API calls w/ PHP

File	Modified
eTrusted_demo.zip	17.09.2019 by Carsten Hellweg

PHP Client for the eTrusted API

PHP Wrapper class - Integration

In order to encapsulate the access-token, I created a wrapper-class. This is what you need to do in order to get the code running (taken from the README.txt file):

Quick-and-Dirty: Uncomment Code in TS_CurlWrapper and fill out values

The top of the file should look like this afterwards. Note: If the customer does an update, changed values will be overwritten.

TS_CurlWrapper.php

```
<?php

class MyTSCurlWrapper extends TS_CurlWrapper
{
    protected function getCredentials()
    {
        return array(
            // your values below here!
              'client_id' => 'abc'
            , 'client_secret' => 'xyz'
        );
    }
}
// */
```

Update-Safe: Create a localconfig-file and fill out the values

Every democode-file has a block like this, which you can remove,if you created the file (leave only the require_once).

```
if( is_file( __DIR__ . '/../eTrusted_localconfig.php' ) )
{
 require_once __DIR__ . '/../eTrusted_localconfig.php';
}
```

It should look like this (remove is_file in PROD systems to reduce filesystem-access):

```
require_once __DIR__ . '/../eTrusted_localconfig.php';
```

The config-file is placed above the current working-directory, in order to prevent accidential deleting/sharing/overwriting.

The contents of '../eTrusted_localconfig.php' and the TS_CurlWrapper.php code box should be the same.

../eTrusted_localconfig.php

```
<?php

class MyTSCurlWrapper extends TS_CurlWrapper
{
    protected function getCredentials()
    {
        return array(
            // your values below here!
              'client_id' => 'abc'
            , 'client_secret' => 'xyz'
        );
    }
}
```

Update-Safe and most complex: Multiple configurations

Copy the file "multipleAccountSolution_example.php" to "../eTrusted_localconfig.php".

Fill out the shop-data you need/ have and delete the classes that you don't need.

../eTrusted_localconfig.php from multipleAccountSolution_example.php

```
<?php
require_once __DIR__ . '/TS_CurlWrapper.php';

/**
 * Creates an Instance of the relevant cURL-Wrapper.
 * Use instead of "new ....()".
 *
 * @param String $language
 * @return TS_CurlWrapper
 */
function getTSCurlWrapper( $language = 'de' )
{
    if( $language == 'de' )
    {
        return new deShop();
    }
    if( $language == 'fr' )
    {
        return new frShop();
    }

    die( 'cannot find shop-config for:' . $language . ':' );
}

class deShop extends TS_CurlWrapper
{
    protected function getCredentials()
    {
  // your germany-values below here!
   return array( 'client_id' => 'abc', 'client_secret' => 'xyz' );
    }
}

class frShop extends TS_CurlWrapper
{
 protected function getCredentials()
    {
  // your france-values below here!
   return array( 'client_id' => 'abc', 'client_secret' => 'xyz' );
    }
}
```

The demo-code needs to be re-written, if you want to use this option. In the demo-code new MyTSCurlWrapper() should not be used and the factory-function named "getTSCurlWrapper(...)" from the above code should be used to create new objects. This is an example for that.

```
<?php
 $language = 'de';

 $wrapper = getTSCurlWrapper( $language ); // old: new MyTSCurlWrapper();
    $url = 'https://api.etrusted.com/events';

    $payload = array();
    // ************************************************
    // * REQUIRED PARAMETERS IN ORDER OF documentation *
    // ************************************************

    $payload[ 'type' ] = 'checkout';
    // .........

 $result = $wrapper->post( $url, $payload );
```

REST Basics

The eTrusted API is using a REpresentional State Transfer Interface. In order to read and/or modify a ressource, the Items must have an unique URL and there are special means of working on these Items. If you would normally use local means for CRUD (Create Read Update Delete) on an item, CRUD would be transformed into different kinds of HTTP-Calls.

The table bellow shows the different means of REST and the methods of the wrapper classes to use. The REST-Endpoints, which are explained h ere usually follow that logic.

local Means	HTTP/REST	Wrapperclass-Method
C reate	POST	post($url, $payload)
R ead	GET	get($url)
U pdate	PUT	put($url, $payload)
D elete	DELETE	delete($url, $payload)

Wrapper class - Practical examples

Once you have created the Class MyTSCurlWrapper and inserted the correct credentials, you can run the demo-scripts.

POST (Create)

api_Events_create.php

```php
<?php
    require_once __DIR__ . '/TS_CurlWrapper.php';

    TS_CurlWrapper::$DEBUG = true; // turn on the verbose mode

    $wrapper = new MyTSCurlWrapper();
    $url = 'https://api.etrusted.com/events';
    $payload = array();

    // ************************************************
    // * REQUIRED PARAMETERS IN ORDER OF documentation *
    // ************************************************
    $payload[ 'type' ] = 'checkout';
    // ....

    $result = $wrapper->post( $url, $payload );
```

GET (Read)

api_InviteRule_getRules.php

```php
<?php
 require_once __DIR__ . '/TS_CurlWrapper.php';

 TS_CurlWrapper::$DEBUG = true; // turn on the verbose mode

 $wrapper = new MyTSCurlWrapper();
    $url = "https://api.etrusted.com/invite-rules";
    $result = $wrapper->get( $url );
```

PUT (Update)

api_Events_updateType.php

```
<?php
    require_once __DIR__ . '/TS_CurlWrapper.php';

    TS_CurlWrapper::$DEBUG = true;  // turn on the verbose mode

    $eventId = 'ety-xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx';

 $wrapper = new MyTSCurlWrapper();
    $url = 'https://api.etrusted.com/event-types/' . $eventId;
    $payload = array();
    $payload[ 'active' ] = 'true';

    $result = $wrapper->put( $url, $payload );
```

DELETE (Delete)

no such file, but this is how it would work

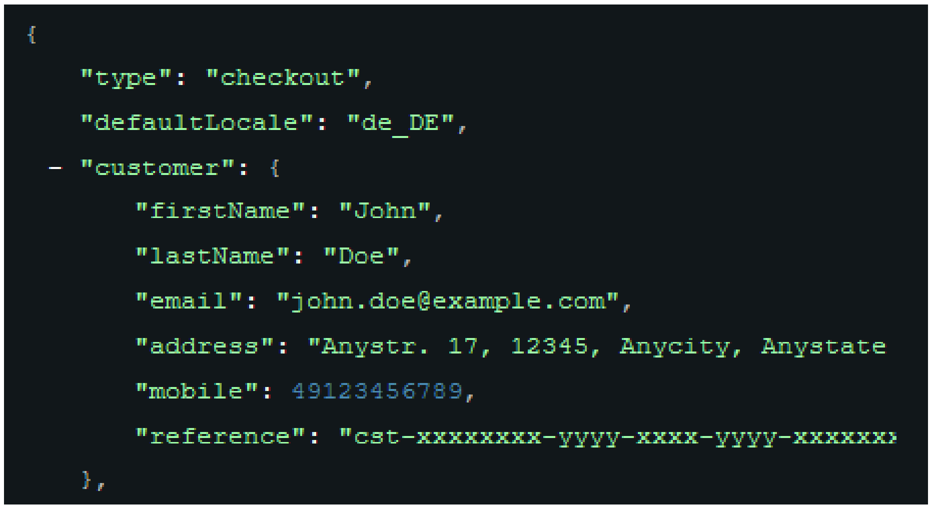
```
<?php
    require_once __DIR__ . '/TS_CurlWrapper.php';

    TS_CurlWrapper::$DEBUG = true;  // turn on the verbose mode

    $eventId = 'ety-xxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx';

 $wrapper = new MyTSCurlWrapper();
    $url = '......' . $eventId;
    $payload = array();
 // ...

    $result = $wrapper->delete( $url, $payload );
```

PHP cURL Basics

Input - reducing complexity

In order to create a JSON input string, you can also use blank arrays in PHP.

Take a look at this example:



```
{
    "type": "checkout",
    "defaultLocale": "de_DE",
  - "customer": {
        "firstName": "John",
        "lastName": "Doe",
        "email": "john.doe@example.com",
        "address": "Anystr. 17, 12345, Anycity, Anystate
        "mobile": 49123456789,
        "reference": "cst-xxxxxxxx-yyyy-xxxx-yyyy-xxxxxx
    },
```

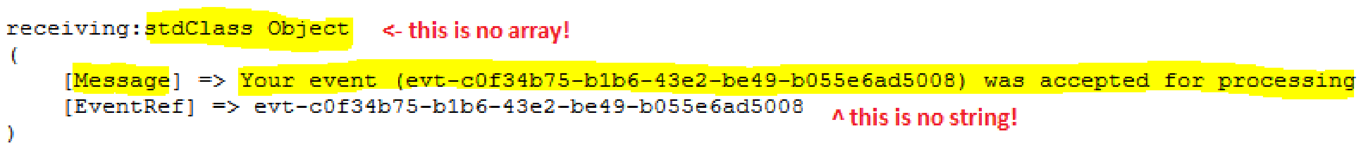
In the above example, "type" is a single-value item, "customer" has multiple entries.

In order to create the payload in PHP, you can use this:

```
<?php
$payload = array();              // init the values

// single value
$payload[ 'type' ] = 'checkout';
$payload[ 'defaultLocale' ] = 'de_DE';

// multiple values, init sub-section
$customer = array();
$customer[ 'firstName' ] = 'John';
$customer[ 'lastName' ] = 'Doe';
$customer[ 'email' ] = 'john.doe@example.com';
$customer[ 'address' ] = 'Anystr. 17, 12345, Anycity, Anystate';
$payload[ 'customer' ] = $customer; // add subsection to $payload

// in order to test the results, you can print_r the $payload
print_r( $payload );

/*
Output may look like
Array
(
    [type] => checkout
    [defaultLocale] => de_DE
    [customer] => Array
        (
            [firstName] => John
            [lastName] => Doe
            [email] => john.doe@example.com
            [address] => nystr. 17, 12345, Anycity, Anystate
        )
)
check, if the indention is similar to the one in json

// */
```

use the Output in your own code

This is an output of the Wrapper Class:



```
receiving:stdClass Object
(
    [Message] => Your event (evt-c0f34b75-b1b6-43e2-be49-b055e6ad5008) was accepted for processing
    [EventRef] => evt-c0f34b75-b1b6-43e2-be49-b055e6ad5008
)
```

stdClass Object is not an array!
Access the entries of a call like this:

```
<?php
// ... init the payload

$wrapper = new MyTSCurlWrapper();
$result = $wrapper->post( $url, $payload );

// THIS DOES WORK
echo $result->Message;

// THIS DOES NOT(!!!!) WORK
echo $result[ 'Message' ];
```

Entries are not Strings!
The entries may look like strings, but they are NOT!

If you want to use the entries in arrays, use implicit type-casts to string!

```
<?php
// ... init the payload

$wrapper = new MyTSCurlWrapper();
$result = $wrapper->post( $url, $payload );

// THIS DOES NOT(!!!!) WORK
$arrayVariable[ $result->EventRef ] = 1;

// THIS DOES WORK
$arrayVariable[ '' . $result->EventRef ] = 1; // use implicit
type-cast to string for
```

PHP Wrapper class - further Integrations

All methods meant to be extended.

```
<?php
abstract class TS_CurlWrapper
{
    /**
     * @Override to configure your password.
     *
     * @return array( 'client_id' => 'abc', 'client_secret' => 'xyz' )
     */
    abstract protected function getCredentials();

    /**
     * @Override, if you want to use a different Logger-class
     *
     * @return bool
     */
    protected function isDebugEnabled()
    {
        // ...
    }

    /**
     * @Override, if you want to use a different Logger-class
     *
     * @param String $message
     */
    protected function log( $message )
    {
  // ...
    }

    /**
     * @Override, if you want to add further curl-options, eg a Proxy
     *
     * @return curl curl-object
     */
    protected function createCurl()
    {
        // ....
    }
}
```

Abstract protected function getCredentials()

This class works with "configuration over inheritance", so this is the only method, that you **NEED** to overwrite.

Just fill out the credentials, you have been sent from Trusted Shops.

Logger

Overwrite both the methods isLoggingEnabled() and log($message), eg. like this for log4php.

```
<?php

class MyLoggingCurlWrapper extends TS_CurlWrapper
{
    protected $logger;

    public function __construct()
    {
  $this->logger = Logger::getLogger( __CLASS__ );
  parent::__construct();
    }

    protected function isDebugEnabled()
    {
        return $this->logger->isDebugEnabled();
    }

    protected function log( $message )
    {
        $this->logger->debug( $message );
    }
}
```

other cURL Options, eg PROXY-Configuration

```
<?php

class MyProxyCurlWrapper extends TS_CurlWrapper
{
 protected function createCurl()
    {
     $curl = parent::createCurl();
     curl_setopt($curl, CURLOPT_PROXY .... );
  return $curl;
 }
}
```