Blueprint Documentation: Admin_Environmental_Assessment

Blueprint Link: Zoho CRM Blueprint

Stages in Blueprint

1. **Project Created**
2. **Survey Scheduled**
3. **Report Completed – Under Review**
4. **Send Report**
5. **Completed**

Transitions & Logic

1. Schedule Survey

- **Before Transition**:
  - Ensure Status is not set to *Paused* or *Cancelled*.
- **During Transition**:
  - Zone (from Service Address): Optional
  - Survey Date: Mandatory
  - **Validation**: Survey Date must be today or a future date (not in the past).
- **After Transition**:
  - Auto-update field: Integrated Stages → *Inspection Scheduled*

2. Confirm Report Completion

- **During Transition**:
  - Mandatory checkbox selection. Options must include:
    - *Site photos and survey map uploaded*
    - *Draft report completed*
- **After Transition**:
  - Auto-update field: Integrated Stages → *Report/Application In Progress*

3. Upload Report

- **During Transition**:
  - Attachment is mandatory.
- **After Transition**:
  - Custom Function: Admin_Get_Attachments_Update_URL
- **CODE :**

```
1 void
  automation.Admin_Get_Attachements_Update_URL(
  Int service_id)
2 {
3 serviceDetails =
  zoho.crm.getRecordById("Service_Request",serv
  ice_id);
4 //     info serviceDetails;
5 CRMresponse =
  zoho.crm.getRelatedRecords("Attachments","Ser
```

```
  vice_Request",service_id);
6 info CRMresponse;
7 if(CRMresponse.size() > 0)
8 {
9    attachment_id = CRMresponse.getJSON("id");
10   info attachment_id;
11   attachment_name =
  CRMresponse.getJSON("File_Name");
12   info attachment_name;
13   serviceWorkdriveUrl =
  serviceDetails.get("Workdrive_folder_url");
14   info serviceWorkdriveUrl;
15   folderId =
  serviceWorkdriveUrl.getSuffix("folder/");
16   info folderId;
17   response = invokeurl
18   [
19       url
  :"https://www.zohoapis.com/crm/v3/Service_Req
  uest/" + service_id + "/Attachments/" +
  attachment_id
20       type :GET
21       connection:"crm"
22   ];
23   info response;
24   //  response.setParamName("file");
25   file_name = attachment_name;
```

```
26   upload =
  zoho.workdrive.uploadFile(response,folderId,f
  ile_name,false,"zohoworkdrive");
27   info upload;
28 }
```

}

Function Purpose:

- Fetch first attachment from the related Service Request.
- Extract and use WorkDrive folder ID from the record.
- Download the attachment via Zoho CRM API.
- Upload it to WorkDrive using the folder ID.
- Ensures proper syncing between CRM and WorkDrive.

4. Send Report/Application

- **After Transition**:
  - Update Integrated Stages → *Service Request Completed*

Custom Function:
Admin_Customer_portal_url_And_Invoice(request_id)

Detailed Function Explanation:

1. **Customer Portal Invitation**

a. Fetches Service Request using request_id.

b. If Customer_portal = False, it sends a portal invite using CRM API.

c. On success, updates Customer_portal to True.

2. **Email Notification (Conditional)**

a. Checks if other *Environmental Assessment* Service Requests exist with same Contact Email and Customer Portal = True.

b. Sends email using Template ID: 61717370000092900 81.

3. **Invoice Creation in Zoho Books**

a. Retrieves related Lead and its Zoho_Book_Customer_Id.

b. Creates invoice using:

i. Unit Price

ii. Service Type

iii. Permit Application Fee line item

c. If Payment Method = Credit Card and Quote_Action_Type = Send quote and request payment method to be saved:

i. Adds 3.1% adjustment charge for Credit Card.

ii. Updates the invoice with:

1. Custom field: Payment Mode
2. Adjustment details

4. **Service Request Update**

a. Saves the following back to the record:

i. Zoho_Book_Invoice_Id

ii. Zoho_Book_Invoice_URL

5. **Billing Event Creation**

a. Creates a Billing Event with:

i. Name

ii. Service Type

iii. Project

iv. Date

v. Amount

vi. Links it to the Service Request

CODE :

```
void automation.Admin_Customer_portal_url_And_Invoice(Int request_id)
{
serviceRequestDetail = zoho.crm.getRecordById("Service_Request",request_id);
if(serviceRequestDetail.isEmpty())
{
   info "Service request not found.";
   return;
}
customerPortal = serviceRequestDetail.get("Customer_portal");
contactName = serviceRequestDetail.get("Contact_Name");
if(customerPortal == "False" && contactName != null)
{
   contactId = contactName.get("id");
   portalUserType = "617173700008301016";
   apiurl = "https://www.zohoapis.com/crm/v5/Contacts/" +
```

```
  contactId +
  "/actions/portal_invite?user_type_id=" +
  portalUserType +
  "&type=invite&language=en_US";
16    response = invokeurl
17    [
18        url :apiurl
19        type :POST
20        connection:"crm"
21    ];
22    if(response.contains("portal_invite"))
23    {
24        portal_invite =
  response.get("portal_invite").get(0).get("cod
  e");
25        if(portal_invite == "SUCCESS")
26        {
27            portalMap = Map();
28
  portalMap.put("Customer_portal","True");
29            update_Service =
  zoho.crm.updateRecord("Service_Request",reque
  st_id,portalMap);
30            info update_Service;
31        }
32    }
33 }
```

```
34 //search record email base pr
35 // emailID = "itsviantest1423@gmail.com"; //
  Replace with actual email ID
36 // searchRecords =
  zoho.crm.searchRecords("Service_Request",
  "Contact_Email:equals:" + emailID + ")");
37 // Fetch records based on Contact Email
38 emailID =
  serviceRequestDetail.get("Contact_Email");
39 info emailID;
40 if(emailID != null)
41 {
42   searchRecords =
  zoho.crm.searchRecords("Service_Request","(Co
  ntact_Email:equals:" + emailID + ") and
  (Category:equals:Environmental Assessment)
  and (Customer_portal:equals:True)");
43   info searchRecords;
44   if(searchRecords.size() > 0)
45   {
46       for each  record in searchRecords
47       {
48           contact_email =
  serviceRequestDetail.get("Contact_Email");
49           owner =
  serviceRequestDetail.get("Owner");
50           //              if(owner != null &&
```

```
  contact_email != null)
51            //              {
52            owner_name = owner.get("name");
53            owner_email = owner.get("email");
54            mailData =
  {"data":{{"from":{"user_name":owner_name,"ema
  il":owner_email},"to":{{"user_name":contactNa
  me.get("name"),"email":contact_email}},"templ
  ate":{"id":"617173700009290081"}}}};
55            requestHeaders = {"Content-
  Type":"application/json"};
56            mailUrl =
  "https://www.zohoapis.com/crm/v7/Service_Requ
  est/" + request_id + "/actions/send_mail";
57            body = mailData.toString();
58            mailRequest = invokeurl
59            [
60                url :mailUrl
61                type :POST
62                parameters:body
63                headers:requestHeaders
64                connection:"crm"
65            ];
66            info mailRequest;
67        }
68    }
69 }
```

```
70 ////////////////////////////// 5 June 2025
   //////////////////////////////////////
71 serviceRequestDetail =
   zoho.crm.getRecordById("Service_Request",requ
   est_id);
72 contactName =
   serviceRequestDetail.get("Contact_Name");
73 projectName =
   serviceRequestDetail.get("Project_Name");
74 unit_price =
   serviceRequestDetail.get("Unit_Price");
75 payment_method =
   serviceRequestDetail.get("Payment_Method");
76 quote_action_type =
   serviceRequestDetail.get("Quote_Action_Type")
   ;
77 invoice_amount = unit_price;
78 serviceTotalFee =
   serviceRequestDetail.get("Total_Fees");
79 service_Type =
   serviceRequestDetail.get("Service_Type");
80 serviceCategoryName =
   serviceRequestDetail.get("Category");
81 service_Type_Name =
   ifnull(service_Type.get("name"),"");
82 if(contactName != NULL)
83 {
```

```
84   contactId = contactName.get("id");
85 }
86 contactRecord =
  ifnull(zoho.crm.getRecordById("Contacts",cont
  actId),Map());
87 leadId =
  ifnull(contactRecord.get("Lead_Name").get("id
  "),null);
88 if(leadId != NULL)
89 {
90   leadRecords =
  zoho.crm.getRecordById("Leads",leadId);
91   ZohoBooksCustomerId =
  ifnull(leadRecords.get("Zoho_Book_Customer_Id
  "),null);
92   if(ZohoBooksCustomerId != NULL)
93   {
94       // Create Invoice Data
95       invoiceData = Map();
96
  invoiceData.put("customer_id",ZohoBooksCustom
  erId);
97
  invoiceData.put("date",zoho.currentdate);
98
  invoiceData.put("expiry_date",zoho.currentdat
  e.addDay(30));
```

```
99        productNames = {"Gopher Tortoise
  Mitigation","Permit Application"};
100       lineItemsList = List();
101       lineItem = Map();
102       lineItem.put("name","Permit
  Application Fee");
103       lineItem.put("rate",invoice_amount);
104       lineItem.put("quantity",1);
105       lineItemsList.add(lineItem);
106
  invoiceData.put("line_items",lineItemsList);
107       // Create Invoice
108       InvoiceCreateResponse =
  zoho.books.createRecord("Invoices","848446319
  ",invoiceData,"zohobooks");
109       info "InvoiceCreateResponse" +
  InvoiceCreateResponse;
110
  if(InvoiceCreateResponse.containKey("invoice"
  ))
111       {
112           Invoice_id =
  InvoiceCreateResponse.get("invoice");
113           id = Invoice_id.get("invoice_id");
114           ////////////////// 6 June 2025
  ////////////////////////////
115           sub_total =
```

```
  Invoice_id.getJSON("sub_total");
116           credit_card_charge = sub_total *
  0.031;
117           custom_field_list = list();
118           payment_mode = Map();
119           payment_mode.put("label","Payment
  Mode");
120
  payment_mode.put("value",payment_method);
121
  custom_field_list.add(payment_mode);
122           if(payment_method == "Credit Card"
  && quote_action_type == "Send quote and
  request payment method to be saved")
123           {
124               invoice_book_map = Map();
125
  invoice_book_map.put("custom_fields",custom_f
  ield_list);
126
  invoice_book_map.put("reason","Credit Card
  Charge");
127
  invoice_book_map.put("adjustment_account_name
  ",payment_method);
128
  invoice_book_map.put("adjustment",credit_card
  _charge);
```

```
129
  invoice_book_map.put("adjustment_description"
  ,"Credit Card Charge");
130               zoho_book_invoice_update =
  zoho.books.updateRecord("invoices","848446319
  ",id,invoice_book_map,"zohobooks");
131               info zoho_book_invoice_update;
132           }
133           ////////////////// 6 June 2025
  ////////////////////////////
134           service_map = Map();
135
  service_map.put("Zoho_Book_Invoice_Id",id);
136
  service_map.put("Zoho_Book_Invoice_URL","http
  s://books.zoho.com/app/848446319#/invoices/"
  + id);
137           // Update Service Request
138           update_service =
  zoho.crm.updateRecord("Service_Request",reque
  st_id,service_map);
139           info update_service;
140       }
141   }
142 }
143 service_request_details =
  zoho.crm.getRecordById("Service_Request",requ
```

```
  est_id);
144 //info service_request_details;
145 name =
  service_request_details.getJSON("Name");
146 //info name;
147 projectId = projectName.get("id");
148 quote_amount =
  serviceRequestDetail.get("Unit_Price");
149 billing_events_map = Map();
150 billing_events_map.put("Name",name);
151 billing_events_map.put("Service_Type","Goph
  er Tortoise");
152 billing_events_map.put("Date",zoho.currentd
  ate);
153 billing_events_map.put("Project",projectId)
  ;
154 billing_events_map.put("Service_Request",re
  quest_id);
155 billing_events_map.put("Amount",quote_amoun
  t);
156 //create Billing Event record /////
157 create_billing_event =
  zoho.crm.createRecord("Billing_Events",billin
  g_events_map);
158 ////////////////////////////// 5 June 2025
  ////////////////////////////////////////////
```

```
}
```

5. Exit

- **Before Transition**:
  - Ensures Created Time is not empty before proceeding.