Admin Gopher Tortoise Blueprint Process

1. Purpose

This blueprint is designed to manage and automate the Gopher Tortoise permit process within the **Service Requests** module. It streamlines the workflow from project initiation to excavation and final permit closure, ensuring regulatory compliance and operational efficiency.

2. Blueprint Trigger Condition

- **Module:** Service Requests
- **Trigger Condition:**
  - **Stage** = `Project Created`
  - **Category** = `Gopher Tortoise`

3. Stages in the Process

The following are the defined **stages** (white boxes) in the process:

1. Project Created
2. Inspection Scheduled
3. Survey Completed
4. Customer Information Pending
5. Customer Information Received
6. Ready for Application Submission
7. Approval Pending
8. Permit Rejected
9. Permit Issued
10. Signed Permit
11. Excavation Scheduled
12. Completed
13. Completed

4. Transitions in the Process

The following are the **transitions** (blue arrows) used to move between stages:

Transition Name	Details
Schedule Survey	- **During:** The field `s` is optional. Confirm survey results. - **After:** Integrated stage updated to `Inspection Scheduled`.
No Active Gopher Tortoise Burrows Found	- **Before:** Checks if `Created Time` is empty. - **After:** Updates integrated stage to `Inspection Completed`. - **Custom Action:** Admin creates survey with service type. **Function Purpose :** This function is designed to **create a new Product record** in Zoho CRM based on data from a **Service Request**, and then update the Service Request with a reference to the newly created Product. `void automation.Admin_Create_Survey_Service_type(Int record_id)` `{` `serviceRequestDetail = zoho.crm.getRecordById("Service_Request",record_id);` `//  info serviceRequestDetail;` `serviceName = "Gopher Tortoise Survey Service Type";` `//  info serviceName;` `surveyServiceType = serviceRequestDetail.get("Survey_Service_Type");`

	8 // info surveyServiceType; 9 servieFee = serviceRequestDetail.get("Total_Fees"); 10 // info servieFee; 11 serviceDescription = serviceRequestDetail.get("Description"); 12 // info serviceDescription; 13 service_typeMap = Map(); 14 service_typeMap.put("Product_Name",serviceName); 15 service_typeMap.put("Unit_Price",servieFee); 16 service_typeMap.put("Description",serviceDescription); 17 // info service_typeMap; 18 createRecord = zoho.crm.createRecord("Products",service_typeMap); 19 info createRecord; 20 createRecordId = createRecord.getJSON("id"); 21 // info createRecordId; 22 service_Map = Map(); 23 service_Map.put("Survey_Service_Type",createRecordId); 24 updateRecord = zoho.crm.updateRecord("Service_Request",record_id,service_Map); 25 info updateRecord; 26 }
Customer Information	During: Notes(optional) Owner Type (mandatory)

	Number of Potentially Occupied Burrows(mandatory) Estimated Number of Tortoises(mandatory) - **After:** Updates integrated stage to Customer Information Pending **- Custom Actions : Admin_Email_Notification_For_Customer_Information** **Function Brief:** This function **sends an email notification** to the **highest-ranked contact** associated with a specific **Service Request**, using a predefined email template based on the **Applicant Type**. `void automation.Admin_Email_Notification_For_Customer_Information (Int request_id)` `{` `// Fetch service request details by ID` `serviceRequestDetail = zoho.crm.getRecordById("Service_Request",request_id);` `// Extract service request name` `service_request_name = serviceRequestDetail.getJSON("Name");` `// Extract owner details` `owner = serviceRequestDetail.get("Owner");`

```
info owner;
owner_name = owner.get("name");
info owner_name;
owner_email = owner.getJSON("email");
info owner_email;
////////////// Owner Type ////////////////////////////////
applicant_type = serviceRequestDetail.getJSON("Applicant_Type");
////////////////// 4/4/2025 ////////////////////////////
// Get related contacts from the Service_Request module
related_contacts = zoho.crm.getRelatedRecords("Contacts","Service_Request",request_id);
info related_contacts;
if(related_contacts.size() > 0)
{
    for each  contact in related_contacts
    {
        // Extract contact ID and details
        contact_id = contact.getJSON("id");
        contact_details = zoho.crm.getRecordById("Contacts",contact_id);
        // Extract contact email
```

```
28          contact_email =
contact_details.getJSON("Email
");
29          info contact_email;
30
////////////////////////
21/4/2025
////////////////////////////
31          // Check if contact is
marked as highest ranked
32          higest_ranked_contact
=
contact_details.getJSON("Highe
st_Ranked_Contact");
33          info
higest_ranked_contact;
34          // Only proceed if the
contact is marked as highest
ranked (not null or empty)
35
if(higest_ranked_contact !=
null && higest_ranked_contact
!= "")
36          {
37
////////////////////////
21/4/2025
////////////////////////////
38              // Prepare email
based on applicant type
39              if(applicant_type
== "Individual Landowner")
40              {
41                  mailData =
{"data":{{"from":{"user_name":
owner_name,"email":owner_email
},"to":{{"user_name":service_r
```

```
equest_name,"email":contact_em
ail}},"template":{"id":"617173
7000009129051"}}}};
42              }
43              if(applicant_type
== "Corporate Landowner")
44              {
45                   mailData =
{"data":{{"from":{"user_name":
owner_name,"email":owner_email
},"to":{{"user_name":service_r
equest_name,"email":contact_em
ail}},"template":{"id":"617173
7000009535149"}}}};
46              }
47              // Setup headers
and endpoint
48              requestHeaders =
Map();
49
requestHeaders.put("Content-
Type","application/json");
50              mailUrl =
"https://www.zohoapis.com/crm/
v7/Service_Request/" +
request_id +
"/actions/send_mail";
51              body =
mailData.toString();
52              // Send the email
via invokeurl
53              mailRequest =
invokeurl
54              [

55                   url :mailUrl
```

	`` 56 type :POST 57 parameters:body 58 headers:requestHeaders 59 connection:"crm" 60]; 61 info mailRequest; 62 } 63 else 64 { 65 // Debug info if highest ranked contact is not present 66 info "Contact skipped – Highest Ranked Contact field is empty or null."; 67 } 68 } 69 } 70 } ```
Customer Information Received	- **During :** Notes(optional) Upload Customer Checklist (message we are showing) Attachments (Optional) **- After :** Integrated Stages update with the Report/Application In Progress

Recipient Site Letter	**During :** Relocation Type (optional) Comment (optional) Upload Recipient Site Letter (showing message) Attachments (optional) - **After** : Custom Actions **Function Name :** Admin_Request_for_Reservation_Letter_for_Service **Function Brief :** This function **automatically sends a reservation letter email** (using a email template) to a vendor when the **relocation type is "Offsite"** for a specific **Service Request**.

```
void automation.Admin_Request_for_Reservation_Letter_for_Service(Int request_id)
{
serviceRequestDetail = zoho.crm.getRecordById("Service_Request",request_id);
// info serviceRequestDetail;
// contact_email = serviceRequestDetail.getJSON("Contact_Email");
// info contact_email;
service_request_name = serviceRequestDetail.getJSON("Name");
owner = serviceRequestDetail.get("Owner");
info owner;
owner_name = owner.get("name");
info owner_name;
owner_email = owner.getJSON("email");
```

```
info owner_email;
relocation_type = serviceRequestDetail.get("On_Site1");
vendor_email = serviceRequestDetail.get("Recipient_Site_Email");
////////////////// 4/4/2025 ////////////////////////
// Get related contacts
// related_contacts = zoho.crm.getRelatedRecords("Contacts","Service_Request",request_id);
// info related_contacts;
// if(related_contacts.size() > 0)
// {
//  for each  contact in related_contacts
//  {
//       contact_id = contact.getJSON("id");
//       contact_details = zoho.crm.getRecordById("Contacts",contact_id);
//       contact_email = contact_details.getJSON("Email");
//       info contact_email;
////////////////// 4/4/2025 ////////////////////////
if(relocation_type == "Offsite")
{
    mailData = {"data":{{"from":{"user_name":owner_name,"email":owner_email},"to":{{"user_name":service_request
```

	_name,"email":vendor_email}},"template":{"id":"617173700000939358"}}}}; requestHeaders = Map(); requestHeaders.put("Content-Type","application/json"); mailUrl = "https://www.zohoapis.com/crm/v7/Service_Request/" + request_id + "/actions/send_mail"; body = mailData.toString(); mailRequest = invokeurl [url :mailUrl type :POST parameters:body headers:requestHeaders connection:"crm"]; info mailRequest; } // } // } }
Submit Application	1. **During :** Mitigation Fee paid by Client (mandatory) Permit Application Number (mandatory) Mitigation Fee Amount (mandatory) Additional permits required (mandatory) **After:**We are updating the Integrated stages with value "Permit Approval Pending" **Custom Action:** Function **Function Name:** Admin_Create_Invoice_On_Submit_Application_Gopher

	Function Brief: Creating an **invoice in Zoho Books** for a Gopher Tortoise service request, 2. Updating the **Service Request record** with invoice details, 3. Creating a **CRM Task** for the Finance team, 4. Creating a **Billing Event** `void automation.Admin_Create_Invoice_On_Submit_Application_Gopher (Int request_id)` `{` `serviceRequestDetail = zoho.crm.getRecordById("Service_Request",request_id);` `contactName = serviceRequestDetail.get("Contact_Name");` `projectName = serviceRequestDetail.get("Project_Name");` `// Tortoise_Mitigation_Fee = serviceRequestDetail.get("Tortoise_Mitigation_Fee");` `unit_price = serviceRequestDetail.get("Unit_Price");` `payment_method = serviceRequestDetail.get("Payment_Method");` `quote_action_type = serviceRequestDetail.get("Quote_Action_Type");` `////////////////////// 5 June 2025 ///////////////`

```
fwc_mitigation_fee = serviceRequestDetail.getJSON("Mitigation_amount_input1");
mitigation_fee_checkbox = serviceRequestDetail.getJSON("Mitigation_fee_checkbox");
////////////////////// 5 June 2025 //////////////
if(projectName != null)
{
    projectId = projectName.get("id");
    project_details = zoho.crm.getRecordById("Deals",projectId);
    account_name = project_details.getJSON("Account_Name");
    if(account_name != null)
    {
        account_id = account_name.getJSON("id");
        account_details = zoho.crm.getRecordById("Accounts",account_id);
        is_strategic_account = ifnull(account_details.getJSON("Is_strategic_Account"),"");
        payment_mode = ifnull(account_details.get("Payment_mode"),"");
    }
}
serviceTotalFee = serviceRequestDetail.get("Total_Fees");
service_Type =
```

```
serviceRequestDetail.get("Service_Type");
serviceCategoryName = serviceRequestDetail.get("Category");
service_Type_Name = ifnull(service_Type.get("name"),"");
if(contactName != NULL)
{
    contactId = contactName.get("id");
}
////////////////// Fetching the contact record ////////////////////
contact_details = zoho.crm.getRecordById("Contacts",contactId);
leadId = contact_details.get("Lead_Name").get("id");
if(leadId != NULL)
{
    ////////////////// Fetching the Lead record ////////////////////
    leadRecords = zoho.crm.getRecordById("Leads",leadId);
    ZohoBooksCustomerId = ifnull(leadRecords.get("Zoho_Book_Customer_Id"),null);
    if(ZohoBooksCustomerId != NULL)
    {
        // Create Invoice Data
```

```
46          invoiceData = Map();
47  invoiceData.put("customer_id",
    ZohoBooksCustomerId);
48  invoiceData.put("date",zoho.cu
    rrentdate);
49  invoiceData.put("expiry_date",
    zoho.currentdate.addDay(30));
50          // Creating Line Items
    List
51          productNames =
    {"Gopher Tortoise
    Mitigation","Permit
    Application"};
52  ////////////////////// ADD
    LINE ITEMS
    ////////////////////////
53          lineItemsList =
    List();
54  if(mitigation_fee_checkbox ==
    false)
55          {
56                  // Line Item 1:
    Permit Application Fee
57                  permitFeeItem =
    Map();
58  permitFeeItem.put("name","Perm
    it Application Fee");
59  permitFeeItem.put("rate",unit_
```

```
price);
60
permitFeeItem.put("quantity",1
);
61
lineItemsList.add(permitFeeIte
m);
62                // Line Item 2:
FWC mitigation fee
63                sanctuaryFeeItem
= Map();
64
sanctuaryFeeItem.put("name","F
WC mitigation paid by IVA");
65
sanctuaryFeeItem.put("rate",fw
c_mitigation_fee);
66
sanctuaryFeeItem.put("quantity
",1);
67
lineItemsList.add(sanctuaryFee
Item);
68            }
69
if(mitigation_fee_checkbox ==
true)
70            {
71                permitFeeItem =
Map();
72
permitFeeItem.put("name","Perm
it Application Fee");
73
permitFeeItem.put("rate",unit_
price);
74
```

```
permitFeeItem.put("quantity",1);
lineItemsList.add(permitFeeItem);
        }
        custom_field_list = list();
        custom_field_map = Map();
custom_field_map.put("label","Is strategic Account?");
custom_field_map.put("value",is_strategic_account);
        payment_mode = Map();
payment_mode.put("label","Payment Mode");
payment_mode.put("value",payment_method);
custom_field_list.add(payment_mode);
custom_field_list.add(custom_field_map);
invoiceData.put("custom_fields",custom_field_list);
invoiceData.put("line_items",lineItemsList);
////////////////////////
```

```
Create Invoice
//////////////////////////
89          InvoiceCreateResponse =
zoho.books.createRecord("Invoices","848446319",invoiceData,"zohobooks");
90          info "InvoiceCreateResponse" + InvoiceCreateResponse;
91
if(InvoiceCreateResponse.containKey("invoice"))
92          {
93                  Invoice_id = InvoiceCreateResponse.get("invoice");
94                  id = Invoice_id.get("invoice_id");
95                  ////////////////
6 June 2025
//////////////////////////////
96                  sub_total = Invoice_id.getJSON("sub_total");
97
credit_card_charge = sub_total * 0.031;
98                  custom_field_list = list();
99                  payment_mode = Map();

100
payment_mode.put("label","Payment Mode");
```

```
101
   payment_mode.put("value",payme
   nt_method);
102
   custom_field_list.add(payment_
   mode);
103             if(payment_method
   == "Credit Card" &&
   quote_action_type == "Send
   quote and request payment
   method to be saved")
104             {
105
   invoice_book_map = Map();
106
   invoice_book_map.put("custom_f
   ields",custom_field_list);
107
   invoice_book_map.put("reason",
   "Credit Card Charge");
108
   invoice_book_map.put("adjustme
   nt_account_name",payment_metho
   d);
109
   invoice_book_map.put("adjustme
   nt",credit_card_charge);
110
   invoice_book_map.put("adjustme
   nt_description","Credit Card
   Charge");
111
   zoho_book_invoice_update =
   zoho.books.updateRecord("invoi
   ces","848446319",id,invoice_bo
   ok_map,"zohobooks");
112                 info
```

```
zoho_book_invoice_update;
113             }
114             ////////////////
6 June 2025
////////////////////////////

115             service_map =
Map();
116
service_map.put("Zoho_Book_Inv
oice_Id",id);
117
service_map.put("Zoho_Book_Inv
oice_URL","https://books.zoho.
com/app/848446319#/invoices/"
+ id);
118             // Update Service
Request
119             update_service =
zoho.crm.updateRecord("Service
_Request",request_id,service_m
ap);
120             info
update_service;
121         }
122    }
123 }
124 ////////////////////// 31
March 2025
////////////////////////////
125 ////////////////// Fetching
the service request details
////////////////////////////
126 service_request_details =
zoho.crm.getRecordById("Servic
e_Request",request_id);
127 //info
```

```
service_request_details;
name = service_request_details.getJSON("Name");
//info name;
contact_id = service_request_details.get("Contact_Name").getJSON("id");
owner_id = service_request_details.getJSON("Owner").getJSON("id");
task_map = Map();
task_map.put("Subject","Task Creation for Finance Team " + name + "");
task_map.put("Owner",owner_id);
task_map.put("$se_module","Service_Request");
task_map.put("Due_Date",zoho.currentdate.addDay(2));
task_map.put("What_Id",request_id);
task_map.put("Who_Id",contact_id);
task_map.put("Status","Not Started");
task_map.put("Priority","High");
/////////////// Creating a task in zoho crm ////////////////////////////
create_task = zoho.crm.createRecord("Tasks",task_map);
info create_task;
```

	144 //----------30/05/2025---------- 145 projectId = projectName.get("id"); 146 quote_amount = serviceRequestDetail.get("Unit_Price"); 147 billing_events_map = Map(); 148 billing_events_map.put("Name",name); 149 billing_events_map.put("Service_Type","Gopher Tortoise"); 150 billing_events_map.put("Date",zoho.currentdate); 151 billing_events_map.put("Project",projectId); 152 billing_events_map.put("Service_Request",request_id); 153 billing_events_map.put("Amount",quote_amount); 154 //create Billing Event record ///// 155 create_billing_event = zoho.crm.createRecord("Billing_Events",billing_events_map); 156 }
Upload Issued Permit	**During:** Attach Permit File and onsite relocation map(show message) Atatchments(mandatory) Permit Issued date(mandatory) Permit number (mandatory) Permit expiry date (mandatory)

	After : Integrated stages update with Permit Issued
Permit Rejected	**During :** Notes (optional) **After:** Integrated stages update with Service Request completed
Signed Permit	**During :** Notes (optional) Attachments (optional)
Schedule Excavation	**During :** No of adult(s) relocated (mandatory) No of Juvenile(s) relocated (mandatory) Excavation Date (mandatory) **After:** Integrated stages update with Excavation Scheduled
Upload Gopher Tortoise Relocation Results	**During :** Recipient Site Name *(Mandatory)* Gopher Tortoise Relocation Site *(Mandatory)* Number of Adult Tortoises *(Mandatory)* Number of Adult Tortoises Relocated *(Mandatory)* Number of Juveniles *(Mandatory)* Number of Adults *(Mandatory)* Number of Subadults *(Mandatory)* ***After :*** Integrated stages update with Service Request completed **Custom Actions:** Function **Function Name:** Admin_Create_Invoice_using_Vendors_Tortoise_Information **Function Brief:** This **Deluge function** automates the creation of a **Zoho Books invoice** for a *Gopher Tortoise service request*, using data from Zoho CRM modules like **Service_Request**, **Deals**, **Accounts**, **Vendors**, and **Leads**.

```
1  void
   automation.Admin_Create_Invoic
   e_using_Vendors_Tortoise_Infor
   mation(Int record_id)
2  {
3  serviceRequestDetail =
   zoho.crm.getRecordById("Servic
   e_Request",record_id);
4  projectName =
   serviceRequestDetail.get("Proj
   ect_Name");
5  unit_price =
   serviceRequestDetail.get("Unit
   _Price");
6  payment_method =
   serviceRequestDetail.get("Paym
   ent_Method");
7  quote_action_type =
   serviceRequestDetail.get("Quot
   e_Action_Type");
8  if(projectName != NULL)
9  {
10    projectId =
   projectName.get("id");
11    project_details =
   zoho.crm.getRecordById("Deals"
   ,projectId);
12    account_name =
   project_details.getJSON("Accou
   nt_Name");
13    if(account_name != null)
14    {
15          account_id =
   account_name.getJSON("id");
16          account_details =
```

```
zoho.crm.getRecordById("Accounts",account_id);
        is_strategic_account = ifnull(account_details.getJSON("Is_strategic_Account"),"");
        payment_mode = ifnull(account_details.get("Payment_mode"),"");
    }
}
vendorId = serviceRequestDetail.getJSON("Vendor").getJSON("id");
vendor_map = Map();
vendor_map.put("Gopher_Tortoise_Relocation_Results",serviceRequestDetail.get("Gopher_Tortoise_Relocation_Results"));
vendor_map.put("Number_of_Adult_Tortoises_Fee",serviceRequestDetail.get("Number_of_Adult_Tortoises_Fee"));
vendor_map.put("Number_of_Adult_Tortoises",serviceRequestDetail.get("Number_of_Adult_Tortoises"));
number_of_juveniles = serviceRequestDetail.get("Number_of_Juveniles");
info number_of_juveniles;

vendor_map.put("Number_of_Juveniles",number_of_juveniles);

number_of_adults = serviceRequestDetail.get("Number_of_Adults");
```

```
info number_of_adults;
vendor_map.put("Number_of_Adults",number_of_adults);
number_of_subadults = serviceRequestDetail.get("Number_of_Subadults");
info number_of_subadults;
vendor_map.put("Number_of_Subadults",number_of_subadults);
////////////////////////// 5 june 2025 ////////////////////////////
vendor_fees = serviceRequestDetail.get("Vendor_Fees");
info vendor_fees;
vendor_map.put("Vendor_Fee",vendor_fees);
update_vendor = zoho.crm.updateRecord("Vendors",vendorId,vendor_map);
//////////////////Fetch tortoise information //////////////
vendorDetails = zoho.crm.getRecordById("Vendors",vendorId);
tortoiseResult = vendorDetails.get("Gopher_Tortoise_Relocation_Results");

info tortoiseResult;

adultTortoiseFee = ifnull(vendorDetails.get("Number_of_Adult_Tortoises_Fee"),"");
```

```
adultTortoise = ifnull(vendorDetails.get("Number_of_Adult_Tortoises"),"");
JuvenilesValue = ifnull(vendorDetails.get("Number_of_Juveniles"),"");
serviceTotalFee = 0.0;
if(vendor_fees != null && number_of_adults != null && number_of_juveniles != null && number_of_subadults != null)
{
    sanctuary_fee = vendor_fees * (number_of_adults + number_of_juveniles + number_of_subadults);
}
info "sanctuary_fee" + sanctuary_fee;
on_site = serviceRequestDetail.get("On_Site1");
if(serviceRequestDetail != NULL)
{
    projectId = serviceRequestDetail.get("id");
}
contactName = serviceRequestDetail.get("Contact_Name");
service_Type = serviceRequestDetail.get("Service_Type");
serviceCategoryName = serviceRequestDetail.get("Cate
```

```
gory");
service_Type_Name =
service_Type.get("name");
if(contactName != Null)
{
    contactId =
contactName.get("id");
}
///////////////// Fetching the
contact record
//////////////////////
contact_details =
zoho.crm.getRecordById("Contac
ts",contactId);
leadId =
contact_details.get("Lead_Name
").get("id");
///////////////// Fetching the
Lead record
//////////////////////
lead_details =
zoho.crm.getRecordById("Leads"
,leadId);
ZohoBooksCustomerId =
lead_details.getJSON("Zoho_Boo
k_Customer_Id");
invoiceData = Map();
///////////////// Fetching the
zoho book customer record id
//////////////////////

invoiceData.put("customer_id",
ZohoBooksCustomerId);

invoiceData.put("date",zoho.cu
rrentdate);
invoiceData.put("expiry_date",
```

```
    zoho.currentdate.addDay(30));
77  custom_field_list = list();
78  custom_field_map = Map();
79  custom_field_map.put("label","
    Is strategic Account?");
80  custom_field_map.put("value",i
    s_strategic_account);
81  payment_mode = Map();
82  payment_mode.put("label","Paym
    ent Mode");
83  payment_mode.put("value",payme
    nt_method);
84  custom_field_list.add(payment_
    mode);
85  custom_field_list.add(custom_f
    ield_map);
86  invoiceData.put("custom_fields
    ",custom_field_list);
87  ////////////////////// ADD
    LINE ITEMS
    //////////////////////
88  lineItemsList = List();
89  // Line Item 1: Permit
    Application Fee
90  permitFeeItem = Map();
91  permitFeeItem.put("name","Perm
    it Application Fee");
92  permitFeeItem.put("rate",unit_
    price);
93  permitFeeItem.put("quantity",1
    );
94  lineItemsList.add(permitFeeIte
    m);
95  // Line Item 2: Sanctuary Fee
96  sanctuaryFeeItem = Map();
97  sanctuaryFeeItem.put("name","S
    anctuary Fee");
```

```
sanctuaryFeeItem.put("rate",sanctuary_fee);
sanctuaryFeeItem.put("quantity",1);
lineItemsList.add(sanctuaryFeeItem);
invoiceData.put("line_items",lineItemsList);
//////////////// Create record  in zoho book invoice module ////////////////////
InvoiceCreateResponse = zoho.books.createRecord("Invoices","848446319",invoiceData,"zohobooks");
info InvoiceCreateResponse;
if(InvoiceCreateResponse.containKey("invoice"))
{
  Invoice_id = InvoiceCreateResponse.get("invoice");
  id = Invoice_id.get("invoice_id");
  /////////////// 6 June 2025 ////////////////////////////
  sub_total = Invoice_id.getJSON("sub_total");
  credit_card_charge = sub_total * 0.031;
  custom_field_list = list();
  payment_mode = Map();
  payment_mode.put("label","Payment Mode");
```

```
115
   payment_mode.put("value",payme
   nt_method);
116
   custom_field_list.add(payment_
   mode);
117   if(payment_method ==
   "Credit Card" &&
   quote_action_type == "Send
   quote and request payment
   method to be saved")
118   {
119         invoice_book_map =
   Map();
120
   invoice_book_map.put("custom_f
   ields",custom_field_list);
121
   invoice_book_map.put("reason",
   "Credit Card Charge");
122
   invoice_book_map.put("adjustme
   nt_account_name",payment_metho
   d);
123
   invoice_book_map.put("adjustme
   nt",credit_card_charge);
124
   invoice_book_map.put("adjustme
   nt_description","Credit Card
   Charge");
125         ////////////////
   Update zoho book invoice
   record ////////////////////
126
   zoho_book_invoice_update =
   zoho.books.updateRecord("invoi
```

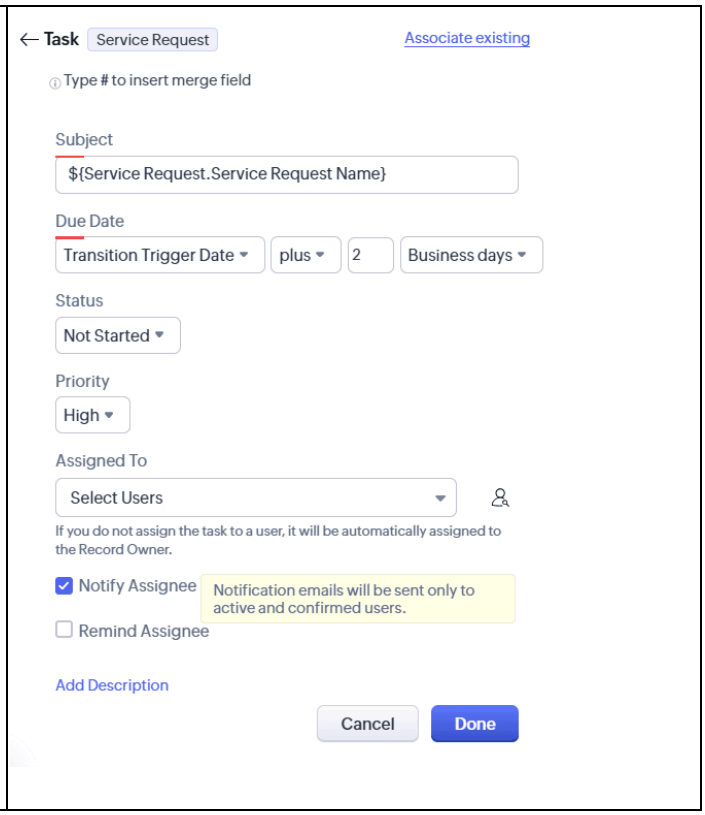
```
ces","848446319",id,invoice_book_map,"zohobooks");
        info zoho_book_invoice_update;
    }
    ///////////////// 6 June 2025 ////////////////////////////

    ///////////////// Update the custom fields in zoho crm service request module ////////////////////////////
    service_map = Map();
    service_map.put("Zoho_Book_Invoice_Id",id);
    service_map.put("Zoho_Book_Invoice_URL","https://books.zoho.com/app/848446319#/invoices/" + id);
    // Update Service Request
    update_service = zoho.crm.updateRecord("Service_Request",record_id,service_map);

    info update_service;

}
service_request_details = zoho.crm.getRecordById("Service_Request",record_id);
//info service_request_details;
name =
```

	service_request_details.getJSON("Name"); 141 //info name; 142 projectId = projectName.get("id"); 143 quote_amount = serviceRequestDetail.get("Unit_Price"); 144 billing_events_map = Map(); 145 billing_events_map.put("Name",name); 146 billing_events_map.put("Service_Type","Gopher Tortoise"); 147 billing_events_map.put("Date",zoho.currentdate); 148 billing_events_map.put("Project",projectId); 149 billing_events_map.put("Service_Request",record_id); 150 billing_events_map.put("Amount",quote_amount); 151 /////// Create Billing Event record in zoho crm //////////////////// 152 create_billing_event = zoho.crm.createRecord("Billing_Events",billing_events_map); 153 }
Close Permit	**After :** Creating a task record


Task
Service Request
Associate existing
Type # to insert merge field
Subject
${Service Request.Service Request Name}
Due Date
Transition Trigger Date
plus
2
Business days
Status
Not Started
Priority
High
Assigned To
Select Users
If you do not assign the task to a user, it will be automatically assigned to the Record Owner.
Notify Assignee
Notification emails will be sent only to active and confirmed users.
Remind Assignee
Add Description
Cancel
Done