PHP Client library

Overview

Client library for PHP provides tools for easier integration with tinypass. It can be used to do server-side access checking; processing encrypted data; encrypting data; making API requests; etc.

Basic usage

Configuration

Tinypass library requires connection credentials to be configured. To do that - create instance of configuration object.

```
// Include Tinypass - make sure to use the correct path
include_once "/path/to/api/TinypassClient.php";

$tinypassConfig = new TPConfig( $applicationId, $apiToken, $privateKey,
$isSandbox );
```

Parameters:

- applicationId – 10 character application ID, can be retrieved from tinypass dashboard
- apiToken – 40 character API token, can be retrieved from tinypass dashboard
- privateKey – 40 character private key, can be retrieved from tinypass dashboard, or by making API request.
- isSandbox – To do queries to sandbox.tinypass.com - set to "true"; To do queries to api.tinypass.com - set to "false"; To do queries to dedicated environment - set value as string url to api root.

Preparing for query

Before executing any queries - client object should be initialised with the created configuration object.

```
$tinypassClient = new TinypassClient( $tinypassConfig );
```

Executing the query

All queries to API can be executed through methods of client object. The query can be executed using builder-pattern code.

```
// The resulting object of this operation will be an instance of
TPPublisherApp
$result = $tinypassClient
    ->PublisherApp()  // Get Publisher Application API
                ->getRequest()   // Create get application request
                ->aid( $applicationId ) // Set application id
                ->execute();   // Execute the query
```

The resulting object differs from which method was called. Refer to API documentation to see complete list of API methods and their returning data types.

Handling errors

If the client encounters connection errors during execution - it throws standard exception. The resulting object of query execution will contain error information In case of encountering API errors. It can be handled as follows:

```
// In case of error - the resulting object will contain "code"
attribute,
// which will be greater than 0, as well as "message" attribute,
containing error message.
if ( isset( $result->code ) && $result->code ) {
 throw new Exception( $result->message, $result->code );
}
```

Helper classes

Library provides several helper classes for easier tinypass integration.

UserRef token builder

To provide data about logged in user (their email, name, registration date and any other) tinypass uses special UserRef token. Library provides helper class for generation of this token.

```
<?php
// Include Tinypass - make sure to use the correct path
include_once "/path/to/api/TinypassClient.php";
$userRef = TPUserRefBuilder::create( $userId, $userEmail ) // User id
and email are required parameters
   ->setFirstName( $userFirstName )    // Set user's first name
   ->setLastName( $userLastName )     // Set user's last name
   ->setCreateDate( $userRegistrationDate )  // Set user's registration
date (in seconds)
   ->set( $name, $value )       // Set additional parameter
   ->build( $privateKey );       // Encrypt the data using private key
and return userRef token
?>
<script type="text/javascript">
 // Set userRef token for use in javascript library
 tp.push(["setUserRef", <?php echo json_encode( $userRef ) ?> ]);
</script>
```

For more information about UserRef token - refer to API documentation.

Webhook parser

For easier decryption and usage of webhooks, library provides webhook parser.

```
// Encrypted webhook data is provided as GET parameter
$encryptedData = strval( @$_GET['data'] );

// Decrypt and parse data
$webhookEvent = Webhook::parse( $encryptedData, $privateKey );

switch ($webhookEvent->getEventType()) {
 case TPEvent::ACCESS_EVENT:
  // Process as access event webhook
  break;
 case TPEvent::SUBSCRIPTION_EVENT:
  // Process as subscription event webhook
  break;
  case TPEvent::TEST_EVENT:
  // Process as test event webhook
  break;
}
```

For more information about webhooks – refer to API documentation.

Access token parser

To minimize interaction with API for access checking, tinypass can store access information as a cookie. The library provides access token parser

to check access data from this cookie.

```
// Include Tinypass - make sure to use the correct path
include_once "/path/to/api/TinypassClient.php";
// Init configuration object. This can be omitted if configuration
object was initialised earlier.
$tinypassConfig = new TPConfig( $applicationId, $apiToken, $privateKey,
$isSandbox );
// Init Acccess token store object with created configuration
$tokenStore = new TPAccessTokenStore( $tinypassConfig );
// Parse cookies to get tokens
$tokenStore->loadTokensFromCookie( $_COOKIE );
// Get access token for some resource
$accessToken = $tokenStore->getAccessToken( $resourceId );

if ( $accessToken->isAccessGranted() ) {
 // Access is granted, content can be displayed
} else {
 // Access is denied, content should be hidden, and offer should be
displayed
}
```