How To Extend User Interface of Process E-commerce Returns
## Document History

<table>
<thead>
<tr>
<th>Document Version</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>First official release of this guide</td>
</tr>
</tbody>
</table>
# Table of Contents

1. Business Scenario
2. Background Information
3. Prerequisites
4. Step-by-Step Procedure
   - 4.1 Extend Structure in Back-End System
   - 4.2 Extend OData Service in Back-End System
   - 4.3 Register the Extension Service in Gateway
   - 4.4 Extend Fiori UI
      - 4.4.1 View Extension
      - 4.4.2 Controller Extension
1 Business Scenario

This guide provides detailed introduction about how to configure the extended features of SAP Process E-Commerce Returns app.

SAP Process E-Commerce Returns app provides the following standard features, for example:

- Identify a return delivery with a reference number, for example, return order, RMA, ERP delivery number
- Search for a return delivery with additional information, for example, product contained, customer name, and customer address
- Identify a returned product with product number, EAN, product picture, and so on
- Inspect the returned products
- Sort the products into different intermediate locations based on different quality conditions or based on different putaway areas to optimize the follow-up execution
- Trigger follow-up warehouse tasks for the sorted products, for example, putaway or scrap

Target Groups
Technical Consultants
Support Specialists
System Administrators
Solution Consultants
Key Users

2 Background Information

You want to customize the user interface of SAP Process E-Commerce Returns app based on your own business needs, for example, to add a Business Partner field.

3 Prerequisites

You have installed and correctly configured the following application:

- SAP EWM 9.5 or higher has been set up and integrated with SAP ERP for warehouse operations (different deployment modes are supported),
- SAP FIORI Front-end Server 2.0 or higher

You have set up that the inspection document is created at the activation of the delivery as follows:

- Choose Inspection Planning at Activation of Delivery for Action LF field for your warehouse and choose 3 for the IOT field (3 means Q-Inspection Returns Delivery).
- Inbound delivery of the customer returns should be unpacked, that is, there is no planned handling unit in inbound delivery.

Authorization

- Before you use Process E-Commerce Returns, you need to be assigned with SAP_EWM_BCR_WOP_T role in the front-end system.)

4 Step-by-Step Procedure

If extending oData is not needed, please start directly from chapter 4.4.
4.1 Extend Structure in Back-End System

1. Open transaction code: SE80, choose the structure you want to extend.
   - Extend delivery header: /SCWM/S_GW_CURE_DLV_H_INCL
   - Extend returned item: /SCWM/S_GW_CURE_QIE_I_INCL

   For example: Append a structure to /SCWM/S_GW_CURE_DLV_H_INCL with 2 fields: F1 and F2.

   Structure: /SCWM/S_GW_CURE_DLV_H_INCL
   Short Description: oData Customer Returns: Customer

   Attributes |
   Component | Typing Method | Component Type |
   ODATA_DUMMY | Types | ▼ DUMMY |
   APPEND | Types | ▼ ZUESTALEX |
   F1 | Types | ▼ INT4 |
   F2 | Types | ▼ CHAR2 |

2. Choose Append Structure

3. Add custom fields in the appended structure

4.2 Extend OData Service in Back-End System

1. Open transaction code: SEGW.
2. Create a new project:
3. Select Data Model entry in the newly created project. Choose Redefine → OData Service (SAP GW)

5. Select all Entity Type,
6. Import additional properties

For example: Import property “F2”.

7. Right click your project, choose *Generate Runtime*.
Warning: Do not check Overwrite Base/Extended Service

8. Record your service name somewhere.
9. Open transaction code: SE80; search your Data Provider Class; for example, ZCL_ZMEGAN_SERV_04_DPC_EXT in above screenshot. Redefine method, and your redefined logic:

   - DLVHEADSET_GET_ENTITY redefine header data
   - DLVQIEITEMSET_GET_ENTITYSET redefine item data

   ![Diagram]

For example:
Redefine method DLVHEADSET_GET_ENTITY and give a fixed value (22) to property “F2”.

10. Search your Model Provider Base Class: for example, ZCL_ZMEGAN_SERV_04_MPC. Go to Property tab, change Superclass to /SCWM/CL_CUST_RETURNS_MPC_EXT.
11. Check your code. There would be errors “The statement ‘TYPES’ must be followed by other additions.”. Delete duplicated lines of “types: ...”. Save and check your class again. There would be errors. Like duplicated methods, types etc. Delete the duplicated methods or types till there is no error when you check the class.

### 4.3 Register the Extension Service in Gateway

1. In gateway(front-end) system, open transaction code: /IWFND/MAINT_SERVICE  
2. Choose Add Service:

   ![Gateway Transaction Code](image)

   ![Service Catalog](image)

3. Search Technical Service Name in the back-end system  
4. Add Selected Services  
5. Open transaction code: SICF. Check if your service is in route /sap/opu/odata/sap/

### 4.4 Extend Fiori UI

#### 4.4.1 View Extension

1. Open WebIDE  
2. Choose **File → New → Extension Project**
3. Select Application → SAPUI5 ABAP Repository

4. Choose your front-end system, Application: EWM_CURE_MAN. OK, then Next

5. Check Open extension project in extensibility pane.

6. In the right side, Outline pane, choose Show extension points
7. Choose the extension point, add customer field
8. Code would be generated in the extension project → webapp → view →
    <yourextensioncustomname>.fragment.xml.
9. Add the new property F2 in the view just created.
    For example:
    ```xml
    <verticalLayout xmlns:ui="sap.ui.core.layout" width="200">
        <objectStatus text="{F2}" title="F2 Value is ">
    </verticalLayout>
    </core:FragmentDefinition>
    ```
10. Open manifest.json file and add following data source script into “sap.app” section.
    ```json
    "sap.app": {
        "dataSources": {
            "mainService": {
                "uri": "/sap/opu/odata/SAP/YourTechnicalExtendedServiceName",
                "type": "OData",
                "settings": {
                    "odataVersion": "2.0",
                    "localUri": "localService/metadata.xml"
                }
            }
        }
    }
    ```
    Please replace the URI highlight part with your technical extended service name.
    The technical extended service name should be same as the extended service name in “4.2 Extend OData Service in Back-End System” step 7.
11. Run your extension project.
Controller Extension

1. Open WebIDE
2. You can choose to extend the original controller or replace it by a custom controller. Config in manifest.json:

An example to extend:

```
"sap.ui": {
  "version": "1.1.0",
  "dependencies": {
    "minUI5Version": "1.49.0-SNAPSHOT"
  },
  "extends": {
    "component": "ti.ewm.customer retourss1",
    "extensions": {
      "sap.ui.controllerExtensions": {
        "ti.ewm.customer retourss1.controllerDetail": {
          "controllerName": "ti.ewm.customer retourss1.EXTENSION.controllerDetailCustom"
        }
      }
    }
  }
},
```

“The SAPUI5 controller extension concept does not use inheritance. Instead, methods of the custom controller override standard methods with the same name. The following controller lifecycle methods are, however, an exception to this rule: onInit, onExit, onBeforeRendering, onAfterRendering. For these methods, the controller methods of your custom application are called either after (for onInit and onAfterRendering), or before (for onExit and onBeforeRendering) the standard lifecycle methods.”

(Reference: https://wiki.wdf.sap.corp/wiki/display/ERPRTSOH/Extensibility)