

Node JS Version 0



Table of Contents

1. Overview	3
2. Registering a client	3
3. Installation of Node CRM SDK	5
a. Installing the SDK	
b. Upgrade the SDK	
4. Configurations	5
5. Token Storage Mechanisms	7
a. MySQL Persistence	
b. Custom Persistence	
c. Sample Code for Custom Token Management	
d. Configuration Array	
6. Initialization	11
a. Generating self-authorized grant and refresh token	
b. Sample API Call for getting leads	
c. Hierarchy	
7. Response Handling	15
8. Error Handling	15
9. Sample Codes	16
a. Organisation and User Operations	
b. Settings Operations	
c. Modules Operations	
d. Action and Attachment Operations	
10. Release Notes	3



Overview

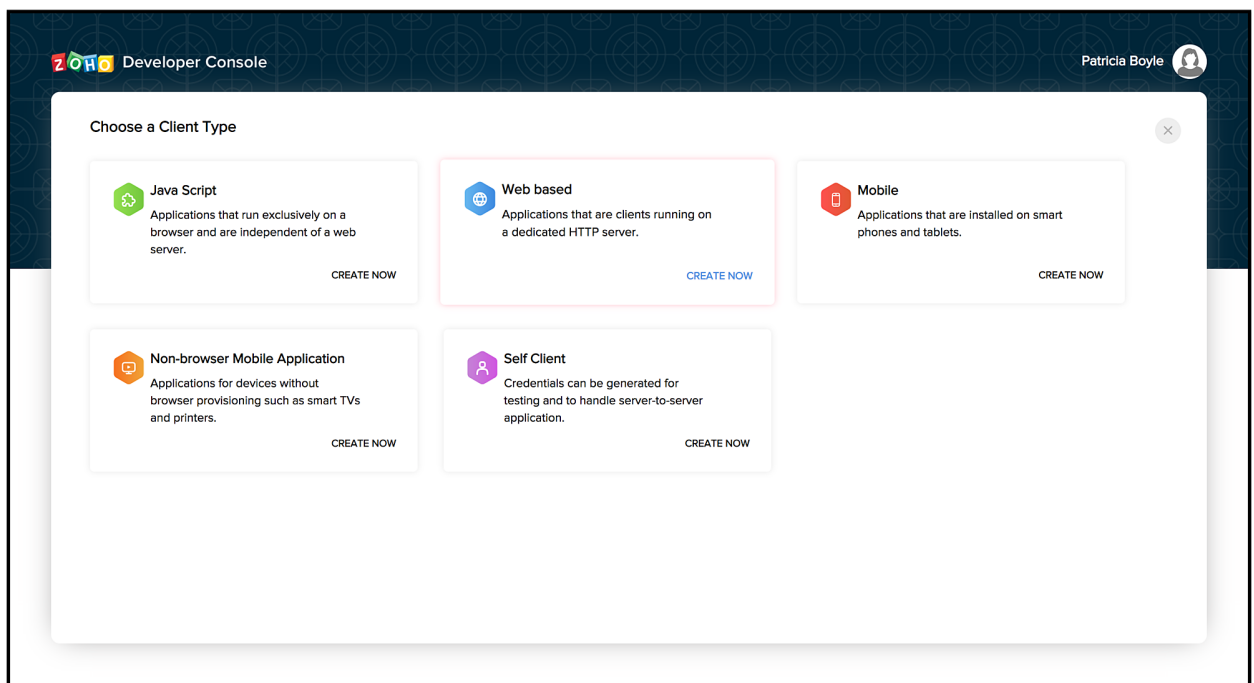
Node SDK is a wrapper for Zoho CRM APIs. Hence invoking a Zoho CRM API from your Node application is just a function call which provide the most appropriate response.

This SDK supports both single user as well as multi user authentication.

Registering a Zoho Client

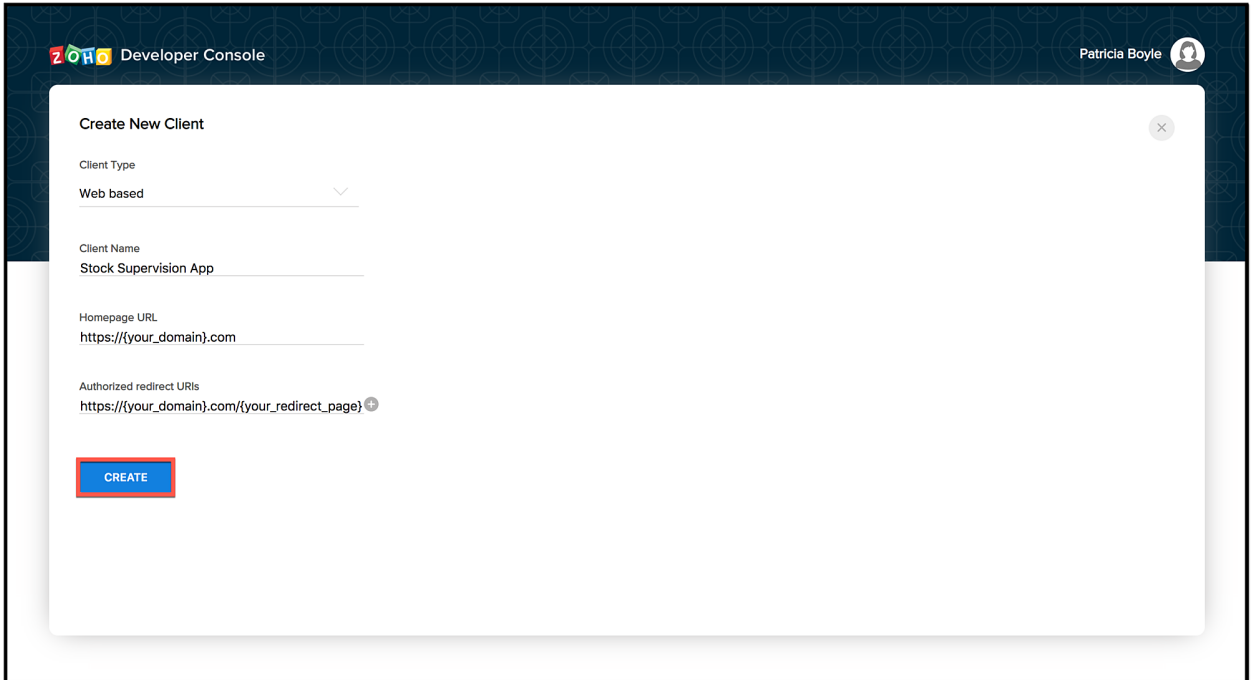
Since Zoho CRM APIs are authenticated with OAuth2 standards, you should register your client app with Zoho. To register your app:

1. Go to Zoho Developer Console.
2. Click ADD CLIENT.
3. Choose the client as Web based and click CREATE NOW.

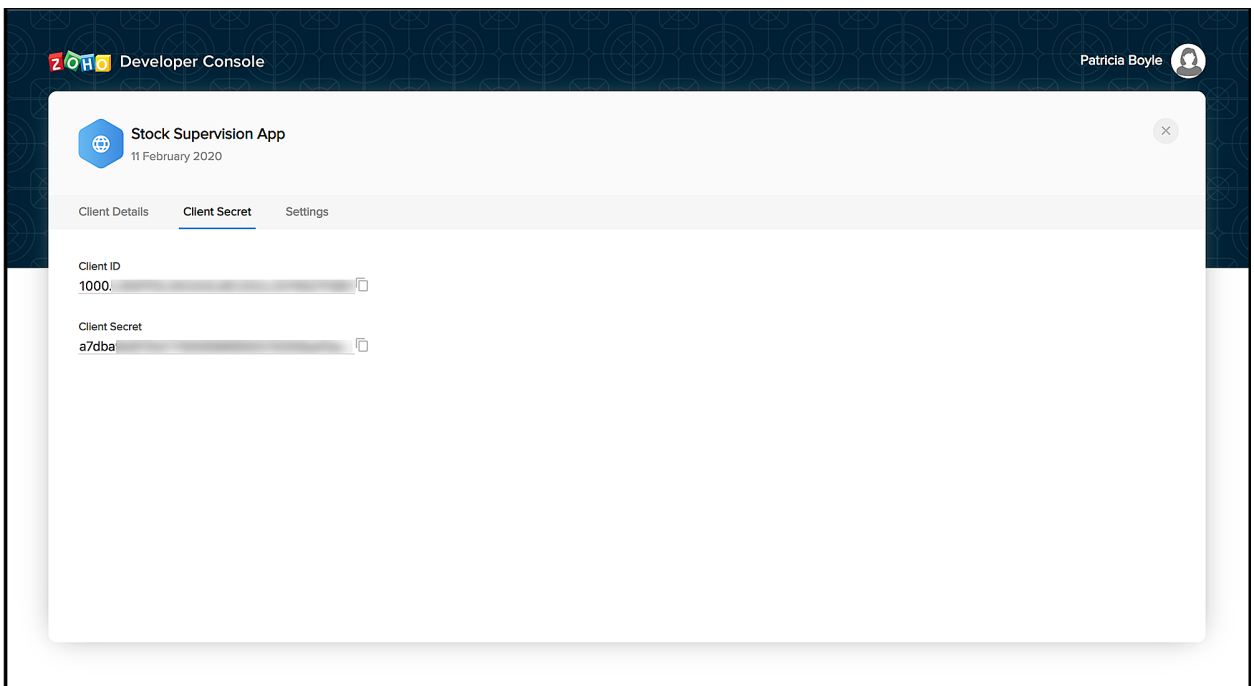


4. Specify the client name, homepage URL of your application's UI, and a redirect URI to which you want to redirect the users after they grant consent to your application.





5. Click Create.
6. Your Client ID and Client Secret will be displayed.



Installation of Node CRM SDK

Node JS SDK will be installed and a package named 'zcrmsdk' will be created in the local machine.

The package can be added using the following code:

```
1 var ZCRMRestClient = require('zcrmsdk')
```

Installing the SDK

Here's how you install the Node JS SDK

- Run the command below:

```
1 npm install zcrmsdk
```

Another method to install the SDK is to add it in dependencies to the **package.json** of the node server with the latest version (recommended) and to run npm install in the directory which installs all the dependencies mentioned in package.json.

Upgrade the SDK

- Run this command to upgrade the Node JS SDK to the latest version.

```
1 npm install --upgrade zcrmsdk
```

Configurations

Your OAuth Client details should be given to the SDK as a property file or must be passed while initializing the SDK as JSON key-value pairs. In the SDK, you need to configure a file named `oauth_configuration.properties`. Please place the respective values in that file. You can place it under resources folder from where the SDK is used.

Please fill the values for the following keys alone.

Based on your domain(EU,CN, IN, AU or JP), please change the value of `crm.iamurl`. Default value is set as US domain.



In **oauth_configuration.properties** file:

```
1 [zoho]
2 crm.iamurl=
3 crm.clientid=
4 crm.clientsecret=
5 crm.redirecturl=
```

- `crm.clientid`, `crm.clientsecret` and `crm.redirecturl` are your OAuth client's configurations that you get after registering your Zoho client.
- `crm.iamurl` is the accounts URL. It may be `accounts.zoho.com`, `accounts.zoho.eu`, or `accounts.zoho.jp`. If the `crm.iamurl` is not specified, by default the URL will be `accounts.zoho.com`.

In **configuration.properties** file:

```
1 [crm]
2 api.url=
3 api.user_identifier=
4 api.tokenmanagement=
5 [mysql]
6 username=
7 password=
```

- **api.url** is the URL used to call APIs. By default, the URL is `www.zohoapis.com`.
- **api.user_identifier** will be empty by default. For single user authentication, this key can be filled with the respective email id, so that all calls happens by using this user's authentication.
- **api.tokenmanagement** is given as a measure to manage and maintain tokens. If `tokenmanagement` is not provided, then the SDK's default implementation(`mysql`) will be followed.
- **username** and **password** can be given here if you already have one created for your MySQL.

The above keys specified in `configuration.properties` file are all optional.



Token Storage Mechanism

MySQL Persistence

To use the default token storage provided by the SDK, the following are to be done:

MySQL should be running at the **default port** in **localhost**.

Database with name **zohooauth** should be created and a table with below configurations should be present in the database. Table, named "**oauthtokens**", should have the columns "**useridentifier**" (varchar) "**accesstoken**" (varchar), "**refreshtoken**" (varchar) and "**expirytime**" (bigint).

Once the configuration is set, storage and retrieval of tokens will be handled by the SDK.

Custom Persistence

If the user wants to utilize their own mechanism, they can mention it in **configuration.properties** by providing the respective module in **api.tokenmanagement**.

This module should contain the below methods,

1. saveOAuthTokens(token_obj)
2. updateOAuthTokens(token_obj)
 - Irrespective of the response, the next execution happens. So care should be taken by the user in handling their module.
3. getOAuthTokens()
 - The expected response for this method : JSON array containing json response with expirytime, refreshtoken and accesstoken fields.

Note:

All methods should return promise.

saveOAuthToken & **updateOAuthTokens** will be called with JSON parameters, which contain fields given below,



```
1 access_token
2 refresh_token
3 expires_in
```

Sample code for custom token management methods

saveOAuthTokens()

```
1 const fs = require('fs');
2 var path = require('path');
3 var file_persistence = {};
4 file_persistence.saveOAuthTokens = function(tokenobject){
5   return new Promise(function(resolve, reject){
6
7     file_persistence.updateOAuthTokens(tokenobject).then(function()
8       resolve();
9     })
10  }
```

getOAuthTokens()

```
1 file_persistence.getOAuthTokens =
2   function(user_identifier){
3     return new Promise(function(resolve, reject){
4       var found = 0;
5       if
6         (fs.existsSync(path.dirname(require.main.filename)+'/zcrm_o
7
8         var token_as_string =
9         fs.readFileSync(path.dirname(require.main.filename)+'/zcrm_
10
11       tokens = JSON.parse(token_as_string);
12     })
13   }
```




```

9   for(token in tokens.tokens){
10  if(tokens.tokens[token].user_identifier ==
    user_identifier){
11    found = 1;
12    resolve(tokens.tokens[token]);
13  }
14  }
15
16  if(found == 0){
17    resolve();
18  }
19  }
20 })
21 }

```

updateOAuthTokens()

```

1  file_persistence.updateOAuthTokens = function(tokenobject){
2  return new Promise(function(resolve, reject){
3    var tokens = {}
4    if
      (fs.existsSync(path.dirname(require.main.filename)+' /zcrm_o

5    var token_as_string =
      fs.readFileSync(path.dirname(require.main.filename)+' /zcrm_

6    tokens = JSON.parse(token_as_string);
7
8    for(token in tokens.tokens){
9      if(tokens.tokens[token].user_identifier ==
      tokenobject.user_identifier){
10     tokens.tokens.splice(token)
11   }
12 }
13 tokens.tokens.push(tokenobject);
14 }

```



```

15  else{
16    tokens.tokens = [tokenobject]
17  }
18  token_as_string = JSON.stringify(tokens);
19
20
    fs.writeFile(path.dirname(require.main.filename)+' /zcrm_oau

21  if (err) throw err;
22  resolve();
23  });
24  })
25 }
26
27 module.exports = file_persistence;

```

Custom persistence will create the token file named zcrm_oauthtokens.txt and this file will contain the below keys with their values.

```

1  {"tokens":
2   [{"access_token":"1000.xxxxx",
3    "expires_in":1582804396157,
4    "user_identifier":"Patricia.boyle@zohocorp.com",
5    "refresh_token":"xxx"}]}

```

Configuration array

The Node JS SDK also accepts the configuration details as a JSON array. You can pass this array when you initialize the SDK, as below (after the 'require' statement).

```

1  var configJson={
2    "client_id":"{client_id}",//mandatory
3    "client_secret":"{client_secret}",//mandatory
4    "redirect_url":"{redirect_url}",//mandatory

```



```

5  "user_identifier":"{user_identifier}",
6  "mysql_username":"{mysql_username}",//optional ,"root" is
   default value
7  "mysql_password":"{mysql_password}",//optional ,"" is
   default value
8  "base_url":"{api_base_url}",//optional ,"www.zohoapis.com"
   is default value
9  "iamurl":"{accounts_url}",//optional ,"accounts.zoho.com"
   is default value
10 "version":"{api_version}",//optional ,"v2" is default
    value
11 "tokenmanagement":"{token_management_module}"//optional
    ,mysql module is default
12 }

```

Initialization

Whenever the app is started, the below code snippet is to be called for initialization. Note that the snippet mentioned below is only when you specify the configuration details in the properties files.

```

1  var ZCRMRestClient = require('zcrmsdk');
2  ZCRMRestClient.initialize().then(function()//Use
   ZCRMRestClient.initialize(configJson).then(function() for
   configuration array
3  {
4  //do whatever required after initialize
5  })

```

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. Go to [Zoho Developer Console](#).
2. Select your Self Client.
3. Provide the necessary [scopes](#) separated by commas, along with the scope `aaaserver.profile.READ`.
4. Select the *Time Duration* from the drop-down. This is the time the grant token is valid for.
5. Provide a description for the scopes. Click *GENERATE*.
6. Copy the grant token.

Please note that 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.

All functions return promises in zcrm node sdk.

Getting access and refresh tokens from grant token through method calls

```
1 ZCRMRestClient.generateAuthTokens(user_identifier,grant_tok
2 console.log("access token :"+auth_response.access_token);
3 console.log("refresh token :"+auth_response.refresh_token);
4 console.log("expires in :"+auth_response.expires_in);
5 });
```

The access and refresh tokens are generated. In case the access token is expired, the SDK will refresh it automatically.

If the user has refresh token and need to generate access token below function can be used,

```
1 ZCRMRestClient.generateAuthTokenfromRefreshToken(user_ident
```



```
2 console.log("access token :"+auth_response.access_token);
3 console.log("refresh token :"+auth_response.refresh_token);
4 console.log("expires in :"+auth_response.expires_in);
5 });
```

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.

Sample API call for getting Leads:

```
1 var input ={};
2 input.module = "Leads";
3 var params = {};
4 params.page = 0;
5 params.per_page = 5;
6 input.params = params;
7 zcrmsdk.API.MODULES.get(input).then(function(response){
8     var result = "<html><body><b>Leads</b>";
9     var data = response.body;
10    data = JSON.parse(data);
11    data = data.data;
12    for (i in data){
13        var record = data[i];
14        var name = record.Full_Name;
```



```
15         result+="<span>" + name + "</span>";
16     }
17     result+="</body></html>";
18 }
```

Hierarchy

zcrmsdk

```
1     API
2     ORG
3         get
4     MODULES
5         get
6         post
7         put
8         delete
9         getAllDeletedRecords
10        getRecycleBinRecords
11        getPermanentlyDeletedRecords
12        search
13    SETTINGS
14        getFields
15        getLayouts
16        getCustomViews
17        updateCustomViews
18        getModules
19        getRoles
20        getProfiles
21        getRelatedLists
22    ACTIONS
23        convert
24    USERS
25        get
26    ATTACHMENTS
```



```
27     uploadFile
28     deleteFile
29     downloadFile
30     uploadLink
31     uploadPhoto
32     downloadPhoto
33     deletePhoto
34     FUNCTIONS
35     executeFunctionsInGet
36     executeFunctionsInPost
```

As appearing in the hierarchy, **zcrmsdk** entity module has variables to fetch its own and other modules properties.

For example, to call an API to get module data, the request should be **zcrmsdk.API.MODULES.{operation_type}**. The operation types can be GET, POST, PUT, DELETE or CREATE.

Response Handling

All API calls will give the actual API response given by Zoho APIs, except file download.

For file download, the response will contain an extra field filename.

Error Handling

All errors will be thrown explicitly and care should be taken in catching the same.



Sample Codes

All of Zoho CRM's APIs can be used through the Python 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.

Organization & User Operations

These methods involve actions that can be performed in your application, to fetch the data of 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
crmclient.API.ORG.get(params)	To fetch information about your CRM account organization.
crmclient.API.USERS.get(input)	To fetch information about all the users in your CRM account.
crmclient.API.USERS.get(input)	To fetch information about a particular user in your CRM account. [Specify the id of the user].

Settings Operations

These methods involve actions that can be performed in your application, to fetch information regarding the resources available in Zoho CRM. In other words, you can get the meta data of all the resources.



Methods	Description
crmclient.API.SETTINGS.getModules(input)	To fetch the list of all the modules available in your CRM account.
crmclient.API.SETTINGS.getModules(input)	To fetch information about a particular module in your CRM account. [Specify the name of the module]
crmclient.API.SETTINGS.getRoles(input)	To fetch the list of all the roles that were created in your CRM account.
crmclient.API.SETTINGS.getRoles(input)	To fetch information about a particular role in your CRM account. [Specify the id of the role]
crmclient.API.SETTINGS.getProfiles(input)	To fetch the list of all the profiles that are available in your CRM account.
crmclient.API.SETTINGS.getProfiles(input)	To fetch information about a particular role available in your CRM account. [Specify the id of the profile]
crmclient.API.SETTINGS.getFields(input)	To fetch the list of all the fields available in a module in your CRM account.
crmclient.API.SETTINGS.getLayouts(input)	To fetch the list of all the layouts of a module in your CRM account.
crmclient.API.SETTINGS.getLayouts(input)	To fetch information about a particular layout of a particular module in your



	CRM account. [Specify the id of the layout]
<code>crmclient.API.SETTINGS.getRelatedLists(input)</code>	To fetch the list of all the related lists of a particular module in your CRM account.
<code>crmclient.API.SETTINGS.getCustomViews(input)</code>	To fetch the list of all the custom views of a particular module in your CRM account.
<code>crmclient.API.SETTINGS.getCustomViews(input)</code>	To fetch information about a particular custom view of a particular module in your CRM account. [Specify the id of the custom view]

Modules 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
<code>crmclient.API.MODULES.get(input)</code>	To fetch the list of all the records that are present in a module.
<code>crmclient.API.MODULES.get(input)</code>	To fetch information about a particular record present in a module. [Specify the id of the record]



<code>crmclient.API.MODULES.post(input)</code>	To create records in a module.
<code>crmclient.API.MODULES.put(input)</code>	To update existing records in a module.
<code>crmclient.API.MODULES.put(input)</code>	To update a particular record in a module. [Specify the id of the record]
<code>crmclient.API.MODULES.delete(input)</code>	To delete a particular record in a module.
<code>crmclient.API.MODULES.delete(input)</code>	To delete records in a module.
<code>crmclient.API.MODULES.getAllDeletedRecords(input)</code>	To fetch the list of all the deleted records of a module.
<code>crmclient.API.MODULES.getRecycleBinRecords(input)</code>	To fetch the list of all the records that are deleted and are in the Recycle Bin.
<code>crmclient.API.MODULES.getPermanentlyDeletedRecords(input)</code>	To fetch the list of all the permanently deleted records of a module.
<code>crmclient.API.MODULES.search(input)</code>	To search for records in a module.

Actions & Attachments Operations

These methods involve actions that can be performed in your application, to perform certain actions such as converting a lead, uploading attachments, photos, etc.

Methods	Description
---------	-------------



<code>crmclient.API.ACTIONS.convert(input)</code>	To convert a record(Leads to Contacts/Deals).
<code>crmclient.API.ATTACHMENTS.uploadFile(input)</code>	To upload files(attachments) to a record in a module.
<code>crmclient.API.ATTACHMENTS.downloadFile(input)</code>	To download files(attachments) that were attached to a record in a module.
<code>crmclient.API.ATTACHMENTS.deleteFile(input)</code>	To delete files(attachments) that were attached to a record in a module.
<code>crmclient.API.ATTACHMENTS.uploadPhoto(input)</code>	To upload photos/images to a record in a module.
<code>crmclient.API.ATTACHMENTS.downloadPhoto(input)</code>	To download photos/images that were attached to a record in a module.
<code>crmclient.API.ATTACHMENTS.deletePhoto(params)</code>	To delete photos/images that were attached to a record in a module.

Release Notes

Current Version:

1. ZCRMSDK -VERSION 0.0.20

- Install command

```
1 npm install zcrmsdk@0.0.20
```

Notes

Fixed the mandatory check for `redirect_url` in configuration JSON.



Previous Versions

1. ZCRMSDK -VERSION 0.0.19

- Install command

```
1 npm install zcrmsdk@0.0.19
```

Notes

- Support provided for the `getPersistenceModule()` in `ZCRMRestClient.js`.

2. ZCRMSDK -VERSION 0.0.18

- Install command

```
1 npm install zcrmsdk@0.0.18
```

Notes

Custom token management.

3. ZCRMSDK -VERSION 0.0.17

- Install command

```
1 npm install zcrmsdk@0.0.17
```

Notes

- Configuration details can be sent as JSON while initializing the SDK.
- Handled the error Response.

4. ZCRMSDK -VERSION 0.0.16

Install command

```
1 npm install zcrmsdk@0.0.16
```

Notes

Handled the change in the OAuth token response.

5. ZCRMSDK -VERSION 0.0.15

- Install command



```
1 npm install zcrmsdk@0.0.15
```

Notes

- User identifier default value handled in get method.

6. ZCRMSDK -VERSION 0.0.14

- Install command

```
1 npm install zcrmsdk@0.0.14
```

Notes

- Default user identifier set internally in case user doesn't set it.

7. ZCRMSDK -VERSION 0.0.13

- Install command

```
1 npm install zcrmsdk@0.0.13
```

Notes

- Custom headers support added.
- Unwanted code removed.

8. ZCRMSDK -VERSION 0.0.12

- Install command

```
1 npm install zcrmsdk@0.0.12
```

Notes

- Issue fixes

9. ZCRMSDK -VERSION 0.0.11

- Install command

```
1 npm install zcrmsdk@0.0.11
```



Notes

- Refresh token key added in result.

10. ZCRMSDK -VERSION 0.0.10

- Install command

```
1 npm install zcrmsdk@0.0.10
```

Notes

- Header added.

11. ZCRMSDK -VERSION 0.0.7 - 0.0.9

- Install command

```
1 npm install zcrmsdk@0.0.x
```

Notes

- Issue fixes.

12. ZCRMSDK -VERSION 0.0.6

- Install command

```
1 npm install zcrmsdk@0.0.6
```

Notes

- Token management taken from oauth_configuration.properties fixed.

13. ZCRMSDK -VERSION 0.0.1 - 0.0.5

Install command

```
1 npm install zcrmsdk@0.0.x
```

Notes

- Issue fixes.



14. ZCRMSDK -VERSION 0.0.0

Install command

```
1 npm install zcrmsdk@0.0.0
```

Notes

- Initial release.

