Version 1

PHP SDK

ZOHO CRM

zoho.com/crm

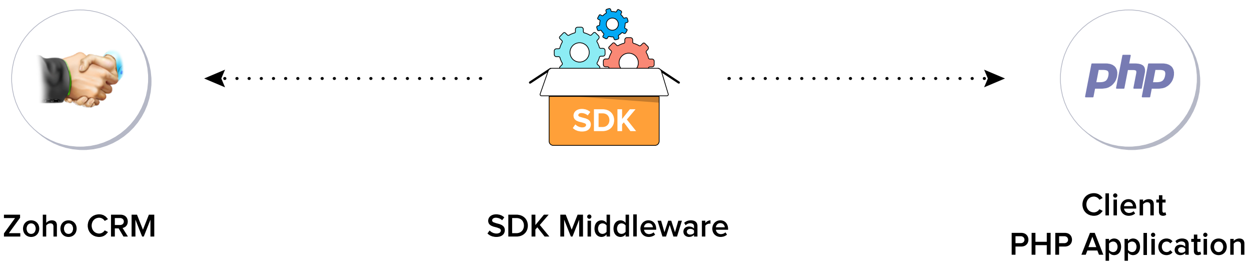
Table of Contents

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 configuration-related properties files, interface implementations, instances or objects. Authentication to access Zoho CRM APIs is through Oauth authentication mechanism. Invariably, HTTP requests and responses are taken care by SDK.

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



PHP SDK allows you to

- Exchange data between Zoho CRM and the client application where the CRM entities are modelled as classes.
- CRM API equivalents are declared and defined as simple functions in your PHP application.
- Push data into Zoho CRM, by accessing appropriate APIs of the CRM Service.

Environmental Setup

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

- Client app must have **PHP 5.6** or above with curl extension enabled.
- PHP SDK must be installed into client app though **composer.**
- The function *ZCRMRestClient::initialize()* must be called on startup of app.
- MySQL should run in the same machine.
- The database name should be **zohooauth.**
- There must be a table **oauthtokens** with columns
    - **useridentifier**(varchar(100))
    - **accesstoken**(varchar(100))
    - **refreshtoken**(varchar(100)),
    - **expirytime**(bigint)

Note:
If token_persistence_path is provided in the oauth_configuration.properties file, then persistence happens only in the file. In this case, there is no need of MySQL. Create a empty file with name, zcrm_oauthtokens.txt, in the mentioned token_persistence_path.

Using the SDK

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

```
require ‘vendor/autoload.php’
```

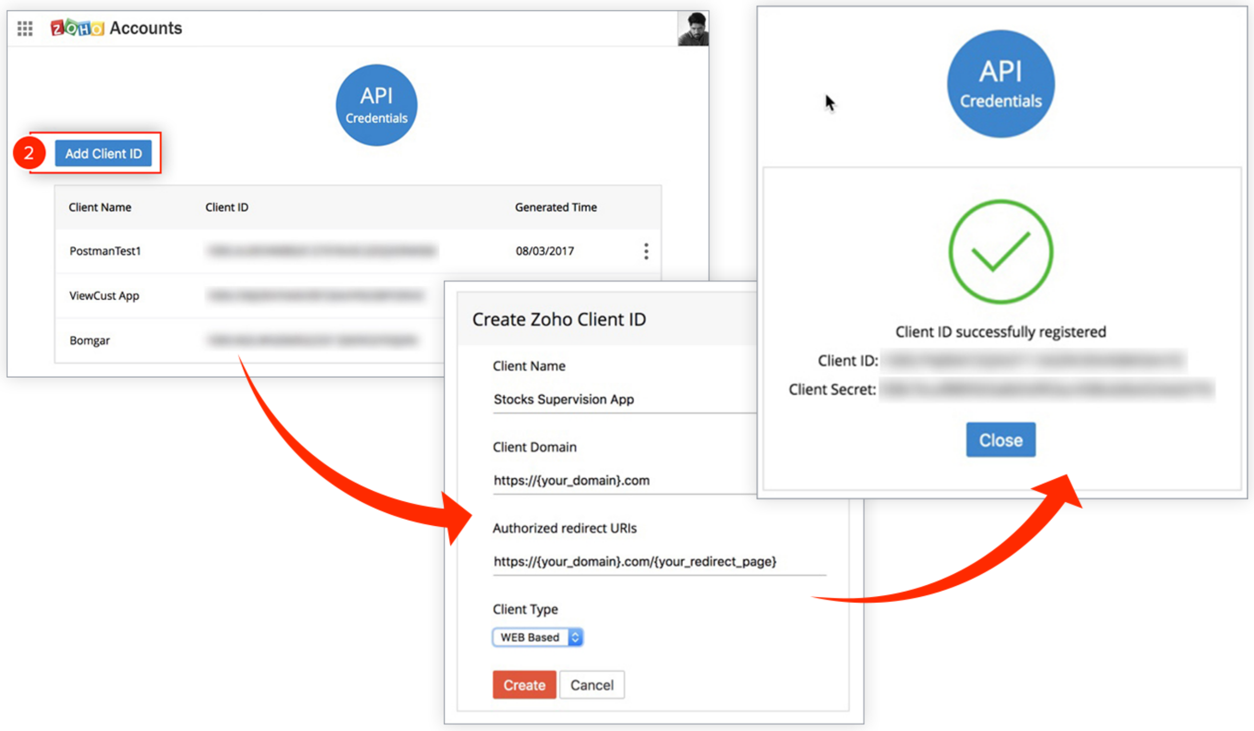
Through this line, you can access all the functionalities of the PHP SDK.

Register your application

All the Zoho CRM APIs are authenticated with OAuth2 standards, so it is mandatory to register and authenticate your client app with Zoho.

To register your application

1. Go to [accounts.zoho.com/developerconsole.](accounts.zoho.com/developerconsole)
2. Click **Add Client ID.**
3. Enter the **Client Name, Client Domain,** and **Redirect URL.**
4. For **Client Type,** select **Web based.**
5. Click **Create.**
6. Your Client app would have been created and displayed by now.
7. The newly registered app's Client ID and Client Secret can be found by clicking **Options → Edit.**



Client Name	Client ID	Generated Time
PostmanTest1	[redacted]	08/03/2017
ViewCust App	[redacted]	
Bomgar	[redacted]	

Create Zoho Client ID

Client Name: Stocks Supervision App

Client Domain: https://{your_domain}.com

Authorized redirect URIs: https://{your_domain}.com/{your_redirect_page}

Client Type: WEB Based

Create Cancel

Client ID successfully registered

Client ID: [redacted]

Client Secret: [redacted]

Note:
Options is the three dot icon at the right corner.

Registered applications 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.
- **Redirect URI –** The Callback URL that you registered during the app registration.

Installation

Install Composer (if not installed)

Run this command to install the composer

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

Info:
To install composer on mac/ linux machine:
https://getcomposer.org/doc/00-intro.md#installation-linux-unix-osx
To install composer on windows machine:
https://getcomposer.org/doc/00-intro.md#installation-windows

Install PHP SDK

Here's how you install the SDK:

1. Navigate to the workspace of your client app
2. **Run the command below:**

```
composer require zohocrm/php-sdk
```

Hence, the PHP SDK would be installed and a package named 'vendor' would be created in the workspace of your client app.

Configuration

Before you get started with creating your php application, you need to first authenticate the app with Zoho. And to do that there are some configuration procedures that need to be in place.

There are two methods in which you can authenticate your application:

- Passing a configuration array - and then call ZCRMRestClient::initialize($configuration);
  - The array option is available only for **version 1.1.0** and above.
- Using properties files in resources folder- and then call ZCRMRestClient::initialize();

Properties file

Your OAuth Client details should be given to the PHP SDK as a property file. In PHP SDK, we have placed a configuration file **(oauth_configuration.properties)**. Please place the respective values in that file. You can find that file under **' vendor/zohocrm/php-sdk/src/resources '**.

Based on your domain(EU,CN) please change the value of 'accounts_url'. Default value set as US domain. Please fill only the following keys.

```
client_id=
client_secret=
redirect_uri=
accounts_url= https://accounts.zoho.com/
token_persistence_path=
db_port=
db_username=root
db_password=
```

- **client_id** , **client_secret** and **redirect_uri** are your OAuth client's configurations that you get after registering your Zoho client.
- **access_type** must be set to **offline** only because online OAuth client is not supported by the PHP SDK as of now.
- **persistence_handler_class** is the implementation of the **ZohoOAuthPersistenceInterface** i.e. ZohoOAuthPersistenceHandler.
- **token_persistence_path** is the path to store the OAuth related tokens in file. If this is set, no need of database for persistence.
- In case you're using DB persistence, the keys db_port, db_usernam and db_password must be provided. The default value for db_username will be "root", db_password will be left empty and the db_port will be "3306".
    - Please replace the default values with your db credentials.

The **“configuration.properties”** file contains some additional properties. Place the respective values of the properties in that file. You can find that file under **‘vendor/zohocrm/php-sdk/src/resources’.**

Run this command to install the composer

```
apiBaseUrl=  www.zohoapis.com
apiVersion=v2
sandbox=false
applicationLogFilePath=
currentUserEmail=
```

- **apiBaseUrl -**
  Url to be used when calling an API. It is used to denote the domain of the user. Url may be:
  - https://www.zohoapis.com/
  - https://www.zohoapis.eu/
  - https://www.zohoapis.com.cn/
- apiVersion is “v2”.
- **sandbox -**
  To make API calls to sandbox account , please change the value of following key to true. By default the value is false.
- **applicationLogFilePath -**
  Represents the file to which the SDK can log.
- **currentUserEmail -**
  In case of single user, this configuration can be set.

Create a file named **“ZCRMClientLibrary.log”** in your client app machine and mention the absolute path of the created file in configuration.properties for the key **“applicationLogFilePath”.** This file is to log the exceptions occurred during the usage of PHP SDK.

In order to work with multi user authentication, you need to set the user EmailId in PHP super global variable ‘$_SERVER’ as given below:

```
$_SERVER[‘user_email_id’]=“  user@email.com  ”
```

You can use $_SERVER variable for single user authentication as well, but it is recommended to go with setting up of email Id in configuration.properties file.

If you do not set the user email as a super global variable, then SDK expects it from configuration. properties file. If user email is not set in any of these two then PHP SDK will throw exception.

Configuration Array

You can now pass the configuration values as php array(key-value pair) as argument when you call the ZCRMRestclient::initialize() function. Below is the list of keys that are to be in the array.

Mandatory Keys	Optional Keys
client_id	applicationLogFilePath
client_secret	sandbox
redirect_uri	apiBaseUrl
currentUserEmail	apiVersion
client_secret	access_type
	accounts_url
	persistence_handler_class
	token_persistence_path
	db_port
	db_username
	db_password

★ **Note:**

1. The key **“currentUserEmail”** must be specified if not present globally (existing functionality).
2. If the Optional keys are not specified, their default values will be assigned automatically.
3. The **‘apiBaseUrl’** and **‘accounts_url’** are mandatory in case the user is not in the “com” domain.

Below is an example of a PHP array containing the mandatory keys.

```
$configuration = array(“client id” => ”value”,”client_secret” =>
”value”,”redirect_uri” => ”value”,”currentUserEmail” => ”value”);
```

Below is an example of a PHP array containing all the keys.

```
$configuration=array(“client_id” => ”value”,”client_secret” =>
”value”,”redirect_uri” => ”value”,”currentUserEmail” =>
”value”,”applicationLogFilePath” => ”value”,“sandbox” =>
”value”,”apiBaseUrl” => ”value”,”apiVersion” => ”value”,
“access_type” => ”value”,”accounts_url” => ”value”,
”persistence_handler_class” => ”value”,
”token_persistence_path” => ”value”);
```

Initialization

The app would be ready to be initialized after defining the OAuth configuration file.

Generating self-authorized grant and refresh token

For self client apps, the self authorized grant token should be generated from the Zoho Developer Console (https://accounts.zoho.com/developerconsole)

1. Visit https://accounts.zoho.com/developerconsole
2. Click **Options** → **Self Client** of the client for which you wish to authorize.
3. Enter one or more(comma separated) valid **Zoho CRM scopes**, that you wish to authorize, in the "**Scope**" field and choose a time of expiry. Provide "**aaaserver.profile.READ**" scope along with Zoho CRM scopes.
4. Copy the **grant token** for backup.
5. Generate **refresh_token** from grant token by making a **POST** request with the URL below https://accounts.zoho.com/oauth/v2/token?code={grant_token}&redirect_uri={redirect_uri}&client_id={client_id}&client_secret={client_secret}&grant_type=authorization_code
6. Copy the refresh token for backup

Note:
Please note that the generated grant token is valid only for the stipulated time you choose while generating it. Hence, the access and refresh tokens should be generated within that time.

Generating access token

Access token can be generated by grant token or refresh token. Following any one of the two methods is sufficient.

Generating Access token from Grant token

The following code snippet should be executed from your main class to get access and refresh tokens. Please paste the copied grant token in the string literal mentioned below.

This is a one-time process.

```
ZCRMRestClient::initialize();
$oAuthClient = ZohoOAuth::getClientInstance();
$grantToken = “paste_the_self_authorized_grant_token_here”;
$oAuthTokens = $oAuthClient->generateAccessToken($grantToken);
```

Please note that the above code snippet is valid only once per grant token.
Upon successful execution of the above code snippet, the generated access and refresh tokens would have been persisted through our persistence handler class.

Generating Access token from Refresh token

The following code snippet should be executed from your main class to get access and refresh tokens. Please paste the generated refresh token in the string literal mentioned below.

This is a one-time process.

```
ZCRMRestClient::initialize();
$oAuthClient = ZohoOAuth::getClientInstance();
$refreshToken = “paste_the_refresh_token_here”;
$userIdentifier = “provide_user_identifier_like_email_here”;
$oAuthTokens = $oAuthClient->generateAccessTokenFromRefreshToken
($refreshToken,$userIdentifier);
```

Upon successful execution of the above code snippet, the generated access token and given refresh token would have been persisted through our persistence handler class.

Once the OAuth tokens have been persisted, subsequent API calls would use the persisted access and refresh tokens.

The PHP SDK will take care of refreshing the access token using refresh token, as and when required.

App Startup

The PHP SDK requires the following line of code invoked every time your client app is started.

```
class RestClient{
 public function __construct()
 {
 $configuration = array(“client_id”=>{client_id},”client_secret”=>{-
client_secret},”redirect_uri”=>{redirect_url},”currentUserE-
mail”=>{user_email_id});
 ZCRMRestClient::initialize($configuration);
 }
 public function getAllModules(){
 }
}
$obj =new RestC();
$obj->getAllModules();
```

Once the PHP SDK has been initialized by the above line, you could use any APIs of the SDK to get proper results.

Below is an example:

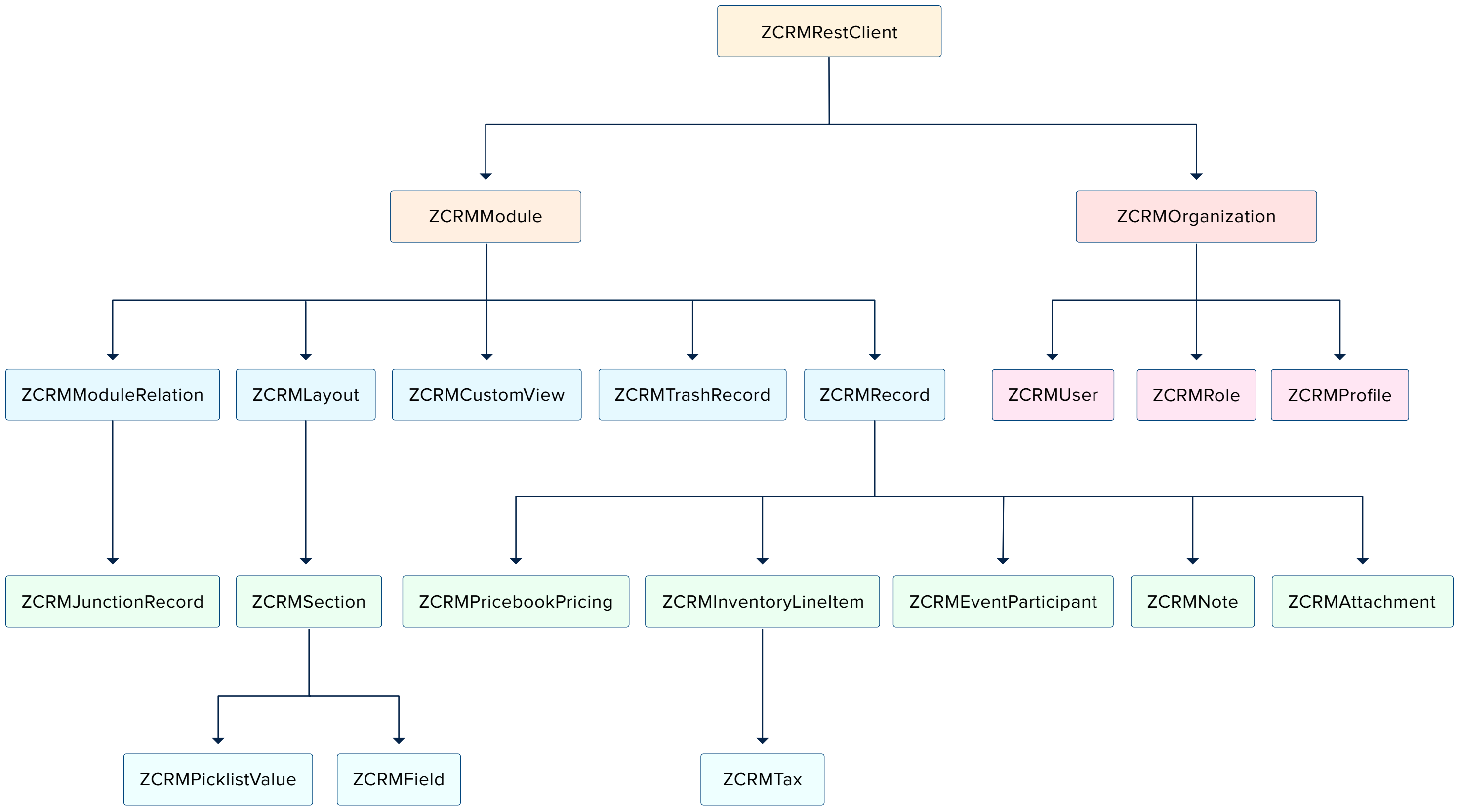
```
public function __construct()
    { ZCRMRestClient::initialize(); }
```

Class Hierarchy

All Zoho CRM entities are modelled as classes having members and methods applicable to that particular entity.

- ZCRMRestClient is the base class of the SDK.
- This class has, methods to get instances of various other Zoho CRM entities.
- The class relations and hierarchy of the SDK follows the entity hierarchy inside Zoho CRM.
- Each class entity has functions to fetch its own properties and to fetch data of its immediate child entities through an API call. For example, a Zoho CRM module (ZCRMModule) object will have member functions to get a module's properties like display name , module Id , etc, and will also have functions to fetch all its child objects (like ZCRMLayout).

The class hierarchy of various Zoho CRM entities is depicted as:



Instance Objects

It is not always effective to follow the complete class hierarchy from the top to fetch the data of an entity at some lower level, since this would involve API calls at every level.

In order to handle this, every entity class will have a “**getInstance()**” method to get its own dummy object and methods to get dummy objects of its child entities.

★ **Note:**
getInstance() methods would not have any of its properties filled since it would not fire an API call. This would just return a dummy object that shall be only used to access the non-static methods of the class.

Summing it up,

- **ZCRMRestClient.getModule(“Contacts”)** would return the actual Contacts module, that has all the properties of the Contacts module filled through an API call.
- **ZCRMRestClient.getModuleInstance(“Contacts”)** would return a dummy ZCRMModule object that would refer to the Contacts module, with no properties filled, since this doesn’t make an API call.

Hence, to get records from a module, there’s no need to start from ZCRMRestClient. Instead, you could get a ZCRMModule instance with ZCRMModule.getInstance() and then invoke its non-static getRecords() method from the created instance.

This would avoid the API call that would otherwise have been triggered to populate the ZCRMModule object.

Accessing record properties

Since record properties are dynamic across modules, we have only given the common fields like createdTime, createdBy, owner etc, as ZCRMRecord’s default members. All other record properties are available as a map in ZCRMRecord object.

To access the individual field values of a record, use the getter and setter methods available. The keys of the record properties map are the API names of the module's fields. API names of all fields of all modules are available under,

Setup → Marketplace → APIs → CRM API → API Names.

- To get a field value, use **record.getFieldValue(field_api_name);**
- To set a field value, use **record.setFieldValue(field_api_name, new_value);**

i **Info:**
While setting a field value, please make sure of that the set value is of the data type of the field to which you are going to set it.

Responses & Exceptions

APIResponse, BulkAPIResponse and FileAPIResponse are the wrapper objects for Zoho CRM APIs' responses. All API calling methods would return one of these two objects.

- A method-seeking entity would return **APIResponse** object, whereas a method-seeking list of entities would return **BulkAPIResponse** object.
- **FileAPIResponse** will be returned for file download APIs to download a photo or an attachment from a record or note such as record.downloadPhoto() , **record.downloadAttachment()** etc.
- Use the function **getData()** to get the entity data from the response wrapper objects for both APIResponse and BulkAPIResponse.
    - **APIResponse.getData()** would return a single Zoho CRM entity object, while **BulkAPIResponse.getData()** would return a list of Zoho CRM entity objects.
- **FileAPIResponse** has two defined methods namely FileAPIResponse.getFileName() which returns the name of the file that is downloaded and FileAPIResponse.getFileContent() that gives the file content as String.

Note:
BulkAPIResponse is a generic class. Hence, to get the records, the corresponding type has to be used.

```
“ZCRMModule module = ZCRMModule.getInstance(“Contacts”);
BulkAPIResponse<ZCRMRecord> response = module.getRecords();
List<ZCRMRecord> records = response.getData();”
```

Other than data, these response wrapper objects have the following properties:

ResponseHeaders -
This method is used to get the response headers. It is available thorugh

```
response.getResponseHeaders()
```

ResponseInfo -
any other information, if provided by the API, in addition to the actual data.

```
response.getInfo()
```

List<EntityResponse> -
status of individual entities in a bulk API. For example: an insert records API may partially fail because of a few records. This dictionary gives the individual records’ creation status. It is available through:

```
response.getEntityResponses()
```

Check Exceptions

All unexpected behaviors like faulty API responses, SDK anomalies are handled by the SDK and are thrown only as a single exception — **ZCRMException**. Hence, it's enough to catch this exception alone in the client app code.

Samples 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.

Rest Client Operations

These methods involve authentications procedures that are to be included in your application, to provide access to Zoho CRM's data.

Methods	**Description**
getAllModules	To fetch the list of all the modules available in your CRM account.
getModule	To fetch information about a particular module in your CRM account.
getRecordInstance	To get a dummy record.
getModuleInstance	To fetch details of a dummy module.
getOrganizationInstance	To fetch details of a dummy organization.

Methods	Description
getCurrentUser	To fetch information about the user who is currently accessing Zoho CRM's data through your application.
getCurrentUserEmailID	To fetch the email ID of the user, who's is currently accessing CRM's data.
getOrganizationDetails	To fetch all the details regarding your organization in your CRM account.

Organization Operations

These methods involve actions that can be performed in your application, to modify the data that pertains to your Zoho CRM's organization. For instance, you can get the list of all the users (employees) that are present in your organization at any point of time.

Methods	Description
getUser	To fetch information about a specific user in your CRM account.
getAllUsers	To fetch the list of all the users from your CRM account.
getAllActiveUsers	To fetch the list of all the active users in your CRM account.
getAllDeactiveUsers	To fetch the list of all the non-active users in your CRM account.

Methods	Description
getAllConfirmedUsers	To fetch the list of all the confirmed users in your CRM account.
getAllNotConfirmedUsers	To fetch the list of all the non-confirmed users in your CRM account.
getAllDeletedUsers	To fetch the list of all the users who were deleted from your CRM account.
getAllActiveConfirmedUsers	To fetch the list of all the active and confirmed users in your CRM account.
getAllAdminUsers	To fetch the list of all the users who have admin level permissions in your CRM account.
getAllActiveConfirmedAdmins	To fetch the list of all the users who have admin level permissions and are confirmed, in your CRM account.
getCurrentUser	To fetch the information about the users who is currently accessing CRM's data though your application.
deleteUser	To delete a user from your CRM account.
createUser	To create a new user in your CRM account.
updateUser	To update details of an existing user in your CRM account.
getAllProfiles	To fetch the list of all the profiles that were created in your CRM account.

Methods	Description
getProfile	To fetch information about a particular profile in your CRM account.
getAllRoles	To fetch the list of all the roles that were created in your CRM account.
getRole	To fetch information about a particular role in your CRM account.

Module Operations

These methods involve actions that can be performed in your application, to modify the data in your CRM at the module level. For instance, you can get all the records from a module, search for specific ones, delete them, and do more.

Methods	Description
getFieldDetails	To fetch the details about a particular field present in a module, in your CRM.
getAllFields	To fetch the list of all the fields that are available in a module.
getLayoutDetails	To fetch information about a particular layout of a module.
getAllLayouts	To fetch the list of all the layouts that are available for a module.

Methods	Description
getCustomView	To fetch information about a particular custom view of a module.
getAllCustomViews	To fetch the list of all the custom views that are available for a module.
updateCustomView	To update a custom view of a module.
getRelatedListDetails	To fetch information about a particular related list of a module.
getAllRelatedLists	To fetch the list of all the related lists that are available for a module.
getRecords	To fetch the list of all the records that are available in a module.
getRecord	To fetch information about a particular record in a module.
searchRecords	To search for records in a module.
searchRecordsByPhone	To search for records in a module based on the Phone number.
searchRecordsByEmail	To search for records in a module based on Email address.
searchRecordsByCriteria	To search for records in a module based on a criteria specified by the user.
massUpdateRecords	To update content in a particular field for multiple records in a module.

Methods	**Description**
updateRecords	To update details of multiple records in a module.
createRecords	To create a new record in a module.
deleteRecords	To delete existing records from a module.
getAllDeletedRecords	To fetch the list of all the records that were deleted from a module.
getRecycleBinRecords	To fetch the list of all the records that were deleted from a module and stored in the recycle bin.
getPermanentlyDeletedRecords	To fetch the list of all the records that were permanently deleted from a module.
getTags	To fetch the list of all the tags that were created for a module.
getTagCount	To fetch total count of the tags that were created for a module.
createTags	To create new tags for a module.
updateTags	To update details of existing tags for a module.
addTagsToRecords	To associate tags to records in a module.
removeTagsFromRecords	To disassociate tags from records in a module.

Record Operations

These methods involve actions that can be performed in your application, to access or modify data that are stored in a particular record. You could fetch the details of a record, create new ones, update existing ones, upload notes, attachments, photos, etc.

Methods	Description
create	To create new records.
update	To update existing records.
delete	To delete existing records.
convert	To convert records(Leads to Contacts/ Deals).
getRelatedListRecords	To fetch list of records in Related Lists.
getNotes	To fetch the notes that were attached to a record.
addNote	To add a note to a record.
updateNote	To update a note that was previously added to a record.
deleteNote	To delete a note from a record.
getAttachments	To fetch the list of attachments of a record.
uploadAttachment	To upload an attachment to a record.
uploadLinkAsAttachment	To upload a link as an attachment to a record.

Methods	Description
downloadAttachment	To download an attachment that was uploaded to a record.
deleteAttachment	To delete an attachment that was added to a record.
uploadPhoto	To upload a photo to a record.
downloadPhoto	To download a photo that was added to a record.
deletePhoto	To delete a photo that was added to a record.
addRelation	To make a relation between two records.
removeRelation	To remove a relation between two records.
addTags	To add tags to a record.
removeTags	To remove tags from a record.

ZOHO CRM

Contact Us

USA

- **California**
  Zoho Corporation
  4141 Hacienda Drive, Pleasanton,
  California 94588, USA
  Phone : +1 877 834 4428 | +1 615 671 9025

- **Austin**
  Zoho Corporation
  3910 S, IH 35, Suite 100, Austin,
  Texas 78704, USA

INDIA

- **Chennai**
  Zoho Corporation Pvt. Ltd.,
  Estancia IT Park, Plot No. 140 & 151, GST Road,
  Vallancherry Village, Chengalpattu Taluk,
  Kanchipuram District 603 202, INDIA
  Phone : +91 (44) 71817070 | +91 (44) 71817000
  +91 (44) 67447000

- **Tenkasi**
  Zoho Technologies Pvt. Ltd.,
  Silaraipuravu Village, Mathalamparai,
  Tenkasi, Tirunelveli District 627 814, INDIA

Zoho CRM Resources

www.zoho.com/crm/resources