Vingd API for PHP Documentation

Release 1.7

Radomir Stevanovic, Vingd Inc.

April 30, 2015

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 PHP

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

The last stable release of PHP Vingd API is available on GitHub:

```
$ git clone https://github.com/vingd/vingd-api-php
$ cd vingd-api-php
$ php example/test.php
```

1.3 Examples

Client initialization and account balance fetching:

```
require_once('/path/to/vingd-api-php/vingd/vingd.php');

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

// Initialize vingd client.
$v = new Vingd($VINGD_USERNAME, $VINGD_PASSWORD, Vingd::URL_ENDPOINT_SANDBOX, Vingd::URL_FRONTEND_

// Fetch user balance.
$balance = $v->getUserBalance();
```

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 = 2.00;

// Register vingd object (once per selling item).
$oid = $v->createObject($OBJECT_NAME, $OBJECT_URL);

// Prepare vingd order.
$order = $v->createOrder($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 fronted 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

// Verify purchase with received parameters.
$purchase = $v->verifyPurchase(array('oid' => $oid, 'tid' => $tid));

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

// Content is successfully served, commit vingd transaction.
$commit = $v->commitPurchase($purchase);
```

1.3.2 Reward user with vingd

Transfer vingd directly on users account:

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

// Reward user.
$reward = $v->rewardUser($REWARD_HUID, $REWARD_AMOUNT, $REWARD_DESCRIPTION);
```

1.3.3 Reward user with voucher

Redirect user to redeem his reward on vingd frontend:

```
$VOUCHER_AMOUNT = 1.00;
$VOUCHER_VALID_PERIOD = '+7 days';

// Create vingd voucher.
$voucher = $v->createVoucher($VOUCHER_AMOUNT, $VOUCHER_VALID_PERIOD);

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

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

1.4 Documentation

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

1.5 Copyright and License

CHAPTER 2

Vingd interaction overview

You communicate with Vingd system thru Vingd Broker, using the `Vingd` PHP class. Direct communication with Vingd Broker over HTTP is also simple (see A few technical details), but discouraged.

Vingd enables you to monetize your digital goods by "selling" the access rights to your content/service. You host the content and you control the access, and we handle the access rights. As the first approximation, Vingd can be seen as a payment gateway.

Thru Vingd, you can also reward users with vingds via *Vingd Vouchers*, or directly transfering vingds to user account in backend.

2.1 Purchase

The content or service of yours has an abstract representation on Vingd, called *Vingd Object*. Object enrollment (aka registration) is a one-time process, after which an object can be "sold" to Vingd users. Terms of purchase (namely, the price) are defined by you, the seller, in what is called *Purchase Order*. Orders can be viewed as bills you give to users. User pays that bill on Vingd user frontend and it gets object access rights assigned for the object you referenced in the order. Each time a user tries to access your object (the location you specified upon object enrollment and which you control), you should demand an access token from your user, issued by Vingd Broker. With the access token we shall vouch that the specific user has paid for the access, and you should allow him to view the object (content/service).

2.1.1 Registering objects

In order to sell anything thru Vingd, you must *enroll* (i.e. register) an abstract representation of that item, as an object in our *Object Registry* (thru Vingd Broker). Objects are enrolled with `Vingd::createObject` function. During object enrollment, a unique *Object ID* (`oid`) is assigned to that object. Store that `oid`, since you'll be using it as the object handle.

Object description is a dictionary (associative array) with only two mandatory keys: `name` and `url`. Make sure the `url` points to a valid location, because users will be redirected there upon object purchase (consider this URL to represent the object location, according to the REST paradigm). More precisely, the complete URL where user will be redirected is object's `url` with *Token ID* and *Object ID* glued as GET parameters (`tid` and `oid`).

2.1.2 Selling an object

To sell any of your enrolled objects, you must create an *Object Order* which shall encapsulate the terms under which the object is being sold to the customer (e.g. price). `Vingd::createOrder` method facilitates order creation. Object and price are defined with *$oid* and *$price* in vingds (rounded to cents). Expiry date of the very order is defined with *$expires*.

It is important to note that orders are not bound to a specific user. That means you could (and preferably would) generate one order with expiry time longer than the default 15mins and offer that same order to more than one user

(e.g. you could have special order for each "class" of your users). The advantage of having pre-generated orders is you are cutting down the overhead of object purchase on your site - you don't have to generate a new order in background for every user's "buy-click", but simply direct the user to order-purchase-link on Vingd (as returned upon the initial order making).

On a successful order creation, you shall be given an URL pointing to your order on Vingd frontend (see `['urls']['redirect']` element of a value returned from `Vingd::createOrder`). To enable an user to pay the access to your object, you should direct her exactly to that very URL (if you're using Vingd Popup Library you should use the `order['urls']['popup']` URL).

2.1.3 Verifying user access rights

Upon successful confirmation of an order (specified amount of vingds is reserved from user's account), a *Vingd Token* is issued by Vingd Broker. User is redirected to the (callback) *object url* with the *Token* (`tid`) and *Object ID* (`oid`) appended. This token is short-lived (typically 15 minutes) and during its lifespan it represents a voucher that can be checked to see if the carrier of the `(tid, oid)` pair has rights for accessing object `oid`.

To verify the token, use `Vingd::verifyPurchase`. If user access is granted, token verification interface returns *Token Description*.

In a popup mode, you will also want to verify the token when user returns from Vingd frontend (see Vingd Popup onSuccess() callback).

2.1.4 Wrapping up the purchase

Once you have successfully served the content user has paid for, you should notify the Vingd Broker, referencing the *Order ID*, *Purchase ID* and *Transfer ID* (the last two are returned in *Token Description* upon token verification). This completes the purchase.

If you don't do `Vingd::commitPurchase`, Vingd Broker assumes a seller (you) failed to deliver the content or service to the user and does an automatic refund.

2.2 Rewarding

`Vingd::createVoucher` allows you to allocate a certain amount of vingds from your account and offer it to a Vingd user. Voucher code looks like `LXKG-TBR-HQV` (only letters A-Z and digits 0-9 are significant; all other characters are ignored). Upon voucher creation, `Vingd::createVoucher` returns a structure with an URL pointing to Vingd frontend which user can follow to use the voucher. Vingd popup can also be used, see Vingd Popup Voucher example.

2.3 A few technical details

Vingd Broker (https://api.vingd.com/broker/v1/) has a very simple REST interface to a complete Vingd backend. For example, to retrieve a list of objects you registered, execute a `GET` request on the Vingd Objects resource (https://api.vingd.com/broker/v1/registry/objects/) authenticating using HTTP Basic Auth (with Vingd username and password SHA1 hash). The response should be a JSON list of object descriptions.

Creating (enrolling, or registering) a new object is slightly more complex, but nevertheless still trivial: `POST` a JSON-encoded description of the object to that same URL (which, btw, represents a collection of your objects).

However, if you are using Python/PHP/.NET/Java there should never be a need for you to manually implement a client for our REST backend, since we already support libraries for those environments.

CHAPTER 3

Vingd library reference

3.1 Vingd class

class `Vingd`
Vingd API interface class.

3.1.1 Constants

constant `Vingd::URL_ENDPOINT`
https://api.vingd.com/broker/v1

constant `Vingd::URL_FRONTEND`
https://www.vingd.com

constant `Vingd::URL_ENDPOINT_SANDBOX`
https://api.vingd.com/sandbox/broker/v1

constant `Vingd::URL_FRONTEND_SANDBOX`
http://www.sandbox.vingd.com

constant `Vingd::EXP_ORDER`
Default order expiry, '+15 minutes'. The order expires in 15 minutes, relative to the time of creation.

constant `Vingd::EXP_VOUCHER`
Default voucher expiry, '+1 month'. The voucher expires in one month, relative to the time of creation. When it expires, allocated funds are refunded to the issuer's Vingd account.

3.1.2 Purchase

`Vingd::createObject` (*$name*, *$url*)
Creates (registers) an object in the Vingd Object Registry.

Parameters

- **$name** (*string*) – Object's name.
- **$url** (*string*) – Object's callback URL.

Returns (integer) Object ID assigned in Vingd.

Throws VingdException, Exception

`Vingd::updateObject` (*$oid*, *$name*, *$url*)
Updates an object enrolled in the Vingd Object Registry.

Parameters

- **$oid** (*integer*) – Object ID, as returned by `Vingd::createObject`.
- **$name** (*string*) – Object's new name.
- **$url** (*string*) – Object's new callback URL.

Returns (integer) Object ID assigned in Vingd.

Throws VingdException, Exception

`Vingd::`**`getObject`**(*$oid*)
Fetches object with id *$oid*.

Parameters

- **$oid** (*integer*) – Object ID, as returned by `Vingd::createObject`.

Returns (array) Vingd object description.

Throws VingdException, Exception

`Vingd::`**`getObjects`**()
Fetches all objects for the authenticated user.

Returns (array) A list of Vingd objects.

Throws VingdException, Exception

`Vingd::`**`createOrder`**(*$oid*, *$price*, *$context* = *null*, *$expires* = *Vingd::EXP_ORDER*)
Creates an order for object *$oid*, with price set to *$price* and validity until *$expires*.

Parameters

- **$oid** (*integer*) – Identifier of the object to be sold, as returned by `Vingd::createObject`.
- **$price** (*float*) – Object's price in VINGDs. Rounded to two decimal digits.
- **$context** (*string*) – Arbitrary (user-defined) context handle of this purchase. $context shall be retrieved upon purchase/token verification. (Usage discouraged for sensitive data.) Default: no context associated with order.
- **$expires** (*string*) – Expiry timestamp / validity period of the order being generated (accepts any PHP/strtotime-parsable date/time string, including ISO 8601, RFC 822, and most English date formats, as well as relative dates). Default: `Vingd::EXP_ORDER` ('+15 minutes', i.e. order expires in 15 minutes).

Returns (array) Vingd order description.

Throws VingdException, Exception

`Vingd::`**`verifyPurchase`**(*$token*)
Verifies purchase token *$tid* and returns token data associated with it (and bound to object *$oid*).

Note If token was invalid (purchase can not be verified), a VingdException is thrown.

Parameters

- **$token** (*array/string*) – Access token user brings in, as returned from Vingd user frontent on object's callback URL.

  `verifyPurchase` accepts *$token* as either *string* (as read from `$_GET['token']` on callback processor), or as *array* (json-decoded from the URL).

  You should always verify the token the user brings in from Vingd frontend to ensure the user has access rights for your object and/or service. Successful token verification guarantees Vingd Broker has reserved user vingds for the seller. Those vingds will be transfered after seller commits the purchase (see `commitPurchase`).

Returns

(array) Purchase details:

- 'object' key: Object name.
- 'huid' key: Buyer's seller-bound user id (user id unique for the authenticated owner/seller of *$oid*).
- 'purchaseid': Vingd Purchase ID.
- 'transferid': Vingd transfer ID.

Throws VingdException, Exception

`Vingd::`**`commitPurchase`**(*$purchase*)

Commits the purchase (defined with *$purchaseid* and *$transferid*) as finished.

Call `commitPurchase` upon successful delivery of paid content to the user. If you do not call `commitPurchase`, the user shall be refunded automatically.

Parameters

- **$purchase** (*array*) – User purchase description, as returned by `verifyPurchase` upon successful token verification.

Returns (array) ('ok => true), or fails with VingdException.

Throws VingdException, Exception

3.1.3 Rewarding

`Vingd::`**`createVoucher`**(*$amount*, *$until = Vingd::EXP_VOUCHER*, *$message = ''*, *$gid = null*, *$description = null*)

Makes a new voucher.

Parameters

- **$amount** (*float*) – Voucher amount in VINGDs.
- **$until** (*date*) – Timestamp of voucher expiry (accepts any PHP/strtotime-parsable date/time string, including ISO 8601, RFC 822, and most English date formats, as well as relative dates; e.g. '+1 day', '+1 month', etc.). Default: `Vingd::EXP_VOUCHER` ('+1 month', i.e. voucher expires in one month). When it expires, allocated funds are refunded to the issuer's Vingd account.
- **$message** (*string*) – A message user shall be presented with after submitting the voucher (on Vingd frontend).
- **$gid** (*string*) – Voucher Group ID (alphanumeric string: `[-_a-zA-Z0-9]{1,32}`). User can use only one voucher per group.
- **$description** (*string*) – Voucher internal description. Optional, but can be helpful for tracking.

Returns (array) Voucher description. The most interesting keys being: `code` and `urls` (which branches to `redirect` URL and `popup` URL).

Throws VingdException, Exception

`Vingd::`**`getActiveVouchers`**()

Fetches a list of all active (non-expired) vouchers for the authenticated user.

Returns (array) A list of voucher descriptions.

Throws VingdException, Exception

`Vingd::`**`getVouchers`**()

Fetches a complete vouchers history (for the authenticated user). The list includes **active**, **expired**, **used** and **revoked** vouchers (discriminated via `action` key in voucher description).

Returns (array) A list of voucher descriptions.

Throws VingdException, Exception

`Vingd::`**`rewardUser`**(*$huid*, *$amount*, *$description = null*)
Rewards user defined with *$huid* with *$amount* vingds, transfered from the account of the authenticated user.

Parameters

- **$amount** (*float*) – Voucher amount in VINGDs.
- **$until** (*date*) – Timestamp of voucher expiry (accepts any PHP/strtotime-parsable date/time string, including ISO 8601, RFC 822, and most English date formats, as well as relative dates; e.g. '+1 day', '+1 month', etc.). Default: `Vingd::EXP_VOUCHER` ('+1 month', i.e. voucher expires in one month). When it expires, allocated funds are refunded to the issuer's Vingd account.
- **$message** (*string*) – A message user shall be presented with after submitting the voucher (on Vingd frontend).
- **$gid** (*string*) – Voucher Group ID (alphanumeric string: `[-_a-zA-Z0-9]{1,32}`). User can use only one voucher per group.
- **$description** (*string*) – Voucher internal description. Optional, but can be helpful for tracking.

Returns (array) Voucher description. The most interesting keys being: `code` and `urls` (which branches to `redirect` URL and `popup` URL).

Throws VingdException, Exception

3.1.4 Account-related

`Vingd::`**`getUserProfile`**`()`
Fetches profile of the authenticated user.

Returns (array) User profile.

Throws VingdException, Exception

`Vingd::`**`getUserId`**`()`
Shorthand to fetch authenticated user's id.

Returns (integer) User ID.

Throws VingdException, Exception

`Vingd::`**`getUserBalance`**`()`
Fetches authenticated user's account balance.

Returns (float) Account balance.

Throws VingdException, Exception

CHAPTER 4

Indices and tables

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

C

G

R

U

V