

PHP SDK Version 3.x.x



Zoho CRM
-zoho.com/crm-

Table of contents

1. Overview.....	3
a. Environmental Setup	
b. Including SDK in your Project	
c. Using the SDK	
2. Register your Application.....	5
3. Configurations.....	7
4. Token Persistence.....	10
a. Implementing OAuth Persistence	
b. Database Persistence	
c. File Persistence	
d. Custom Persistence	
5. Initializing the Application.....	13
a. Generating the Grant Token	
b. Initialization	
6. Class Hierarchy.....	17
7. Responses and Exception.....	18
a. For GET Requests	
b. For POST, PUT, DELETE Requests	
8. Multi-user Support.....	20
9. Sample Codes.....	25
10. Release Notes.....	45
a. Current Version	
b. Previous Version(s)	

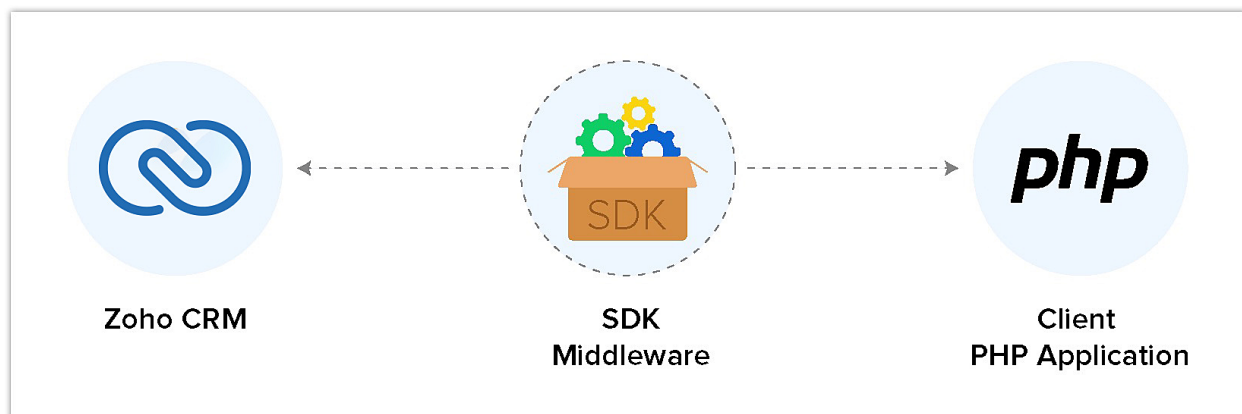


Overview

PHP SDK offers a way to create client PHP applications that can be integrated with Zoho CRM. This SDK makes the access and use of necessary CRM APIs with ease. In other words, it serves as a wrapper for the REST APIs, making it easier to use the services of Zoho CRM.

A point to note would be that the developer of the client application should create programming code elements along with interface implementations, instances or objects. Authentication to access Zoho CRM APIs is through OAuth2.0 authentication mechanism. Invariably, HTTP requests and responses are taken care of by the SDK.

A sample of how an SDK acts a middle ware or interface between Zoho CRM and a client PHP application.



Note

- For the sake of better explanation, we have used Eclipse to describe how to get started on using the SDK.
- While using PHP SDK, you must add the SSL Certificate in your system. Download the certificate bundle.
- In the PHP folder in your system, locate the php.ini file and add these lines to your certificate.

```
1 curl.cainfo="<filepath>/cacert.pem"  
2 openssl.cafile="<filepath>/cacert.pem"
```



PHP SDK allows you to

1. Exchange data between Zoho CRM and the client application where the CRM entities are modeled as classes.
2. Declare and define CRM API equivalents as simple functions in your PHP application.
3. Push data into Zoho CRM by accessing appropriate APIs of the CRM Service.

Environmental Setup

PHP SDK is installable through the composer. A composer is a tool for dependency management in PHP. SDK expects the following from the client app

- Client app must have **PHP(version 7 and above)** or above with curl extension **enabled**.
- PHP SDK must be installed into client app through composer.

Including the SDK in your project

You can include the SDK to your project using:

1. Install Composer (if not installed)
 - Run this command to install the composer.

```
a. curl -sS https://getcomposer.org/installer | php
```

- To install composer on mac/linux machine:

```
a. https://getcomposer.org/doc/00-intro.md#installation-linux-unix-osx
```

- To install composer on windows machine:

```
a. https://getcomposer.org/doc/00-intro.md#installation-windows
```

2. Install PHP SDK Here's how you install the SDK

- Navigate to the workspace of your client app.
- Run the command below:

```
1 composer require zohocrm/php-sdk:3.0.0
```

- The PHP SDK will be installed and a package named vendor will be created in the workspace of your client app.



Using the SDK

Add the below line in PHP files of your client app , where you would like to make use of PHP SDK.

```
1 require 'vendor/autoload.php'
```

Through this line, you can access all the functionalities of the PHP SDK. The namespaces of the class to be used must be included within the "use" statement.

Note

- It is mandatory for the client to have **ZohoCRM.settings.fields.ALL** to access all the record operations API. Otherwise, the system returns the **OAuth-SCOPE-MISMATCH error**
- **The access and refresh tokens are environment-specific and domain-specific.** When you handle various environments and domains such as Production, Sandbox, or Developer and IN, CN, US, EU, or AU, respectively, you must use the access token and refresh token generated only in those respective environments and domains. The SDK throws an error, otherwise.
- For example, if you generate the tokens for your Sandbox environment in the CN domain, you must use only those tokens for that domain and environment. You cannot use the tokens generated for a different environment or a domain.

Register your Application

Before you get started with authorization and make any calls using the Zoho CRM APIs, you need to register your application with Zoho CRM.

To register,

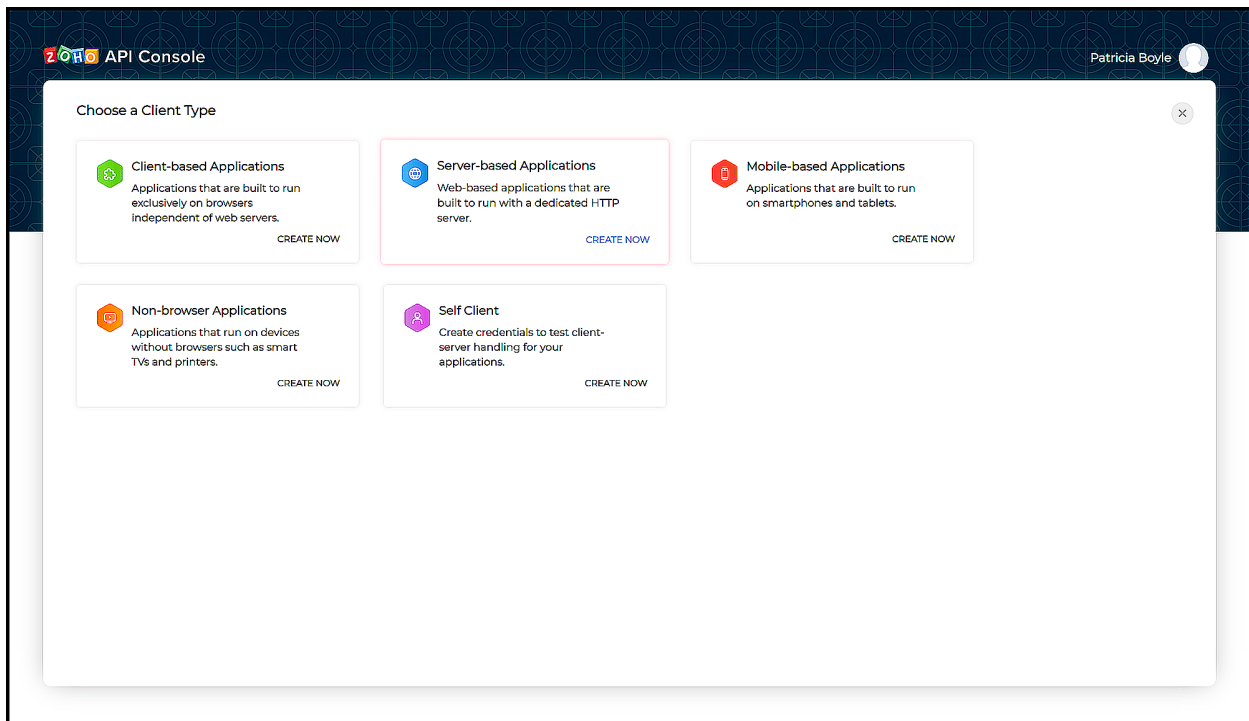
- Go to [Zoho Developer Console](#).



Zoho CRM
-zoho.com/crm-

- Choose a client type:
 - **Client-based:** Applications that are built to run exclusively on browsers independent of web servers.
 - **Server-based:** Web-based applications that are built to run with a dedicated HTTP server.
 - **Mobile:** Applications that are installed on smart phones and tablets.
 - **Non-browser Mobile Applications:** Applications for devices without browser provisioning such as smart TVs and printers.
 - **Self Client:** Stand-alone applications that perform only back-end jobs (without any manual intervention) like data sync.

For more details, refer to [OAuth Overview](#).



- Enter the following details:
 - **Client Name:** The name of your application you want to register with Zoho.
 - **Homepage URL:** The URL of your web page.
 - **Authorized Redirect URIs:** A valid URL of your application to which Zoho Accounts redirects you with a grant token(code) after successful authentication.



The screenshot shows the 'Create New Client' form in the Zoho API Console. The form is titled 'Create New Client' and has a close button (X) in the top right corner. It contains the following fields:

- Client Type:** A dropdown menu with 'Server-based Applications' selected.
- Client Name:** A text input field containing 'ABC App'.
- Homepage URL:** A text input field containing 'https://{your_domain}.com'.
- Authorized Redirect URIs:** A text input field containing 'https://{your_domain}.com/{your_redirect_page}'.

At the bottom of the form is a blue button labeled 'CREATE'.

- Click **CREATE**.
- You will receive the following credentials:
 - **Client ID:** The consumer key generated from the connected app.
 - **Client Secret:** The consumer secret generated from the connected app.

The screenshot shows the 'Client Secret' page for the 'ABC App' in the Zoho API Console. The page is titled 'ABC App' and has a close button (X) in the top right corner. It contains the following information:

- Client ID:** A text input field containing '1000'.
- Client Secret:** A text input field containing a long, alphanumeric string.

The page also has a header with 'Client Details', 'Client Secret', and 'Settings' tabs, and a sub-header with 'ABC App' and '09 March 2021'.



Note

If you don't have a domain name and a redirect URL, you can use dummy values in their place and register your client.

Configuration

Before you get started with creating your PHP application, you need to register your client and authenticate the app with Zoho.

Follow the below steps to configure the SDK.

1. Create an instance of the **Logger** Class to log exception and API information.

```
1 /*
2 * Create an instance of Logger Class that takes two parameters
3 * 1 -> Level of the log messages to be logged. Can be configured
  by typing Levels "::" and choose any level from the list
  displayed.
4 * 2 -> Absolute file path, where messages need to be logged.
5 */
6 $logger = Logger::getInstance(Levels::info,
  "/Users/user_name/Documents/php_sdk_log.log");
```

2. Create an instance of **UserSignature** that identifies the current user.

```
1 //Create an UserSignature instance that takes user Email as
  parameter
2 $user = new UserSignature("abc@zoho.com");
```

3. Configure the **API environment** which decides the domain and the URL to make API calls.

```
1 /*
2 * Configure the environment
3 * which is of the pattern Domain.Environment
4 * Available Domains: USDataCenter, EUDataCenter, INDataCenter,
  CNDataCenter, AUDataCenter
5 * Available Environments: PRODUCTION(), DEVELOPER(), SANDBOX()
```




```
6 */
7 $environment = USDataCenter::PRODUCTION();
```

4. Create an instance of **OAuthToken** with the information that you get after registering your Zoho client.

```
1 /*
2 * Create a Token instance
3 * 1 -> OAuth client id.
4 * 2 -> OAuth client secret.
5 * 3 -> OAuth redirect URL.
6 * 4 -> REFRESH/GRANT token.
7 * 5 -> Token type(REFRESH/GRANT).
8 */
9 $token = new OAuthToken("clientId", "clientSecret",
    "REFRESH/GRANT token", TokenType::REFRESH/GRANT, "redirectURL");
```

5. Create an instance of **TokenStore** to persist tokens used for authenticating all the requests.

```
1 /*
2 * Create an instance of DBStore.
3 * 1 -> DataBase host name. Default value "localhost"
4 * 2 -> DataBase name. Default value "zohooauth"
5 * 3 -> DataBase user name. Default value "root"
6 * 4 -> DataBase password. Default value ""
7 * 5 -> DataBase port number. Default value "3306"
8 */
9 $tokenstore = new DBStore("hostName", "DataBaseName",
    "userName", "password", "portNumber");
10 // $tokenstore = new FileStore("absolute_file_path");
11 // $tokenStore = new CustomStore();
```

6. Create an instance of **SDKConfig** containing the SDK configuration.

```
1 /*
2 * autoRefreshFields (default value is false)
3 * true - all the modules' fields will be auto-refreshed in the
    background, every hour.
```



```

4 * false - the fields will not be auto-refreshed in the
  background. The user can manually delete the file(s) or refresh
  the fields using methods from
  ModuleFieldsHandler(com\zoho\crm\api\util\ModuleFieldsHandler)
5 *
6 * pickListValidation (default value is true)
7 * A boolean field that validates user input for a pick list field
  and allows or disallows the addition of a new value to the list.
8 * true - the SDK validates the input. If the value does not exist
  in the pick list, the SDK throws an error.
9 * false - the SDK does not validate the input and makes the API
  request with the user's input to the pick list
10 *
11 * enableSSLVerification (default value is true)
12 * A boolean field to enable or disable curl certificate
  verification
13 * true - the SDK verifies the authenticity of certificate
14 * false - the SDK skips the verification
15 */
16
17 $autoRefreshFields = false;
18 $pickListValidation = false;
19 $enableSSLVerification = true;
20 $connectionTimeout = 2; //The number of seconds to wait while
  trying to connect. Use 0 to wait indefinitely.
21
22 $timeout = 2; //The maximum number of seconds to allow cURL
  functions to execute.
23
24 $sdkConfig = (new SDKConfigBuilder())-
  >setAutoRefreshFields($autoRefreshFields)-
  >setPickListValidation($pickListValidation)-
  >setSSLVerification($enableSSLVerification)-
  >connectionTimeout($connectionTimeout)->timeout($timeout)-
  >build();

```

7. Set the absolute directory path to store user specific files containing module fields information in **resourcePath**



```
1 $resourcePath = "/Users/user_name/Documents/phpsdk-application";
```

8. Create an instance of **RequestProxy** containing the proxy properties of the user.

```
1 $requestProxy = new RequestProxy("proxyHost", "proxyPort",  
    "proxyUser", "password");
```

9. [Initialize](#) the SDK and make API calls.

Token Persistence

Token persistence refers to storing and utilizing the authentication tokens that are provided by Zoho. There are three ways provided by the SDK in which persistence can be applied. They are file persistence, DB persistence (default) and Custom persistence.

Implementing OAuth Persistence

Once the application is authorized, OAuth access and refresh tokens can be used for subsequent user data requests to Zoho CRM. Hence, they need to be persisted by the client app.

The persistence is achieved by writing an implementation of the inbuilt TokenStore interface, which has the following callback methods.

- **getToken(\$user, \$token)** - invoked before firing a request to fetch the saved tokens. This method should return implementation Token interface object for the library to process it.
- **saveToken(\$user, \$token)** - invoked after fetching access and refresh tokens from Zoho.
- **deleteToken(\$token)** - invoked before saving the latest tokens.
- **getTokens()** - The method to retrieve all the stored tokens.
- **deleteTokens()** - The method to delete all the stored tokens.

Note

- \$user instance of UserSignature class
- \$token instance of Token interface implementing class.



There are three ways provided by the SDK in which you can achieve persistence. They are:

- Database Persistence
- File Persistence
- Custom Persistence

Database Persistence

If you want to use database persistence, you can use MySQL. The DB persistence mechanism is the default method.

- The database name should be zohooauth.
- There must be a table oauthtokens with columns
 - **id**(int(11))
 - **user_mail** (varchar(255))
 - **client_id** (varchar(255))
 - **refresh_token** (varchar(255))
 - **grant_token** (varchar(255))
 - **access_token** (varchar(255))
 - **expiry_time**(varchar(20))

MySQL Query

```
1 create table oauthtoken(id int(11) not null auto_increment,
  user_mail varchar(255) not null, client_id varchar(255),
  refresh_token varchar(255), access_token varchar(255),
  grant_token varchar(255), expiry_time varchar(20), primary key
  (id));
2 alter table oauthtoken auto_increment = 1;
```

Here is the code to create a DBStore object:

```
1 /*
2 * 1 -> DataBase host name. Default value "localhost"
3 * 2 -> DataBase name. Default value "zohooauth"
4 * 3 -> DataBase user name. Default value "root"
5 * 4 -> DataBase password. Default value ""
6 * 5 -> DataBase port number. Default value "3306"
```



```
7 */
8 $tokenstore = new DBStore();
9 $tokenstore = new DBStore("hostName", "DataBaseName",
    "userName", "password", "portNumber");
```

File Persistence

In case of file persistence, you can set up persistence the tokens in the local drive, and provide the absolute file path in the FileStore object. This file must contain the following:

- **user_mail**
- **client_id**
- **refresh_token**
- **access_token**
- **grant_token**
- **expiry_time**

Here is the code to create a FileStore object:

```
1 //Parameter containing the absolute file path to store
  tokens
2 $tokenstore = new
  FileStore("/Users/username/Documents/php_sdk_token.txt");
```

Custom Persistence

To use Custom Persistence, you must implement the TokenStore interface (com\zoho\api\authenticator\store\TokenStore) and override the methods.

Here is the code:

```
1 namespace store;
2 use com\zoho\api\authenticator\Token;
3 use com\zoho\crm\api\exception\SDKException;
4 use com\zoho\crm\api\UserSignature;
5 use com\zoho\api\authenticator\store\TokenStore;
6 class CustomStore implements TokenStore
```



```

7 {
8     /**
9         * @param user A UserSignature class instance.
10        * @param token A Token
11        (com\zoho\api\authenticator\OAuthToken) class instance.
12        * @return A Token class instance representing the
13        user token details.
14        * @throws SDKException if any problem occurs.
15    */
16    public function getToken($user, $token)
17    {
18        // Add code to get the token
19        return null;
20    }
21    /**
22        * @param user A UserSignature class instance.
23        * @param token A Token
24        (com\zoho\api\authenticator\OAuthToken) class instance.
25        * @throws SDKException if any problem occurs.
26    */
27    public function saveToken($user, $token)
28    {
29        // Add code to save the token
30    }
31    /**
32        * @param token A Token
33        (com\zoho\api\authenticator\OAuthToken) class instance.
34        * @throws SDKException if any problem occurs.
35    */
36    public function deleteToken($token)
37    {
38        // Add code to delete the token
39    }
40    /**
41        * @return array An array of Token
42        (com\zoho\api\authenticator\OAuthToken) class instances

```



```
38  */
39  public function getTokens()
40  {
41      //Add code to retrieve all the stored tokens
42  }
43  public function deleteTokens()
44  {
45      //Add code to delete all the stored tokens.
46  }
47 }
```

Initializing the Application

To access the CRM services through the SDK, you must first authenticate your client app.

Generating the grant token

For a Single User

The developer console has an option to generate grant token for a user directly. This option may be handy when your app is going to use only one CRM user's credentials for all its operations or for your development testing.

1. Login to your Zoho account.
2. Visit <https://api-console.zoho.com>
3. Click Self Client option of the client for which you wish to authorize.
4. Enter one or more (comma-separated) valid Zoho CRM scopes that you wish to authorize in the "Scope" field and choose the time of expiry.
5. Copy the grant token that is displayed on the screen.

Note

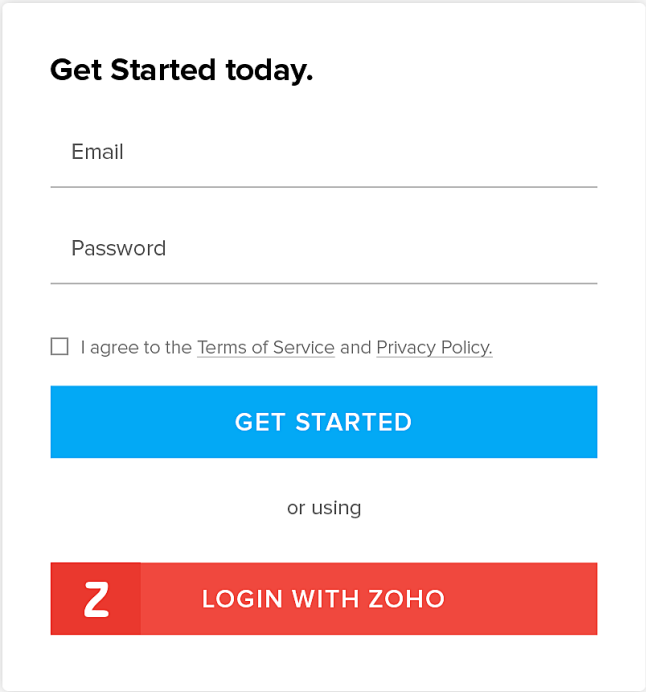
- The generated grant token is valid only for the stipulated time you chose while generating it. Hence, the access and refresh tokens should be generated within that time.
- The OAuth client registration and grant token generation must be done in the same Zoho account's (meaning - login) developer console.



For Multiple Users

For multiple users, it is the responsibility of your client app to generate the grant token from the users trying to login.

- Your Application's UI must have a "Login with Zoho" option to open the grant token URL of Zoho, which would prompt for the user's Zoho login credentials.



Get Started today.

Email

Password

I agree to the [Terms of Service](#) and [Privacy Policy](#).

GET STARTED

or using

Z **LOGIN WITH ZOHO**

- Upon successful login of the user, the grant token will be sent as a param to your registered redirect URL.

Note

- **The access and refresh tokens are environment-specific and domain-specific.** When you handle various environments and domains such as Production, Sandbox, or Developer and IN, CN, US, EU, or AU, respectively, you must use the access token and refresh token generated only in those respective environments and domains. The SDK throws an error, otherwise.
- For example, if you generate the tokens for your Sandbox environment in the CN domain, you must use only those tokens for that domain and environment. You cannot use the tokens generated for a different environment or a domain.

Initialization

You must pass the following details to the SDK and initialize it before you can make API calls.

1. **UserSignature** - The email ID of the user that is making the API calls. The tokens are also specific to this user.
2. **Environment** - The environment such as Production, Developer, or Sandbox from which you want to make API calls. This instance also takes the domain (data center) in which the tokens are generated. The format is `USDataCenter::PRODUCTION()`, `EUDataCenter::SANDBOX()` and so on.
3. **Token** - The grant or refresh token. The token must be specific to the user that makes the call, and specific to the org and the environment the token was generated in. Besides the token, the token instance also takes the client ID, client secret, and the redirect URI as its parameters.
4. **Tokenstore** - The token persistence method. The possible methods are DB persistence and file persistence. For file persistence, you must specify the absolute file path to the file where you want to store the tokens. For DB persistence, you must specify the host, database name, user name, password and the port at which the server runs.
5. **Logger** - To log the messages. You can choose the level of logging of messages through `Logger.Levels`, and provide the absolute file path to the file where you want the SDK to write the messages in.
6. **SDKConfig** - The class that contains the values of `autoRefresh` and `pickListValidation` fields.
7. **resourcePath** - The absolute directory path to store user-specific files containing information about the fields of a module.
8. **RequestProxy** - An instance containing the proxy details of the request.

Note

- From version 3.x.x, initialization of the SDK happens through the `Initializer` class. This class contains instances of the current user, environment, token, tokenstore, and logger.
- Initializing the SDK does not generate a token. A token is generated only when you make an API call.



```

1 <?php
2 use com\zoho\api\authenticator\OAuthToken;
3 use com\zoho\api\authenticator\TokenType;
4 use com\zoho\api\authenticator\store\DBStore;
5 use com\zoho\api\authenticator\store\FileStore;
6 use com\zoho\crm\api\Initializer;
7 use com\zoho\crm\api\UserSignature;
8 use com\zoho\crm\api\dc\USDataCenter;
9 use com\zoho\api\logger\Logger;
10 use com\zoho\api\logger\Levels;
11 use com\zoho\crm\api\SDKConfigBuilder;
12 class Initialize
13 {
14     public static function initialize()
15     {
16         /*
17          * Create an instance of Logger Class that takes two
18          parameters
19          * 1 -> Level of the log messages to be logged. Can be
20          configured by typing Levels "." and choose any level from the
21          list displayed.
22          * 2 -> Absolute file path, where messages need to be
23          logged.
24          */
25         $logger = Logger::getInstance(Levels::INFO,
26             "/Users/user_name/Documents/php_sdk_log.log");
27
28         //Create an UserSignature instance that takes user Email
29         as parameter
30         $user = new UserSignature("abc@zoho.com");
31
32         /*
33          * Configure the environment
34          * which is of the pattern Domain.Environment
35          * Available Domains: USDataCenter, EUDataCenter,
36          INDataCenter, CNDataCenter, AUDataCenter
37          * Available Environments: PRODUCTION, DEVELOPER, SANDBOX
38          */
39         $environment = USDataCenter::PRODUCTION();
40     }
41 }

```



```

34         * Create a Token instance
35         * 1 -> OAuth client id.
36         * 2 -> OAuth client secret.
37         * 3 -> OAuth redirect URL.
38         * 4 -> REFRESH/GRANT token.
39         * 5 -> Token type(REFRESH/GRANT).
40     */
41     $token = new OAuthToken("clientId", "clientSecret",
    "REFRESH/GRANT token", TokenType::REFRESH/GRANT, "redirectURL");
42     /*
43     * Create an instance of DBStore.
44     * 1 -> DataBase host name. Default value "localhost"
45     * 2 -> DataBase name. Default value "zohooauth"
46     * 3 -> DataBase user name. Default value "root"
47     * 4 -> DataBase password. Default value ""
48     * 5 -> DataBase port number. Default value "3306"
49     */
50     //$tokenstore = new DBStore();
51     $tokenstore = new DBStore("hostName", "dataBaseName",
    "userName", "password", "portNumber");
52     // $tokenstore = new FileStore("absolute_file_path");
53     $autoRefreshFields = false;
54
55     $pickListValidation = false;
56
57     // Create an instance of SDKConfig
58     $connectionTimeout = 2; //The number of seconds to wait
    while trying to connect. Use 0 to wait indefinitely.
59     $timeout = 2; //The maximum number of seconds to
    allow curl functions to execute.
60     $sdkConfig = (new SDKConfigBuilder())-
    >setAutoRefreshFields($autoRefreshFields)-
    >setPickListValidation($pickListValidation)-
    >setSSLVerification($enableSSLVerification)-
    >connectionTimeout($connectionTimeout)->timeout($timeout)-
    >build();
61
62     $resourcePath = "/Users/user_name/Documents/phpsdk-
    application";
63

```



```

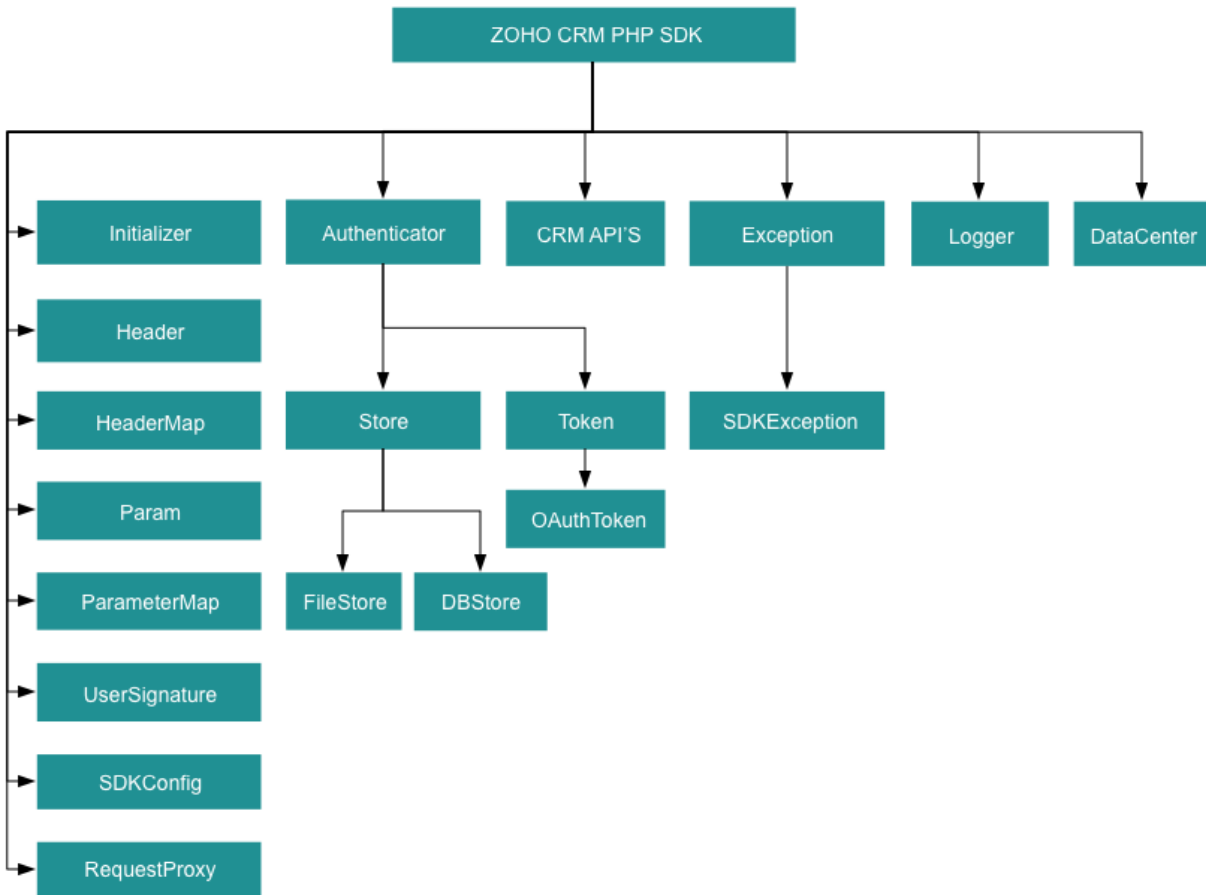
64     /*
65     * Call static initialize method of Initializer class that
    takes the arguments
66     * 1 -> UserSignature instance
67     * 2 -> Environment instance
68     * 3 -> Token instance
69     * 4 -> TokenStore instance
70     * 5 -> autoRefreshFields
71     * 6 -> The path containing the absolute directory path to
    store user specific JSON files containing module fields
    information.
72     * 7 -> Logger instance
73     */
74     Initializer::initialize($user, $environment, $token,
    $tokenstore, $sdkConfig, $resourcePath, $logger);
75
76     // $tokenstore->deleteTokens();
77 }
78 }
79 ?>

```

Class Hierarchy

All Zoho CRM entities are modeled as classes having members and methods applicable to that particular entity. The class hierarchy of various Zoho CRM entities in the PHP SDK is depicted in the following image.





Responses and Exceptions

APIResponse<ResponseHandler> and **APIResponse<ActionHandler>** are the wrapper objects for Zoho CRM APIs' responses. All API calling methods would return one of these two objects.

Use the **getObject()** method to obtain the response handler interface depending on the type of request (GET, POST,PUT,DELETE).

Whenever the API returns an error response, the response will be an instance of **APIException class**.

The following are the wrappers with their handlers for the respective APIs:

- Tag API's record and count tag operation:



Zoho CRM
-zoho.com/crm-

-APIResponse<RecordActionHandler>

-APIResponse<CountHandler>

- BaseCurrency operation:
-APIResponse<BaseCurrencyActionHandler>
- Lead convert operation:
-APIResponse<ConvertActionHandler>
- Deleted record operation:
-APIResponse<DeletedRecordsHandler>
- Download record's photo operation:
-APIResponse<DownloadHandler>
- MassUpdate record operation:
-APIResponse<MassUpdateActionHandler>
-APIResponse<MassUpdateResponseHandler>

For GET Requests

- **APIResponse.getObject()** returns the response handler interface.
- This ResponseHandler interface encompasses the **ResponseWrapper class** (for application/json responses), **file body wrapper class** (for file download responses), and the **APIException class**.
- This CountHandler interface encompasses the **CountWrapper class** (for application/json responses) and the **APIException class**.
- This DeletedRecordsHandler interface encompasses the **DeletedRecordsWrapper class** (for application/json responses) and the **APIException class**.
- This DownloadHandler interface encompasses the **FileBodyWrapper class** (for file download responses responses) and the **APIException class**.
- This MassUpdateResponseHandler interface encompasses the **MassUpdateResponseWrapper class** (for application/json responses) and the **APIException class**.



For POST, PUT, DELETE Requests

- **APIResponse.getObject()** returns the ActionHandler interface.
- The ActionHandler interface encompasses the **ActionWrapper class** (for application/json responses) and the **APIException class**.
- The ApiResponse interface encompasses the **SuccessResponse class** (for application/json responses) and the **APIException class**.
- The RecordActionHandler interface encompasses the **RecordActionWrapper class** (for application/json responses), and the **APIException class**.
- The BaseCurrencyActionHandler interface encompasses the **BaseCurrencyActionWrapper class** (for application/json responses), and the **APIException class**.
- The MassUpdateActionHandler interface encompasses the **MassUpdateActionWrapper class** (for application/json responses), and the **APIException class**.
- The ConvertActionHandler interface encompasses the **ConvertActionWrapper class** (for application/json responses), and the **APIException class**.
- If the root key of the response is not "data" (errors such as Internal Server Error), then the **ActionResponse interface** with either the **SuccessResponse class** or **APIException class** is returned.

All other exceptions such as SDK anomalies and other unexpected behaviors are thrown under the **SDKException class**.

Multi-User Support

The PHP SDK (from version 3.x.x) supports both single user and a multi-user app.

Multi-user App

Multi-users functionality is achieved using Initializer's static switchUser().

```
1Initializer::switchUser($user, $environment, $token,  
    $sdkConfig, $proxy);
```

To Remove a user's configuration in SDK. Use the below code

```
1Initializer::removeUserConfiguration($user, $environment);
```



```

1 <?php
2 namespace multiuser;
3 use com\zoho\api\authenticator\OAuthToken;
4 use com\zoho\api\authenticator\TokenType;
5 use com\zoho\api\authenticator\store\FileStore;
6 use com\zoho\crm\api\Initializer;
7 use com\zoho\crm\api\UserSignature;
8 use com\zoho\crm\api\SDKConfigBuilder;
9 use com\zoho\crm\api\dc\USDataCenter;
10 use com\zoho\crm\api\logger\Logger;
11 use com\zoho\crm\api\logger\Levels;
12 use com\zoho\crm\api\HeaderMap;
13 use com\zoho\crm\api\ParameterMap;
14 use com\zoho\crm\api\record\RecordOperations;
15 use com\zoho\crm\api\record\GetRecordsHeader;
16 use com\zoho\crm\api\record\GetRecordsParam;
17 use com\zoho\crm\api\dc\EUDataCenter;
18 require_once 'vendor/autoload.php';
19 class MultiThread
20 {
21     public function main()
22     {
23         $logger = Logger::getInstance(Levels::INFO,
24             "/Users/user_name/Documents/php_sdk_log.log");
25         $environment1 = USDataCenter::PRODUCTION();
26         $user1 = new UserSignature("abc1@zoho.com");
27         $tokenstore = new
28             FileStore("/Users/user_name/Documents/php_sdk_token.txt");
29         $token1 = new OAuthToken("clientId1",
30             "clientSecret1","REFRESH/GRANT token",
31             TokenType.REFRESH/GRANT,"redirectURL1");
32         $autoRefreshFields = false;
33         $pickListValidation = false;
34         // Create an instance of SDKConfig
35         $sdkConfig = new SDKConfigBuilder()-
36             >setAutoRefreshFields($autoRefreshFields)-
37             >setPickListValidation($pickListValidation)->build();
38         $resourcePath = "/Users/user_name/Documents/phpsdk-
39             application";
40         Initializer::initialize($user1, $environment1, $token1,

```




```

$tokenstore, $sdkConfig, $resourcePath, $logger);
34     $this->getRecords("Leads");
35     $environment2 = EUDataCenter::PRODUCTION();
36     $user2 = new UserSignature("abc2@zoho.eu");
37     $token2 = new OAuthToken("clientId2",
"clientSecret2","REFRESH/GRANT token",
TokenType.REFRESH/GRANT,"redirectURL2");
38     $autoRefreshFields = false;
39     Initializer::switchUser($user2, $environment2, $token2,
$sdkConfig);
40     // Initializer::removeUserConfiguration($user1,
$environment1);
41     $this->getRecords("Students");
42     $autoRefreshFields = false;
43     Initializer::switchUser($user1, $environment1, $token1,
$sdkConfig);
44     $this->getRecords("Contacts");
45 }
46 public function getRecords($moduleAPIName)
47 {
48     try
49     {
50         $recordOperations = new RecordOperations();
51         $paramInstance = new ParameterMap();
52         $paramInstance->add(GetRecordsParam::approved(),
"false");
53         $headerInstance = new HeaderMap();
54         $ifmodifiedsince = date_create("2020-06-
02T11:03:06+05:30")->setTimezone(new
\DateTimeZone(date_default_timezone_get()));
55         $headerInstance-
>add(GetRecordsHeader::IfModifiedSince(), $ifmodifiedsince);
56         //Call getRecord method that takes paramInstance,
moduleAPIName as parameter
57         $response = $recordOperations-
>getRecords($moduleAPIName,$paramInstance, $headerInstance);
58         echo($response->getStatusCode() . "\n");
59         print_r($response->getObject());
60         echo("\n");
61     }

```



```

62         catch (\Exception $e)
63         {
64             print_r($e);
65         }
66     }
67 }
68 $obj = new MultiThread();
69 $obj->main();
70 ?>

```

1. The program execution starts from **main()**.
2. The details of **user1** is given in the variables **user1**, **token1**, **environment1**.
3. Similarly, the details of another user **user2** is given in the variables **user2**, **token2**, **environment2**.
4. Then, the **Switch user** function is used to switch between the **User 1** and **User 2** as required.
5. Based on the latest switched user, the **\$this->getRecords(\$moduleAPIName)** will fetch record.

SDK Sample Code

```

1 <?php
2 namespace index;
3 use com\zoho\api\authenticator\OAuthToken;
4 use com\zoho\api\authenticator\TokenType;
5 use com\zoho\api\authenticator\store\DBStore;
6 use com\zoho\api\authenticator\store\FileStore;
7 use com\zoho\crm\api\Initializer;
8 use com\zoho\crm\api\UserSignature;
9 use com\zoho\crm\api\SDKConfigBuilder;
10 use com\zoho\crm\api\dc\USDataCenter;
11 use com\zoho\crm\api\logger\Logger;
12 use com\zoho\crm\api\logger\Levels;
13 use com\zoho\crm\api\record\RecordOperations;
14 use com\zoho\crm\api\HeaderMap;
15 use com\zoho\crm\api\ParameterMap;
16 use com\zoho\crm\api\record\GetRecordsHeader;
17 use com\zoho\crm\api\record\GetRecordsParam;
18 use com\zoho\crm\api\record\ResponseWrapper;

```



```

19 require_once 'vendor/autoload.php';
20 class Record
21 {
22     public static function getRecord()
23     {
24         /*
25             * Create an instance of Logger Class that takes two
                parameters
26             * 1 -> Level of the log messages to be logged. Can be
                configured by typing Levels "::" and choose any level from the
                list displayed.
27             * 2 -> Absolute file path, where messages need to be
                logged.
28         */
29         $logger = Logger::getInstance(Levels::INFO,
"/Users/user_name/Documents/php_sdk_log.log");
30         //Create an UserSignature instance that takes user Email
                as parameter
31         $user = new UserSignature("abc@zoho.com");
32         /*
33             * Configure the environment
34             * which is of the pattern Domain.Environment
35             * Available Domains: USDataCenter, EUDataCenter,
                INDataCenter, CNDataCenter, AUDataCenter
36             * Available Environments: PRODUCTION(), DEVELOPER(),
                SANDBOX()
37         */
38         $environment = USDataCenter::PRODUCTION();
39         //Create a Token instance
40         $token = new OAuthToken("clientId", "clientSecret",
                "redirectURL", "REFRESH/GRANT token", TokenType.REFRESH/GRANT);
41         //Create an instance of TokenStore
42         // $tokenstore = new DBStore();
43         $tokenstore = new
                FileStore("/Users/user_name/Documents/php_sdk_token.txt");
44         $autoRefreshFields = false;
45         $pickListValidation = false;
46         // Create an instance of SDKConfig
47         $sdkConfig = new SDKConfigBuilder()-
                >setAutoRefreshFields($autoRefreshFields)-

```



```

    >setPickListValidation($pickListValidation)->build();
48     $resourcePath ="/Users/user_name/Documents/phpsdk-
application";
49     /*
50     * Call static initialize method of Initializer class that
takes the following arguments
51     * 1 -> UserSignature instance
52     * 2 -> Environment instance
53     * 3 -> Token instance
54     * 4 -> TokenStore instance
55     * 5 -> SDKConfig instance
56     * 6 -> resourcePath -A String
57     * 7 -> Log instance (optional)
58     * 8 -> RequestProxy instance (optional)
59     */
60     Initializer::initialize($user, $environment, $token,
$tokenstore, $sdkConfig, $resourcePath, $logger);
61     try
62     {
63         $recordOperations = new RecordOperations();
64         $paramInstance = new ParameterMap();
65         $paramInstance->add(GetRecordsParam::approved(),
"both");
66         $headerInstance = new HeaderMap();
67         $ifmodifiedsince = date_create("2020-06-
02T11:03:06+05:30")->setTimezone(new
\DateTimeZone(date_default_timezone_get()));
68         $headerInstance-
>add(GetRecordsHeader::IfModifiedSince(), $ifmodifiedsince);
69         $moduleAPIName = "Leads";
70         //Call getRecord method that takes paramInstance,
moduleAPIName as parameter
71         $response = $recordOperations-
>getRecords($moduleAPIName,$paramInstance, $headerInstance);
72         if($response != null)
73         {
74             //Get the status code from response
75             echo("Status Code: " . $response->getStatusCode()
. "\n");
76             //Get object from response

```



```

77     $responseHandler = $response->getObject();
78     if($responseHandler instanceof ResponseWrapper)
79     {
80         //Get the received ResponseWrapper instance
81         $responseWrapper = $responseHandler;
82         //Get the list of obtained Record instances
83         $records = $responseWrapper->getData();
84         if($records != null)
85         {
86             $recordClass =
127         'com\zoho\crm\api\record\Record';
87             foreach($records as $record)
88             {
89                 //Get the ID of each Record
90                 echo("Record ID: " . $record->getId()
127         . "\n");
91                 //Get the createdBy User instance of
127         each Record
92                 $createdBy = $record->getCreatedBy();
93                 //Check if createdBy is not null
94                 if($createdBy != null)
95                 {
96                     //Get the ID of the createdBy
127         User
97                     echo("Record Created By User-ID:
127         " . $createdBy->getId() . "\n");
98                     //Get the name of the createdBy
127         User
99                     echo("Record Created By User-
127         Name: " . $createdBy->getName() . "\n");
100                     //Get the Email of the createdBy
127         User
101                     echo("Record Created By User-
127         Email: " . $createdBy->getEmail() . "\n");
102                 }
103                 //Get the CreatedTime of each Record
104                 echo("Record CreatedTime: ");
105                 print_r($record->getCreatedTime());
106                 echo("\n");
107                 //Get the modifiedBy User instance

```



```

of each Record
108             $modifiedBy = $record-
               >getModifiedBy();
109             //Check if modifiedBy is not null
110             if($modifiedBy != null)
111             {
112                 //Get the ID of the modifiedBy
               User
113                 echo("Record Modified By User-
               ID: " . $modifiedBy->getId() . "\n");
114                 //Get the name of the modifiedBy
               User
115                 echo("Record Modified By User-
               Name: " . $modifiedBy->getName() . "\n");
116                 //Get the Email of the
               modifiedBy User
117                 echo("Record Modified By User-
               Email: " . $modifiedBy->getEmail() . "\n");
118             }
119             //Get the ModifiedTime of each
               Record
120             echo("Record ModifiedTime: ");
121             print_r($record->getModifiedTime());
122             print_r("\n");
123             //Get the list of Tag instance each
               Record
124             $tags = $record->getTag();
125             //Check if tags is not null
126             if($tags != null)
127             {
128                 foreach($tags as $tag)
129                 {
130                     //Get the Name of each Tag
131                     echo("Record Tag Name: " .
               $tag->getName() . "\n");
132                     //Get the Id of each Tag
133                     echo("Record Tag ID: " .
               $tag->getId() . "\n");
134                 }
135             }

```



```

136 //To get particular field value
137 echo("Record Field Value: " .
    $record->getKeyValue("Last_Name") . "\n");// FieldApiName
138 echo("Record KeyValues : \n" );
139 //Get the KeyValue map
140 foreach($record->getKeyValues() as
    $keyName => $value)
141 {
142     echo("Field APIName" . $keyName
    . " \tValue : ");
143     print_r($value);
144     echo("\n");
145 }
146 }
147 }
148 }
149 }
150 }
151 catch (\Exception $e)
152 {
153     print_r($e);
154 }
155 }
156 }
157 Record::getRecord();
158 ?>

```



Record Response

```
Status Code: 200
Record ID: [REDACTED]
Record Created By User-ID: [REDACTED]
Record Created By User-Name: [REDACTED]
Record Created By User-Email: [REDACTED]
Record CreatedTime: DateTime Object
(
  [date] => 2020-07-17 10:30:59.000000
  [timezone_type] => 3
  [timezone] => UTC
)

Record Modified By User-ID: [REDACTED]
Record Modified By User-Name: [REDACTED]
Record Modified By User-Email: [REDACTED]
Record ModifiedTime: DateTime Object
(
  [date] => 2020-07-19 08:39:18.000000
  [timezone_type] => 3
  [timezone] => UTC
)

Record Tag Name: Testtask
Record Tag ID: [REDACTED]
Record Field Value: Last Name
Record KeyValues :
Record ID: [REDACTED]
Record Created By User-ID: [REDACTED]
Record Created By User-Name: [REDACTED]
Record Created By User-Email: [REDACTED]
```

Print X Lite

Sample Codes

All of Zoho CRM's APIs can be used through the PHP SDK, to enable your custom application to perform data sync to the best degree. Here are the sample codes for all the API methods available in our SDK.

Attachment Operations

Constructor	Description
AttachmentsOperations(\$moduleAPIName, \$recordId)	Creates an AttachmentsOperations class instance with the moduleAPIName and recordId.



Method	Description
getAttachments	To fetch the list of attachments of a record.
uploadAttachments	To upload attachments to a record.
deleteAttachments	To delete the attachments that were added to a record.
deleteAttachment	To delete an attachment that was added to a record.
downloadAttachment	To download an attachment that was uploaded to a record.
uploadLinkAttachments	To upload a link as an attachment to a record.

Blueprint Operations

Constructor	Description
BluePrintOperations(\$recordId, \$moduleAPIName)	Creates a BluePrintOperations class instance with the recordId and moduleAPIName.

Method	Description
--------	-------------



getBlueprint	To get the next available transitions for that record, fields available for each transition, current value of each field, and validation(if any).
updateBlueprint	To update a single transition at a time.

Bulk Read Operations

Method	Description
createBulkReadJob	To schedule a bulk read job to export records that match the criteria.
getBulkReadJobDetails	To know the status of the bulk read job scheduled previously.
downloadResult	To download the result of the bulk read job. The response contains a zip file. Extract it to get the CSV or ICS file depending on the "file_type" you specified while creating the bulk read job

Bulk Write Operations

Method	Description
uploadFile	To upload a CSV file in ZIP format. The response contains the "file_id". Use this ID while making the bulk write request.



createBulkWriteJob	To create a bulk write job to insert, update, or upsert records. The response contains the "job_id". Use this ID while getting the status of the scheduled bulk write job.
getBulkWriteJobDetails	To know the status of the bulk write job scheduled previously.
downloadResult	To download the result of the bulk write job. The response contains a zip file. Extract it to get the CSV or ICS file depending on the "file_type" you specified while creating the write read job

Contact Roles Operations

Method	Description
getContactRoles	To get the list of all contact roles.
createContactRoles	To create contact roles.
updateContactRoles	To update contact roles.
deleteContactRoles	To delete contact roles.
getContactRole	To get specific contact role.
updateContactRole	To update specific contact role.



deleteContactRole	To delete specific contact role
-----------------------------------	---------------------------------

Currencies Operations

Method	Description
getCurrencies	To get the list of all currencies available for your org.
addCurrencies	To add new currencies to your org.
updateCurrencies	To update the currencies' details of your org.
enableMultipleCurrencies	To enable multiple currencies for your org.
updateBaseCurrency	To update the base currency details of your org.

Custom View Operations

Constructor	Description
CustomViewsOperations(\$module)	Creates a CustomViewsOperations class instance with the moduleAPIName

Method	Description
	To get the list of all custom views in a



getCustomViews	module.
getCustomView	To get the details of specific custom view in a module.

Fields Metadata Operations

Constructor	Description
FieldsOperations(\$module)	Creates a FieldsOperations class instance with the module.

Method	Description
getFields	To get the meta details of all fields in a module.
getField	To get the meta details of specific field in a module.

Files Operations

Method	Description
uploadFiles	To upload files and get their encrypted IDs.
getFile	To get the uploaded file through its



	encrypted ID.
--	---------------

Layouts Operations

Constructor	Description
LayoutsOperations(\$module)	Creates a LayoutsOperations class instance with the moduleAPIName.

Method	Description
getLayouts	To get the details of all the layouts in a module.
getLayout	To get the details (metadata) of a specific layout in a module.

Modules Operations

Method	Description
getModules	To get the details of all the modules.
getModule	To get the details (metadata) of a specific module.
updateModuleByAPIName	To update the details of a module by its



	module API name.
updateModuleById	To update the details of a module by its ID.

Notes Operations

Method	Description
getNotes	To get the list of notes of a record.
createNotes	To add new notes to a record.
updateNotes	To update the details of the notes of a record.
deleteNotes	To delete the notes of a record.
getNote	To get the details of a specific note.
updateNote	To update the details of an existing note.
deleteNote	To delete a specific note.

Notification Operations

Method	Description
enableNotifications	To enable instant notifications of actions



	performed on a module.
getNotificationDetails	To get the details of the notifications enabled by the user.
updateNotifications	To update the details of the notifications enabled by a user. All the provided details would be persisted and rest of the details would be removed.
updateNotification	To update only specific details of a specific notification enabled by the user. All the provided details would be persisted and rest of the details will not be removed.
disableNotifications	To stop all the instant notifications enabled by the user for a channel.
disableNotification	To disable notifications for the specified events in a channel.

Organization Operations

Method	Description
getOrganization	To get the details of your organization.
uploadOrganizationPhoto	To upload a photo of your organization.

Profile Operations



Constructor	Description
ProfilesOperations(\$ifModifiedSince)	Creates a ProfilesOperations class instance with the value of the If-Modified-Since header.

Method	Description
getProfiles	To get the list of profiles available for your organization.
getProfile	To get the details of a specific profile.

Query (COQL) Operation

Method	Description
getRecords	To get the records from a module through a COQL query.

Records Operations

Method	Description
getRecord	To get a specific record from a module.
updateRecord	To update a specific record in a module.



deleteRecord	To delete a specific record from a module.
getRecords	To get all records from a module.
createRecords	To insert records in a module.
updateRecords	To update records in a module.
deleteRecords	To delete records from a module.
upsertRecords	To insert/update records in a module.
getDeletedRecords	To get the deleted records from a module.
searchRecords	To search for records in a module that match certain criteria, email, phone number, or a word.
convertLead	To convert records(Leads to Contacts/Deals).
getPhoto	To get the photo of a record.
uploadPhoto	To upload a photo to a record.
deletePhoto	To delete the photo of a record.
massUpdateRecords	To update the same field for multiple records in a module.
	To get the status of the mass update job



getMassUpdateStatus	scheduled previously.
-------------------------------------	-----------------------

Related List Operations

Constructor	Description
RelatedListsOperations(\$module)	Creates a RelatedListsOperations class instance with the moduleAPIName.

Method	Description
getRelatedLists	To get the details of all the related lists of a module.
getRelatedList	To get the details of a specific related list of a module.

Related Records Operations

Constructor	Description
RelatedRecordsOperations(\$relatedListAPIName, \$recordId, \$moduleAPIName)	Creates a RelatedRecordsOperations class instance with the relatedListAPIName, recordId, and moduleAPIName

Method	Description
--------	-------------



getRelatedRecords	To get list of records from the related list of a module.
updateRelatedRecords	To update the association/relation between the records.
delinkRecords	To delete the association between the records.
getRelatedRecord	To get the records from a specific related list of a module.
updateRelatedRecord	To update the details of a specific record of a related list in a module.
Delink Record	To delete a specific record from the related list of a module.

Role Operations

Method	Description
getRoles	To get the list of all roles available in your organization.
getRole	To get the details of a specific role.

Shared Records Operations



Constructor	Description
ShareRecordsOperations(\$recordId, \$moduleAPIName)	Creates a ShareRecordsOperations class instance with the recordId and moduleAPIName.

Method	Description
getSharedRecordDetails	To get the details of a record shared with other users.
shareRecord	To share a record with other users in the organization.
updateSharePermissions	To <ul style="list-style-type: none"> • Update the sharing permissions of a record granted to users as Read-Write, Read-only, or grant full access. • Revoke access given to users to a shared record. • Update the access permission to the related lists of the record that was shared with the user.
revokeSharedRecord	To revoke access to a shared record.



Tags Operations

Method	Description
getTags	To get the list of all tags in your organization.
createTags	To create tags.
updateTags	To update multiple tags.
updateTag	To update a specific tag.
deleteTag	To delete a specific tag from the module.
mergeTags	To merge two tags.
addTagsToRecord	To add tags to a specific record.
removeTagsFromRecord	To remove tags from a record.
addTagsToMultipleRecords	To add tags to multiple records.
removeTagsFromMultipleRecords	To remove tags from multiple records.
getRecordCountForTag	To get the record count for a tag.

Taxes Operations



Method	Description
getTaxes	To get the taxes of your organization.
createTaxes	To add taxes to your organization.
updateTaxes	To update the existing taxes of your organization.
deleteTaxes	To delete multiple taxes from your organization.
getTax	To get the details of a specific tax.
deleteTax	To delete a specific tax from your organization.

Territory Operations

Method	Return Type
getTerritories	APIResponse<ResponseHandler>
getTerritory	APIResponse<ResponseHandler>

Users Operations



Method	Description
getUsers	To get the list of users in your organization.
createUser	To add a user to your organization.
updateUsers	To update the existing users of your organization.
getUser	To get the details of a specific user.
updateUser	To update the details of a specific user.
deleteUser	To delete a specific user from your organization.

Variable Groups Operations

Method	Description
getVariableGroups	To get the list of all variable groups available for your organization.
getVariableGroupById	To get the details of a variable group by its group ID.
getVariableGroupByAPIName	To get the details of a specific variable group by its API name.



Variables Operations

Method	Description
getVariables	To get the list of variables available for your organization.
createVariables	To add new variables to your organization.
updateVariables	To update the details of variables.
deleteVariables	To delete multiple variables.
getVariableById	To get the details of a specific variable by its unique ID.
updateVariableById	To update the details of a specific variable by its unique ID.
deleteVariable	To delete a specific variable.
getVariableForAPIName	To get the details of a variable by its API name.
updateVariableByAPIName	To update the details of a variable by its API name.



Release Notes

Current Version:

1. ZCRMSDK - VERSION 3.1.0

Install command

```
1 composer require zohocrm/php-sdk:3.1.0
```

Notes

- Supported External ID

Previous Versions

2. ZCRMSDK - VERSION 3.0.3

Install command

```
1 composer require zohocrm/php-sdk:3.0.3
```

Notes

- Fixed oauth expiry_time error, which occurred in some PHP versions.

3. ZCRMSDK - VERSION 3.0.2

Install command

```
1 composer require zohocrm/php-sdk:3.0.2
```

Notes

- Supported Carry-over tags.
- Supported connection timeout.
- Followed PSR4 Standards.

4. ZCRMSDK - VERSION 3.0.1

Install command

```
1 composer require zohocrm/php-sdk:3.0.1
```

Notes



- Supported multiple extensions in file download operations.
- Handled different field types.
- Supported SSL verification toggle.

5. ZCRMSDK - VERSION 3.0.0

Install command

```
1 composer require zohocrm/php-sdk:3.0.0
```

Notes

- Version 3 is a new major version of the SDK that represents a significant effort to improve the capabilities of the SDK, incorporate customer feedback, upgrade our dependencies, improve performance, and adopt the latest PHP standards.
- The basic usage pattern of the SDK has changed from Version 2 to Version 3. Refer to the samples.
- The SDK is highly structured to ensure easy access to all the components.
- Each CRM entity is represented by a package, and each package contains an Operations Class that incorporates methods to perform all possible operations over that entity.
- **SDKException** - A wrapper class to wrap all exceptions such as SDK anomalies and other unexpected behaviours.
- **StreamWrapper** - A wrapper class for File operations.
- **APIResponse** - A common response instance for all the SDK method calls.

