```php
<?php


use PHPUnit\Framework\Assert;
use PHPUnit\Framework\TestCase;
require 'functions.php';


/**
 * Class functionsTest
 */
class functionsTest extends TestCase
{
    /**
     * Tests the call_api function
     */
    public function test_call_api(): void
    {
        $method_post = 'POST';
        $method_get = 'GET';
        $method_different = '1234';
        $wc_order_test_request = '';
        $backend_url = 'https://backend.demo.taler.net';
        $api_key = 'ApiKey sandbox';

        try {
            $wc_order_test_request = 'wc_test_' . random_int(0, 1000);
        } catch (Exception $e) {
        }
        $body_request_1 = array(
            'order' => array(
                'amount' => 'KUDOS:0.1',
                'fulfillment_url' => 'http://gnutaler.hofmd.ch',
                'summary' => 'Test_order',
                'order_id' => $wc_order_test_request,
            )
        );
        $purpose_1 = 'create_order';

        $body_request_2 = $wc_order_test_request;
        $purpose_2 = 'confirm_payment';

        $result_create_order = call_api( $method_post, $backend_url,
        json_encode($body_request_1), $purpose_1, $api_key );
        $result_confirm_payment = call_api( $method_get, $backend_url,
        $body_request_2, $purpose_2, $api_key );
        $result_verify_backend_url = call_api( $method_get, $backend_url, '', '',
        $api_key );
        $result_method_different = call_api( $method_different, $backend_url,
        json_encode($body_request_1), $purpose_1, $api_key );

        Assert::assertTrue($result_create_order['result']);
        Assert::assertEquals($wc_order_test_request,
        json_decode($result_create_order['message'], true)['order_id']);
        Assert::assertTrue($result_confirm_payment['result']);
        Assert::assertEquals(false, json_decode($result_confirm_payment['message'],
        true)['paid']);
        Assert::assertTrue($result_verify_backend_url['result']);
        Assert::assertEquals('Hello, I\'m a merchant\'s Taler backend. This HTTP
        server is not for humans.', trim($result_verify_backend_url['message']));
        Assert::assertFalse($result_method_different['result']);
        Assert::assertEquals('Bad request', $result_method_different['message']);
    }


    /**
     * Tests the create_api_url function
     */
    public function test_create_api_url(): void
    {
        $url_test = 'https://backend.demo.taler.net';
        $wc_test_order = '';
```

```
66              try {
67                  $wc_test_order = 'wc_test_' . random_int(0, 1000);
68              } catch (Exception $e) {
69              }
70              $purpose_create_order = 'create_order';
71              $purpose_confirm_payment = 'confirm_payment';
72              $purpose_create_refund = 'create_refund';
73
74              $create_order_url = create_api_url($url_test, $purpose_create_order, '');
75              $confirm_payment_url = create_api_url($url_test, $purpose_confirm_payment,
                $wc_test_order);
76              $create_refund_url = create_api_url($url_test, $purpose_create_refund, '');
77
78              Assert::assertEquals('https://backend.demo.taler.net/order',
                $create_order_url);
79
                Assert::assertEquals('https://backend.demo.taler.net/check-payment?order_id='
                . $wc_test_order, $confirm_payment_url);
80              Assert::assertEquals('https://backend.demo.taler.net/refund',
                $create_refund_url);
81          }
82
83          /**
84           * Tests the curl_error_handling function with the http status code 200
85           */
86          public function test_curl_error_handling_code_200(): void
87          {
88              $api_key = 'ApiKey sandbox';
89              $test_url = 'https://backend.demo.taler.net';
90
91              $curl_error_message_array = call_api('GET', $test_url, '', '', $api_key);
92
93              Assert::assertTrue($curl_error_message_array['result']);
94              Assert::assertEquals(200, $curl_error_message_array['http_code']);
95          }
96
97          /**
98           * Tests the curl_error_handling function with the http status code 400
99           */
100         public function test_curl_error_handling_code_400(): void
101         {
102             $api_key = 'ApiKey sandbox';
103             $api_key_wrong = 'ApiKey ____***£££èèè';
104             $test_url = 'https://backend.demo.taler.net';
105             $test_url_wrong = 'https://backend.demo.taler.net/test_if_this_exits';
106             $body = json_encode(array(
107                 'order' => array(
108                     'wrong_field' => 'Wrong value',
109                 )
110             ));
111
112             $curl_error_message_array_400 = call_api('POST', $test_url, $body,
                'create_order', $api_key);
113             $curl_error_message_array_401 = call_api('GET', $test_url, '', '',
                $api_key_wrong);
114             $curl_error_message_array_403 = call_api('GET', 'https://httpstat.us/403',
                '', '', '');
115             $curl_error_message_array_404 = call_api('GET', $test_url_wrong, '', '',
                $api_key);
116
117             Assert::assertFalse($curl_error_message_array_400['result']);
118             Assert::assertEquals(400, $curl_error_message_array_400['http_code']);
119             Assert::assertFalse($curl_error_message_array_401['result']);
120             Assert::assertEquals(401, $curl_error_message_array_401['http_code']);
121             Assert::assertFalse($curl_error_message_array_403['result']);
122             Assert::assertEquals(403, $curl_error_message_array_403['http_code']);
123             Assert::assertFalse($curl_error_message_array_404['result']);
124             Assert::assertEquals(404, $curl_error_message_array_404['http_code']);
125
126         }
127
128
```

```php
    /**
     * Tests the curl_error_handling function with the http status code 500
     */
    public function test_curl_error_handling_code_500(): void
    {
        $api_key = 'ApiKey sandbox';
        $test_url = 'https://backend.demo.taler.net';
        $test_url_500 = 'https://httpstat.us/500';
        $test_url_502 = 'https://httpstat.us/502';
        $test_url_503 = 'https://httpstat.us/503';
        $test_url_504 = 'https://httpstat.us/504';

        $curl_error_message_array_500 = call_api('GET', $test_url_500, '', '', $api_key);
        $curl_error_message_array_502 = call_api('GET', $test_url_502, '', '', $api_key);
        $curl_error_message_array_503 = call_api('GET', $test_url_503, '', '', $api_key);
        $curl_error_message_array_504 = call_api('GET', $test_url_504, '', '', $api_key);

        Assert::assertFalse($curl_error_message_array_500['result']);
        Assert::assertEquals(500, $curl_error_message_array_500['http_code']);
        Assert::assertFalse($curl_error_message_array_502['result']);
        Assert::assertEquals(502, $curl_error_message_array_502['http_code']);
        Assert::assertFalse($curl_error_message_array_503['result']);
        Assert::assertEquals(503, $curl_error_message_array_503['http_code']);
        Assert::assertFalse($curl_error_message_array_504['result']);
        Assert::assertEquals(504, $curl_error_message_array_504['http_code']);

    }

}
```