Client Scripting for Zoho CRM

ZOHO CRM

READ ME

Table of Contents

1. What is a client script?

Client script is a piece of JavaScript code that **runs on your web browser** instead of the server, thus returning an **immediate response**. Using the Client script, you can perform **event-triggered UI actions on the client-side**. For example, when a user fills the email address of a Lead/Contact, you can perform email discovery and automatically populate relevant data in the form.

Likewise, you can implement any highly interactive custom business logic with ease.

2. Pre-requisite

You should be on the **Enterprise** or **Ultimate** edition of Zoho CRM to use this feature.

3. Supported Browser Versions

Browser	Recommended Version
Chrome	73
Firefox	69
Safari	13
Opera	60

Note:
Using client script, you can handle only the UI on the client-side. It does not have any effect on the server configurations.

4. Events and Actions

A client script has two primary components, namely:

(i) Events: The client script gets executed when an associated event is triggered in the client.

(ii) Actions: You can perform client-side actions using ZDK. There are various helpers available in ZDK, which you can use to perform client operations, hit APIs, and call functions.

4a. Application Events

Application events are triggered from inside the Zoho CRM application. The event can be triggered either from a page or field.

(i) Page Events : These are the events that occur on a **specific page** in Zoho CRM. It can be a **create page**, **clone page**, or an **edit page** of a module. For instance,

- **onLoad:** The system automatically runs the client script when a page(create/clone/edit) gets **loaded**. For instance, you can load the address details of a lead, in the create lead page, based on the territory of the current user.
- **onChange:** The system automatically runs the client script when you **update any field** in the page.
- **onSave:** The system automatically runs the client script when you **save a record**. For instance, you can display the score points of the current user, when a contact is saved.

(ii) Field Events: These are the events that occur in a **specific field** in Zoho CRM.

- **onChange:** The system automatically runs the client script when you **update** a specific field. For instance, you can update the scale of a company based on the entry in the number of employees field.

4b. Actions

Some of the sample page actions you can achieve using client script are:

- **Read and write data** in any type of **field** on the current page. For instance, you can load the address details of a lead, in the create lead page, based on the territory that the current user belongs to.
- **Show error/alert/warning/info messages** in the native style of the application. For instance, you can throw an error message when the input given for a custom field is invalid.
- **Freeze or unfreeze a page**. For instance, you can freeze a page(create/clone/edit) until an internal process is complete.
- Display **inline error messages** for a field. For instance, you can display an inline error message if the pin code entered is invalid.

- **Mark a field as mandatory** and convey it through a text.

5. Advantages of Client Script

- **No latency** as you do not have to wait for a response from the server as all the processing happens in the client's browser itself.
- Perform **more responsive client-side validation** of fields using the ZDK.
- **Auto-populate field data** while creating or editing a record. Example: Auto-updating the state or county field when a user enters the pin code.
- **Apply advanced formula computation** to the fields. Example: Calculating and updating the age of a person based on the birth date entered in a form.
- Achieve **field validation**, **auto-correction**, and **data enrichment**. Example: Automatically detecting whether the card is a VISA or Master Card.
- Display **custom error messages** for a specific set of users or fields. Example: Throwing an error when the user enters a personal email ID in the organization email ID field.

6. Creating a Client Script

You can add a client script in two ways in Zoho CRM:

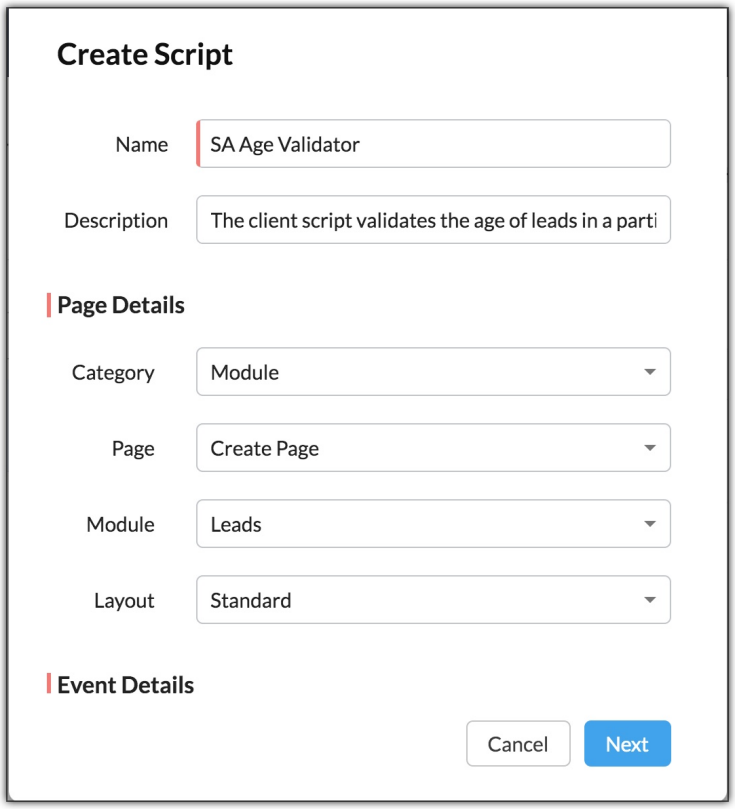
1. Through the setup page
2. Through create/clone/edit page of a record in a module

Let us discuss each one of them in detail.

6a. To add a client script through the setup page

Step 1: Go to *Setup* > *Developer Space* > *Client Script*. Click *+New Script*.

Step 2: In the **Create Script** pop up, enter the *Name* and *Description* of the script.

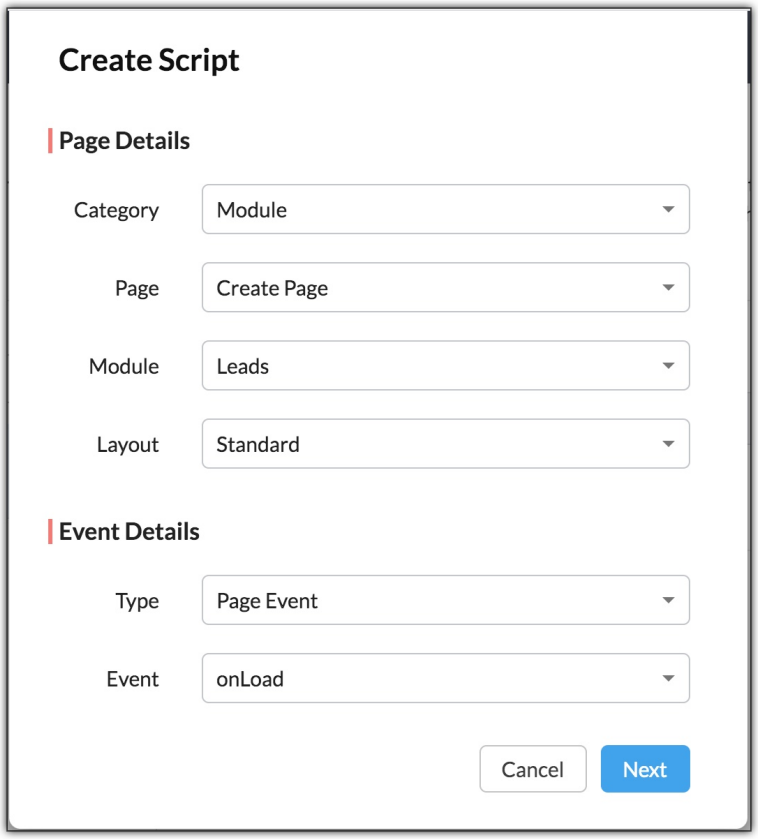


Step 3: In the *Page details* section, choose the following:

- ***Category*** **–** Choose **Module**. In the upcoming enhancements, we will provide option to add the client script to pages other than the module.
- ***Page*** - Choose the page to which you want to add the client script. It can be: **Create Page**, **Clone Page**, or **Edit Page**.
- Choose the module. It can be: Leads, Contacts, Accounts, Deals, Products, Quotes, Sales Orders, Purchase Orders, Invoices, Campaigns, Vendors, Price Books, Cases, Solutions, or Custom Module.
- Choose the layout. It can be Standard or any other custom layout.

Step 4: In the *Event Details* section, select the *Event Type* (page or field event).

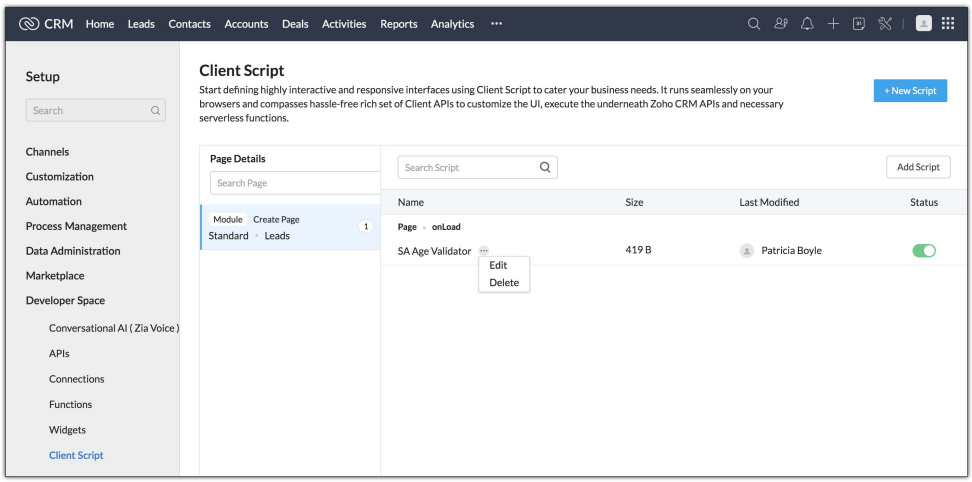
- For a field event, select the *Field* (list of fields on that page) where the event occurs and the *Field Event* i.e., onChange.
- For a page event, select the *Event*. It can be: onLoad, onChange, or onSave.



Step 5: Click *Next*.

Step 6: The client script editor opens. This is similar to the Visual Studio Code IDE.

Step 7: Click *Save and Close* to save your script. You can see the client script setup page that lists all the scripts you have created. It shows the name of the script, the size, the name of the user who last modified the script, and the status (enabled or disabled). You can enable a script by sliding the toggle button.

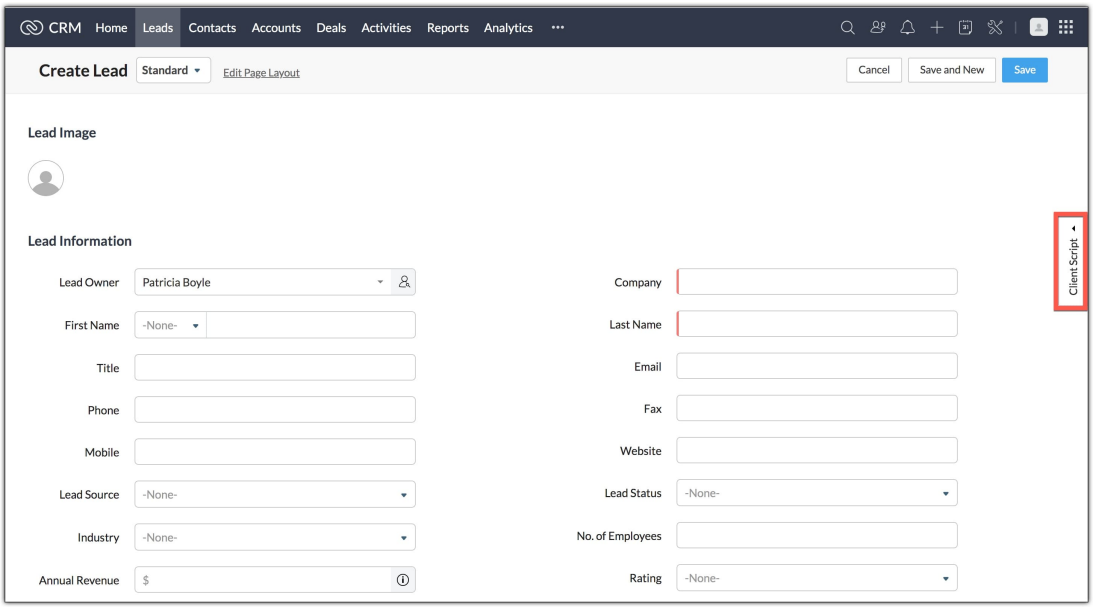


Step 8: You can also edit or delete any script from this setup page.

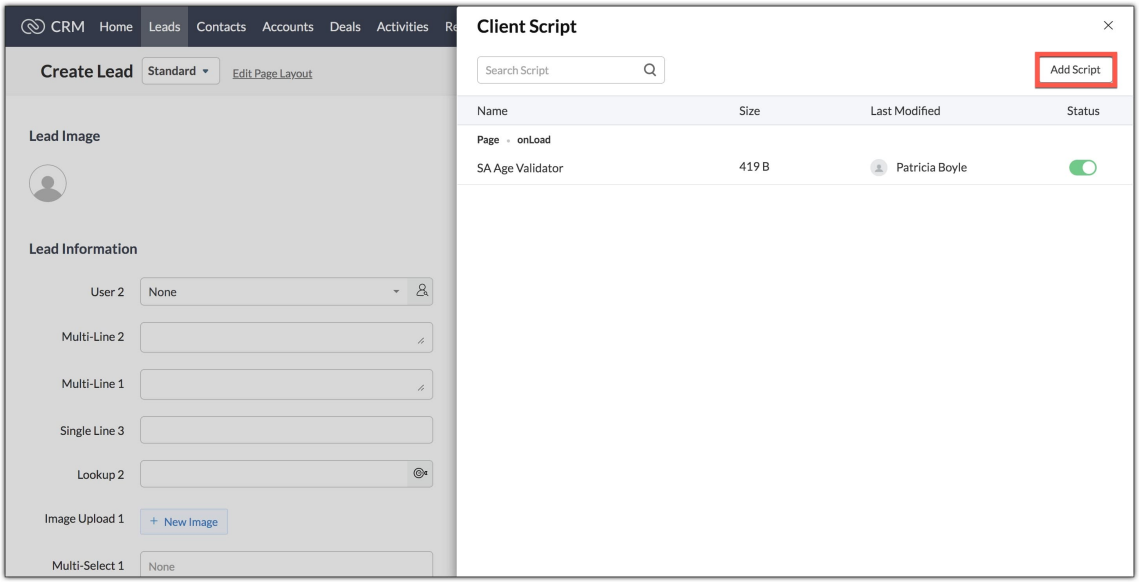
6b. To add a client script through the create/clone/edit page of a record in a module

Step 1: Choose a desired module in Zoho CRM, and open either create, clone, or edit page of any record in that module. For example, consider the create lead page.

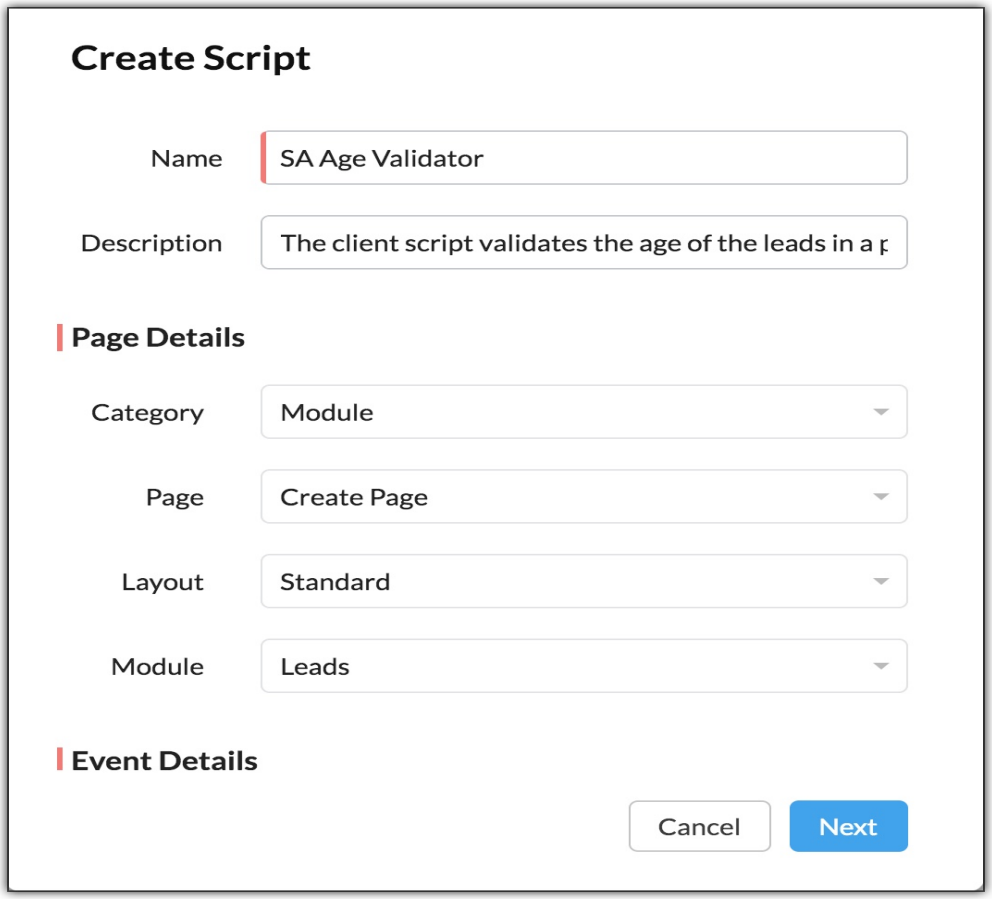
Step 2: In the right-most corner of the page, click *Client Script*.



Step 3: Click Add Script.



Step 4: The page details will be **pre-populated** by the system. Fill the other details of the client script, such as Name, Description, and the Event Details of the client script. Further, click *next* and add your script to the script editor and click *Save and Close*.



7. ZDK Client Apps

The **Zoho Development Kit (ZDK)** javascript library offers a rich set of client and server interfaces that broaden the scope of Zoho Applications. The Dot Notation in ZDK allows you to instantly access various pre-defined functions. You can utilise them to perform UI operations and trigger REST API calls.

7a. UI Actions

Function	Description
ZDK.Message.Info(message)	To display info message in a page.
ZDK.Message.Success(message)	To display success message in a page.
ZDK.Message.Warning(message)	To display warning message in a page.
ZDK.Message.Error(message)	To display error message in a page.
ZDK.Message.Inline(message)	To display inline message below a specific field in a page.
ZDK.Form.Read()	To get list of fields and their corresponding values in JSON format.
ZDK.Form.Write(JSON)	To set values for multiple fields in a page.
ZDK.Form.Field.Read(field_api_name)	To get the value of a specific field in a page.
ZDK.Form.Field.Write(field_api_name,value)	To set the value of a specific field in a page.
ZDK.Form.Field.Mandate(field_api_name)	To set a field as mandatory.
ZDK.UI.Freeze()	To freeze a page.
ZDK.UI.UnFreeze()	To un-freeze a page.
ZDK.App.Page_Info()	To get information about the current page.
ZDK.App.URL()	To get the URL of the current page.

In addition to performing the above UI actions, you can also trigger REST API calls, execute functions, and invoke connections using ZDK. It gets auto-bundled in the client script environment. Also, the system implicitly handles both authentication and authorization. Here are the sample codes for all the API methods available in our ZDK.

7b. Record Operations (REST API)

a. SingleCreate: To create a record in a module.

```
var lead = new ZDK.Apps.CRM.Leads();

lead.Last_Name = "dot sdK";

var response = lead.create();

log(response);
```

b. **SingleUpdate:** To update a record in a module.

```
var lead = ZDK.Apps.CRM.Leads.fetchById("4312588000000381015");

lead.Last_Name = "updated Name"

var response = lead.update();

log(response);
```

c. **SingleDelete:** To delete a record in a module.

```
var lead = ZDK.Apps.CRM.Leads.fetchById("4312588000000381015")

var response = lead.remove();

log(response);
```

d. **Create:** To create multiple records in a module.

```
var leads = new Array();

var lead1 = new ZDK.Apps.CRM.Leads();

lead1.Last_Name = "first lead";

leads.push(lead1);

var lead2 = new ZDK.Apps.CRM.Leads();
```

```
lead2.Last_Name = "second lead";

leads.push(lead2);

var response = ZDK.Apps.CRM.Leads.create(leads);

log(response);
```

e. **FetchRecords:** To retrieve all the records in a module.

```
var leads = ZDK.Apps.CRM.Leads.fetch()

for(var i=0; i<leads.length; i++){

        log(leads[i].Last_Name);

}
```

f. **FetchUsingPageNumber:** To retrieve all the records in a page.

```
var leads = ZDK.Apps.CRM.Leads.fetch(1);

for(var i=0; i<leads.length; i++){

        log(leads[i].Last_Name);

}
```

g. **FetchParams:** To filter records based on parameters.

```
var leads = ZDK.Apps.CRM.Leads.fetch(1, 20, "Last_Name", "desc");

for(var i=0; i<leads.length; i++){

        log(leads[i].Last_Name);

}
```

h. **Update:** To update multiple records in a module.

```
var leadsArray = new Array();
```

```
var lead1 = ZDK.Apps.CRM.Leads.fetchById("423383000005261048");

lead1.Last_Name = "Updated Name";

leadsArray.push(lead1);

var lead2 = ZDK.Apps.CRM.Leads.fetchById("423383000005267521");

lead2.Last_Name = "Updated Name2";

leadsArray.push(lead2);

var response = ZDK.Apps.CRM.Leads.update(leadsArray);

log(response[0].status);
```

i. **Delete:** To delete multiple records in a module.

```
var idsArray = new Array();

idsArray.push("431258800000381013");

var response = ZDK.Apps.CRM.Leads.deleteByIds(idsArray);

log(response);
```

j. **UsageOfPicklist:** To add values to picklist fields of a record.

```
var lead = new ZDK.Apps.CRM.Leads();

lead.Last_Name = "picklist test";

lead.Lead_Status = ZDK.Apps.CRM.Leads.Models.Lead_StatusPicklist.CONTACTED;

var response = lead.create();

log(response);
```

k. UsageOfMultiselectPicklist: To add values to multi-select picklist fields of a record.

```
var lead = new ZDK.Apps.CRM.Leads();
```

```
lead.Last_Name = "test";

var multi = new Array();

multi.push("Option 1");

multi.push("Option 2");

lead.Mutli_Select_1 = multi;

var response = lead.create()

log(response);
```

l. **UsageOfLookup:** To add values to lookup fields of a record. The getters and setters are methods **only** for lookup fields.

```
var lead = ZDK.Apps.CRM.Leads.fetchById("4312588000000381015");

var user = lead.Created_By();

log(user.state);
```

m. **LookupCreate:** To create a record and add it to the lookup field of a record.

```
var account = new ZDK.Apps.CRM.Accounts();

account.Account_Name = "test";

var parent_account = ZDK.Apps.CRM.Accounts.fetchById("2000000065057");

account.Parent_Account(parent_account);

account.create();
```

n. **UsageOfMultiselectLookup:** To add values to the multi-select lookup of a record.

```
var multi = new ZDK.Apps.CRM.LinkingModule3();

var lead = ZDK.Apps.CRM.Leads.fetchById("2000000074021");
```

```
multi.lead(lead);

var contact = ZDK.Apps.CRM.Contacts.fetchById("2000000065057");

multi.Alternate_Contacts(contact);

var response = multi.create();

log(response);
```

n. **UsageOfLookupDirectly:** To directly assign lookup_id rather than fetching an object through id and assigning an object using the field (field_name)_Lookup_Id.

```
var account = new ZDK.Apps.CRM.Accounts();

account.Account_Name = "test";

account.Parent_Account_Lookup_Id = "2000000065057";

account.create();
```

o. **CreateSubforms:** To create subforms.

```
var lead = new ZDK.Apps.CRM.Leads();

lead.Last_Name = "name";

var subform = new ZDK.Apps.CRM.Subform_1();

subform.Single_Line_1 = "single line";

var subforms = new Array();

subforms.push(subform);

lead.Subform_1 = subforms;

var response = lead.create();

log(response);
```

p. **FetchSubforms:** To read subforms.

```
var subforms = ZDK.Apps.CRM.Subform_1.fetch();

log(subforms);
```

q. **ConvertLeads:** To convert leads into contacts or accounts.

```
var convertLead = new ZDK.Apps.CRM.Leads.Models.ConvertLeads();

convertLead.Accounts = "735220000004375005";

convertLead.Deals = {

        "Pipeline":
ZDK.Apps.CRM.Deals.Models.PipelinePicklist.STANDARDSTANDARD,

        "Deal_Name": "Robert",

        "Closing_Date": "2020-11-20",

        "Stage": "Closed Won",

        "Amount": 56.6

    };

var response = ZDK.Apps.CRM.Leads.convert(convertLead, "735220000003635975");
```

r. **AddingSpecialInputsToRecord:** To add Line_Items to records in the inventory modules, Pricing_Model in price books, and Participants in Events.

```
var invoice = new ZDK.Apps.CRM.Invoices();

invoice.Subject = "testing2";

var account = ZDK.Apps.CRM.Accounts.fetchById("2000000136022");

invoice.Account_Name(account);

var line_item = new ZDK.Apps.CRM.Invoices.Models.Line_Items();

var product = ZDK.Apps.CRM.Products.fetchById("2000000136027");
```

```
line_item.product(product);

line_item.quantity=3.0;

var line_items = new Array();

line_items.push(line_item);

invoice.Product_Details=line_items;

invoice.$line_tax = [{ "percentage": 8.7, "name": "VAT"} ]

var response = invoice.create();

log(response);
```

s. **CreatePriceBooks:** To create PriceBooks.

```
var book = new ZDK.Apps.CRM.Price_Books();

book.Price_Book_Name = "test";

book.Pricing_Model = ZDK.Apps.CRM.Price_Books.Models.Pricing_ModelPicklist.FLAT;

var pricing_detail = new ZDK.Apps.CRM.Price_Books.Models.Pricing_Details();

pricing_detail.from_range = 10;

pricing_detail.to_range = 100;

pricing_detail.discount = 5;

var pricing_detail2 = new ZDK.Apps.CRM.Price_Books.Models.Pricing_Details();

pricing_detail2.from_range = 1000;

pricing_detail2.to_range = 2000;

pricing_detail2.discount = 10;

var details = new Array();

details.push(pricing_detail);
```

```
details.push(pricing_detail2);

book.Pricing_Details = details;

var response = book.create();

log(response);
```

t. **AddParticipants:** To add participants to events.

```
var event = new ZDK.Apps.CRM.Events();

event.Event_Title = "testing";

event.Start_DateTime = "2020-10-22T21:00:00+05:45";

event.End_DateTime = "2020-10-24T21:00:00+06:45";

event.Description = "test";

event.Currency = "USD";

var participant = new ZDK.Apps.CRM.Events.Models.Participants();

participant.participant= "423383000004997180";

participant.type= "lead";

var participants = new Array();

participants.push(participant);

event.Participants = participants;

event.Tag = [{'id': "735220000001193166"}];

var response = event.create();

log(response);
```

7c. Execute Functions

a. **ExecuteFunctions:** To trigger functions that are enabled as REST API.

```
var response = ZDK.Apps.CRM.Functions.execute("dealtoquote2")

log(response);
```

b. **ExecuteFunctionsWithParams:** To trigger functions along with parameters.

```
//parameters is of json type - {"amount": 100, "name": "test"}

var response = ZDK.Apps.CRM.Functions.execute("dealtoquote2", parameters);

log(response);
```

7d. InvokeConnections

a. **InvokeConnections:** To invoke connections.

```
//headers and parameters are json, if they are empty need to pass empty json {}

var response = ZDK.Apps.CRM.Connections.invoke("mailchimp4",
"http://mailchimp.api/sample_api",  "POST", 1, parameters, headers);

log(response);
```