Oracle Fusion Service

Integrating Fusion Service with Field Service Questions and Answers

Oracle Fusion Service
Integrating Fusion Service with Field Service Questions and Answers

F98410-08

Author: JKOLB

Contents

Get Help

There are a number of ways to learn more about your product and interact with Oracle and other users.

Get Help in the Applications

Some application pages have help icons to give you access to contextual help. If you don't see any help icons on your page, click your user image or name in the global header and select Show Help Icons. If the page has contextual help, help icons will appear.

Get Support

You can get support at *My Oracle Support*. For accessible support, visit *Oracle Accessibility Learning and Support*.

Get Training

Increase your knowledge of Oracle Cloud by taking courses at *Oracle University*.

Join Our Community

Use *Cloud Customer Connect* to get information from industry experts at Oracle and in the partner community. You can join forums to connect with other customers, post questions, suggest *ideas* for product enhancements, and watch events.

Learn About Accessibility

For information about Oracle's commitment to accessibility, visit the *Oracle Accessibility Program*. Videos included in this guide are provided as a media alternative for text-based topics also available in this guide.

Share Your Feedback

We welcome your feedback about Oracle Applications user assistance. If you need clarification, find an error, or just want to tell us what you found helpful, we'd like to hear from you.

You can email your feedback to *oracle_fusion_applications_help_ww_grp@oracle.com*.

Thanks for helping us improve our user assistance!

1 Questions and Answers

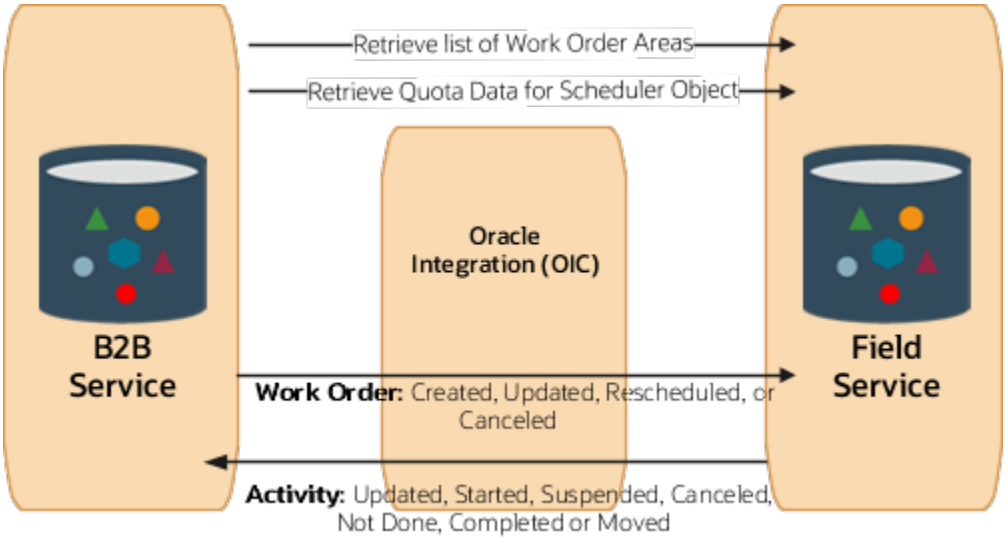
Integration Component Architecture Between Oracle Fusion Service and Oracle Field Service

Service work order management is the primary use case handled in the Oracle Fusion Service and Oracle Field Service integration.

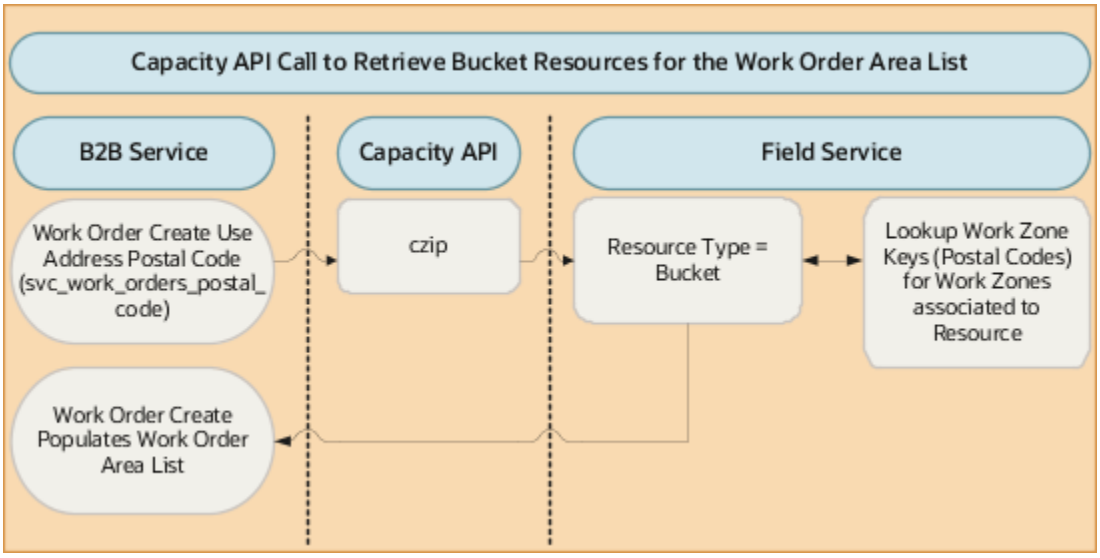
Service work order management has both work order creation and updates in Oracle Fusion Service, and updates in Oracle Field Service. To get this integration, a combination of point-to-point and bidirectional integrations are used. The point-to-point integrations are used for getting data from Oracle Field Service that's used to create and reschedule work orders. Bidirectional integration is used for synchronizing Oracle Fusion Service work orders with Oracle Field Service activities. Oracle Fusion Service and Oracle Field Service Bi-Directional integration uses Oracle Integration as the integration component. Oracle Integration is a complete, secure, and lightweight integration solution where you can connect your applications in the cloud. It simplifies connectivity between your applications, and can connect both your applications that exist in the cloud, and your applications that are still maintained on-premise.

The integration manages error handling and guaranteed delivery by introducing concrete fault handling and prevention measures in the integration layer. This is gained through Oracle Integration. The integration domain covers typical elements and integration functionality such as adapters for connectivity to back-end systems, routing, transformation, and filtering.

The following figure shows the process flow of information between Fusion Service, Oracle Integration, and Field Service.



The following figure shows the point-to-point components of the Oracle Fusion Service and Oracle Field Service integration using the Oracle Field Service Capacity API to retrieve the data work order area list in Oracle Fusion Service from Oracle Field Service.



Oracle Fusion Service Integration Services

The Fusion Service web services `CustomerWorkOrderService` is used in the integration. This SOAP API is called from the Event Handling Framework to retrieve a work order and Oracle Integration to create, update, reschedule, and cancel a work order in Fusion Service.

Oracle Field Service Integration Services

These Oracle Field Service web services are used in the integration:

- `BulkUpdateActivity` REST API. Use this web service through `OIC` to create, update, and reschedule an activity in Oracle Field Service.
- `CancelActivity` REST API. Use this web service through `OIC` to cancel an activity in Oracle Field Service.
- Capacity SOAP API. Use this web service in the point-to-point integration when creating and scheduling a work order to retrieve the list of work order areas based on postal code and time zone and the scheduler data based on work order area and work order type.

Oracle Integration

The prebuilt integrations are available through Oracle Marketplace. You can sign in and install the package directly into your Oracle Integration instance. The installation includes the following:

- Connection: Oracle Field Service
- Connection: Oracle Fusion Service
- Connection: Oracle REST OFS Attachment
- Integration: Oracle `B2BSVC` `OFS` Work Order Created
- Integration: Oracle `B2BSVC` `OFS` Work Order Updated
- Integration: Oracle `B2BSVC` `OFS` Work Order Canceled
- Integration: Oracle `OFS` `B2BSVC` Activity Updated
- Integration: Oracle `OFS` `B2BSVC` Attachment

To access the integrations in Oracle Marketplace, do the following:

1. Access Oracle Marketplace.

   You can either use the **Search** field and enter criteria such as Oracle Fusion Service to Oracle Field Service, or do the following steps:
2. Select PLATFORM (PaaS) from the **Products** drop-down list.
3. Select **Oracle Integration**.
4. In the Oracle Integration window, scroll and select **Oracle Fusion Service to Oracle Field Service**.
5. Click **Get App**.
6. Read and accept the Terms and click **Next**.

   The My Oracle Support page Integrating Create and Update Processes for Service Work Orders (Document ID 2247612.1) opens. This is where you can download the file

> **Note:** If Oracle Marketplace isn't available, you can download the prebuilt files from My Oracle Support. To access the prebuilt integration flow, see Integrating Oracle Fusion Service with Oracle Field Service on My Oracle Support. Oracle Support Document 2247612.1 In the Attachments section, select the appropriate attachment for your implementation. Save the orcl.r.b2bsvc_ofs_work_order_sync.20_10_0_developed.par file to a local computer.

Overview of Echo Suppression and Bi-Directional Synchronization

During bidirectional synchronization, work order activity generates synchronization echos between Oracle Fusion Service and Oracle Field Service. This means that when an event is triggered in Oracle Fusion Service it's synchronized through Oracle Integration to Oracle Field Service, which then fires an event in Oracle Field Service and then back to Oracle Fusion Service, on and on. The Oracle Integration-based integration uses an echo suppression mechanism, which stops unwanted update or create events (the echoes) from going back to the source application.

> **CAUTION:** You must follow the user name guidelines for the Oracle Fusion Service integration and Oracle Field Service integration because they're used for echo suppression in the prebuilt integration flows. If you use different user names, you must modify the prebuilt integration flows in Oracle Integration for echo suppression to work.

How do I create a Fusion Service integration user account?

All inbound requests from Oracle Field Service to Fusion Service are routed through Oracle Integration. To make the update in Fusion Service, Oracle Integration initiates the SOAP APIs for Fusion Service that are exposed in the Oracle CX Sales and Fusion Service Catalog.

To initiate the Oracle CX Sales and Fusion Service Catalog, you must create a unique user called the Integration User Account user.

> **Note:** To do this task, you must have the IT Security Manager job role.

Create the Integration User

First, create the new user:

1. Sign in to Oracle CX Sales using administrator privileges.
2. Using Navigator, navigate to My Team > Users and Roles.
3. In the Manage Users page, click **Create**.
4. Complete the fields as shown on the following table.

Field	Value
Last Name	SERVICE_APP_ICS_ID
Email	Enter a valid email.
Hire Date	Enter the current date.
User Name	SERVICE_APP_ICS_ID
Person Type	Employee
Legal Employer	Select a valid legal organization from the list of values.
Business Unit	Select a valid business unit from the list of values.

5. Click **Save and Close**.

CAUTION: Unless you don't intend to change the prebuilt integration in Oracle Integration, Oracle requires that you use the user name SERVICE_APP_ICS_ID to connect from Oracle Integration to Oracle Fusion because it's used for echo suppression in the prebuilt integration flows. If you use a different user name, you must change the prebuilt integration flows in Oracle Integration for echo suppression to work.

Create the SOA Operator Job Role

Now that the user is created, you create the new job role:

1. Using Navigator, select **Security Console** in the **Tools** section.
2. Click **Create Role**.
3. Complete the fields as shown on the following table.

Field	Value
Role Name	SVC `Soa` Operator

Field	Value
Role Code	SVC_SOA_OPERATOR
Role Category	CRM - Job Roles

4. Navigate to the Role Hierarchy train stop and click **Create Role**.
5. Search for the SOA operator role and click **Add Role Membership**.
6. Click **Close**.
7. Navigate to the Summary train stop and verify the SOA operator role shows up in the Role Hierarchy section.
8. Click **Save and Close**.
9. Click **OK** on the confirmation message.

Assign Job Roles and Setting Password for Integration User

Users must be associated with roles and privileges in Oracle Authorization Policy Manage APM on the Oracle Elements Server

1. Using Navigator, navigate to the Users tab in Security Console.
2. Search for the SERVICE_APP_ICS_ID user.
3. Open SERVICE_APP_ICS_ID and click **Edit**.
4. Click **Add Role**.
5. Search and select Customer Service Representative.
6. Click **Add Role Membership**.
7. Search and select Employee.
8. Click **Add Role Membership**.
9. Search and select Resource.
10. Click **Add Role Membership**.
11. Search and select SVC SOA Operator.
12. Click **Add Role Membership**.
13. Click **Done**.
14. Click **Save and Close**.
15. Click **Reset Password**.
16. Update the password then click **Reset Password**.
17. Click **Done** to sign out of the Security Console.

The Integration User is now set up and is used in the Oracle Integration User connection to Oracle Fusion Service. To verify the integration user was set up correctly, sign in to Oracle Fusion Service using the user credentials.

How do I use Service Logistics Parts Order with work orders?

To order parts on work orders, you need to expose the **Parts Order** region and the **Service Request Work Order Parts Order** tab on the Work Order page. If you're using Service Logistics, you must expose the Service Logistics Parts Order region from Setup and Maintenance.

Note: You must have a role that contains the following privileges:

- Setup and Maintain Applications
- Setup Service
- Setup Service Work Order

To expose the Service Logistics **Parts Order** region and **Service Request Work Order Parts Order** tab, do the following:

1. Go to Setup and Maintenance:
    - Offering: Service
    - Functional Area: Change Feature Opt-in
2. In the Service row, click the **Edit** icon in the Features column.
3. Click the **Enable** icon for **Service Logistics Parts Order**.
4. In the Feature Name: Service Logistics Parts Order window: select the following:
    - **Service Request Parts Order** check box to enable only part orders (no field service work order).
    - **Service Request Work Order Parts Order** check box for both parts and work orders.
5. Click **Save and Close**.
6. Click **Done**.
7. Click **Done** on the Opt In page.

The Service Request Parts Order region and tab now appear on the Work Order page. Refer to the *Getting Started with Service Logistics Cloud Implementation* guide to continue setup of Service Logistics.

How do I enable Installed Base Assets for service requests and work orders?

If you use Installed Base Assets for processes such as Supply Chain, Service Logistics, Service Contracts, or IOT, you can opt-in to use the same asset model for SRs and Work Orders. Use Application Composer to add the Installed Base Asset fields to the SR and Work Order pages.

Enable Installed Base Assets

To opt in, do the following:

1. In the Setup and Maintenance work area, go to the following:
    - Offering: Service
    - Change Feature Opt in link
2. Click the **Features** icon for Service in the first row.
3. Select Enable for **Manage Assets Using Common Asset Model**.
4. Click **Done**.
5. Click Done on the Opt In page.

Note: This is a global setting where you choose whether to use Installed Base Assets or the default Asset object for the Service Request and Work Order process. You can't use both asset objects in Fusion Service so you should carefully consider the impact if you have requirements to support asset management outside these processes. For example, Installed Base Asset doesn't currently support sales processes in Fusion Sales and Service and has limited support for extensibility.

How do I map work order attachment fields?

The table in this topic details field mappings for attachments in Oracle Field Service. The fields are updated in Oracle Fusion Service through Oracle Integration.

Attachment Field Mapping

The following table shows the Oracle Field Service to Oracle Fusion Service field mappings for attachments.

Oracle Fusion Service Property	Data Type	Oracle Field Service Property	Data Type	Condition
Attachment	file	wo_attachment	file	Enter the allowed MIME type.
Photo	image	wo_photo	file	NA

How can I map more fields from the SR to work orders?

This example shows you how you can map extra fields from service request to a work order. For example map the title of the SR to the work order.

Note: This example assumes that custom fields are already mapped in the layouts.

Create a New Action Chain on VB Enter

1. Navigate to `AppUIs tab > Expand Oracle CX Service UI Extension App > Service > fieldsvc > Create page .`
2. Navigate to `Event Listeners > Create a new Event Listener on VBEnter > Create page Action chain` (with JSON format). For example, vbEnterChangeListener.json

Note: By default VB will generate in Javascript format. Change it to the JSON format. Create-page.json-x file generates lines as shown in the following code: **Create-page.json-x**

```
"vbEnter": {
 "chains": [
 {
 "parameters": {},
 "chainId": "vbEnterChangeListener"
 }
 ]
}
```

Assign Fields in Action Chain

1. Navigate to the action chain created in the previous step and add this logic:
   a. Assign 'fetchSRDetails' = true
   b. Assign 'fetchOtherSRFields' = <SR Field names with coma separated>

Note: The field name should match with the service request object. For example: CustomText_c,CustomText_c_2. The Action chain JSON file looks like the following code: **vbEnterChangeListener.json**

```
"actions": {
 "assignVariablesFetchSRDetails": {
 "module": "vb/action/builtin/assignVariablesAction",
 "parameters": {
 "$base.page.variables.fetchSRDetails": {
 "source": true
 }
 },
 "outcomes": {
 "success": "assignVariables"
 }
 },
 "assignVariables": {
 "parameters": {
 "$base.page.variables.fetchOtherSRFields": {
 "source": "{{'CustomText_c','CustomText_c_2'}}"
 }
 }
 }
}
```

Create Action Chain for Page Event and Add Input Parameters

1. Navigate to **Event Listeners > Create a new Event Listener** for the page Event 'fetchSRDetails' > Create page Action chain with JSON format. For example fetchSRDetailsChangeListener.json

Note: By default VB will generate Javascript format. You need to change this to JSON format. Declare input parameter as 'srObj' for action chain > $event.srObj (map 'event.srOb'j object to input parameter). **create-page-x.json**

```
"oracle_cx_serviceUI/page:fetchSRDetails": {
 "chains": [
 {
 "parameters": {
 "srObj": "{{ $event.srObj }}"
 },
 "chainId": "fetchSRDetailsChangeListener"
 }
 ]
}
```

Add Input Parameter for the Action Chain and Map Event Object

1. Navigate to the Action chain created in the previous step and enter the logic as follows:
   a. Create input parameter for event listener "srObj" as 'Any' type and mark as input required from the caller.

fetchSRDetailsChangeListener.json

```
"variables": {
 "srObj": {
 "type": "any",
 "input": "fromCaller"
 }
}
```

Create a new JS Function and Pass Parameters

1. Invoke the Call function method and create a new function 'assignFieldsFromSRtoWO' and add the parameters 'srObj' and 'customerWorkOrders' object as shown in the following code sample:

fetchSRDetailsChangeListener.json

```
"assignFieldsFromSRtoWO": {
 "module": "vb/action/builtin/callModuleFunctionAction",
 "parameters": {
 "module": "[[ $functions ]]",
 "functionName": "assignFieldsFromSRtoWO",
 "params": [
 "{{ $variables.srObj }}",
 "{{ $base.page.variables.customerWorkOrders }}"
 ]
 }
```

2. In the 'assignFieldsFromSRtoWO' function add logic to map the SR object fields to WO object fields and return added WO object from function. The code will look like the following sample:

create-page-x.js

```
 /**
 * assign Fields from SR to WO Object
 * @param {Object} srObject
 * @param {Object} woObject
```

```
* @return {Object} woObject
*/
assignFieldsFromSRtoWO(srObject,woObject) {
woObject.WorkOrderCustText_c = srObject.CustomText_c; // Add relevant object mapping example :
WorkOrderCustText_c is wo custom field where 'CustomText_c' is SR custom field
return woObject;
}
```

Assign Function Result to Work Order Object

1. Assign 'customerWorkOrders' object to function result obtained from the previous step. The Action chain code will look like the following sample:

fetchSRDetailsChangeListener.json

```
"actions": {
 "assignVariablesCustomerWorkOrders": {
 "module": "vb/action/builtin/assignVariablesAction",
 "parameters": {
 "$base.page.variables.customerWorkOrders": {
 "source": "{{ $chain.results.assignFieldsFromSRtoWO }}"
 }
 }
 }
```

Optional Step: If at runtime an error is displayed, for example, 'create-page-template-x.html' not found in browser console:

Navigate to Create page and create a test section to generate 'create-page-template-x.html'. This is required to load the `customization` page.

1. Go to the `fieldsvc` section and find the Create page.
2. In the page designer, go to the **Structure** tab and click the **Container Ruleset** section.
3. A properties window opens.
4. Click the **CreateFormTemplate** link.
5. In the **CreateFormTemplate** window, select the option **Create Section** from the drop-down list.
6. This step guides you through creating a sample section, which will ultimately generate the file named "create-page-template-x.html".

How do I set field values on value change of predefined fields?

Here's how you make dependent fields auto populate when the default or custom field's value changes:

1. Expose the fields for which you need to set the value in a dynamic form layout.
2. Navigate to the Create Work order Variables section.
3. Add an event listener with an action chain to transientCustomerWO(create page)/ `NewWoTransient` on the Edit page.
4. Add a condition if the condition to check that the custom property is changed (By comparing $variables.event.oldValue.<Fieldname> !== $variables.event.value. <Fieldname>)

5. Add an assign variable to set the new value to Field in dynamic form transient variable (transientCustomerWO.<Fieldname> = Value & customerWorkOrders.<Fieldname> = value (create page)/ NewWoTransient.<Fieldname> = value & NewWO.<Fieldname>= value (edit page)).

Why is the layout not rendering even though the rule set is correct?

Here's what you do if you need to view the Work Order page based on the Work Order's completed status and the page layout isn't rendering correctly:

Prerequisite step:

1. Verify in GET customerWorkOrders/<WONumber> has “EditModeFlag”: "false" > To enable Detail Work order page (layout contains read only fields) & “EditModeFlag”: "true" > To enable Edit Work order page (layout contains edit fields).

Uptake steps for Detail work order page (Read only fields):

2. Navigate to “DetailViewLayout” and duplicate layout “Oracle Field Service Cloud” for Field service work order or “Standard work order” for Generic work order.
3. Apply rule set:
   - if $fields.WoStatusCdvalue() strictly equals 'ORA_SVC_WO_SUBMITTED'
   - return Oracle_Field_Service_Cloud_Detail_Page

   The generated code should look like the following:

```
"addLayouts": {
 "Oracle_Field_Service_Cloud_Detail_Page": {

 "expression": "{{ $fields.WoStatusCd.value() === 'ORA_SVC_WO_SUBMITTED' ?
 'Oracle_Field_Service_Cloud_Detail_Page' : null }}"
 }
```

   Make sure the detail section has the layout mapped correctly. If any other layout is referred, check the rule set section.

```
"chainRules": {
"/detailViewLayout": {
 "rules": [
 "Oracle_Field_Service_Cloud_Detail_Page"
 ]
 }
}
```

4. Drill down into the duplicate layout. For example, “Oracle_Field_Service_Cloud_Detail_Page”.
5. Add “WoStatusCd” field from the field section to the layout and map the field into “Dummy” template which doesn’t have any code look like “<template id="dummy"> </template>”
6. To get the values in runtime, run the application to see the changes.

How do I show or hide the create work order button?

Here's how you manage the visibility of the Create Work Order button:

This applies to the following pages:

- SR Foldout: Work Order Panel (both main panel and sub view section)
- SR Detail Page: Work Order Section

Visibility Control:

A variable named `$base.flow.variables.woCreateBtnOptIn` is used to control the button's visibility. Set the variable to:

- `true`: Makes the Create Work Order button visible.
- `false`: Hides the Create Work Order button.

For the work order list page only, setting `$base.flow.variables.woCreateBtnOptIn` to `false` on page load will hide the Create Work Order button.

How do I update the attachment component in the work order edit page?

Follow these steps to update the attachment component in the work order.

To use the **oj-sp-attachments** component (used in the SR), instead of using the **oj-sp-attachments-simple** component (used in the work order edit/detail page), follow these steps:

1. Navigate to the Work Order page.
2. Click your User Profile.
3. Select **Edit page in Visual Builder Studio** in the Administration section.
4. Log on to VSB.
5. Enter a project name and click **Create**.
6. In VBS, navigate to WO detail page by expanding `Oracle CX Service UI Extension App > Service > fieldsvc > detail .`
7. Click the **Detail** tab.
8. Click the **Structure** tab.
9. In the HTML DOM structure, select **containerLayout1**.
10. You'll see the layout structure in the properties panel.
11. Create a duplicate layout for the container layout.
12. Rename **case1** to **case2**.
13. Click the add icon and add a new section called **NewAttachments**.
14. Use the up arrow to move the **NewAttachments** section and place it below the **PartsSummaryTemplate**.
15. Remove the old **AttachmentTemplate** section by clicking the delete button next to it.
16. Click the new **NewAttachments** section to open it.
17. Add the **oj-collapsible** components to the section template.
18. Add the **Attachment** component.
19. Add the following code within the template:

```
<oj-collapsible>
```

```
 <div slot="header" class="oj-typography-heading-xs oj-sm-padding-3x-bottom">
 <span tabindex="0">
 <oj-bind-text value="[[$translations.fsvc['attachments']]]">
 </oj-bind-text>
 </span>
 </div>
 <oj-defer>
 <oj-sp-attachments class="oj-flex-item oj-sm-12 oj-md-12"
 category="MISC"
 view.endpoint="[[ 'oracle_cx_serviceUI:fsRestApiGroup/getall_customerWorkOrders-Attachment' ]]"
 view.endpoint-params="[[$variables.attachmentsEndpointParams1]]"

 background-tracker.endpoint="[[ 'oracle_cx_serviceUI:applcoreApi/docTracker']]"
 background-upload.endpoint="[[ 'oracle_cx_serviceUI:applcoreApi/upload' ]]"

 create.endpoint="[[ 'oracle_cx_serviceUI:fsRestApiGroup/create_customerWorkOrders-Attachment' ]]"
 create.endpoint-params="[[$variables.attachmentsEndpointParams1]]"

 delete.endpoint="[[ 'oracle_cx_serviceUI:fsRestApiGroup/delete_customerWorkOrders-Attachment' ]]"
 delete.endpoint-params="[[$variables.attachmentsEndpointParams1]]"
 delete.endpoint-attachment-param-name="customerWorkOrders_Attachment_Id"

 download.endpoint="[[ 'oracle_cx_serviceUI:fsRestApiGroup/get_customerWorkOrders-Attachment-
FileContents' ]]"
 download.endpoint-params="[[$variables.attachmentsEndpointParams1]]"
 download.endpoint-attachment-param-name="customerWorkOrders_Attachment_Id"

 edit.endpoint="[[ 'oracle_cx_serviceUI:fsRestApiGroup/update_customerWorkOrders-Attachment' ]]"
 edit.endpoint-params="[[$variables.attachmentsEndpointParams1]]"
 edit.endpoint-attachment-param-name="customerWorkOrders_Attachment_Id"

 categories.endpoint="[[ 'oracle_cx_serviceUI:applcoreApi/getAttachCategory' ]]"

 preview.endpoint="[[ 'oracle_cx_serviceUI:fsRestApiGroup/getall_customerWorkOrders-Attachment-
AttachmentsPreview' ]]"
 preview.endpoint-params="[[$variables.attachmentsEndpointParams1]]"
 preview.endpoint-attachment-param-name="customerWorkOrders_Attachment_Id"

 display-options.category-for-create="SELECT_REQUIRED"
 display-options.preview-visibility="[[ 'hidden' ]]"

 display-options.add-visibility="[[ ($functions.canAddAttachment($application.user.permissions) &&
 $functions.isWoEditFlagEnabled($base.variables.NewWO)) ? 'visible' : 'hidden']]"
 display-options.remove-visibility="[[ ($functions.canAddAttachment($application.user.permissions) &&
 $functions.isWoEditFlagEnabled($base.variables.NewWO))? 'visible' : 'hidden' ]]"
 display-options.update-visibility="[[ ($functions.canAddAttachment($application.user.permissions) &&
 $functions.isWoEditFlagEnabled($base.variables.NewWO)) ? 'visible' : 'hidden' ]]"
 display-options.download-visibility="[[ ($functions.canAddAttachment($application.user.permissions)) ?
 'visible' : 'hidden' ]]"
 display-options.remove = "[[ !($functions.canAddAttachment($application.user.permissions)) || !
$functions.canEditWO($application.user.permissions) || !
$functions.isWoEditFlagEnabled($base.variables.NewWO)]]"
 display-options.add = "[[ !($functions.canAddAttachment($application.user.permissions)) || !
$functions.canEditWO($application.user.permissions) || !
$functions.isWoEditFlagEnabled($base.variables.NewWO)]]"
 display-options.update = "[[!($functions.canAddAttachment($application.user.permissions)) || !
$functions.canEditWO($application.user.permissions) || !
$functions.isWoEditFlagEnabled($base.variables.NewWO)]]"
 display-options.preview = "[[ !($functions.canAddAttachment($application.user.permissions)) ]]"

 entity-name="SVC_WORK_ORDERS">
 </oj-sp-attachments>
 </oj-defer>
 </oj-collapsible>
```

20. Add the following code in the **detail-page-x.js**:

```
define([], () => {
 'use strict';

 class PageModule {

 /**
 * Method to get the WONumber from CustomerWorkOrder object
 * @param wo : CustomerWorkOrder object
 */
 getWoNumber(wo) {
 let woNum;
 if(wo !== null && wo['@context'] && wo['@context'].key ){
 woNum = wo['@context'].key;
 }
 return woNum;
 }

 /**
 * Method to get the WONumber from CustomerWorkOrder object
 * @param wo : CustomerWorkOrder object
 */
 isWoEditFlagEnabled(wo) {
 let flag = false;
 if(wo !== null && wo.EditModeFlag ){
 flag = wo.EditModeFlag;
 }
 return flag;
 }

 /**
 * Method to get check the Add Attachment access
 * @param permissions
 */
 canAddAttachment(permissions){
 return permissions.indexOf('SVC_VBCS_Add_Attachment_Access') > 0 ;
 }

 /**
 * Method to get check the WO Edit access
 * @param permissions
 */
 canEditWO(permissions){
 return permissions.indexOf('SVC_VBCS_Edit_Service_Work_Order_Access') > 0 ;
 }
 }

 return PageModule;
});
```

21. Add a new variable attachmentsEndpointParams1 of the object type with the following code:

```
{
 "customerWorkOrders_Id": "[[$functions.getWoNumber($base.variables.NewWO)]]"
}
```

22. Add the following code to the translation bundle **detail-page-x.json**:

```
"translations": {
 "fsvc": {
 "path": "faResourceBundle/nls/oracle.apps.crm.service.fieldservice.resource"
 }
 }
```

23. Preview the changes before saving.

How do I manage the default address source for service work orders?

Here are instructions on how you can change, add, or remove default sources from the list of values in the work order layout such as not allowing one-time addresses.

The address source can be based on the asset, service profile, account, contact, or a one-time address for the service request and work order. By default, the address source is set to asset.

Here's how you change the default to be based on service profile and remove the one-time address from the list of available sources:

1. Open the Oracle CX Service UI Extension App template in Visual Builder Studio.
2. Choose the **Layout Section** icon.
3. In the Dependence tree, find **CustomerWorkOrders** in Oracle CX Service UI Extension App.
4. Select **CreateForm** in the **Dynamic Form** section.
5. Select the **Event Listeners** tab.
6. Click the **+ Event Listener** icon.
7. On the Create Event Listener page, click **vbEnter**.
8. Click **Next**.
9. Click **Finish**.
10. Click **vbEnterListener Go to Action Chain**.
11. Click Code and add these two lines in the following order:
    - `$base.variables.addressSourceContext = {"ORA_SVC_ASSET_ADDRESS":true,"ORA_SVC_PROFILE_ADDRESS": true, "ORA_SVC_CONTACT_ADDRESS": true,"ORA_SVC_ACCOUNT_ADDRESS": true, "ORA_SVC_ONETIME_ADDRESS":false};`
    - `$base.variables.addressDefaultSource = 'ORA_SVC_PROFILE_ADDRESS';`

12. The following image shows how the action chain appears:

```
define([
  'vb/action/actionChain',
  'vb/action/actions',
  'vb/action/actionUtils',
], (
  ActionChain,
  Actions,
  ActionUtils
) => {
  'use strict';

  class vbEnterListener extends ActionChain {

    /**
     * @param {Object} context
     */
    async run(context) {
      const { $layout, $base, $extension, $responsive, $user, $constants, $variables } = context;
      $base.variables.addressSourceContext = {"ORA_SVC_ASSET_ADDRESS":true,"ORA_SVC_PROFILE_ADDRESS
      $base.variables.addressDefaultSource = 'ORA_SVC_PROFILE_ADDRESS';
    }
  }

  return vbEnterListener;
});
```

13. Test to verify Service Profile is defaulted after selecting a service profile.
14. Open the Address Source to verify that the One-time address isn't displayed.

How can I make the time zone auto populate to reflect that of the person entering the work order?

To auto populate the time zone of the person entering the work order, use the following code: `def user_time_zone = oracle.apps.fnd.applcore.common.ApplSessionUtil.getTimeZone();`