SMS Verification APIs with PHP

What Is SMS Verification?

SMS verification is a method of authentication system that enables users to receive a shortcode on their mobile phones, which is used to verify their identity when logging into an application.
SMS is among the most widespread types of multi-factor authentication (MFA), requiring no apps or digital keys, making it independent of a specific ecosystem and helping to streamline various user workflows in applications.

It provides enhanced security because a hacker would need to have the user's username, password, and physical access to their phone, which likely has its own authentication, to log into an application. Although, nowadays, OTP SMS is still being hacked and hence it is very necessary to keep in mind the OTP SMS fraud prevention methods.

MESSAGE CENTRAL

Benefits of SMS Verification

Despite alternatives like FIDO2 and mobile authenticator applications gaining popularity, SMS verification remains widely used due to its benefits:

- Secure: Increases security significantly as it makes it hard for malicious actors to access user accounts.
- Ease of use: Almost everyone has access to a mobile phone today, making SMS verification a popular choice with no additional costs on software or hardware devices.
- Convenience: Reduces the burden of remembering passwords by sending a unique code to the user's mobile for verification.

Setup for SMS Verification with Message Central

Message Central is a CPaaS solution with a suite of products including OTP SMS APIs, SMS APIs and WhatsApp marketing platform.

a. Create a MessageCentral Account: Sign up for a Message Central account to get started. You will receive a customer ID, which you will use in your application.

b. To install PHP, follow these steps:

Download PHP:

Go to the PHP download page on the official PHP website: https://www.php.net/downloads.

Download the latest version of PHP for your operating system (Windows, macOS, or Linux).

Install PHP:

Windows:

Run the downloaded PHP installer.

Follow the on-screen instructions to install PHP. You may need to choose the installation directory and configure some options.

macOS/Linux:

Extract the downloaded PHP archive to a location on your system.

Configure PHP by editing the php.ini file. You may need to set up extensions and other settings based on your requirements.

MESSAGE CENTRAL

Verify Installation:

Open a command prompt (on Windows) or terminal (on macOS/Linux).

Run the following command to verify the installation:

```
php --version
```

This should print the installed PHP version. If installed correctly, you should see something like:

```
PHP x.x.x (cli) (built: xxxxxx) (...)
```

Set PATH Environment Variable (Windows):

Right-click on "This PC" or "My Computer" and select "Properties".

Click on "Advanced system settings" and then "Environment Variables".

In the "System variables" section, select the "Path" variable and click "Edit".

Add the path to the PHP installation directory (e.g., C:\PHP) if it's not already present.

Click "OK" to save the changes.

Set PATH Environment Variable (macOS/Linux):

Open a terminal and open or create the .bash_profile file in your home directory.

Add the following line, replacing /path/to/php with the actual path to your PHP installation:

```
export PATH=$PATH:/path/to/php
```

Save the file and restart the terminal for the changes to take effect.

After following these steps, you should have PHP installed and ready to use on your system.

Integration Steps

This process involves the following three significant steps:

a. Message Central Details
b. Adding Message Central Details in Code
c. Send a Test Otp for verification

MESSAGE CENTRAL

a. Message Central Details

After creating an account on Message Central, you need the following details:

- Customer Id - You can get the customer ID from Message Central Home Console
- Login Credentials: You'd require an email and would need to create a password.

b. Adding MessageCentral Details in Code

To add MessageCentral details on the code find out the variable definition with name "customerId", "email" and "password":

```
$customerID = "C-45539F39A449437";
$email = "abhishekkaushik68@gmail.com";
$password = "Wesee@123";
$baseURL = "https://cpaas-staging.messagecentral.com";
$authToken = null;
$verificationId = null;
$verificationStatus = null;
```

Now, you need to create a PHP file named Program.php and add the provided PHP code into this file.

```php
<?php

require __DIR__ . '/vendor/autoload.php';

use Http\Discovery\HttpClientDiscovery;
use Http\Discovery\Psr17FactoryDiscovery;
use Http\Discovery\Psr18ClientDiscovery;
use Psr\Http\Client\ClientExceptionInterface;
use Psr\Http\Client\ClientInterface;
use Psr\Http\Message\RequestFactoryInterface;
use Psr\Http\Message\StreamFactoryInterface;
use Psr\Http\Message\UriFactoryInterface;

$customerID = "C-45539F39A449437";
$email = "abhishekkaushik68@gmail.com";
$password = "Wesee@123";
$baseURL = "https://cpaas-staging.messagecentral.com";
$authToken = null;
$verificationId = null;
$verificationStatus = null;

function parseJSON($jsonString, $key)
{
    $jsonObject = json_decode($jsonString, true);
    return $jsonObject[$key] ?? null;
}

function generateAuthToken($httpClient, $requestFactory)
{
    global $password, $baseURL, $customerID, $email,
$authToken;

    $base64String = base64_encode($password);
    $url =
"{$baseURL}/auth/v1/authentication/token?country=IN&customerId
={$customerID}&email={$email}&key={$base64String}&scope=NEW";
    $request = $requestFactory->createRequest('GET', $url);

    try {
        $response = $httpClient->sendRequest($request);
        $responseString = $response->getBody()->getContents();
        $authToken = parseJSON($responseString, "token");
        return $authToken;
    } catch (ClientExceptionInterface $e) {
```

MESSAGE CENTRAL

```
        echo "Error generating auth token. Response code:
{$e->getCode()}\n";
        return null;
    }
}

function sendOtp($httpClient, $requestFactory, $authToken,
$countryCode, $mobileNumber)
{
    global $baseURL, $customerID, $verificationId;

    $url =
"{$baseURL}/verification/v2/verification/send?countryCode={$co
untryCode}&customerId={$customerID}&flowType=SMS&mobileNumber=
{$mobileNumber}";
    $request = $requestFactory->createRequest('POST', $url,
['authToken' => $authToken]);

    try {
        $response = $httpClient->sendRequest($request);
        $responseString = $response->getBody()->getContents();
        $verificationId = parseJSON(parseJSON($responseString,
"data"), "verificationId");
        return $verificationId;
    } catch (ClientExceptionInterface $e) {
        echo "Error sending OTP. Response code:
{$e->getCode()}\n";
        return null;
    }
}

function validateOtp($httpClient, $requestFactory, $authToken,
$otpCode, $countryCode, $mobileNumber)
{
    global $baseURL, $customerID, $verificationId,
$verificationStatus;

    $url =
"{$baseURL}/verification/v2/verification/validateOtp?countryCo
de={$countryCode}&mobileNumber={$mobileNumber}&verificationId=
{$verificationId}&customerId={$customerID}&code={$otpCode}";
    $request = $requestFactory->createRequest('GET', $url,
['authToken' => $authToken]);

    try {
        $response = $httpClient->sendRequest($request);
        $responseString = $response->getBody()->getContents();
```

MESSAGE CENTRAL

```
        $verificationStatus = parseJSON($responseString,
"verificationStatus");
        return $verificationStatus;
    } catch (ClientExceptionInterface $e) {
        echo "Error validating OTP. Response code:
{$e->getCode()}\n";
        return null;
    }
}

$httpClient = Psr18ClientDiscovery::find();
$requestFactory = Psr17FactoryDiscovery::findRequestFactory();

$server = new \React\Http\Server(function
(\Psr\Http\Message\ServerRequestInterface $request) use
($httpClient, $requestFactory) {
    global $customerID, $email, $password, $baseURL,
$authToken, $verificationId, $verificationStatus;

    $path = $request->getUri()->getPath();
    $method = $request->getMethod();
    $query = $request->getQueryParams();
    $body = json_decode($request->getBody()->getContents(),
true);

    if ($method === "POST" && $path === "/sendotp" &&
isset($body["countryCode"]) && isset($body["mobileNumber"])) {
        $countryCode = $body["countryCode"];
        $mobileNumber = $body["mobileNumber"];

        generateAuthToken($httpClient, $requestFactory);
        $verificationId = sendOtp($httpClient,
$requestFactory, $authToken, $countryCode, $mobileNumber);

        return new \React\Http\Message\Response(
            200,
            ['Content-Type' => 'application/json'],
            json_encode(["message" => $verificationId ? "Otp
sent! Successfully" : "Bad Request"])
        );
    } elseif ($method === "GET" && $path === "/validateotp" &&
isset($query["countryCode"]) && isset($query["mobileNumber"])
&& isset($query["otpCode"])) {
        $countryCode = $query["countryCode"];
        $mobileNumber = $query["mobileNumber"];
        $otpCode = $query["otpCode"];
```

MESSAGE CENTRAL

```
        generateAuthToken($httpClient, $requestFactory);
        $verificationStatus = validateOtp($httpClient,
$requestFactory, $authToken, $otpCode, $countryCode,
$mobileNumber);

        return new \React\Http\Message\Response(
            200,
            ['Content-Type' => 'application/json'],
            json_encode(["message" => $verificationStatus ===
"VERIFICATION_COMPLETED" ? "Otp verification Done!" : "Bad
Request"])
        );
    } else {
        return new \React\Http\Message\Response(
            404,
            ['Content-Type' => 'text/plain'],
            "Not Found"
        );
    }
});

$socket = new \React\Socket\Server('127.0.0.1:3000');
$server->listen($socket);

echo "Server running at http://localhost:3000/\n";

$loop = \React\EventLoop\Factory::create();
$loop->run();
```

MESSAGE CENTRAL

c. Send a Test Otp Sms for Verification

If you need to test the service without code, you can go to the free SMS verification page on the Message Central's website.

To ensure that the integration is successful, send a test OTP SMS as follows:

1. To run PHP code, you don't need to compile it. PHP is an interpreted language, so you can run PHP scripts directly.

   Here's how you can run a PHP script:
   Save your PHP code in a file with a .php extension, for example, my_script.php.
   Open a terminal or command prompt.

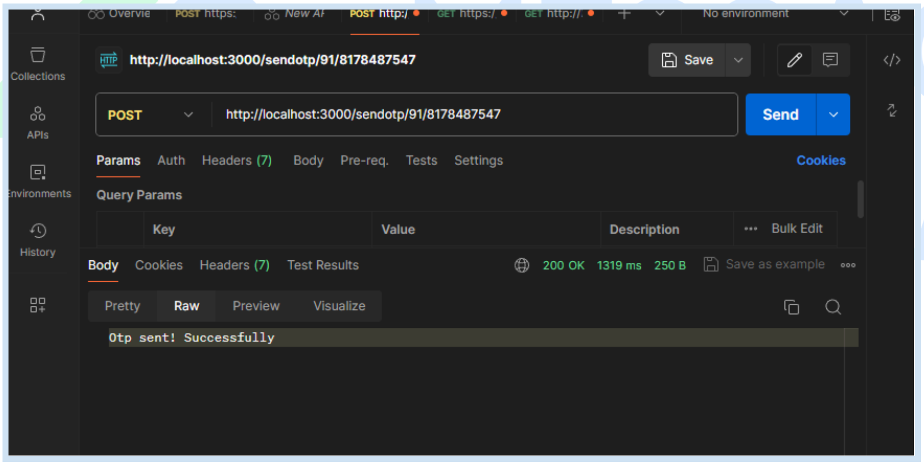
Navigate to the directory where your PHP file is located.
Run the PHP script using the PHP interpreter. For example, to run my_script.php, you would use the following command:

```
php my_script.php
```

Replace my_script.php with the name of your PHP file.
This will execute your PHP script and display any output it generates.

2. Open the Postman and set Request Method as POST and URL as `http://localhost:3000/sendotp/<countryCode>/<phone_number>`
   Here's the port 3000, it is default and defined in code, can be changed.
   Example, For Indian Phone Number URL : `http://localhost:3000/sendotp/91/123****123`

- Image 1 : Sending Otp



http://localhost:3000/sendotp/91/8178487547

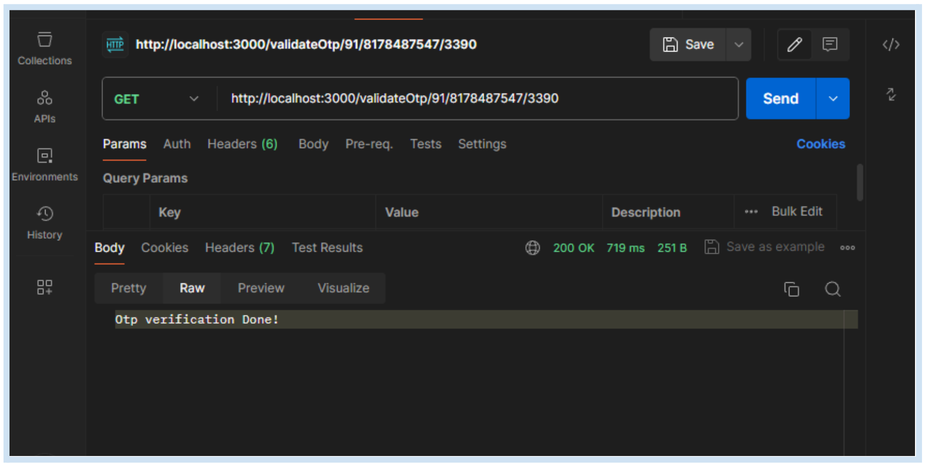
POST http://localhost:3000/sendotp/91/8178487547 Send

200 OK 1319 ms 250 B

```
Otp sent! Successfully
```

3. Now You have an otp on your sms inbox. Try your own validation Otp API to validate OTP.
   Open the Postman and set Request Method as GET and URL as `http://localhost:3000/validateOtp/<countryCode>/<phone_number>/<otp>`
   Here's the port 3000, it is default and defined in code, can be changed.
   Example, For Indian Phone NumberURL : `http://localhost:3000/validateOtp/91/123****123/****`

- Image 2: Verifying Otp



http://localhost:3000/validateOtp/91/8178487547/3390

GET http://localhost:3000/validateOtp/91/8178487547/3390

Query Params

Key	Value	Description

200 OK 719 ms 251 B

```
Otp verification Done!
```

MESSAGE CENTRAL