase, the user (buyer) shall be refunded full amount paid (reserved).

Parameters

- **purchaseid** (`bigint`) – Purchase ID, as returned in purchase description, upon token/purchase verification.
- **transferid** (`bigint`) – Transfer ID, as returned in purchase description, upon token/purchase verification.

Return type `dict`

Returns `{'ok': <boolean>}`.

Raises

- **InvalidData** – invalid format of input parameters
- **NotFound** – non-existing order/purchase/transfer
- **GeneralException** – depends on details of error
- **InternalError** – Vingd internal error (network, server, app)

See *verify_purchase*.

Resource `purchases/<purchaseid>`

Access authorized users (ACL flag: `type.business`)

`create_object`(*name*, *url*)
CREATES a single object in Vingd Object registry.

Parameters

- **name** (`string`) – Object's name.
- **url** (`string`) – Callback URL (object's resource location - on your server).

Return type *bigint*

Returns Object ID for the newly created object.

:raises GeneralException:s

Resource `registry/objects/`

Access authorized users

`create_order` (*oid*, *price*, *context=None*, *expires=None*)

CREATES a single order for object `oid`, with price set to `price` and validity until `expires`.

Parameters

- **oid** (`bigint`) – Object ID.
- **price** (`bigint`) – Vingd amount (in cents) the user/buyer shall be charged upon successful purchase.
- **context** (`string`) – Purchase (order-related) context. Retrieved upon purchase verification.
- **expires** (`datetime/dict`) – Order expiry timestamp, absolute (`datetime`) or relative (`dict`). Valid keys for relative expiry timestamp dictionary are same as keyword arguments for *datetime.timedelta* (`days`, `seconds`, `minutes`, `hours`, `weeks`). Default: *Vingd.EXP_ORDER*.

Return type `dict`

Returns

Order dictionary:

```
order = {
    'id': <order_id>,
    'expires': <order_expiry>,
    'context': <purchase_context>,
    'object': {
        'id': <oid>,
        'price': <amount_in_cents>
    },
    'urls': {
        'redirect': <url_for_failsafe_redirect_purchase_mode>,
        'popup': <url_for_popup_purchase_mode>
    }
}
```

Raises GeneralException

Resource `objects/<oid>/orders/`

Access authorized users

`create_voucher` (*amount*, *expires=None*, *message='‘*, *gid=None*)

CREATES a new preallocated voucher with `amount` vingd cents reserved until `expires`.

Parameters

- **amount** (`bigint`) – Voucher amount in vingd cents.
- **expires** (`datetime/dict`) – Voucher expiry timestamp, absolute (`datetime`) or relative (`dict`). Valid keys for relative expiry timestamp dictionary are same as

keyword arguments for *datetime.timedelta* (`days`, `seconds`, `minutes`, `hours`, `weeks`). Default: *Vingd.EXP_VOUCHER*.

- **message** (`string`) – Short message displayed to user when she redeems the voucher on Vingd frontend.
- **gid** (`alphanum(32)`) – Voucher group id. An user can redeem only one voucher per group.

Return type `dict`

Returns

Created voucher description:

```
voucher = {
    'vid': <voucher_integer_id>,
    'vid_encoded': <voucher_string_id>,
    'amount_allocated': <int_cents | None if not allocated>,
    'amount_vouched': <int_cents>,
    'id_fort_transfer': <id_of_allocating_transfer |
                         None if not allocated>,
    'fee': <int_cents>,
    'uid_from': <source_account_uid>,
    'uid_proxy': <broker_id>,
    'uid_to': <destination_account_id | None if not given>,
    'gid': <voucher_group_id | None if undefined>,
    'ts_valid_until': <iso8601_timestamp_absolute>,
    'description': <string | None>,
    'message': <string | None>
}
```

combined with voucher redeem urls on Vingd frontend.

Raises GeneralException

Resource `vouchers/`

Access authorized users (ACL flag: `voucher.add`)

`get_account_balance`()

FETCHES the account balance for the authenticated user.

Return type `bigint`

Returns `<amount_in_cents>`

Raises GeneralException

Resource `fort/accounts/`

Access authorized users; authenticated user's account data will be fetched

`get_object`(*oid*)

FETCHES a single object, referenced by its `oid`.

Parameters **oid** (`bigint`) – Object ID

Return type `dict`

Returns The object description dictionary.

Raises GeneralException

Note *get_objects* can be used instead, but then specifying any other (conflicting) constraint (except `oid`) yields a non-existing resource exception (*NotFound*).

Resource `registry/objects/<oid>`

Access authorized users (only objects owned by the authenticated user are returned)

`get_objects`(*oid=None*, *since=None*, *until=None*, *last=None*, *first=None*)
FETCHES a filtered collection of objects created by the authenticated user.

Parameters

- **oid** (`bigint`) – Object ID
- **since** (`datetime`/`dict`) – Object has to be newer than this timestamp (absolute `datetime`, or relative `dict`). Valid keys for relative *since* timestamp dictionary are same as keyword arguments for *datetime.timedelta* (`days`, `seconds`, `minutes`, `hours`, `weeks`).
- **until** (`datetime`/`dict`) – Object has to be older than this timestamp (for format, see the *since* parameter above).
- **last** (`bigint`) – The number of newest objects (that satisfy all other criteria) to return.
- **first** (`bigint`) – The number of oldest objects (that satisfy all other criteria) to return.

Return type `list`/`dict`

Returns A list of object description dictionaries. If `oid` is specified, a single dictionary is returned instead of a list.

Raises GeneralException

Resource `registry/objects[/<oid>] [/since=<since>][/until=<until>][/last=<last>`

Access authorized users (only objects owned by the authenticated user are returned)

`get_order`(*orderid*)
FETCHES a single order defined with `orderid`, or fails if order is non-existing (with *NotFound*).

Parameters **orderid** (`bigint`) – Order ID

Return type `dict`

Returns The order description dictionary.

Raises GeneralException

See *get_orders* `(orderid=...)`

Resource `orders/<orderid>`

Access authorized users (authenticated user MUST be the object/order owner)

`get_orders`(*oid=None*, *include_expired=False*, *orderid=None*)
FETCHES filtered orders. All arguments are optional.

Parameters

- **oid** (`bigint`) – Object ID.
- **include_expired** (`boolean`) – Fetch also expired orders.
- **orderid** (`bigint`) – Order ID. If specified, exactly one order shall be returned, or *NotFound* exception raised. Otherwise, a LIST of orders is returned.

Return type `list`/`dict`

Returns (A list of) order(s) description dictionary(ies).

Raises GeneralException

Resource `[objects/<oid>/]orders/[<all>/]<orderid>`

Access authorized users (authenticated user MUST be the object/order owner)

`get_user_profile`()
FETCHES profile dictionary of the authenticated user.

Return type `dict`

Returns A single user description dictionary.

Raises GeneralException

Resource `/id/users/<uid>`

Access authorized users; only authenticated user's metadata can be fetched (UID is automatically set to the authenticated user's UID)

`get_vouchers`(*vid_encoded=None*, *uid_from=None*, *uid_to=None*, *gid=None*, *valid_after=None*, *valid_before=None*, *last=None*, *first=None*)
FETCHES a filtered list of vouchers.

Parameters

- **vid_encoded** (`alphanumeric(64)`) – Voucher ID, as a string with CRC.
- **uid_from** (`bigint`) – Filter by source account UID.
- **uid_to** (`bigint`) – Filter by destination account UID.
- **gid** (`alphanumeric(32)`) – Filter by voucher Group ID. GID is localized to *uid_from*.
- **valid_after** (`datetime`/`dict`) – Voucher has to be valid after this timestamp. Absolute (`datetime`) or relative (`dict`) timestamps are accepted. Valid keys for relative timestamp dictionary are same as keyword arguments for *datetime.timedelta* (`days`, `seconds`, `minutes`, `hours`, `weeks`).
- **valid_before** (`datetime`/`dict`) – Voucher was valid until this timestamp (for format, see the *valid_after* above).
- **last** (`bigint`) – The number of newest vouchers (that satisfy all other criteria) to return.
- **first** (`bigint`) – The number of oldest vouchers (that satisfy all other criteria) to return.

Note If *first* or *last* are used, the vouchers list is sorted by time created, otherwise it is sorted alphabetically by *vid_encoded*.

Return type `list`/`dict`

Returns A list of voucher description dictionaries. If *vid_encoded* is specified, a single dictionary is returned instead of a list.

Raises GeneralException

Resource `vouchers[/<vid_encoded>][/from=<uid_from>][/to=<uid_to>][/valid_after=<valid_after>][/valid_before=<valid_before>][/last=<last>][/first=<first>]`

Access authorized users (ACL flag: `voucher.get`)

`get_vouchers_history`(*vid_encoded=None*, *vid=None*, *action=None*, *uid_from=None*, *uid_to=None*, *gid=None*, *valid_after=None*, *valid_before=None*, *create_after=None*, *create_before=None*, *last=None*, *first=None*)
FETCHES a filtered list of vouchers log entries.

Parameters

- **vid_encoded** (`alphanumeric(64)`) – Voucher ID, as a string with CRC.
- **vid** (`bigint`) – Voucher ID.
- **action** (`string` (add | use | revoke | expire)) – Filter only these actions on vouchers.
- **uid_from** (`bigint`) – Filter by source account UID.
- **uid_to** (`bigint`) – Filter by destination account UID.

- **gid** (`alphanumeric(32)`) – Filter by voucher Group ID. GID is localized to *uid_from*.
- **valid_after** (`datetime`/`dict`) – Voucher has to be valid after this timestamp. Absolute (`datetime`) or relative (`dict`) timestamps are accepted. Valid keys for relative timestamp dictionary are same as keyword arguments for *datetime.timedelta* (`days`, `seconds`, `minutes`, `hours`, `weeks`).
- **valid_before** (`datetime`/`dict`) – Voucher was valid until this timestamp (for format, see the *valid_after* above).
- **create_after** (`datetime`/`dict`) – Voucher has to be created after this timestamp (for format, see the *valid_after* above).
- **create_before** (`datetime`/`dict`) – Voucher was created until this timestamp (for format, see the *valid_after* above).
- **last** (`bigint`) – The number of newest voucher entries (that satisfy all other criteria) to return.
- **first** (`bigint`) – The number of oldest voucher entries (that satisfy all other criteria) to return.

Note If *first* or *last* are used, the vouchers list is sorted by time created, otherwise it is sorted alphabetically by *id*.

Return type `list`/`dict`

Returns A list of voucher log description dictionaries.

Raises GeneralException

Resource `vouchers/history[/<vid_encoded>][/from=<uid_from>][/to=<uid_to>] [/vid=<vid>][/action=<action>][/last=<last>][/first=<first>] [/valid_after=<valid_after>][/valid_before=<valid_before>] [/create_after=<create_after>][/create_before=<create_before>] [/gid=<group_id>]`

Access authorized users (ACL flag: `voucher.history`)

`kvpath`(*base*, **pa*, ***kw*)

Key-value query url builder (of the form: “base/v0/k1=v1/k2=v2”).

`request`(*verb*, *subpath*, *data=''*)

Generic Vingd-backend authenticated request (currently HTTP Basic Auth over HTTPS, but OAuth1 in the future).

Returns Data `dict`, or raises exception.

`revoke_vouchers`(*vid_encoded=None*, *uid_from=None*, *uid_to=None*, *gid=None*, *valid_after=None*, *valid_before=None*, *last=None*, *first=None*)

REVOKES/INVALIDATES a filtered list of vouchers.

Parameters

- **vid_encoded** (`alphanumeric(64)`) – Voucher ID, as a string with CRC.
- **uid_from** (`bigint`) – Filter by source account UID.
- **uid_to** (`bigint`) – Filter by destination account UID.
- **gid** (`alphanumeric(32)`) – Filter by voucher Group ID. GID is localized to *uid_from*.
- **valid_after** (`datetime`/`dict`) – Voucher has to be valid after this timestamp. Absolute (`datetime`) or relative (`dict`) timestamps are accepted. Valid keys for relative timestamp dictionary are same as keyword arguments for *datetime.timedelta* (`days`, `seconds`, `minutes`, `hours`, `weeks`).

- **valid_before** (`datetime/dict`) – Voucher was valid until this timestamp (for format, see the *valid_after* above).
- **last** (`bigint`) – The number of newest vouchers (that satisfy all other criteria) to return.
- **first** (`bigint`) – The number of oldest vouchers (that satisfy all other criteria) to return.

Note As with *get_vouchers*, filters are restrictive, narrowing down the set of vouchers, which initially includes complete voucher collection. That means, in turn, that a naive empty-handed *revoke_vouchers()* call shall revoke **all** un-used vouchers (both valid and expired)!

Return type `dict`

Returns A dictionary of successfully revoked vouchers, i.e. a map `vid_encoded: refund_transfer_id` for all successfully revoked vouchers.

Raises GeneralException

Resource `vouchers[/<vid_encoded>][/from=<uid_from>][/to=<uid_to>] [/valid_after=<valid_after>][/valid_before=<valid_before>] [/last=<last>][/first=<first>]`

Access authorized users (ACL flag: `voucher.revoke`)

`reward_user` (*huid_to*, *amount*, *description=None*)
PERFORMS a single reward. User defined with *huid_to* is rewarded with *amount* cents, transfered from the account of the authenticated user.

Parameters

- **huid_to** (`alphanumeric(40)`) – Hashed User ID, bound to account of the authenticated user (doing the request).
- **amount** (`integer`) – Amount in cents.
- **description** (`string`) – Transaction description (optional).

Return type `dict`

Returns `{'transfer_id': <transfer_id>}` Fort Transfer ID packed inside a dict.

Raises

- **Forbidden** – consumer has to have `transfer.outbound` ACL flag set.
- **GeneralException** –

  raises NotFound

Resource `rewards/`

Access authorized users (ACL flag: `transfer.outbound`)

`update_object` (*oid*, *name*, *url*)
UPDATES a single object in Vingd Object registry.

Parameters

- **oid** (`bigint`) – Object ID of the object being updated.
- **name** (`string`) – New object's name.
- **url** (`string`) – New callback URL (object's resource location).

Return type *bigint*

Returns Object ID of the updated object.

Raises GeneralException

Resource `registry/objects/<oid>/`

Access authorized user MUST be the object owner

`verify_purchase`(*oid*, *tid*)

VERIFIES token `tid` and returns token data associated with `tid` and bound to object `oid`. At the same time decrements entitlement validity counter for `oid` and `uid` bound to this token.

Parameters

- **oid** (`bigint`) – Object ID.
- **tid** (`alphanumeric(40)`) – Token ID.

Return type `dict`

Returns

A single token data dictionary:

```
token = {
    "object": <object_name>,
    "huid": <hashed_user_id_bound_to_seller> / None,
    "context": <order_context> / None,
    ...
}
```

where:

- `object` is object's name, as stored in object's `description['name']` Registry entry, at the time of token creation/purchase (i.e. if object changes its name in the meantime, `object` field will hold the old/obsolete name).
- `huid` is Hashed User ID - The unique ID for a user, bound to the object owner/seller. Each user/buyer of the `oid` gets an arbitrary (random) identification alphanumeric handle associated with her, such that `huid` is unique in the set of all buyers (users) of all of the seller's objects. In other words, each seller can treat a retrieved `huid` as unique in his little microcosm of all of his users. On the other hand, that same `huid` has absolutely no meaning to anyone else – and user's privacy is guaranteed. Also, note that the value of `huid` **will be** `null` iff buyer chose anonymous purchase.
- `context` is an arbitrary purchase context defined when creating order.

Raises

- **GeneralException** –
- **Forbidden** – User no longer entitled to `oid` (count-wise).

See *commit_purchase*.

Resource `objects/<oid>/tokens/<tid>`

Access authenticated user MUST be the object's owner

2.3 Exceptions

exception `vingd.exceptions.`**`GeneralException`**(*msg*, *context='Error'*, *code=409*)

General exception signifies that an Vingd error has been caught, but reasons/details were not understood/propagated well enough.

exception `vingd.exceptions.`**`InvalidData`**(*msg*, *context='Invalid data'*, *code=400*)

Verification of user data failed.

exception `vingd.exceptions.`**`Forbidden`**(*msg*, *context='Forbidden'*, *code=403*)
User's request resulted with a forbidden action and was therefore cancelled.

exception `vingd.exceptions.`**`NotFound`**(*msg*, *context='Not found'*, *code=404*)
User's request did not yield any reasonable result.

exception `vingd.exceptions.`**`InternalError`**(*msg*, *context='Internal error'*, *code=500*)
Internal server error: it's our fault. :)

CHAPTER 3

Examples

(from *example/test.py*)

```
#!/usr/bin/env python

# path hack
import sys
import os
sys.path.insert(0, os.path.abspath('..'))

from vingd import Vingd

# sandbox backend:
v = Vingd(username="test@vingd.com", password="123", endpoint=Vingd.URL_ENDPOINT_SANDBOX, fronten

# in production use:
#v = Vingd(username="<vingd-login-username>", password="<vingd-login-password>")

#
# profile/account
#

profile = v.get_user_profile()
print 'I (%s) registered on %s.' % (profile['name'], profile['timestamp_created'])

balance = v.get_account_balance()
print 'My balance is VINGD %.2f.' % (balance/100.0)

#
# voucher rewarding
#

voucher = v.create_voucher(amount=100, expires={'days':14})
print "I'm rewarding you with this 1 vingd voucher (%s): %s." % (voucher['raw']['vid_encoded'], v

vouchers = v.get_vouchers()
print "Now I have %d active vouchers." % len(vouchers)

used = v.get_vouchers_history(action='use')
print "Also, %d of my vouchers have been redeemed." % len(used)

expired = v.get_vouchers_history(action='expire')
print "And, %d of my vouchers have expired before anybody used them." % len(expired)

#
# selling
#
```

```
oid = v.create_object("My test object", "http://localhost:666/")
print "I've just created an object, just for you. OID is %d." % oid

oid2 = v.update_object(oid, "New object name", "http://localhost:777/")
print "Object updated."

object = v.get_object(oid)
print "Object last modified at %s, new url is %s" % (object['timestamp_modified'], object['descri

objects = v.get_objects()
print 'I have %d active objects.' % len(objects)

order = v.create_order(oid, 200, context='optional purchase details')
print "I've also created an order (id=%d) for the object (oid=%d): %s" % (order['id'], order['obj

tid = raw_input("After you buy it, enter the Token ID here ('tid' param on callback url): ")
purchase = v.verify_purchase(oid, tid)
huid_buyer = purchase['huid']
context = purchase['context']
print "Purchase verified (buyer's HUID = %s, context = '%s')." % (huid_buyer, context)

commit = v.commit_purchase(purchase['purchaseid'], purchase['transferid'])
print "Content served, and purchase committed."

#
# direct rewarding
#

reward = v.reward_user(huid_to=huid_buyer, amount=75, description='Testing direct rewarding')
print "User rewarded (transfer id = %s)." % reward['transfer_id']
```

CHAPTER 4

Indices and tables

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

Python Module Index