Oracle Banking Extensibility Workbench

Getting Started User Guide

Release 14.8.0.0.0
G29517-01
April 2025

Oracle Banking Extensibility Workbench Getting Started User Guide, Release 14.8.0.0.0

G29517-01

Contents

1 Preface

1.1 Purpose

This guide is designed to help acquaint you with the Getting Started User Guide application. This guide provides answers to specific features and procedures that the user need to be aware of the module to function successfully.

This user guide would help you to understand the functioning of the Oracle Banking Extensibility Workbench – OBX and the types of extensions it provides. It provides the steps required to be followed for implementing the extensibility to the Base product. It is assumed that all the prior setup is already done related with Base product/ Kernel. In this document it is also assumed that installation will be done on Windows 10 operating system with minimum 8GB Ram and available/free space of 5GB.

1.2 Introduction

This user guide would help you to understand the functioning of the Oracle Banking Extensibility Workbench – OBX and the types of extensions it provides. It provides the steps required to be followed for implementing the extensibility to the Base product. It is assumed that all the prior setup is already done related with Base product/ Kernel. In this document it is also assumed that installation will be done on Windows 10 operating system with minimum 8GB Ram and available/free space of 5GB.

1.3 Audience

This document is intended for the teams and developers who are responsible for creating extensions like services and web components for products which are developed using Oracle Banking Microservices Architecture.

1.4 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

1.5 Critical Patches

Oracle advises customers to get all their security vulnerability information from the Oracle Critical Patch Update Advisory, which is available at Critical Patches, Security Alerts and Bulletins. All critical patches should be applied in a timely manner to ensure effective security, as strongly recommended by Oracle Software Security Assurance.

1.6 Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

1.7 Related Resources

For more information, see these related user guides:

- Oracle Banking Extensibility Workbench Installation Guide
- Oracle Banking Extensibility Workbench Release Notes

1.8 Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1.9 Screenshot Disclaimer

Personal information used in the interface or documents is dummy and does not exist in the real world. It is only for reference purposes.

1.10 Acronyms and Abbreviations

The list of the acronyms and abbreviations that are used in this guide are as follows:

Table 1-1 Acronyms and Abbreviations

Abbreviation	Description
DDA	Demand Deposit Accounts
ECA	External Credit Approval
EOD	End of Day
IBAN	International Bank Account Number

1.11 Basic Actions

The basic actions performed in the screens are as follows:

Table 1-2 Basic Actions

Actions	Description
New	Click **New** to add a new record. The system displays a new record to specify the required data. The fields marked with asterisk are mandatory. • This button is displayed only for the records that are already created.
Save	Click **Save** to save the details entered or selected in the screen.
Unlock	Click **Unlock** to update the details of an existing record. The system displays an existing record in editable mode. • This button is displayed only for the records that are already created.
Authorize	Click **Authorize** to authorize the record created. A maker of the screen is not allowed to authorize the same. Only a checker can authorize a record. • This button is displayed only for the already created records. For more information on the process, refer Authorization Process.
Approve	Click **Approve** to approve the initiated record. • This button is displayed once the user click **Authorize**.
Audit	Click **Audit** to view the maker details, checker details of the particular record. • This button is displayed only for the records that are already created.
Close	Click **Close** to close a record. This action is available only when a record is created.
Confirm	Click **Confirm** to confirm the action performed.
Cancel	Click **Cancel** to cancel the action performed.

Table 1-2 (Cont.) Basic Actions

Actions	Description
Compare	Click **Compare** to view the comparison through the field values of old record and the current record. • This button is displayed in the widget once the user click **Authorize**.
View	Click **View** to view the details in a particular modification stage. • This button is displayed in the widget once the user click **Authorize.**
View Difference only	Click **View Difference only** to view a comparison through the field element values of old record and the current record, which has undergone changes. • This button is displayed once the user click **Compare**.
Expand All	Click **Expand All** to expand and view all the details in the sections. • This button is displayed once the user click **Compare**.
Collapse All	Click **Collapse All** to hide the details in the sections. • This button is displayed once the user click **Compare**.
OK	Click **OK** to confirm the details in the screen.

1.12 Symbols and Icons

This guide has the following list of symbols and icons.

Table 1-3 Symbols and Icons - Common

Symbol/Icon	Function
┘└ ┐┌	Minimize
┌┐ └┘	Maximize
✕	Close
🔍	Perform Search
▼	Open a list
+	Add a new record
|<	Navigate to the first record

Table 1-3 (Cont.) Symbols and Icons - Common

Symbol/Icon	Function
>|	Navigate to the last record
◀	Navigate to the previous record
▶	Navigate to the next record
⊞	Grid view
☰	List view
⟳	Refresh
+	Click this icon to add a new row.
-	Click this icon to delete a row, which is already added.
📅	Calendar
🔔	Alerts

Table 1-4 Symbols and Icons – Audit Details

Symbol/Icon	Function
👤	A user
📅	Date and time

Table 1-4 (Cont.) Symbols and Icons – Audit Details

Symbol/Icon	Function
⚠	Unauthorized or Closed status
✓	Authorized or Open status
⊘	Rejected status

Table 1-5 Symbols and Icons - Widget

Symbol/Icon	Function
🔓	Open status
🗋	Unauthorized status
🔒	Closed status
🗋✓	Authorized status
🗋✗	Rejected status
✎	Modification Number

2 Welcome to Oracle Banking Extensibility Workbench

This guide provides an overview and detailed instructions for using the Oracle Banking Extensibility Workbench (OBX), enabling users to efficiently configure and customize banking workflows.

It provides the complete solution to create extensions for products based and developed on Oracle Banking Microservices Architecture (OBMA). It helps in generating the services and UI web components artifacts. This guide is designed to help you create all these types of service and UI artifacts. It also has complete life cycle management incorporated for all the extensions generated from tool.

- Introduction
  Oracle Banking Extensibility Workbench (OBX) is a combination of GUI and command line tool, intended to create different type of extensions for Oracle Banking Micro services Architecture.
- OBX and Base artifacts compatibility
  This topic provides the systematic instruction to perform OBX and Base artifacts compatibility.
- Setting up OBX for first time use
  This topic provides the systematic instruction to perform OBX setup for first time use.
- OBX Maintenance
  This topic provides the systematic instructions to execute OBX Maintenance operations.
- OBX UI
  This topic provides information about OBX UI details.

2.1 Introduction

Oracle Banking Extensibility Workbench (OBX) is a combination of GUI and command line tool, intended to create different type of extensions for Oracle Banking Micro services Architecture.

OBX support generation of following types of Extensions:

1. Service Extensions
   - Simple sub domain service
   - Maintenance sub domain service
   - Data/Resource Segment sub domain service
   - Simple Publisher/Subscriber Event Service
   - Custom Validation Service
2. UI Extensions – Web Component
   - Simple Standalone
   - Virtual Page
   - Maintenance Detail and Summary

- Data Segment
- Dashboard Widget

3. Modification of Base Web Component
    - Additions of Fields on Existing component
    - Hiding fields from screen
    - Defaulting values on screen
    - Disable field
    - Making Non-mandatory field

2.2 OBX and Base artifacts compatibility

This topic provides the systematic instruction to perform OBX and Base artifacts compatibility.

OJET version compatibility:

The implementation team must ensure that the OJET version of the app shell used aligns with the OJET version present in the OBX tool.

Note:

As part of OJET upgrade some older libraries may not be supported. If consulting / implementation team is using any of the unsupported libraries for their customizations, compatibility issues may arise if the app-shell version they are using doesn't include those OJET libraries.

All the UI customizations/extensions are bundled into `extended-components` war which ultimately refer to the app-shell OJET libraries only.

Please find the compatibility matrix of app-shell OJET versions and OBX OJET versions below.

Table 2-1 OBX - Compatibility

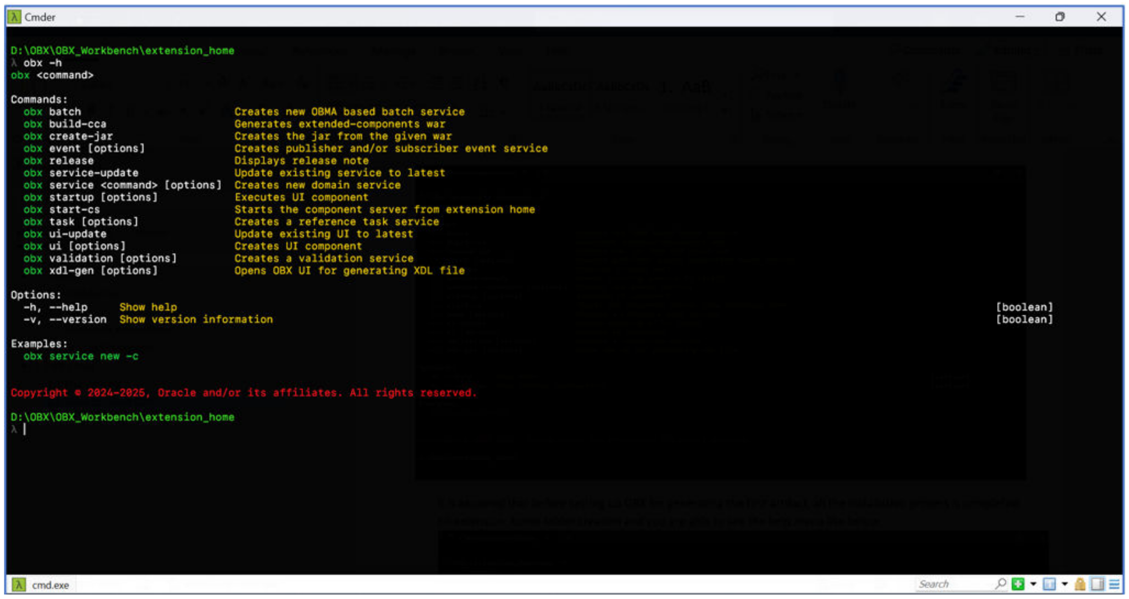
OBX version	OJET version
14.7.0.0.0	Appshell version xxxx (has 13.0.0 OJET version)
14.7.5.0.0	Appshell version 9.5.0 (has 15.1.8 OJET version)
14.8.0.0.0	Appshell version 9.6.0 (has OJET Version 17.xxx)

2.3 Setting up OBX for first time use

This topic provides the systematic instruction to perform OBX setup for first time use.

To generate the first artifact, user must first complete the installation process, including the creation of the **extension_home** folder, and then you should be able to see the help menu as shown below.

Figure 2-1   Setting up OBX

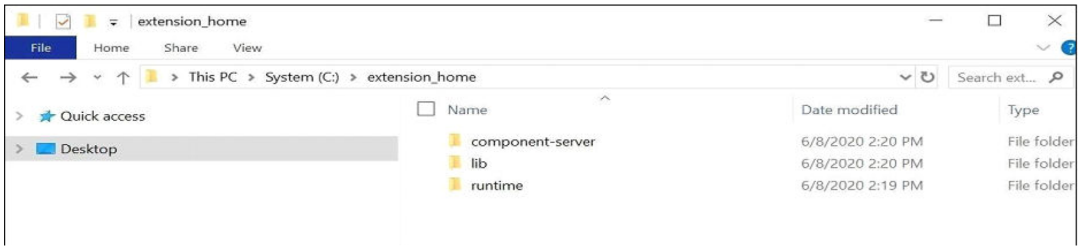


```
D:\OBX\OBX_Workbench\extension_home
λ obx -h
obx <command>

Commands:
  obx batch                         Creates new OBMA based batch service
  obx build-cca                     Generates extended-components war
  obx create-jar                    Creates the jar from the given war
  obx event [options]               Creates publisher and/or subscriber event service
  obx release                       Displays release note
  obx service-update                Update existing service to latest
  obx service <command> [options]   Creates new domain service
  obx startup [options]             Executes UI component
  obx start-cs                      Starts the component server from extension home
  obx task [options]                Creates a reference task service
  obx ui-update                     Update existing UI to latest
  obx ui [options]                  Creates UI component
  obx validation [options]          Creates a validation service
  obx xdl-gen [options]             Opens OBX UI for generating XDL file

Options:
  -h, --help     Show help                                                                   [boolean]
  -v, --version  Show version information                                                    [boolean]

Examples:
  obx service new -c


Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

D:\OBX\OBX_Workbench\extension_home
λ |
```

Once that is done, we will proceed to next step which is setting up libraries and components from base product. Follow the below process to setup libraries and components:

1. Create a folder **component-server** inside **extension_home** directory.
2. Use 7zip or other similar tool to extract **app-shell-9.5.0.war** from base product to copy the **common & js** folders and put it inside the **component-server** folder.
3. Navigate inside the **js** folder and copy the **components** folders and place it in the **component- server** folder.
4. Create a folder **lib** inside **extension_home** directory.
5. To use a service war file like **cmc-datasegment-services-9.5.0.war**, open it using a tool like 7zip. Navigate to the **WEB-INF\lib** folder within the war file and copy all the jars inside. Then, paste them into the **lib** folder of your extension's home directory.
6. Create a folder runtime inside **extension_home** directory.
7. Navigate to the **gradle** folder within the **obx.zip**, then copy the **extra_jars** from the **lib** folder to the runtime folder within the **extension_home** directory.
8. After all the above process **extension_home** folder looks like below.

Figure 2-2   Extension Home Folder



Name	Date modified	Type
component-server	6/8/2020 2:20 PM	File folder
lib	6/8/2020 2:20 PM	File folder
runtime	6/8/2020 2:19 PM	File folder

9. Once all of the above process is done, we cannot now generate the artifact.

2.4 OBX Maintenance

This topic provides the systematic instructions to execute OBX Maintenance operations.

Before generating the artifact, verify the below items from the base installation.

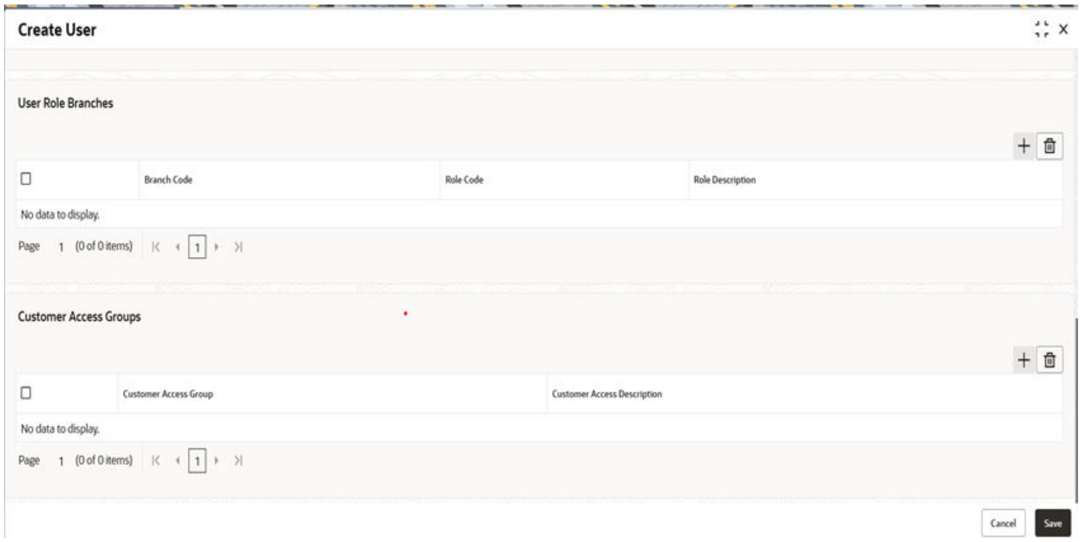
Items for the base installation verification.

- Verify if the **PRODUCT_EXTENDED_LEDGER** table exists in the **plato-ui-config** schema. If it's not present, execute the script below:

```
--------------------------------------------------------
-- DDL for Table PRODUCT_EXTENDED_LEDGER
--------------------------------------------------------
CREATE TABLE "PRODUCT_EXTENDED_LEDGER" ("ID" VARCHAR2(20),
"CCA_NAME"VARCHAR2(100), "CCA_TYPE" VARCHAR2(20), "PARENT_CCA_NAME"
VARCHAR2(100), "PRODUCT_NAME" VARCHAR2(100))
--------------------------------------------------------
-- Constraints for Table PRODUCT_EXTENDED_LEDGER
--------------------------------------------------------
ALTER TABLE "PRODUCT_EXTENDED_LEDGER" ADD CONSTRAINT
"PRODUCT_EXTENDED_LEDGER_PK" PRIMARY KEY ("ID")
ALTER TABLE "PRODUCT_EXTENDED_LEDGER" MODIFY ("CCA_NAME" NOT NULL
ENABLE)
ALTER TABLE "PRODUCT_EXTENDED_LEDGER" MODIFY ("ID" NOT NULL ENABLE)
ALTER TABLE "PRODUCT_EXTENDED_LEDGER" ADD CONSTRAINT
"UNIQUES_CCA_NAME" UNIQUE ("CCA_NAME")
```

- Maintain the product name **OBX** in the table **SMS_TM_APPLICATION** inside SMS schema.
- Grant user **OBX** application access through **SMS_TM_USER_APPLICATION** or preferred use the UI.

Figure 2-3 Create User



Create User

User Role Branches

	Branch Code	Role Code	Role Description

No data to display.

Page 1 (0 of 0 items)

Customer Access Groups

	Customer Access Group	Customer Access Description

No data to display.

Page 1 (0 of 0 items)

2.5 OBX UI

This topic provides information about OBX UI details.

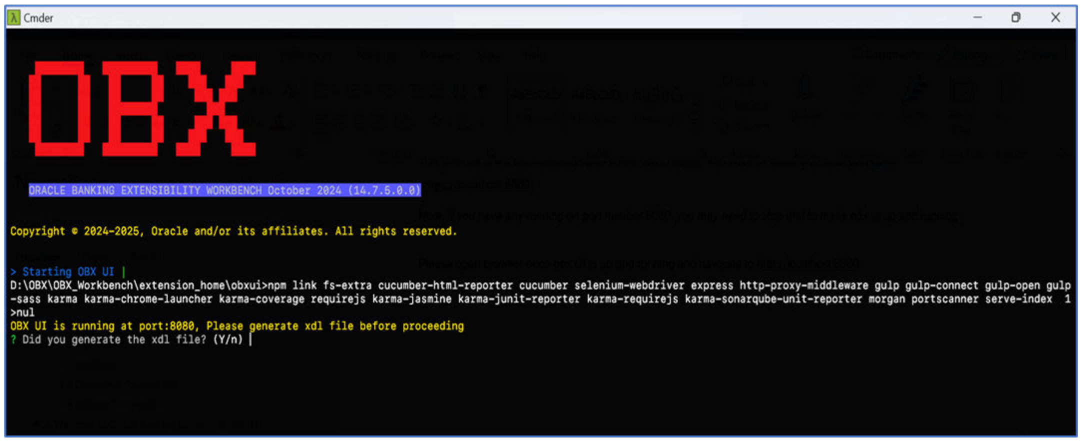
After setting up the OBX, we can now generate the XDL (OBX Domain Language) file, which will be used by the OBX engine to further generate the service and UI artifacts.

To start OBX UI:

1. Navigate to **extension_home** folder from console emulator (cmder).
2. Use the command **obx xdl-gen**.
3. This command will automatically open a new tab in cmder with OBX UI running at local port 8080 (https://localhost:8080).

> **Note:**
>
> If you have running applications on port number 8080, you may need to stop them to start the obx UI.

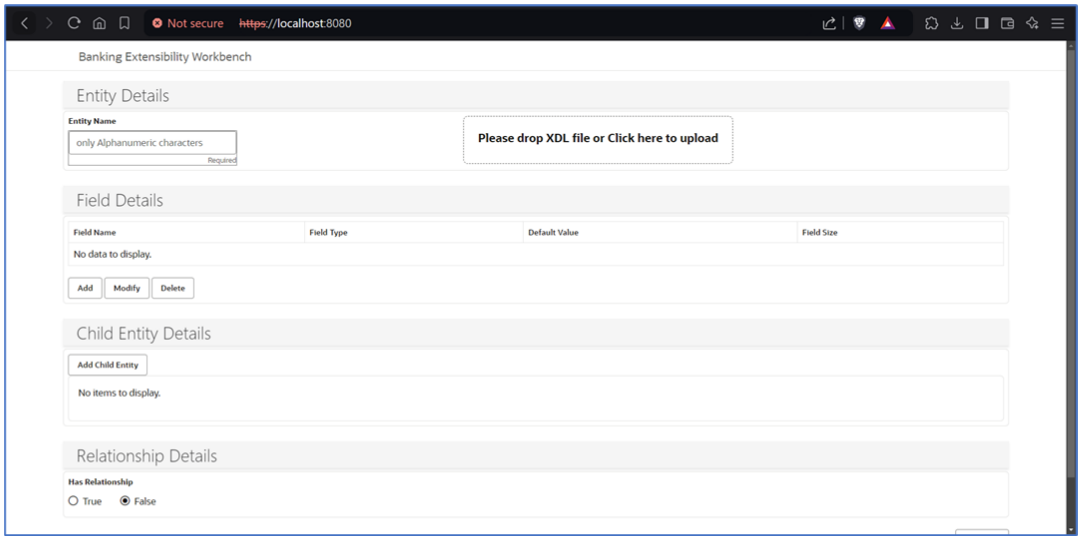
Figure 2-4 OBX UI



```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

> Starting OBX UI |
D:\OBX\OBX_Workbench\extension_home\obxui>npm link fs-extra cucumber-html-reporter cucumber selenium-webdriver express http-proxy-middleware gulp gulp-connect gulp-open gulp
-sass karma karma-chrome-launcher karma-coverage requirejs karma-jasmine karma-junit-reporter karma-requirejs karma-sonarqube-unit-reporter morgan portscanner serve-index  1
>nul
OBX UI is running at port:8080, Please generate xdl file before proceeding
? Did you generate the xdl file? (Y/n) |
```

4. Open the browser and navigate to http://localhost:8080. after the obx UI is running.

Figure 2-5 Banking Extensibility Workbench


Banking Extensibility Workbench

Entity Details

Entity Name: only Alphanumeric characters (Required)

Please drop XDL file or Click here to upload

Field Details

Field Name	Field Type	Default Value	Field Size
No data to display.			

Add Modify Delete

Child Entity Details

Add Child Entity

No items to display.

Relationship Details

Has Relationship: True [✓ False]

Following are sections present on the OBX UI:

- **Entity Details**
- **Field Details**
- **Child Entity Details**
- **Relationship Details**

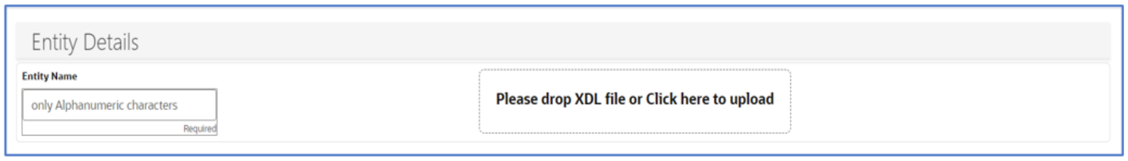
- Entity Details
  This topics helps user to capture the entity name.
- Field Details
  This topic helps user to define the fields for the main entity.
- Child Entity Details
  This topic helps user to define the fields for the Child Entity.
- Relationship Details
  This topic helps user to define the fields for the Relationship Details.

2.5.1 Entity Details

This topics helps user to capture the entity name.

As the Domain Entity pattern an object is primarily defined by its identity is called an Entity.

Figure 2-6 Entity Details


Entity Details

Entity Name: only Alphanumeric characters (Required)

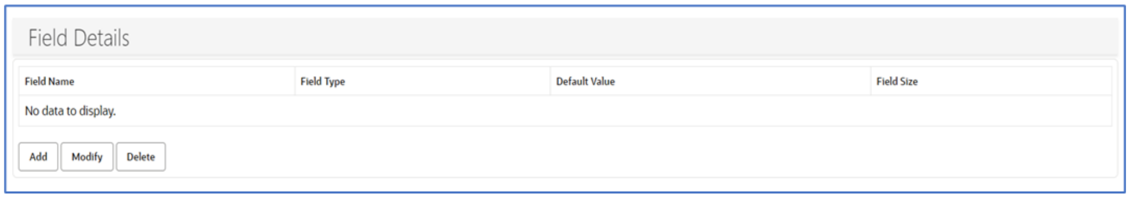
Please drop XDL file or Click here to upload

2.5.2 Field Details

This topic helps user to define the fields for the main entity.

Click the Add button and provide the field details.

Figure 2-7 Field Details



Field Details

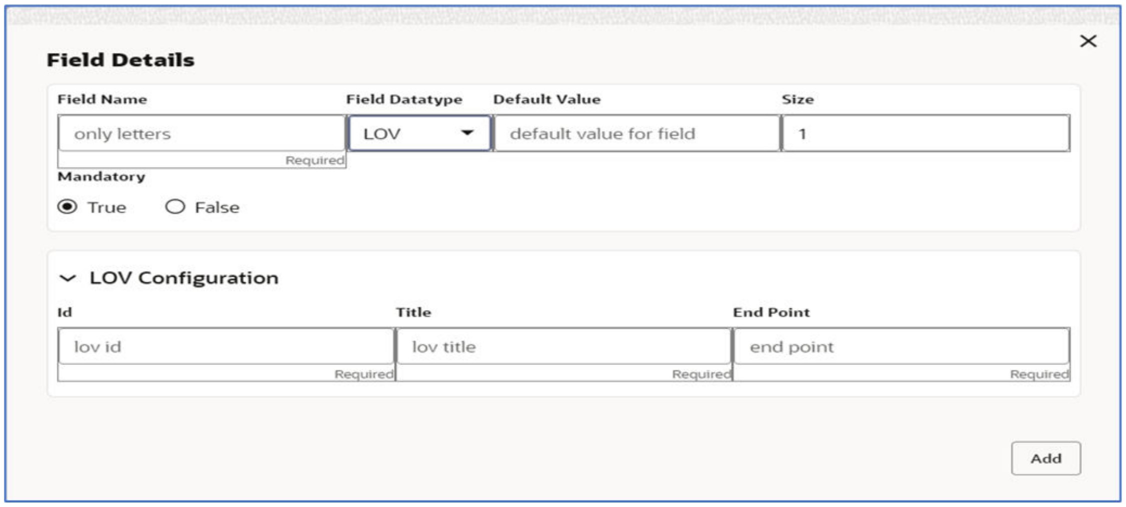
Field Name	Field Type	Default Value	Field Size
No data to display.			

OBX supports the following field types:

Table 2-2 Field types - Field Description

Field	Description
String	The OBX field type is built-in. It's translated to a varchar in SQL scripts, a string type in Java files, and a normal text field in UI components.
Integer	The OBX field type is built-in. It's translated to a number in SQL scripts, a integer type in Java files, and a normal text field in UI components.
Float	The OBX field type is built-in. It's translated to a number in SQL scripts, a float type in Java files, and a normal text field in UI components.
LOV	The OBX field type is inherited from the base product and has its own configuration as below.

Figure 2-8 LOV Configuration



Field Details

Field Name	Field Datatype	Default Value	Size
only letters (Required)	LOV	default value for field	1

Mandatory: [✓ True] False

LOV Configuration

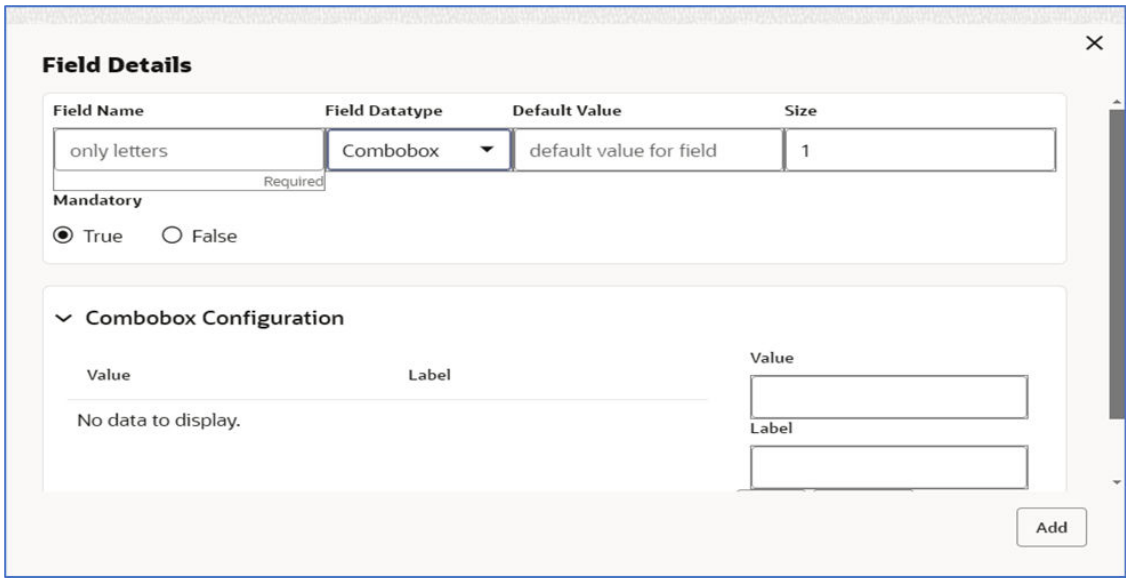
Id	Title	End Point
lov id (Required)	lov title (Required)	end point (Required)

This ID is the specific ID given to this LOV component. The title is displayed on the LOV dialog box, and the endpoint is the service endpoint this field connects to for fetching values.

Table 2-3 LOV component - Field Description

Field Name	Description
Date	This field is also inherited from the base product and add date component on the screen.
Amount	This field is also inherited from the base product and add the amount field on the screen. This field also captures currency along with the amount.
Combobox	This field is taken from Ojet Cookbook and OBX UI provides configurations to needed for this component like value and label.

Figure 2-9 Combobox Configuration



Field Details

Field Name: only letters (Required)
Field Datatype: Combobox
Default Value: default value for field
Size: 1

Mandatory: True [✓] False

Combobox Configuration

Value	Label
No data to display.	

Value
Label

Table 2-4 Combobox Configuration - Field Description

Field	Description
Checkbox	This field type is also taken from Ojet Cookbook and OBX UI provides configurations to needed for this component like value and label.
Toggle Button	This field type is taken from Ojet Cookbook.
Text Area	This field type is taken from Ojet Cookbook.

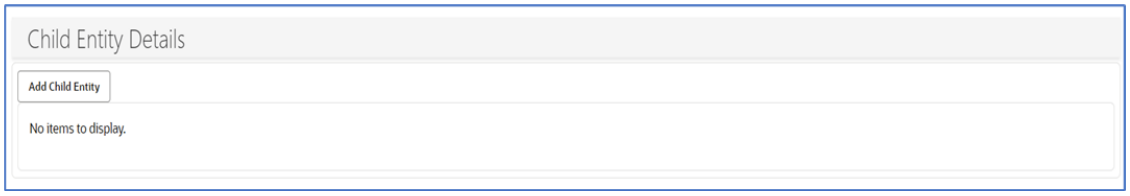
2.5.3 Child Entity Details

This topic helps user to define the fields for the Child Entity.

Use this block for adding the child entities. Once clicked the Add Child Entity Button, it will open a dialog box where we can enter the child entity name. Once clicked ok it will add a child block below with its details.

Add the child entity field details in a similar way like we added for main entity.

Figure 2-10 Child Entity Details



2.5.4 Relationship Details

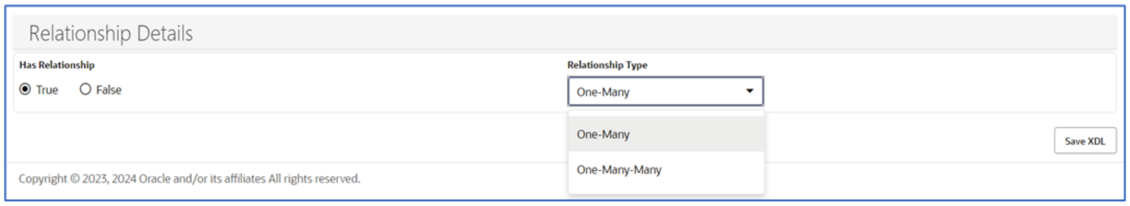
This topic helps user to define the fields for the Relationship Details.

Once all the entity details are added we can define relationship among them. Use this block to define the relationship.

Currently OBX supports two types of relationships:

- **One to Many**
- **One to Many to Many**

Figure 2-11 Relationship Details



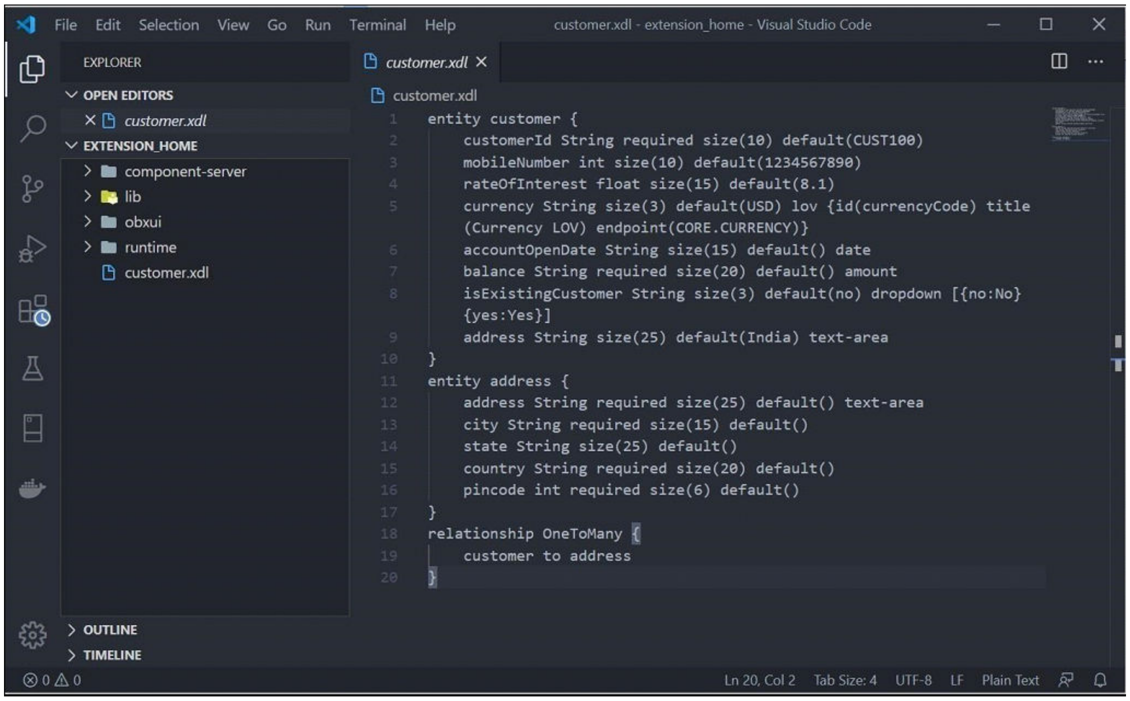
Once all of the above Entity, Field Details & Relationship is created click on the Save XDL button and it will save the xdl file on machine.

Note:

Its recommended to put the xdl file under the same **extension_home** folder and give it proper name (generally main entity name).

The final XDL file looks like below:

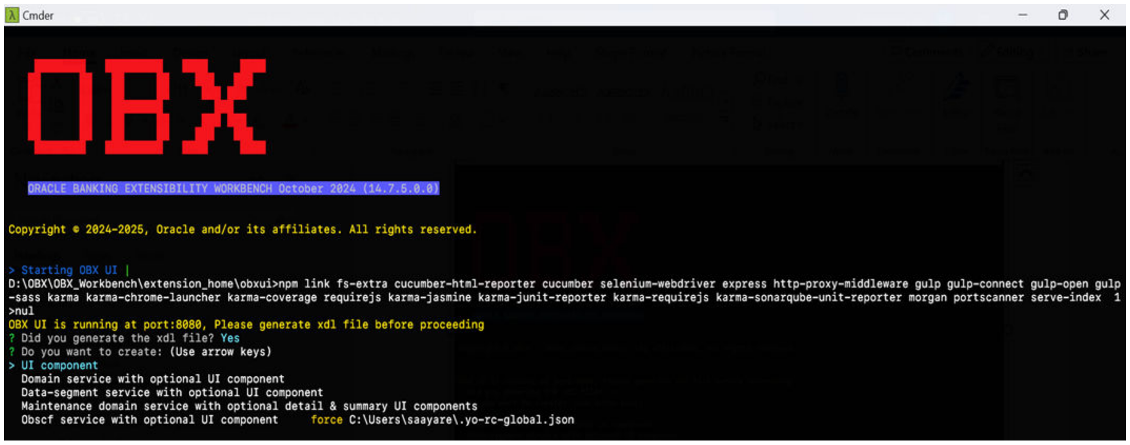
Figure 2-12   XDL File Folder


File Edit Selection View Go Run Terminal Help
customer.xdl - extension_home - Visual Studio Code
EXPLORER
OPEN EDITORS
customer.xdl
EXTENSION_HOME
component-server
lib
obxui
runtime
customer.xdl
OUTLINE
TIMELINE
Ln 20, Col 2   Tab Size: 4   UTF-8   LF   Plain Text

```
entity customer {
    customerId String required size(10) default(CUST100)
    mobileNumber int size(10) default(1234567890)
    rateOfInterest float size(15) default(8.1)
    currency String size(3) default(USD) lov {id(currencyCode) title
    (Currency LOV) endpoint(CORE.CURRENCY)}
    accountOpenDate String size(15) default() date
    balance String required size(20) default() amount
    isExistingCustomer String size(3) default(no) dropdown [{no:No}
    {yes:Yes}]
    address String size(25) default(India) text-area
}
entity address {
    address String required size(25) default() text-area
    city String required size(15) default()
    state String size(25) default()
    country String required size(20) default()
    pincode int required size(6) default()
}
relationship OneToMany {
    customer to address
}
```

Once XDL file is generated you may come back to cmder main tab where it is waiting for the input. You may proceed creating next set of artifacts which are described in next sections.

Figure 2-13   OBX UI



```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

> Starting OBX UI |
D:\OBX\OBX_Workbench\extension_home\obxui>npm link fs-extra cucumber-html-reporter cucumber selenium-webdriver express http-proxy-middleware gulp gulp-connect gulp-open gulp
-sass karma karma-chrome-launcher karma-coverage requirejs karma-jasmine karma-junit-reporter karma-requirejs karma-sonarqube-unit-reporter morgan portscanner serve-index  1
>nul
OBX UI is running at port:8080, Please generate xdl file before proceeding
? Did you generate the xdl file? Yes
? Do you want to create: (Use arrow keys)
> UI component
  Domain service with optional UI component
  Data-segment service with optional UI component
  Maintenance domain service with optional detail & summary UI components
  Obscf service with optional UI component     force C:\Users\saayare\.yo-rc-global.json
```

3

Service Extensions

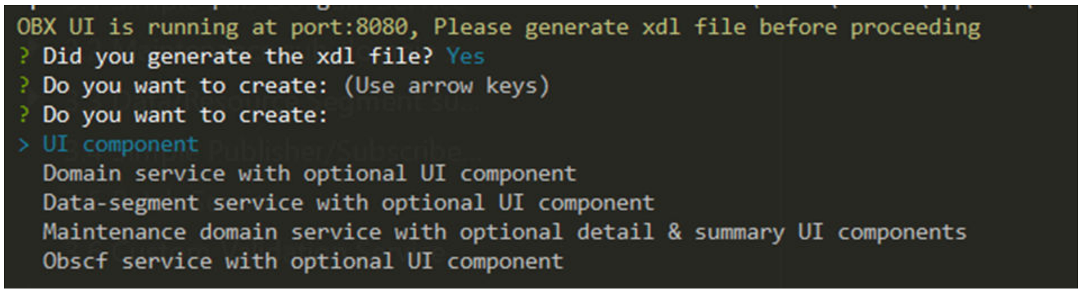
This topic provides the systematic instructions to perform the basic operations on the selected records.

Using OBX we can create multiple types of service extensions. This services extension has complete infrastructure needed to build to service. Also, the source folder generated out the box from OBX follows the package structure which is adopted and used by base/kernel teams to keep it in sync.

There are two ways to generate the service artifact:

1. Select the category immediately after generating the XDL file and proceed.

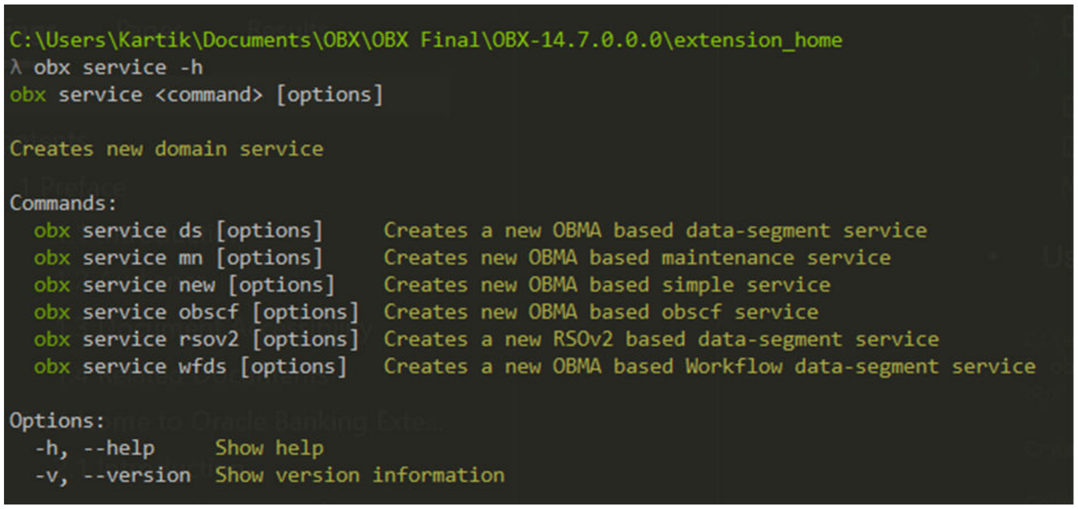
Figure 3-1 XDL File



```
OBX UI is running at port:8080, Please generate xdl file before proceeding
? Did you generate the xdl file? Yes
? Do you want to create: (Use arrow keys)
? Do you want to create:
> UI component
  Domain service with optional UI component
  Data-segment service with optional UI component
  Maintenance domain service with optional detail & summary UI components
  Obscf service with optional UI component
```

2. Use the service specific command to generate different types.

Figure 3-2 Command



```
C:\Users\Kartik\Documents\OBX\OBX Final\OBX-14.7.0.0.0\extension_home
λ obx service -h
obx service <command> [options]

Creates new domain service

Commands:
  obx service ds [options]     Creates a new OBMA based data-segment service
  obx service mn [options]     Creates new OBMA based maintenance service
  obx service new [options]    Creates new OBMA based simple service
  obx service obscf [options]  Creates new OBMA based obscf service
  obx service rsov2 [options]  Creates a new RSOv2 based data-segment service
  obx service wfds [options]   Creates a new OBMA based Workflow data-segment service

Options:
  -h, --help     Show help
  -v, --version  Show version information
```

Note:

Both above ways will generate the same artifacts.

- Simple Sub Domain Service
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- Maintenance Sub Domain Service
  This topic describes the process to generate the Maintenance Sub Domain Service.
- Data/Resource Segment Sub Domain Service
  This topic provides the systematic instructions to perform the basic operations on the Data/ Resource Segment sub domain service.
- Simple Publisher/Subscriber Event Service
  This topic the systematic instructions to perform the basic process to generate simple publisher/subscriber event service.
- Batch Service
  This topic describes the process to generate Oracle Banking Microservices Architecture (OBMA) based Batch service.
- Custom Validation Service
  This topic provides the systematic instructions to generate custom validation service.
- Steps to Adopt Multi in Existing Service
  This topic provides the systematic instruction to adop multi in existing service.
- Service Extensibility
  This topic provides the systematic instructions to perform the basic operations on the selected records.

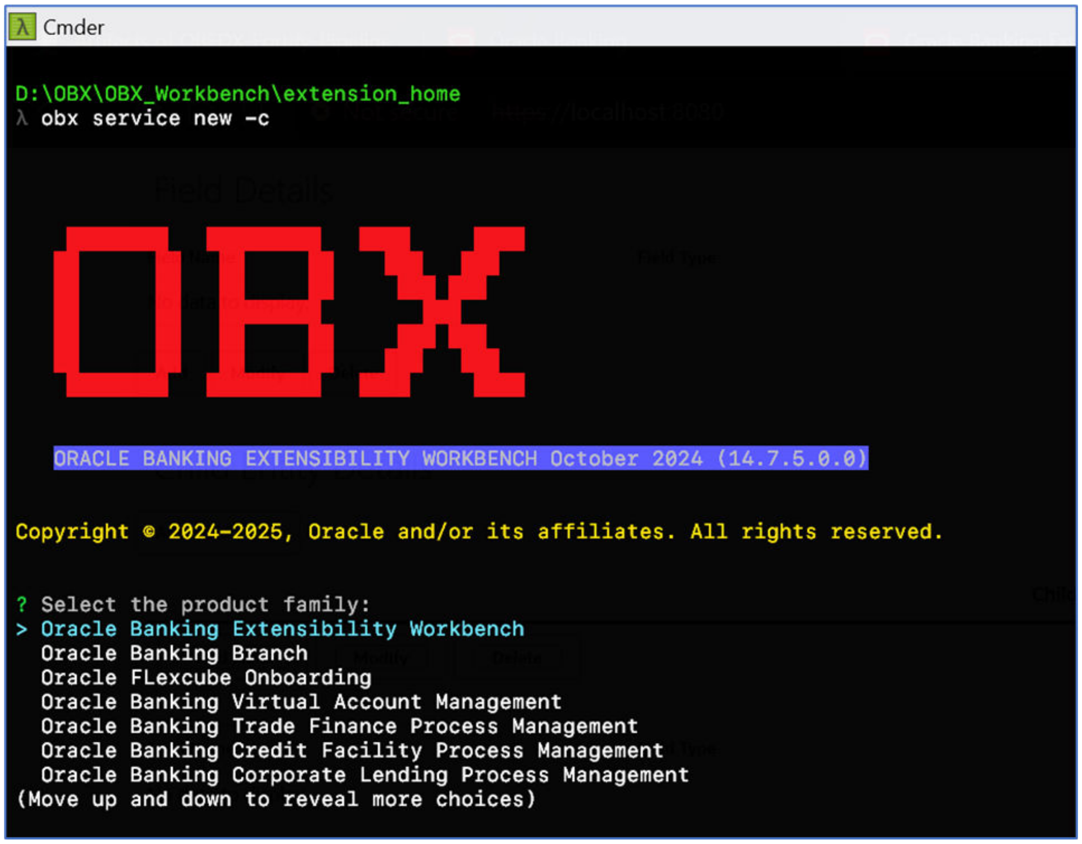
3.1 Simple Sub Domain Service

This topic provides the systematic instructions to perform the basic operations on the selected records.

This is one of the primary use cases in OBX. To generate the simple sub-domain service, follow the below steps:

1. Navigate to same **extension_home** folder using cmder.
2. Use the command **obx service new -c**.

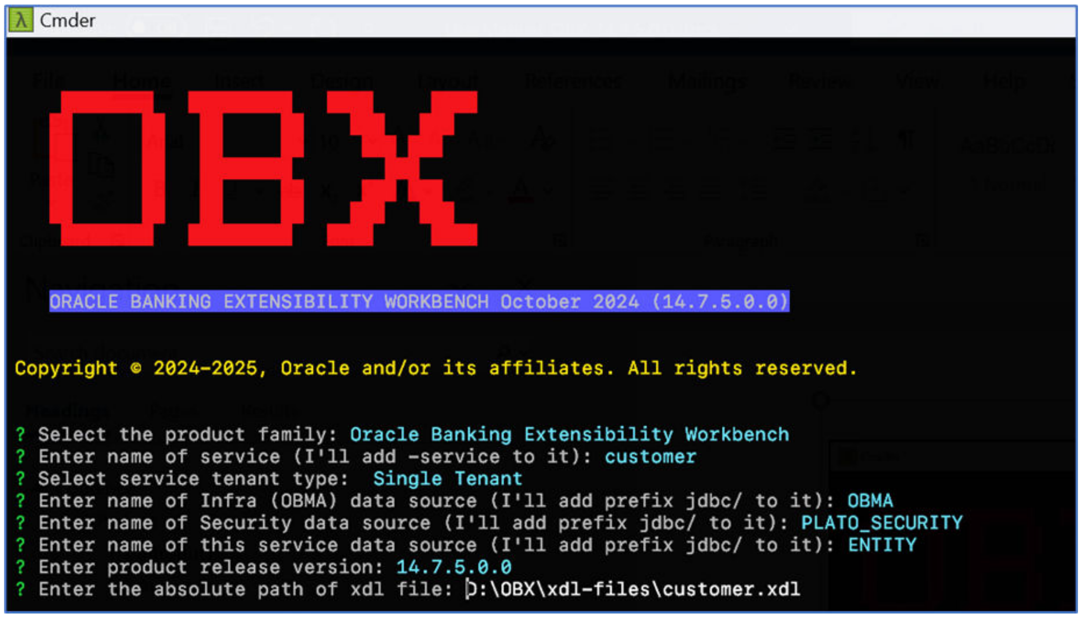
Figure 3-3 OBX Service new -c



```
Cmder

D:\OBX\OBX_Workbench\extension_home
λ obx service new -c

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Select the product family:
> Oracle Banking Extensibility Workbench
  Oracle Banking Branch
  Oracle FLexcube Onboarding
  Oracle Banking Virtual Account Management
  Oracle Banking Trade Finance Process Management
  Oracle Banking Credit Facility Process Management
  Oracle Banking Corporate Lending Process Management
(Move up and down to reveal more choices)
```

3. Once this command is fired, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement.

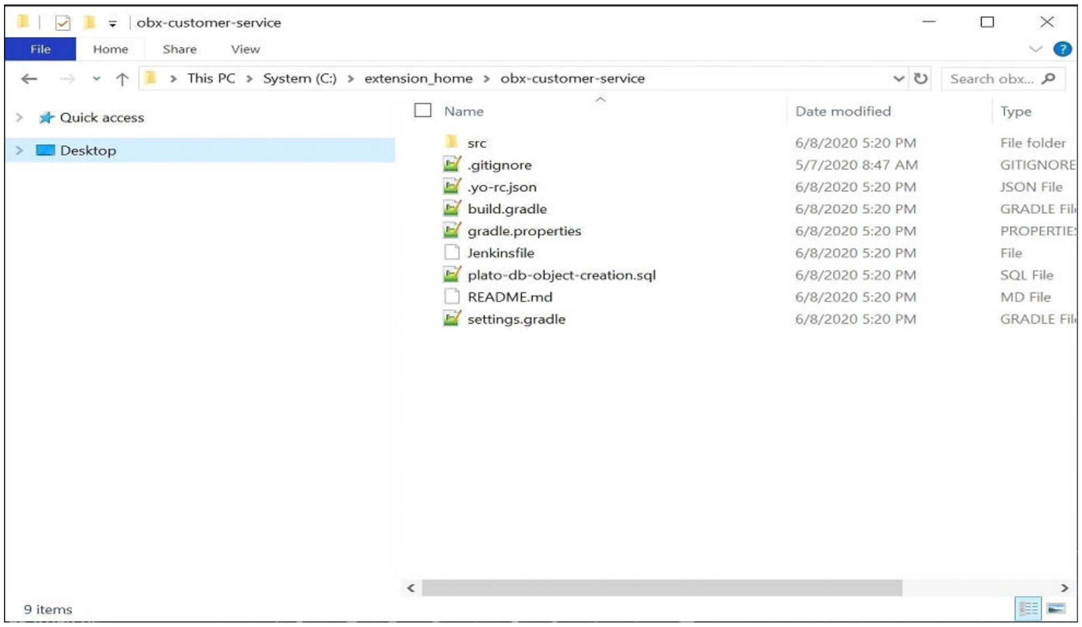
Figure 3-4 OBX UI



```
Cmder

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Select the product family: Oracle Banking Extensibility Workbench
? Enter name of service (I'll add -service to it): customer
? Select service tenant type:  Single Tenant
? Enter name of Infra (OBMA) data source (I'll add prefix jdbc/ to it): OBMA
? Enter name of Security data source (I'll add prefix jdbc/ to it): PLATO_SECURITY
? Enter name of this service data source (I'll add prefix jdbc/ to it): ENTITY
? Enter product release version: 14.7.5.0.0
? Enter the absolute path of xdl file: D:\OBX\xdl-files\customer.xdl
```

4. Once all the questions are answered and path of XDL is given, it will generate a folder inside the **extension_home** folder.

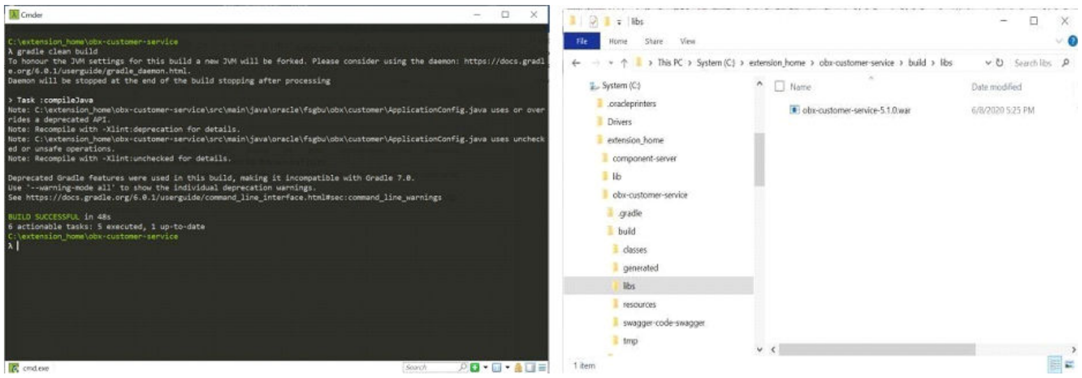
Figure 3-5 OBX Customer Service



Name	Date modified	Type
src	6/8/2020 5:20 PM	File folder
.gitignore	5/7/2020 8:47 AM	GITIGNORE
.yo-rc.json	6/8/2020 5:20 PM	JSON File
build.gradle	6/8/2020 5:20 PM	GRADLE Fil
gradle.properties	6/8/2020 5:20 PM	PROPERTIE
Jenkinsfile	6/8/2020 5:20 PM	File
plato-db-object-creation.sql	6/8/2020 5:20 PM	SQL File
README.md	6/8/2020 5:20 PM	MD File
settings.gradle	6/8/2020 5:20 PM	GRADLE Fil

5. Select the option based on your requirement for question **Do you want to create UI component for this service? (Y/n)**.
6. For building the service go into the service folder from cmder and run the command **gradle clean build**.
7. This will build the service and we can find the war of the service getting created inside the build/libs directory.

Figure 3-6 Lib's Directory



```
C:\extension_home\obx-customer-service
λ gradle clean build
To honour the JVM settings for this build a new JVM will be forked. Please consider using the daemon: https://docs.gradl
e.org/6.0.1/userguide/gradle_daemon.html.
Daemon will be stopped at the end of the build stopping after processing

> Task :compileJava
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses or over
rides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses uncheck
ed or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.0.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 48s
6 actionable tasks: 5 executed, 1 up-to-date
C:\extension_home\obx-customer-service
λ |
```

Name	Date modified
obx-customer-service-5.1.0.war	6/8/2020 5:25 PM

8. Use this service and deploy it in your environment.

Note:

- DB scripts for the service will be generated inside the folder: **\extension_home\obxcustomerservice\src\main\resources\db**
- Compile the Entity script in the entity schema created for extensions only.
- Service created as part of extension should be deployed in separate domain and should not be mixed or co-deployed with any other product specific services.
- Before compiling **CONFIG_SCRIPT**.sql in verify the entries manually and change it according to your setup.
- Also, verify **PLATO_TABLE_SCRIPT.sql** before executing it in the schema it may contain some dummy values.

3.2 Maintenance Sub Domain Service

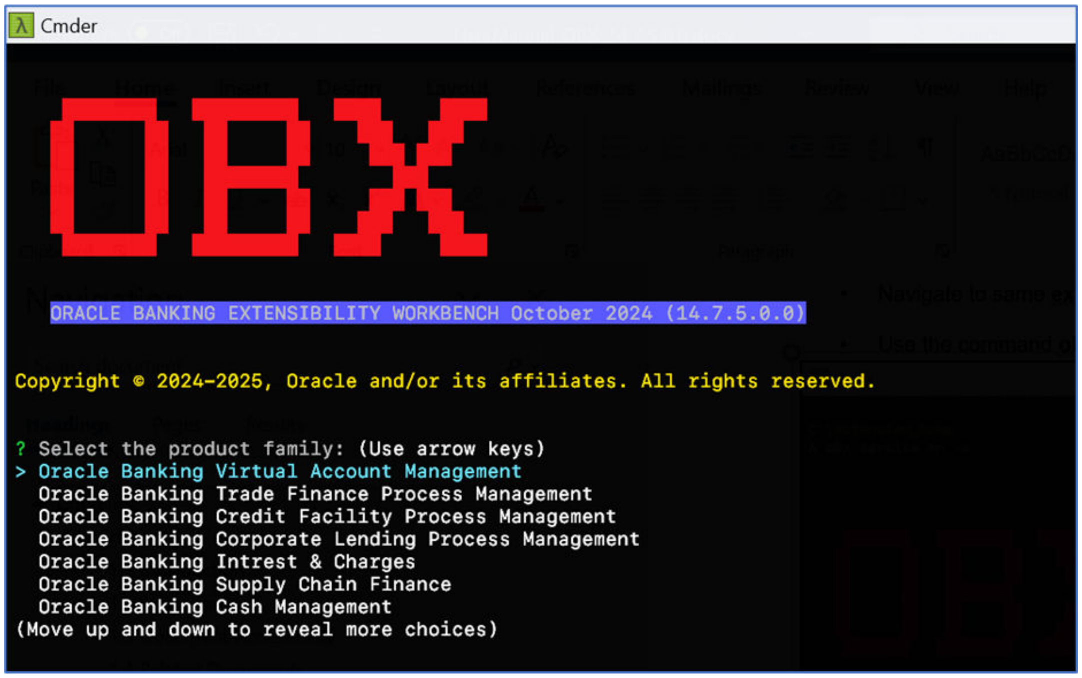
This topic describes the process to generate the Maintenance Sub Domain Service.

Maintenance service generally has concept of main and worktable. This allows enables functionality where all the Authorized records goes to main table and all the unauthorized records goes to worktable. Also, with this type of service we attach audit details to payload.

To generate the maintenance type of service, follow the below steps:

1. Navigate to same **extension_home** folder using cmder.
2. Use the command **obx service mn -c**.

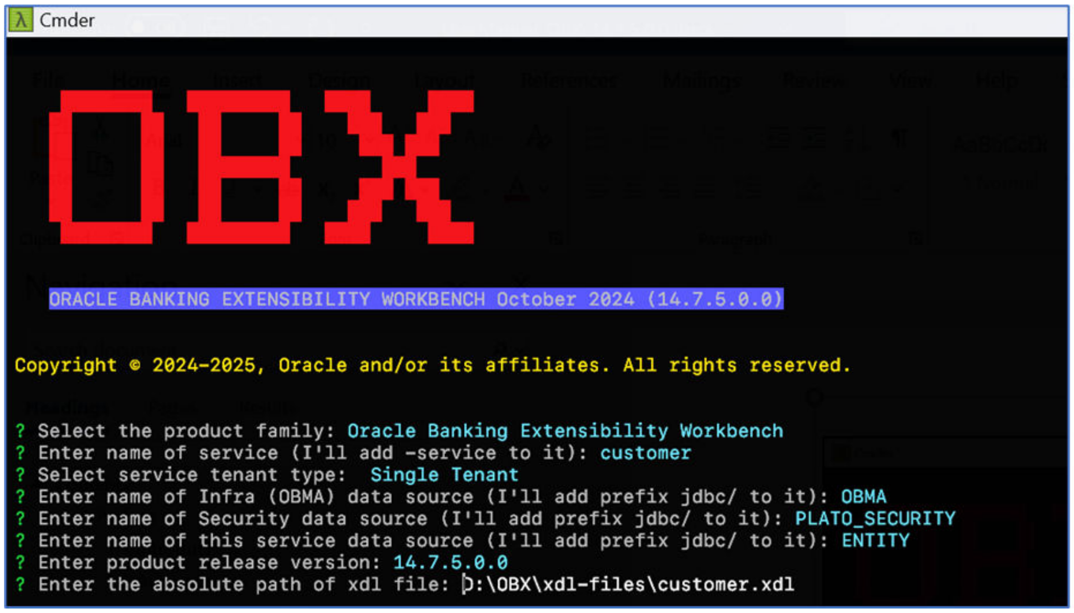
Figure 3-7 OBX Command



```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Select the product family: (Use arrow keys)
> Oracle Banking Virtual Account Management
  Oracle Banking Trade Finance Process Management
  Oracle Banking Credit Facility Process Management
  Oracle Banking Corporate Lending Process Management
  Oracle Banking Intrest & Charges
  Oracle Banking Supply Chain Finance
  Oracle Banking Cash Management
(Move up and down to reveal more choices)
```

3. Once this command is fired, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement.

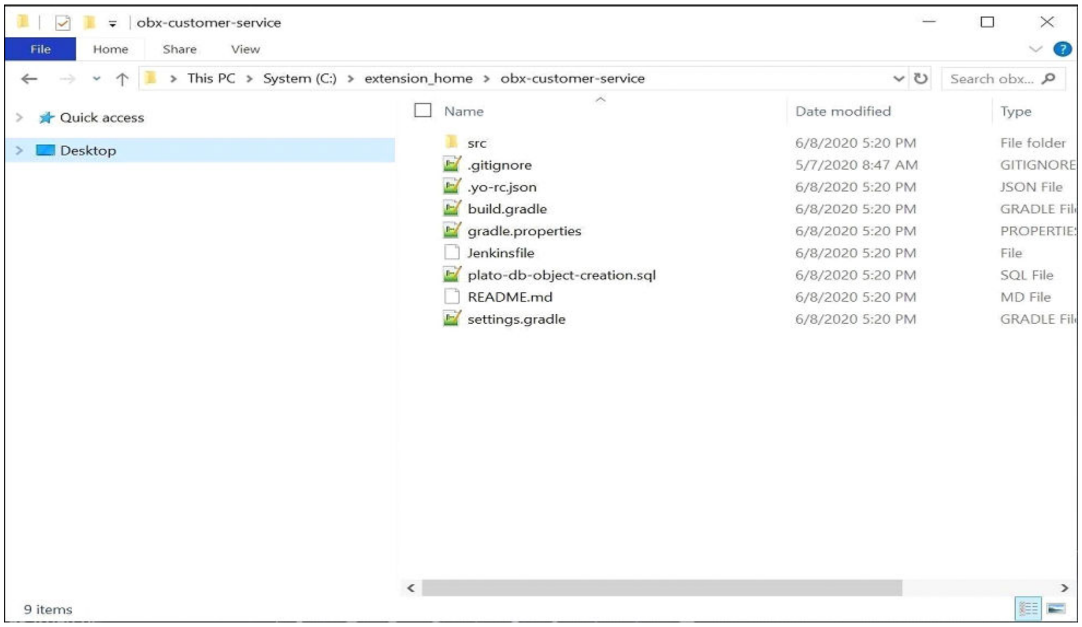
Figure 3-8   OBX Setup



```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Select the product family: Oracle Banking Extensibility Workbench
? Enter name of service (I'll add -service to it): customer
? Select service tenant type:  Single Tenant
? Enter name of Infra (OBMA) data source (I'll add prefix jdbc/ to it): OBMA
? Enter name of Security data source (I'll add prefix jdbc/ to it): PLATO_SECURITY
? Enter name of this service data source (I'll add prefix jdbc/ to it): ENTITY
? Enter product release version: 14.7.5.0.0
? Enter the absolute path of xdl file: D:\OBX\xdl-files\customer.xdl
```

4. Once all the questions are answered and path of XDL is given, it will generate a folder inside the **extension_home** folder.

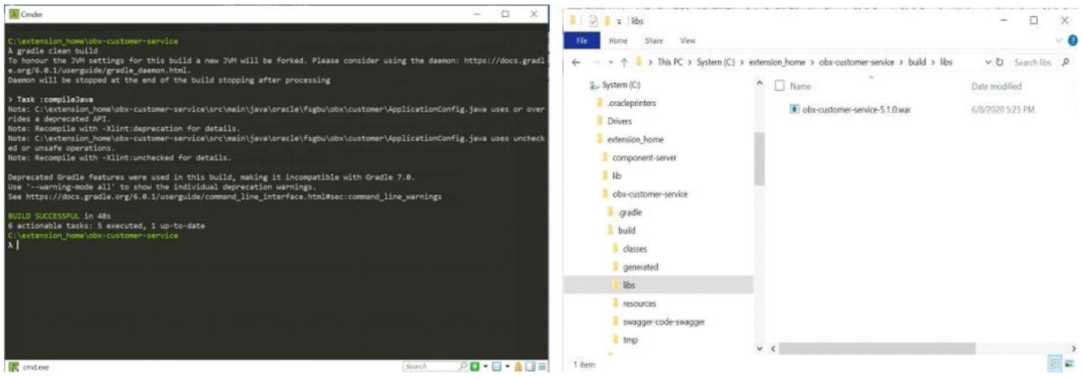
Figure 3-9   Extension Home Folder



Name	Date modified	Type
src	6/8/2020 5:20 PM	File folder
.gitignore	5/7/2020 8:47 AM	GITIGNORE
.yo-rc.json	6/8/2020 5:20 PM	JSON File
build.gradle	6/8/2020 5:20 PM	GRADLE Fil
gradle.properties	6/8/2020 5:20 PM	PROPERTIE
Jenkinsfile	6/8/2020 5:20 PM	File
plato-db-object-creation.sql	6/8/2020 5:20 PM	SQL File
README.md	6/8/2020 5:20 PM	MD File
settings.gradle	6/8/2020 5:20 PM	GRADLE Fil

5. Select the option based on your requirement for question: **Do you want to create a Maintenance and Summary Components for this service? (Y/n)**.
6. For building the service go into the service folder from cmder and run the command **gradle clean build**.
7. This will build the service and we can find the war of the service getting created inside the build/libs directory.

Figure 3-10   Lib's Directory



```
C:\extension_home\obx-customer-service
λ gradle clean build
To honour the JVM settings for this build a new JVM will be forked. Please consider using the daemon: https://docs.gradl
e.org/6.0.1/userguide/gradle_daemon.html.
Daemon will be stopped at the end of the build stopping after processing

> Task :compileJava
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses or over
rides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses uncheck
ed or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.0.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 48s
6 actionable tasks: 5 executed, 1 up-to-date
C:\extension_home\obx-customer-service
λ
```

This PC > System (C:) > extension_home > obx-customer-service > build > libs

Name	Date modified
obx-customer-service-5.1.0.war	6/8/2020 5:25 PM

8. Use this service and deploy it in your environment.

Note:

- DB scripts for the service will be generated inside the folder: **\extension_home\obxcustomerservice\src\main\resources\db**
- Compile the Entity script in the entity schema created for extensions only.
- Service created as part of extension should be deployed in separate domain and should not be mixed or co-deployed with any other product specific services.
- Here Security Management System (SMS) scripts are also generated. **\extension_home\obxcustomer-service\src\main\resources\db\sms**
- Execute the SMS script in sms schema, here we only generate the functional activity of service. Assigning to proper role should be done according to the steps mentioned in base application.

3.3 Data/Resource Segment Sub Domain Service

This topic provides the systematic instructions to perform the basic operations on the Data/ Resource Segment sub domain service.

This topic consists of the following sub-topics:

- RSOV1
  This topic describes the process to generate the data/resource segment type of maintenance service.
- RSOV2 DS
  This topic provides information on RSOV2 DS operations data segment.
- Workflow DS
  This topic provides information on workflow details data segment.

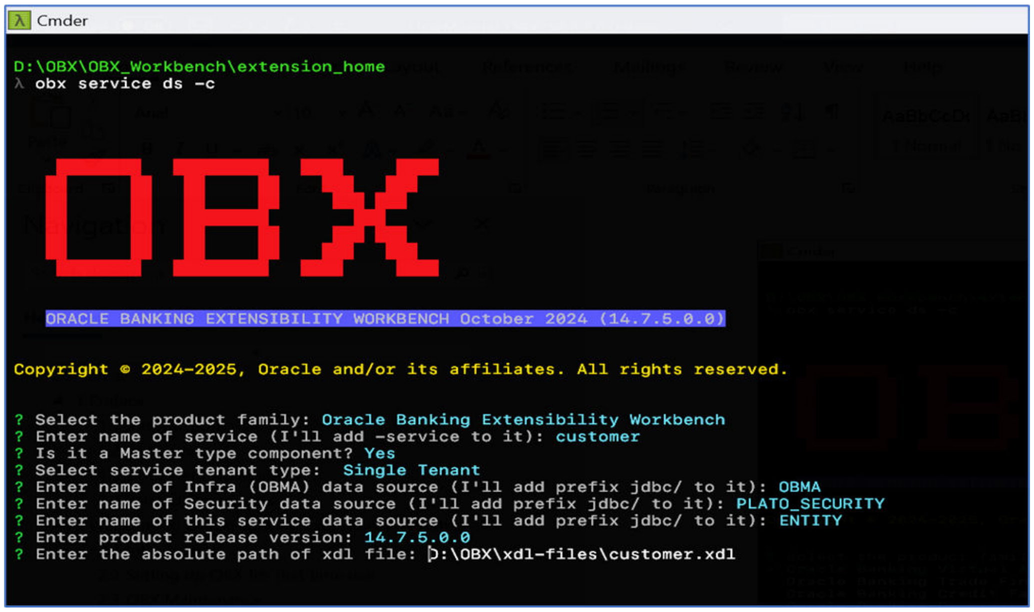
3.3.1 RSOV1

This topic describes the process to generate the data/resource segment type of maintenance service.

Here we can generate Master Type of data segment or child type of data segment.

- **Master Type**: This case is used when user wants to generate the complete flow from scratch. It will generate the new screen class code for the data segments.
- **Child Type**: This is primarily used when user wants to attach a single data-segment in the existing flow/process. Generally, this existing flow/process is available in the base product. We use the same screen class code from base and attach our data segment to it. To generate it please follow the below steps:
  1. Navigate to same **extension_home** folder using cmder
  2. Use the command **obx service ds -c**.

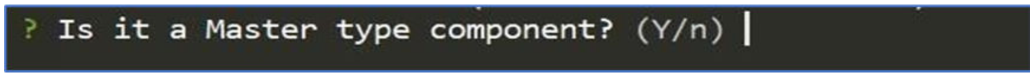
Figure 3-11 OBX service ds - c



```
D:\OBX\OBX_Workbench\extension_home
λ obx service ds -c

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Select the product family: Oracle Banking Extensibility Workbench
? Enter name of service (I'll add -service to it): customer
? Is it a Master type component? Yes
? Select service tenant type:  Single Tenant
? Enter name of Infra (OBMA) data source (I'll add prefix jdbc/ to it): OBMA
? Enter name of Security data source (I'll add prefix jdbc/ to it): PLATO_SECURITY
? Enter name of this service data source (I'll add prefix jdbc/ to it): ENTITY
? Enter product release version: 14.7.5.0.0
? Enter the absolute path of xdl file: D:\OBX\xdl-files\customer.xdl
```

  3. Once this command is fired, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement.
  4. Select the type of component according to your requirement.

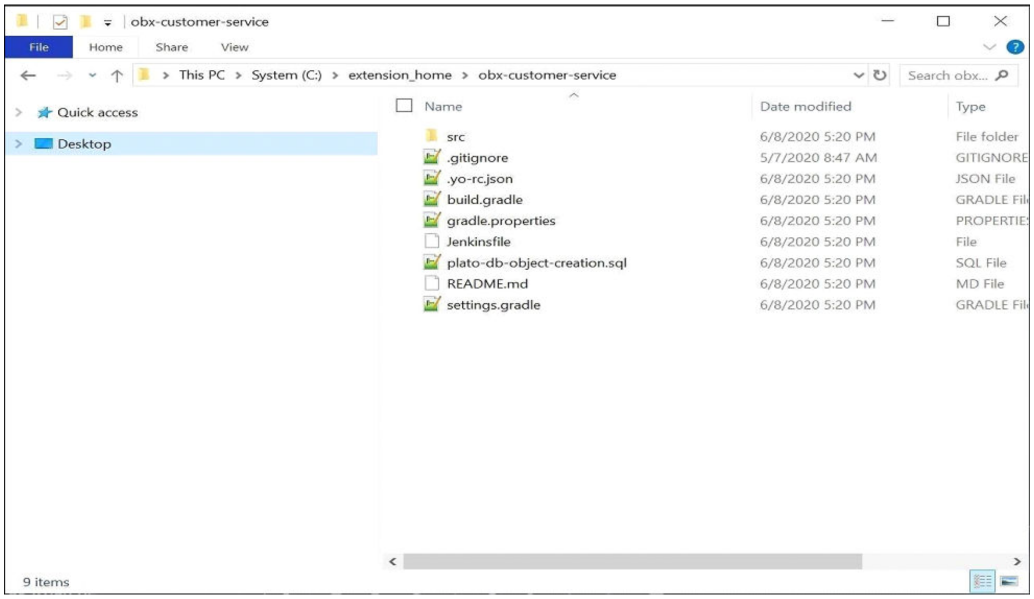
Figure 3-12 Master Type Component



```
? Is it a Master type component? (Y/n)
```

  5. Once all the questions are answered and path of XDL is given, it will generate a folder inside the **extension_home** folder.

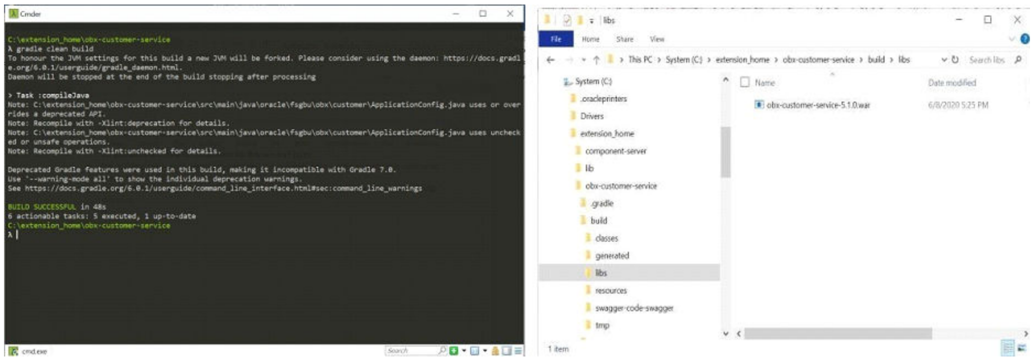
Figure 3-13   Extension Home Folder



```
This PC > System (C:) > extension_home > obx-customer-service

Name                          Date modified        Type
src                           6/8/2020 5:20 PM     File folder
.gitignore                    5/7/2020 8:47 AM     GITIGNORE
.yo-rc.json                   6/8/2020 5:20 PM     JSON File
build.gradle                  6/8/2020 5:20 PM     GRADLE File
gradle.properties             6/8/2020 5:20 PM     PROPERTIES
Jenkinsfile                   6/8/2020 5:20 PM     File
plato-db-object-creation.sql  6/8/2020 5:20 PM     SQL File
README.md                     6/8/2020 5:20 PM     MD File
settings.gradle               6/8/2020 5:20 PM     GRADLE File

9 items
```

6. Select the option based on your requirement for question: **Do you want to create a Data Segment for this service? (Y/n)**.
7. For building the service, go into the service folder from cmder and run the command: **gradle clean build**.
8. This will build the service and we can find the war of the service getting created inside the build/libs directory.

Figure 3-14   Lib's Directory



```
C:\extension_home\obx-customer-service
λ gradle clean build
To honour the JVM settings for this build a new JVM will be forked. Please consider using the daemon: https://docs.gradl
e.org/6.0.1/userguide/gradle_daemon.html.
Daemon will be stopped at the end of the build stopping after processing

> Task :compileJava
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses or over
rides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses uncheck
ed or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.0.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 48s
6 actionable tasks: 5 executed, 1 up-to-date
C:\extension_home\obx-customer-service
λ
```

```
This PC > System (C:) > extension_home > obx-customer-service > build > libs

Name                              Date modified
obx-customer-service-5.1.0.war    6/8/2020 5:25 PM

1 item
```

9. Use this service and deploy it in your environment.

Note:

- DB scripts for the service will be generated inside the folder: **\extension_home\obxcustomerservice\src\main\resources\db**
- Compile the Entity script in the entity schema created for extensions only.
- Service created as part of extension should be deployed in separate domain and should not be mixed or co-deployed with any other product specific services.
- Here Security Management System (SMS) scripts are also generated: **\extension_home\obxcustomer-service\src\main\resources\db\sms**.
- Execute the SMS script in sms schema, here we only generate the functional activity of service. Assigning to proper role should be done according to the steps mentioned in base application.
- Here along with SMS and Entity, CMC scripts are also generated under folder: **\extension_home\obx-customer-service\src\main\resources\db\cmc**.
- Execute them in the CMC schema.
- **Screen Class and Data Segment** has to be maintained from the UI which is present under common core.

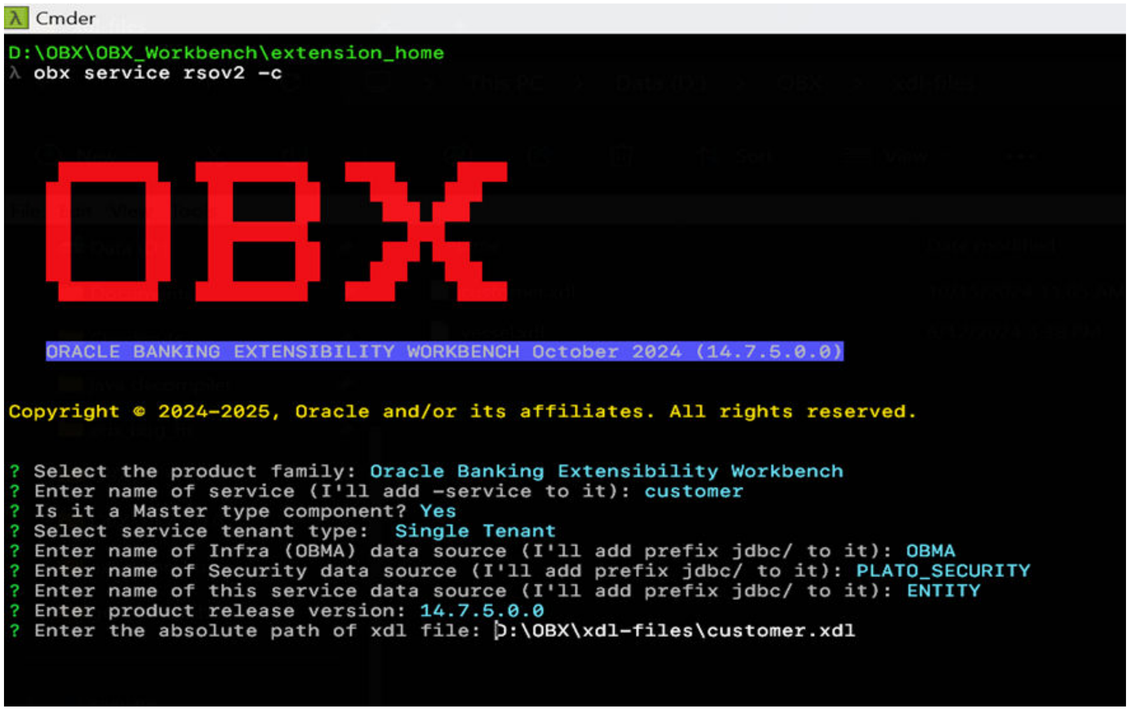
3.3.2 RSOV2 DS

This topic provides information on RSOV2 DS operations data segment.

For Nov patchset innovation - RSOv1 is discontinued and RSOv2 should be adopted for all customizations for maintenance services.

Here we can generate Master Type of data segment or child type of data segment.

Figure 3-15 OBX Service RSOV2-C



```
D:\OBX\OBX_Workbench\extension_home
λ obx service rsov2 -c

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Select the product family: Oracle Banking Extensibility Workbench
? Enter name of service (I'll add -service to it): customer
? Is it a Master type component? Yes
? Select service tenant type:  Single Tenant
? Enter name of Infra (OBMA) data source (I'll add prefix jdbc/ to it): OBMA
? Enter name of Security data source (I'll add prefix jdbc/ to it): PLATO_SECURITY
? Enter name of this service data source (I'll add prefix jdbc/ to it): ENTITY
? Enter product release version: 14.7.5.0.0
? Enter the absolute path of xdl file: D:\OBX\xdl-files\customer.xdl
```

- **Master Type**: This will create two components one would be core component of product services which will contain utility service, the other one would be the master type of component that needs to be included in the core services folder.
- **Child Type**: This will create only one component that needs to be included in the core services (containing utility).

Follow the steps to deploy it in your environment:

1. Navigate to same **extension_home** folder using cmder.
2. Use the command **obx service rsov2 -c**.
3. Once this command is fired, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement.
4. Select the type of component according to your requirement.
5. Once all the questions are answered and path of XDL is given, it will generate the folders accordingly inside the **extension_home**.
6. Select the option based on your requirement for question: **Do you want to create a Data Segment for this service?(Y/N).**
7. Include the folders created either master or child inside the (core-services), folder and make the modifications accordingly.
8. Use this service and deploy it in your environment.

3.3.3 Workflow DS

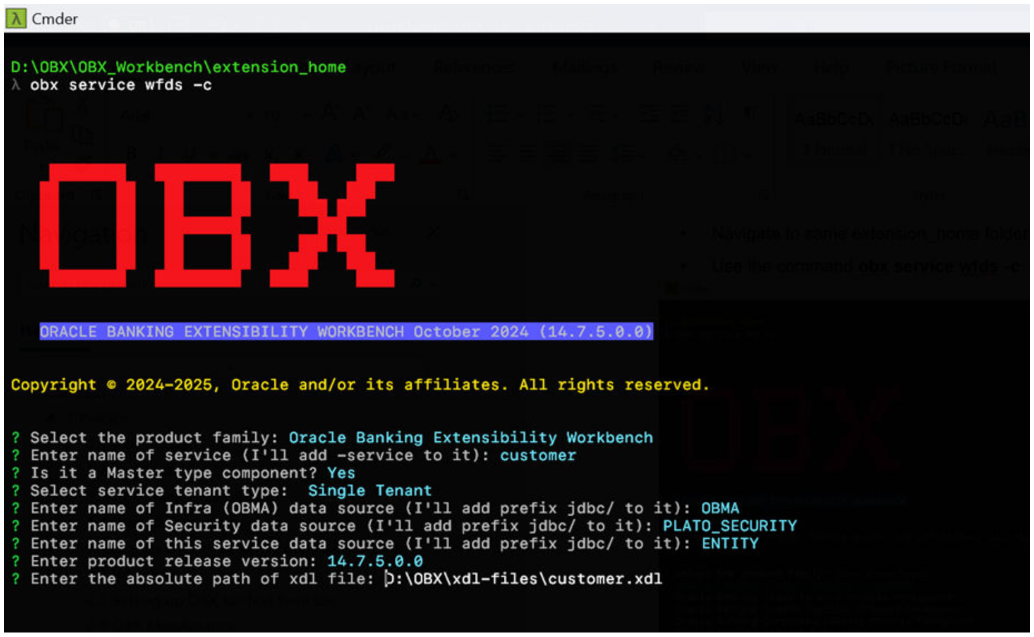
This topic provides information on workflow details data segment.

Here, the user can generate master or child type if data segment.

- **Master Type**: This case is used when user wants to generate the complete flow from scratch. It will generate the new screen class code for the data segments.
- **Child Type**: This is primarily used when user wants to attach a single data-segment in the existing flow/process. Generally, this existing flow/process is available in the base product. We use the same screen class code from base and attach our data segment to child type. To generate master or child type if data segment, follow the below steps:

1. Navigate to same **extension_home** folder using cmder.
2. Use the command **obx service wfds -c**.

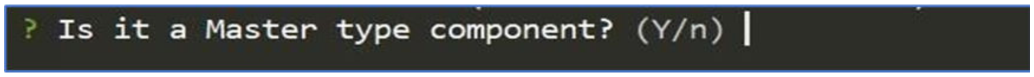
Figure 3-16 OBX service wfds -c



```
D:\OBX\OBX_Workbench\extension_home
λ obx service wfds -c

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Select the product family: Oracle Banking Extensibility Workbench
? Enter name of service (I'll add -service to it): customer
? Is it a Master type component? Yes
? Select service tenant type:  Single Tenant
? Enter name of Infra (OBMA) data source (I'll add prefix jdbc/ to it): OBMA
? Enter name of Security data source (I'll add prefix jdbc/ to it): PLATO_SECURITY
? Enter name of this service data source (I'll add prefix jdbc/ to it): ENTITY
? Enter product release version: 14.7.5.0.0
? Enter the absolute path of xdl file: D:\OBX\xdl-files\customer.xdl
```

3. Once this command is fired, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement.
4. Select the type of component according to your requirement.

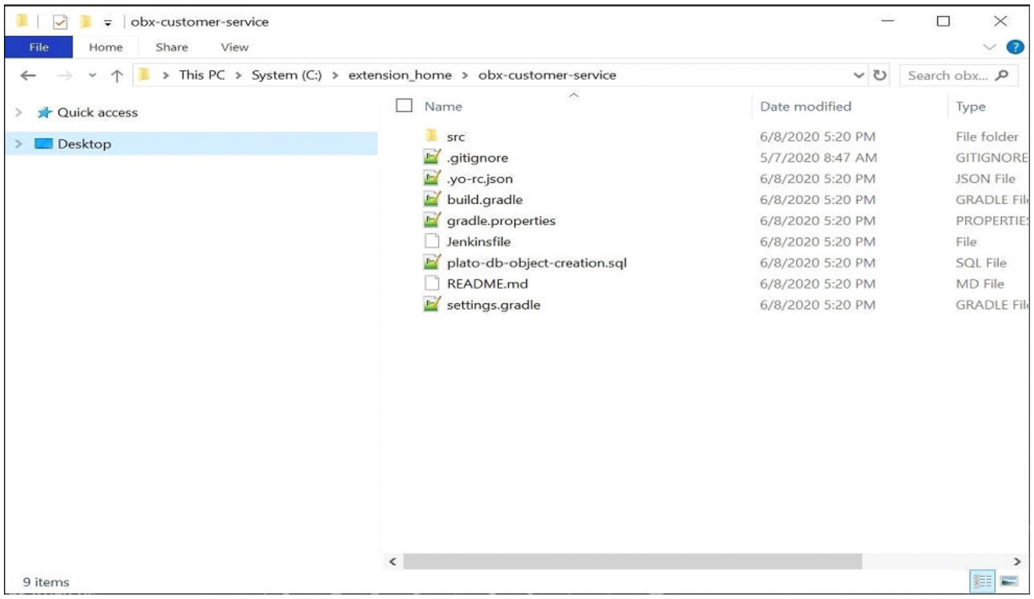
Figure 3-17 Component Type



```
? Is it a Master type component? (Y/n)
```

5. Once all the questions are answered and path of XDL is given, it will generate a folder inside the **extension_home** folder.

Figure 3-18 Extension Home Folder



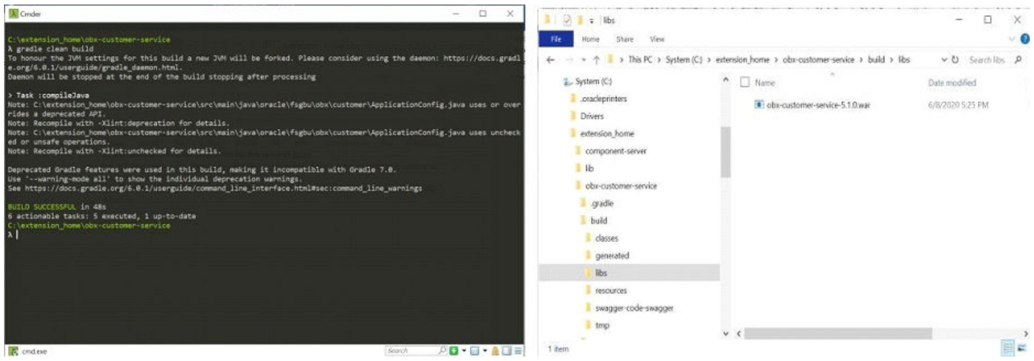
This PC > System (C:) > extension_home > obx-customer-service

Name	Date modified	Type
src	6/8/2020 5:20 PM	File folder
.gitignore	5/7/2020 8:47 AM	GITIGNORE
.yo-rc.json	6/8/2020 5:20 PM	JSON File
build.gradle	6/8/2020 5:20 PM	GRADLE Fil
gradle.properties	6/8/2020 5:20 PM	PROPERTIE
Jenkinsfile	6/8/2020 5:20 PM	File
plato-db-object-creation.sql	6/8/2020 5:20 PM	SQL File
README.md	6/8/2020 5:20 PM	MD File
settings.gradle	6/8/2020 5:20 PM	GRADLE Fil

9 items

6. Select the option based on your requirement for question: **Do you want to create a Data Segment for this service? (Y/n)**.
7. For building the service go into the service folder from cmder and run the command **gradle clean build**.
8. This will build the service and we can find the war of the service getting created inside the build/libs directory.

Figure 3-19 Lib's Directory



```
C:\extension_home\obx-customer-service
λ gradle clean build
To honour the JVM settings for this build a new JVM will be forked. Please consider using the daemon: https://docs.gradl
e.org/6.0.1/userguide/gradle_daemon.html.
Daemon will be stopped at the end of the build stopping after processing

> Task :compileJava
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses or over
rides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses uncheck
ed or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.0.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 48s
6 actionable tasks: 5 executed, 1 up-to-date
C:\extension_home\obx-customer-service
λ
```

This PC > System (C:) > extension_home > obx-customer-service > build > libs

Name	Date modified
obx-customer-service-5.1.0.war	6/8/2020 5:25 PM

1 item

9. Use this service and deploy it in your environment.

Note:

- DB scripts for the service will be generated inside the folder. **\extension_home\obxcustomerservice\src\main\resources\db**
- Compile the Entity script in the entity schema created for extensions only.
- Service created as part of extension should be deployed in separate domain and should not be mixed or co-deployed with any other product specific services.
- Here Security Management System (SMS) scripts are also generated. **\extension_home\obxcustomer-service\src\main\resources\db\sms**
- Execute the SMS script in sms schema, here we only generate the functional activity of service. Assigning to proper role should be done according to the steps mentioned in base application.
- Here along with SMS and Entity, CMC scripts are also generated under folder. **\extension_home\obx-customer-service\src\main\resources\db\cmc**
- Execute them in the CMC schema.
- Screen Class and Data Segment has to be maintained from the UI which is present under common core.

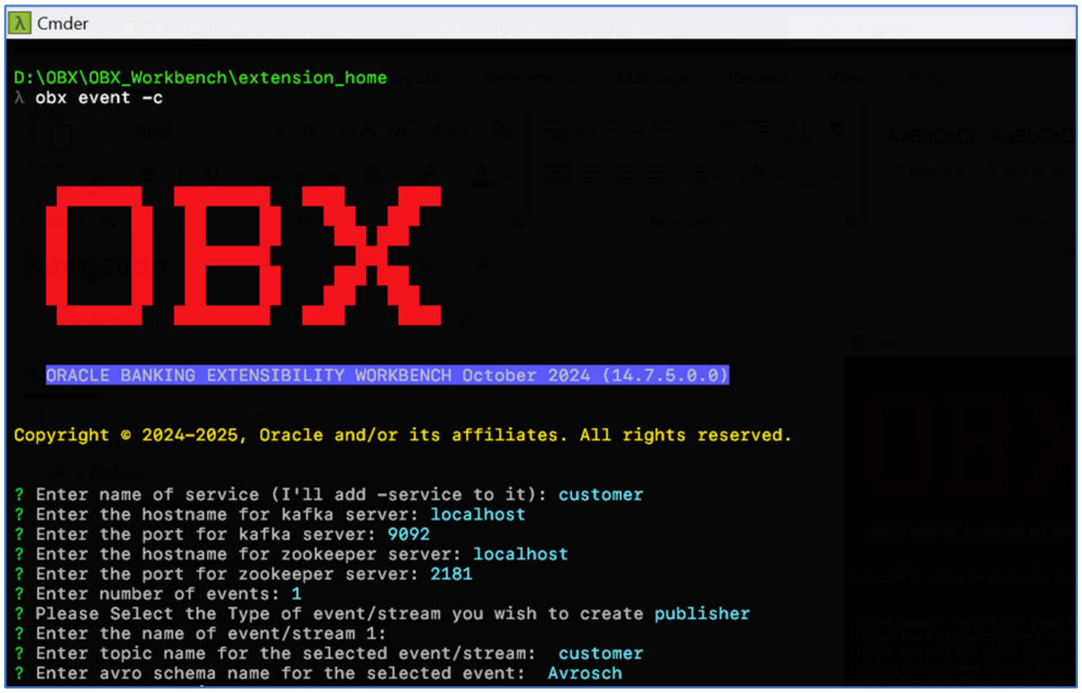
3.4 Simple Publisher/Subscriber Event Service

This topic the systematic instructions to perform the basic process to generate simple publisher/subscriber event service.

To generate simple publisher/subscriber event service, follow the below steps:

1. Navigate to same **extension_home** folder using cmder.
2. Use the command **obx event -c**.
3. Once this command is fired, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement.

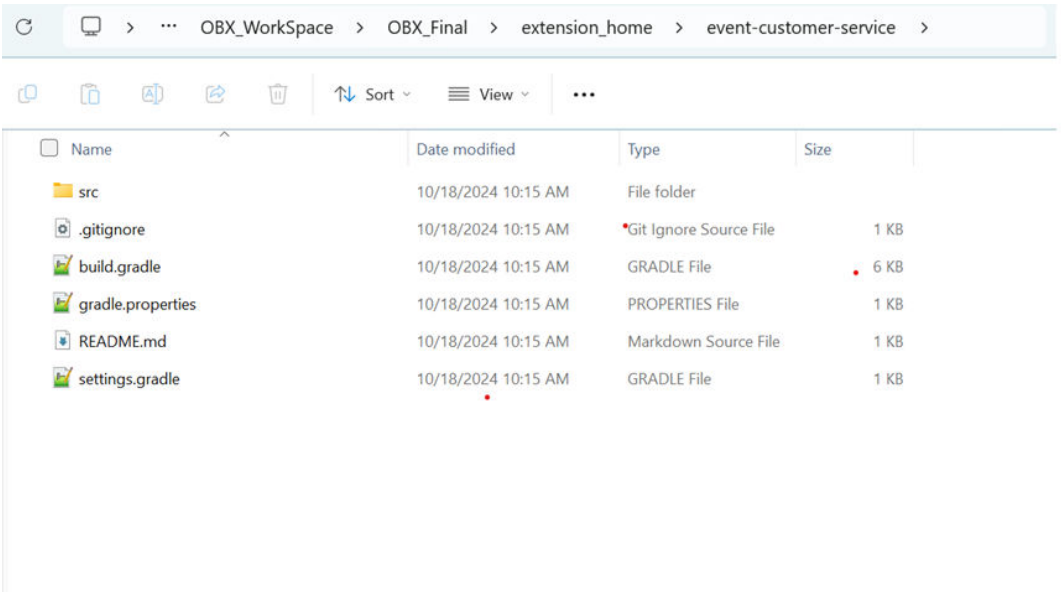
Figure 3-20   OBX event - c



```
D:\OBX\OBX_Workbench\extension_home
λ obx event -c

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Enter name of service (I'll add -service to it): customer
? Enter the hostname for kafka server: localhost
? Enter the port for kafka server: 9092
? Enter the hostname for zookeeper server: localhost
? Enter the port for zookeeper server: 2181
? Enter number of events: 1
? Please Select the Type of event/stream you wish to create publisher
? Enter the name of event/stream 1:
? Enter topic name for the selected event/stream:  customer
? Enter avro schema name for the selected event:  Avrosch
```

4. Once all the questions are answered and path of XDL is given, it will generate a folder inside the **extension_home** folder.

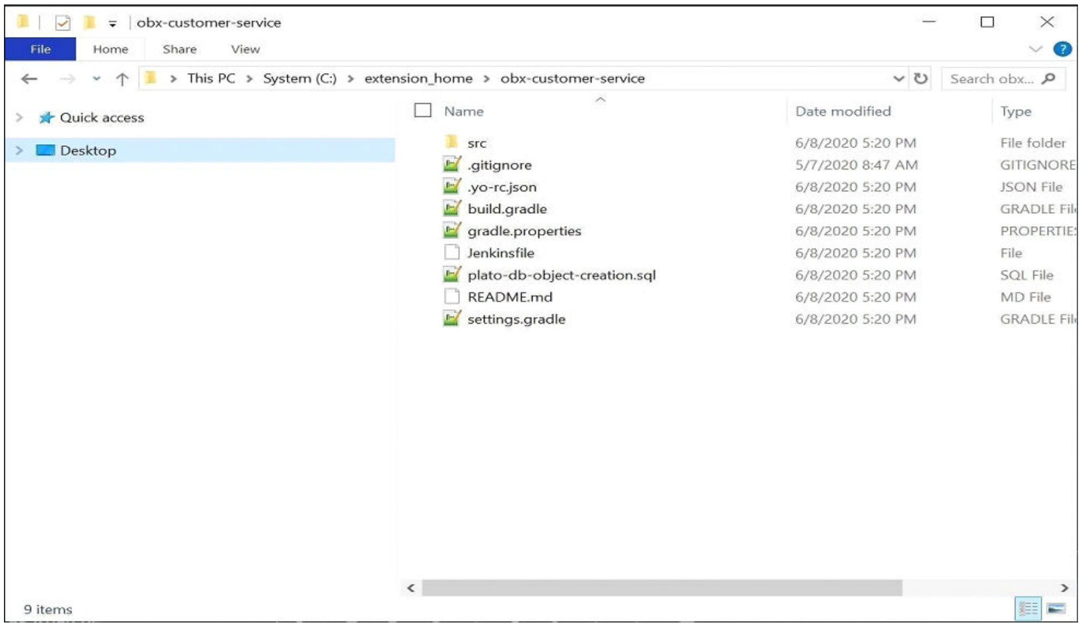
Figure 3-21   Extension Home Folder



OBX_WorkSpace > OBX_Final > extension_home > event-customer-service >
Sort
View

Name	Date modified	Type	Size
src	10/18/2024 10:15 AM	File folder	
.gitignore	10/18/2024 10:15 AM	Git Ignore Source File	1 KB
build.gradle	10/18/2024 10:15 AM	GRADLE File	6 KB
gradle.properties	10/18/2024 10:15 AM	PROPERTIES File	1 KB
README.md	10/18/2024 10:15 AM	Markdown Source File	1 KB
settings.gradle	10/18/2024 10:15 AM	GRADLE File	1 KB

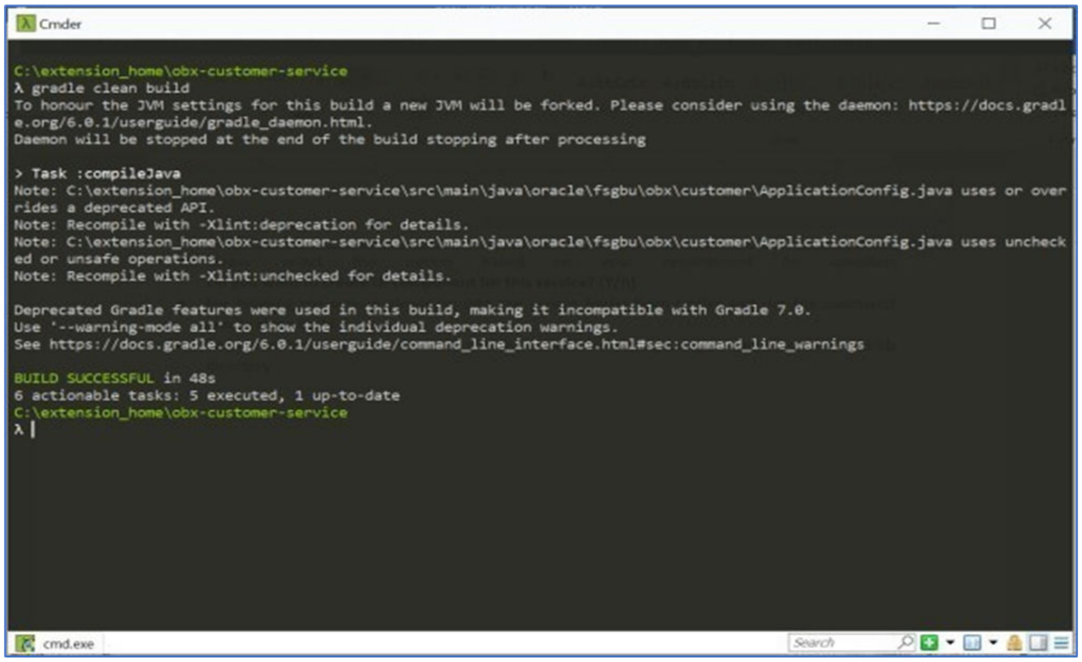
Figure 3-22 Extension Home Folder



Name	Date modified	Type
src	6/8/2020 5:20 PM	File folder
.gitignore	5/7/2020 8:47 AM	GITIGNORE
.yo-rc.json	6/8/2020 5:20 PM	JSON File
build.gradle	6/8/2020 5:20 PM	GRADLE Fil
gradle.properties	6/8/2020 5:20 PM	PROPERTIE
Jenkinsfile	6/8/2020 5:20 PM	File
plato-db-object-creation.sql	6/8/2020 5:20 PM	SQL File
README.md	6/8/2020 5:20 PM	MD File
settings.gradle	6/8/2020 5:20 PM	GRADLE Fil

5. For building the service, go into the service folder from cmder and run the command **gradle clean build**.
6. This will build the service and we can find the war of the service getting created inside the build/libs directory.

Figure 3-23 Libs Directory



```
C:\extension_home\obx-customer-service
λ gradle clean build
To honour the JVM settings for this build a new JVM will be forked. Please consider using the daemon: https://docs.gradl
e.org/6.0.1/userguide/gradle_daemon.html.
Daemon will be stopped at the end of the build stopping after processing

> Task :compileJava
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses or over
rides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses uncheck
ed or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.0.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 48s
6 actionable tasks: 5 executed, 1 up-to-date
C:\extension_home\obx-customer-service
λ |
```

7. Use this service and deploy it in your environment.

3.5 Batch Service

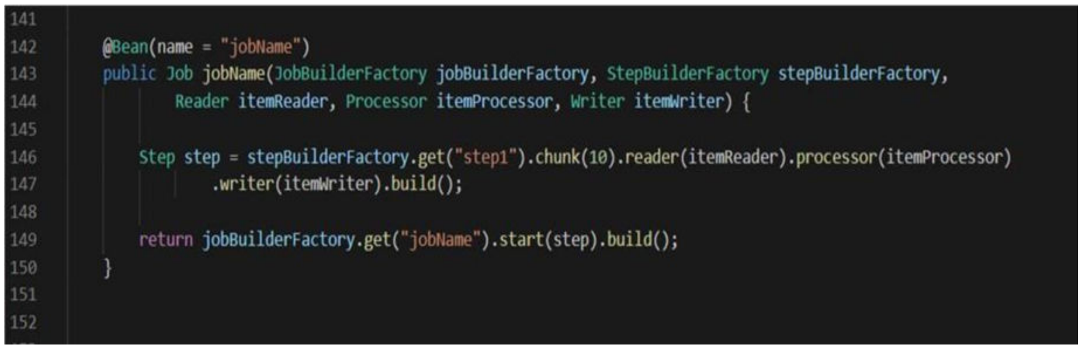
This topic describes the process to generate Oracle Banking Microservices Architecture (OBMA) based Batch service.

The purpose of this service is to create reader, writer and processor in which methods will be written according to business use case.

To generate Oracle Banking Microservices Architecture (OBMA) based Batch service, follow the below steps:

1. Navigate to same **extension_home** folder using cmder.
2. Use the command **obx batch -c**.
3. Inputs to be given after the command:
   - Select the product family.
   - Enter name of the service (I'll construct it as <productFamilyName>-batch<serviceName>- extended-services).
   - Enter product release version.
4. Upon successful creation of batch service, user will find a folder generated with <productFamilyName>-batch-<serviceName>-extended-services having the sample service code generated.
5. The generated code has two types of batch job template inside.o Simple job creation using spring batch features. The method name for this type of job creation is jobName(). The reader, writer, processor etc are taken from spring’s itemReader, itemWriter, itemProcessor.
6. Plato batch type job creation by keeping plato batch into consideration.

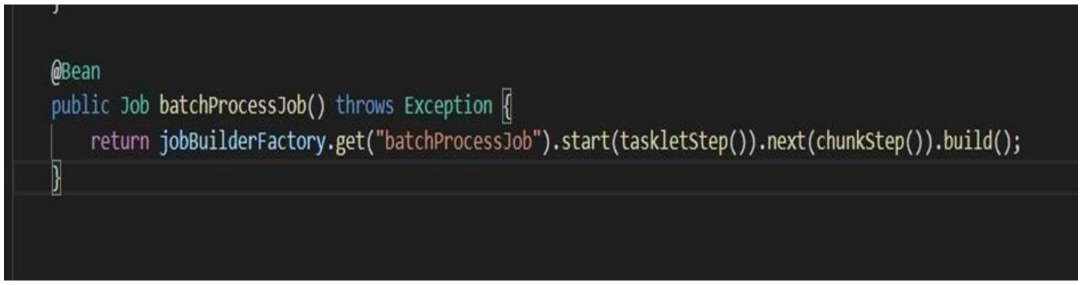
Figure 3-24 Job Name



```

    @Bean(name = "jobName")
    public Job jobName(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory,
            Reader itemReader, Processor itemProcessor, Writer itemWriter) {

        Step step = stepBuilderFactory.get("step1").chunk(10).reader(itemReader).processor(itemProcessor)
                .writer(itemWriter).build();

        return jobBuilderFactory.get("jobName").start(step).build();
    }


```

7. The method name for this type of job creation is batchProcessJob(). In this case reader is specified as EReader, writer as TWriter and processor as ETProcessor. E means the entity to be read for this job; T means the transformed object to be persisted in the database. Hence the names are given in that manner.

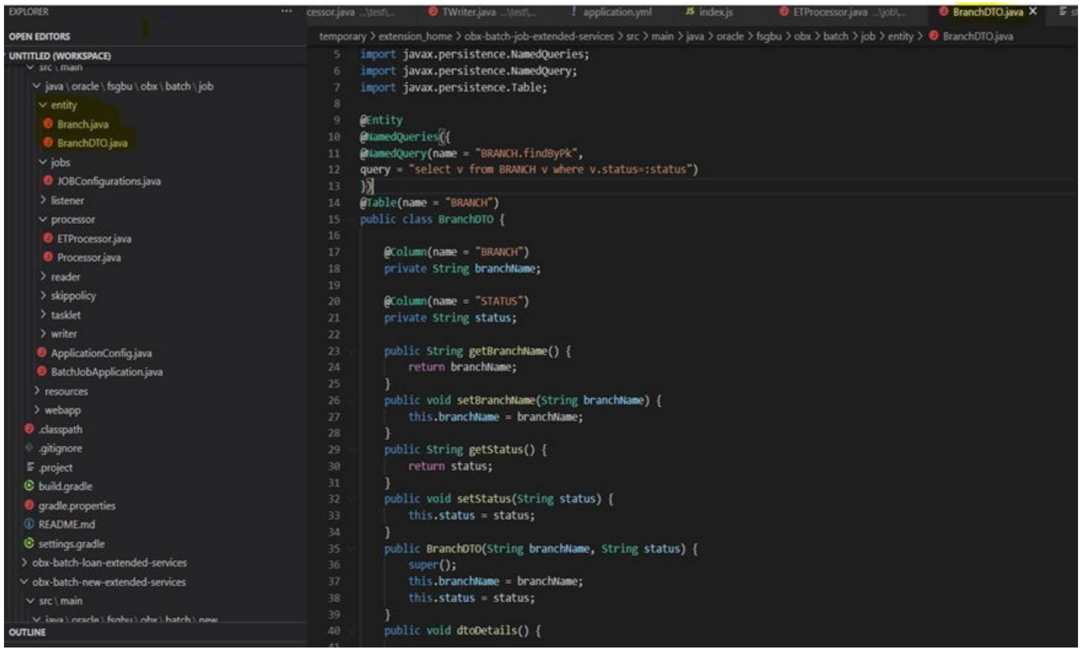
Figure 3-25 Batch Process Job



```
@Bean
public Job batchProcessJob() throws Exception {
    return jobBuilderFactory.get("batchProcessJob").start(taskletStep()).next(chunkStep()).build();
}
```

8. For plato batch type job, user needs to write his/her entity classes in which the business logic will be kept. For example, this is the structure of the entity class highlighted in the left.

Figure 3-26 Plato Batch Type



```
import javax.persistence.NamedQueries;
import javax.persistence.NamedQuery;
import javax.persistence.Table;

@Entity
@NamedQueries({
@NamedQuery(name = "BRANCH.findByPk",
query = "select v from BRANCH v where v.status=:status")
})
@Table(name = "BRANCH")
public class BranchDTO {

    @Column(name = "BRANCH")
    private String branchName;

    @Column(name = "STATUS")
    private String status;

    public String getBranchName() {
        return branchName;
    }
    public void setBranchName(String branchName) {
        this.branchName = branchName;
    }
    public String getStatus() {
        return status;
    }
    public void setStatus(String status) {
        this.status = status;
    }
    public BranchDTO(String branchName, String status) {
        super();
        this.branchName = branchName;
        this.status = status;
    }
    public void dtoDetails() {
```

9. One needs to write methods for reader, writer and processor accordingly.
10. To build the service:
    a. Navigate to the service.
    b. Fire the command gradle clean build.
    c. This will create the war file of the service in the folder structure build/libs/ productFamilyName>-batch-<serviceName>-extended-services.war.

3.6 Custom Validation Service

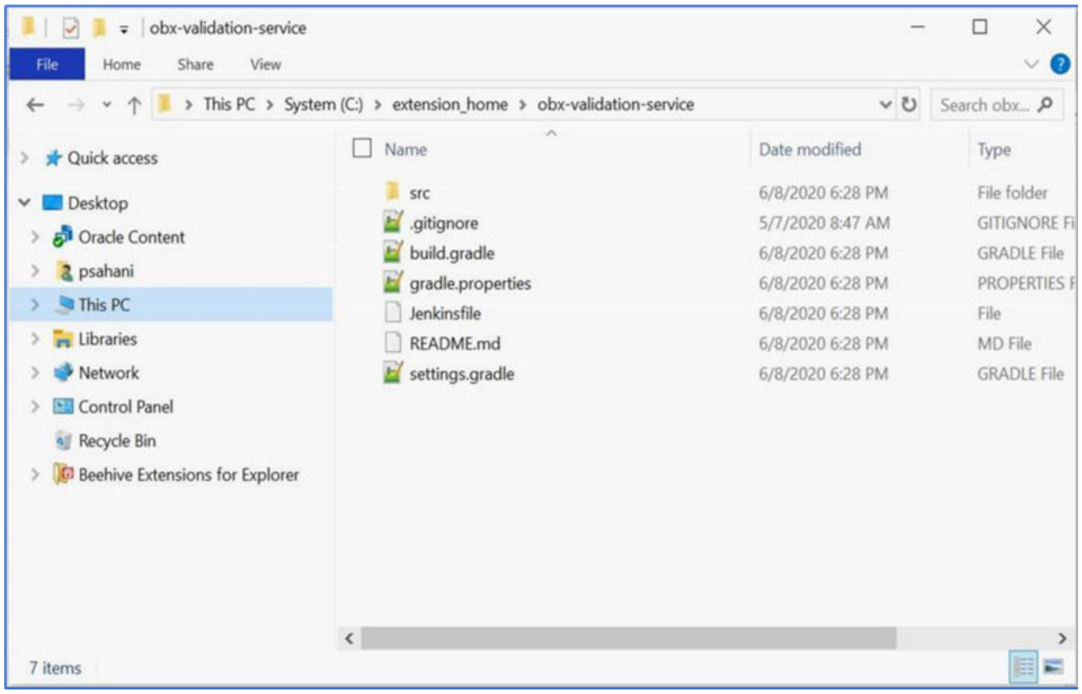
This topic provides the systematic instructions to generate custom validation service.

The purpose of this service is to perform custom validations on the base service. It is important to remember that we will be only able to perform the validation and never modify the payload to change the value.

To generate validation service, follow the below steps:

1. Navigate to same **extension_home** folder using cmder.
2. Use the command **obx validation -c**.
3. It will generate a folder inside the **extension_home** folder with obx-validation-service.

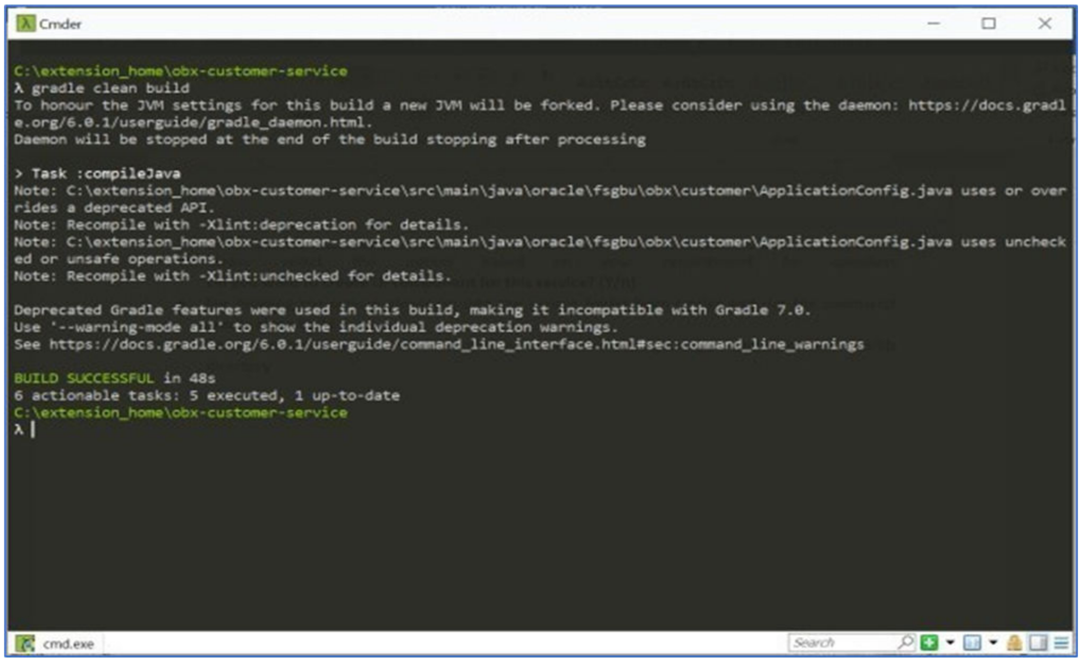
Figure 3-27 OBX validation service



Name	Date modified	Type
src	6/8/2020 6:28 PM	File folder
.gitignore	5/7/2020 8:47 AM	GITIGNORE Fi
build.gradle	6/8/2020 6:28 PM	GRADLE File
gradle.properties	6/8/2020 6:28 PM	PROPERTIES F
Jenkinsfile	6/8/2020 6:28 PM	File
README.md	6/8/2020 6:28 PM	MD File
settings.gradle	6/8/2020 6:28 PM	GRADLE File

4. For building the service, go into the service folder from cmder and run the command **gradle clean build**.
5. This will build the service and we can find the war of the service getting created inside the build/libs directory.

Figure 3-28 Libs Directory



```
C:\extension_home\obx-customer-service
λ gradle clean build
To honour the JVM settings for this build a new JVM will be forked. Please consider using the daemon: https://docs.gradl
e.org/6.0.1/userguide/gradle_daemon.html.
Daemon will be stopped at the end of the build stopping after processing

> Task :compileJava
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses or over
rides a deprecated API.
Note: Recompile with -Xlint:deprecation for details.
Note: C:\extension_home\obx-customer-service\src\main\java\oracle\fsgbu\obx\customer\ApplicationConfig.java uses uncheck
ed or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.

Deprecated Gradle features were used in this build, making it incompatible with Gradle 7.0.
Use '--warning-mode all' to show the individual deprecation warnings.
See https://docs.gradle.org/6.0.1/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 48s
6 actionable tasks: 5 executed, 1 up-to-date
C:\extension_home\obx-customer-service
λ |
```

6. Use this service and deploy it in your environment.

3.7 Steps to Adopt Multi in Existing Service

This topic provides the systematic instruction to adop multi in existing service.

Plato Micro Service Dependencies Changes

```
compile("release.obma.plato.21_0_0.services:plato-microservice-
dependencies:6.0.0")
```

Eventhub dependency changes

```
compile("release.obma.plato.21_0_0.services:plato-eventhub-
dependencies:6.0.0")
```

PlatoInterceptor Changes

```
@Bean  public MappedInterceptor gemInterceptor(PlatoInterceptor
platoInterceptor) {
LOG.info("Added interceptor for fetching the application headers"); return new
MappedInterceptor(new String[] { "/**" }, platoInterceptor);
}
```

Logging (Please include only ,%X{entityId}, change. Rest of them remain as per the old logback.xml)

```
Please include only %X{entityId} in the existing value of the LOG_PATTERN of
```

```
your logba
c k.xml

One sample format is below,

<property name="LOG_PATTERN"         value="%clr(%d{yyyy-MM- dd
HH:mm:ss.SSS}){faint} %clr(%5p [${applicationName},%X{entityId},%X{X-B3-
TraceId:},%X{X-B3-SpanId:-},%X{X-Span-Export:-}]) %clr([%mdc{env:-null}]
[%mdc{tenant:- null}]
[%mdc{user:-null}] [%mdc{branch:-null}]){faint} %clr(${PID:- }){magenta}
%clr(--
-){faint} %clr([%15.15t]){faint} %clr(%-
40.40logger{39}){cyan} %clr(:){faint} %m%n${LOG_EXCEPTION_CONVERSION_WORD:-
%wEx}" />
```

Feed Services

Folder structure should be */parentFolder/<<entityId>>/{fileName}

```
compile("release.obma.plato.21_0_0.services:plato-feed-core:6.0.0")
```

Caching Strategy

```
@Cacheable(value = "customers", key = "{ <<funtionalKeys>>
T(oracle.fsgbu.plato.core.per
sistence.provider.PlatoHolder).getCurrentEntityId() }")
```

Introduce appId in application.yml of individual micro services

If the service is a eventhub based service they should use

```
spring:
  application:
appID:
```

If the service is a non-eventhub based service they can use either

```
spring:
  application:
    appID:
```

or

```
appId: <<appId>>
```

3.8 Service Extensibility

This topic provides the systematic instructions to perform the basic operations on the selected records.

Structure of Service Extensions can be seen in below table.

Table 3-1 Service Extensibility - Field Description

Component Name	Component Description
<< micro - service - name >>- extn.jar	Extension jar
<< micro - service - name >> .war	WAR File which refers to **<< micro - service - name >> - extn .jar** during runtime.

For systematic instructions to retrieve a service extensibility record, follow the steps:

1. Add all the required classes from **<< micro - service - name >>.war** to the classpath of `<< micro - service - name >> - extn.jar` project and then build it.
   For creation of war we can use the command **obx create-jar**
   a. Go to **extension home**.
   b. Run the command **obx create-jar**.
   c. It will prompt you with the location of the extended war file. (After giving the location give enter two times).
   d. On providing the war file, it will create a jar for the same in the same location.
2. The **build.gradle** of the extension project should include the statement. **compileOnly files("classes")**.
3. For shared libraries we follow the optional packages approach. The following entries are expected in the **MANIFEST.MF** of respective war file.
   Extension-List: `<< micro - service - name >> - extn, << micro - service - name >> - extn-Extension-Name : << micro - service - name >> - extn`

   For this, we need to modify the **build.gradle** of war files to include the below statements.

```
war {
  ...
  manifest {
    attributes(
       "Extension-List": "<< micro - service - name >> -extn",
       "<< micro - service - name >>- extn -Extension -Name": "<< micro-
service- name >>-extn"
      )
       }
         ...
      }
```

4. In the extension jar create a new service class that extends the original service class and annotate the class with **@Primary** annotation to give the service class in the extension jar higher precedence.

CustomerServiceImplExt

```
@Primary
@Service
public class CustomerServiceImplExt extends CustomerServiceImpl
    implements CustomerService {.....}
```

If the extension jar is provided the methods in the extension jar will be invoked or else the methods in the original war will be invoked.

5. **Weblogic deployment**
   Deploy the extension jar first in the weblogic then in the same server deploy the war.

   Tomcat deployment

   Modification in server.xml

```
<Context ...>
  <Resources>
    <PreResources className =
"org.apache.catalina.webresources.DirResourceSet" base="<<directory
containing the extension jars "webAppMount="/WEB-INF/lib"/>
  </Resources>
</Context>
```

6. The class names inside the **<< micro - service - name >>- extn.jar**, should have the naming convention as below,
   `<<basePackageNameOf<< micro - service - name >>.war>>.<<service /controller / model>>`

4

UI Extensions – Web Component

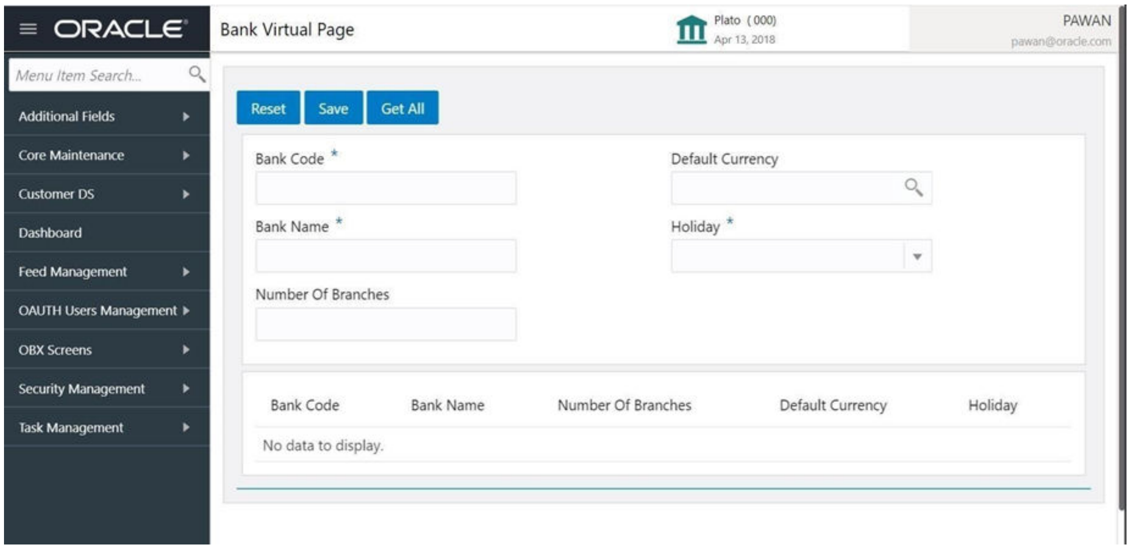
This topic describes the OBX capability to generate to different types of web components. Each Web component is capable of running itself locally.

There are various types of these web components each serving different functionality.

Standalone Component: A standalone component can be thought of as a smallest reusable UI component. They are generally the building blocks of main screens. Components like amount, text fields, lov etc. are part of standalone components.

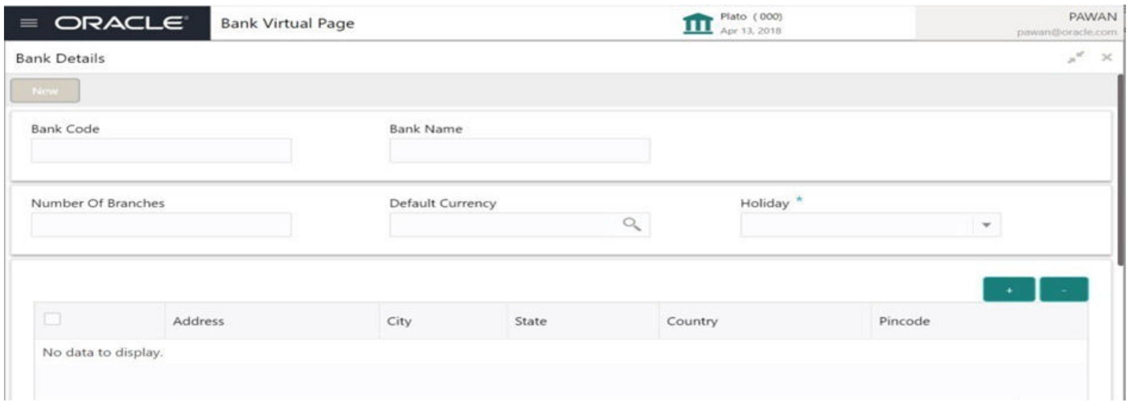
Virtual Page: A virtual page can be thought of as a screen or a web page in single page applications. They are loaded inside the content area next to the left navigation menu. Important point to remember when designing virtual page is, it appends and changes the router (app URL) when navigation is done.

Figure 4-1 Virtual Page



Container Component: These Components are a special type of components which are loaded inside a container called as Wizard. It gives functionality like minimizing the component and open multiple screens simultaneously on the screen. Important point to remove here is that these components never change to router state, so bookmarking is not possible for these screens.

Figure 4-2   Bank Details



Bank Code

Bank Name

Number Of Branches

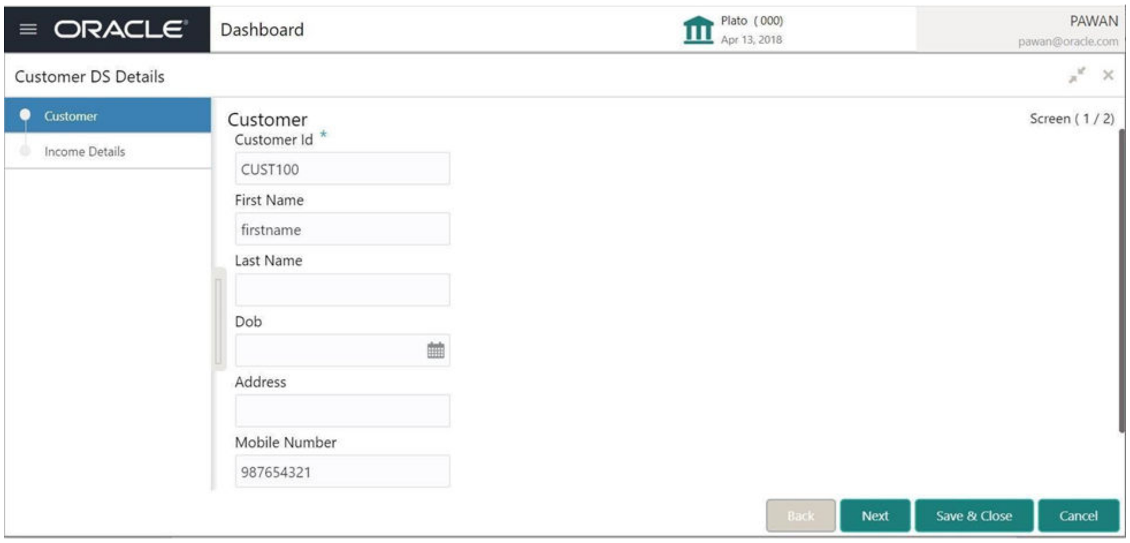
Default Currency

Holiday *

	Address	City	State	Country	Pincode
No data to display.					

Data/Resource Segment: A component designed using data segment approach are similar to that of virtual page but are always part of flow or process and loaded like container components. It is helpful in use cases where data to be captured is huge or is captured in various stages of applications.

Figure 4-3   Customer Dashboard



Customer

Customer Id *: CUST100

First Name: firstname

Last Name:

Dob:

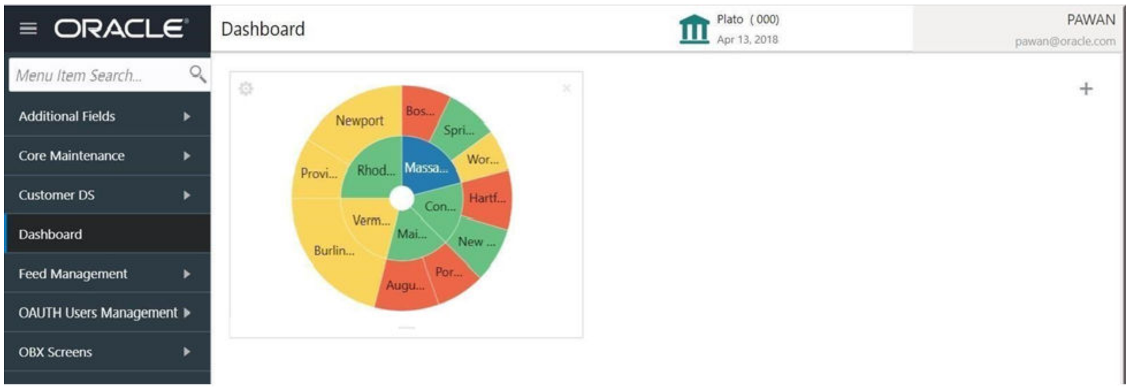
Address:

Mobile Number: 987654321

In above screenshot Customer and Income Details on left are two data segments which is part of Customer DS Details Application.

Widgets: Widgets are special components meant for dashboard. These are generally created in the form of tiles and are attached to the dashboard.

Figure 4-4   Dashboard



Note:

- All the above components except standalone components have SMS applied on it.
- We have to assign functional activity of web components to the role and then only they are integrated with the main application shell.
- Also, it always recommended to try and run the component locally before merging them into main application.
- All web components come bundled with testing framework including unit test cases and functional test. Therefore, it’s a good practice to write them along with the development.

- Component Server
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- Simple Standalone
  This topic describes the process of creating the simple standalone component using OBX.
- Virtual Page
  This topic describes the process of creating the virtual page component using OBX.
- Maintenance Detail and Summary
  This topic describes the process of creating the Maintenance Detail and Summary component using OBX.
- Data Segment
  This topic describes the process of creating the virtual page component using OBX.
- Dashboard Widget
  This topic describes the process of creating the simple standalone component using OBX.
- Running Component after Generation
  This topic describes the steps you need to follow to re-run the component created or generated earlier.
- Creating final Extended Component war for Deployment
  This topic describes the steps to generate Extended Component war for Deployment.

- Understanding DB Scripts for Web Components
  This topic describes the significance of DB folder generate inside the web component folder.

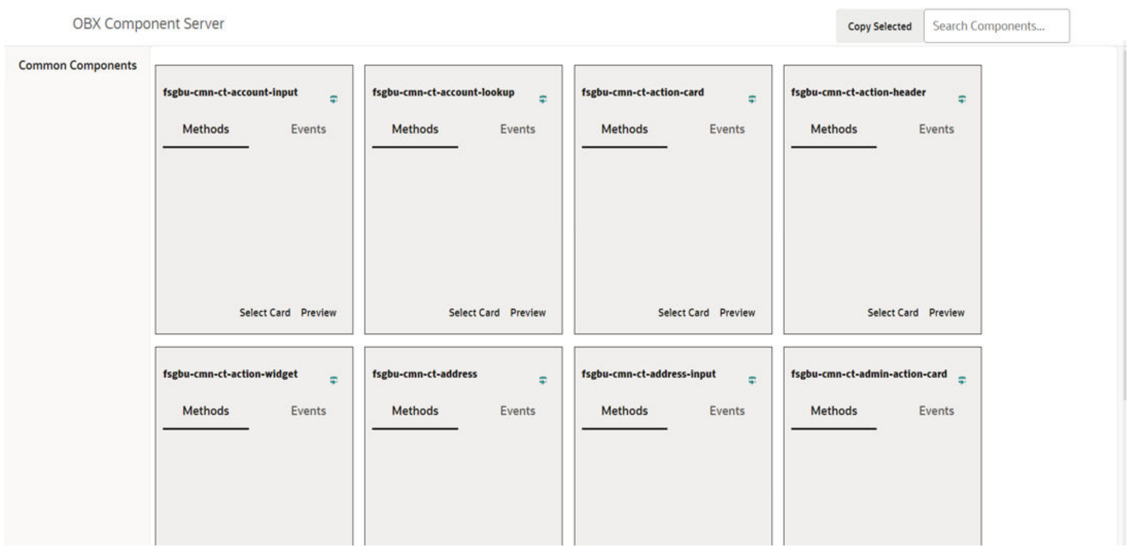
4.1 Component Server

This topic provides the systematic instructions to perform the basic operations on the selected records.

It is one of highlight feature from OBX. A component server is hub of components which are available from the base/kernel application. As each component is developed individually and reusable, we can use this functionality to reuse even the components from base application. It saves time as we don’t have to code same thing again and again. We can reuse as many components as possible from base application into extensions.

Component server is started automatically when you generate the web component. It runs on http://localhost:8002. One can simply go to browser and copy components and put them in a **metadata.js** file which is created inside the component and by doing so it indicated OBX that we have to reuse the component and it generates the code automatically.

Figure 4-5 Component Server


OBX Component Server
Copy Selected
Search Components...
Common Components
fsgbu-cmn-ct-account-input
Methods
Events
Select Card
Preview
fsgbu-cmn-ct-account-lookup
fsgbu-cmn-ct-action-card
fsgbu-cmn-ct-action-header
fsgbu-cmn-ct-action-widget
fsgbu-cmn-ct-address
fsgbu-cmn-ct-address-input
fsgbu-cmn-ct-admin-action-card

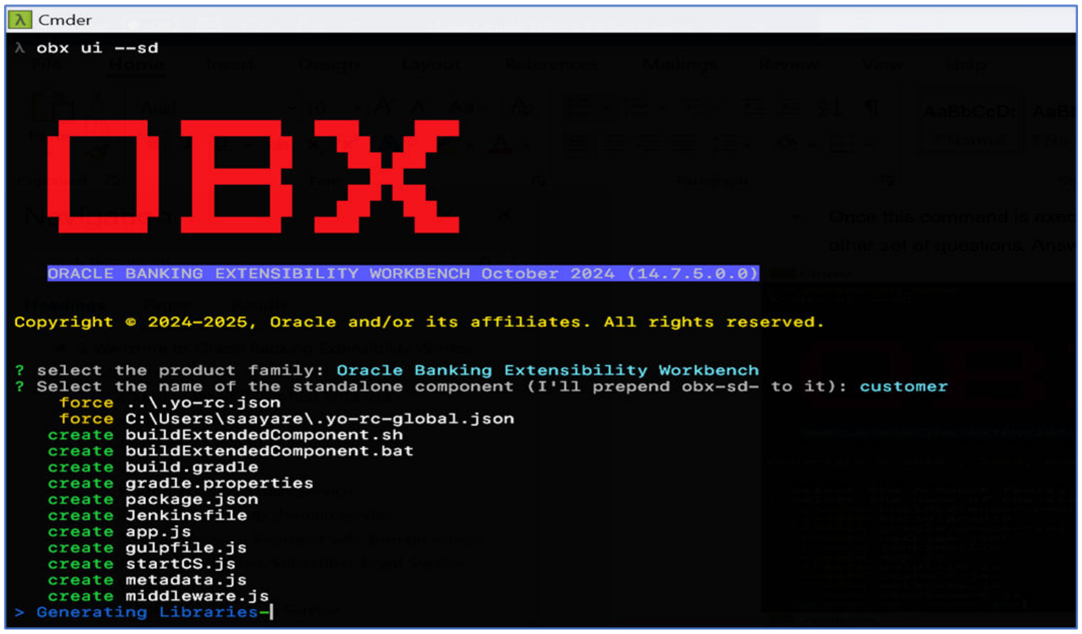
4.2 Simple Standalone

This topic describes the process of creating the simple standalone component using OBX.

Following are the steps needed to be followed:

1. Navigate to **extension_home** folder from cmder.
2. Use the command: **obx ui –sd**.
3. Once this command is executed, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement.

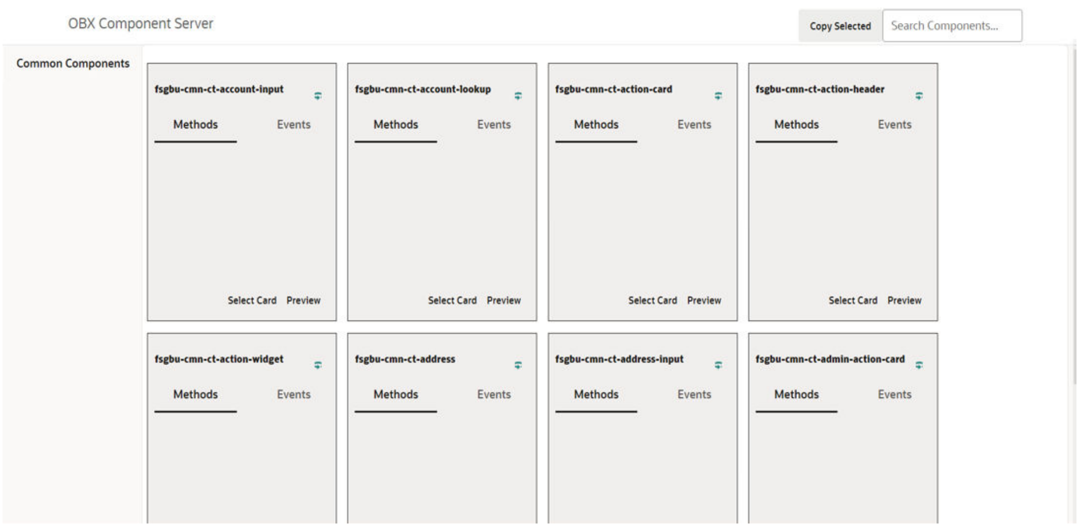
Figure 4-6 OBX UI-sd



```
λ obx ui --sd

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? select the product family: Oracle Banking Extensibility Workbench
? Select the name of the standalone component (I'll prepend obx-sd- to it): customer
   force ..\.yo-rc.json
   force C:\Users\saayare\.yo-rc-global.json
  create buildExtendedComponent.sh
  create buildExtendedComponent.bat
  create build.gradle
  create gradle.properties
  create package.json
  create Jenkinsfile
  create app.js
  create gulpfile.js
  create startCS.js
  create metadata.js
  create middleware.js
> Generating Libraries-|
```

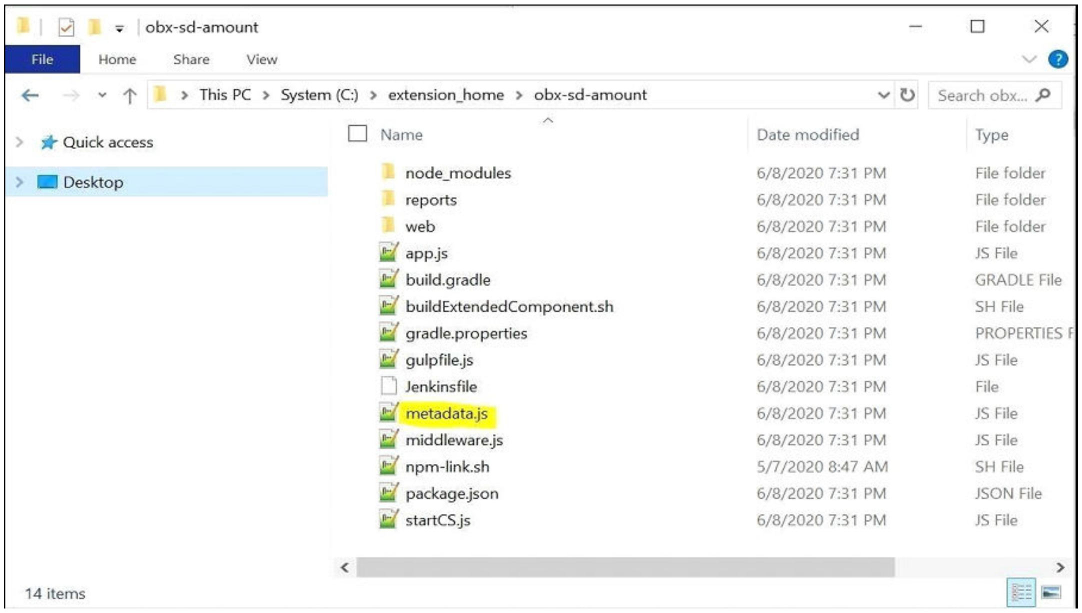
4. It will automatically generate the libraries for the component to run locally and you will be also able to see new cmder tab opened where component server is running.
5. At this point of time go to browser and navigate to http://localhost:8002. You will be able to see component server home page like:

Figure 4-7 OBX Component Server


OBX Component Server
Copy Selected
Search Components...
Common Components
fsgbu-cmn-ct-account-input
Methods
Events
Select Card
Preview
fsgbu-cmn-ct-account-lookup
fsgbu-cmn-ct-action-card
fsgbu-cmn-ct-action-header
fsgbu-cmn-ct-action-widget
fsgbu-cmn-ct-address
fsgbu-cmn-ct-address-input
fsgbu-cmn-ct-admin-action-card

6. Select the component which you want to reuse in your extension and paste it in **module.exports = []** inside the **metadata.js** file.

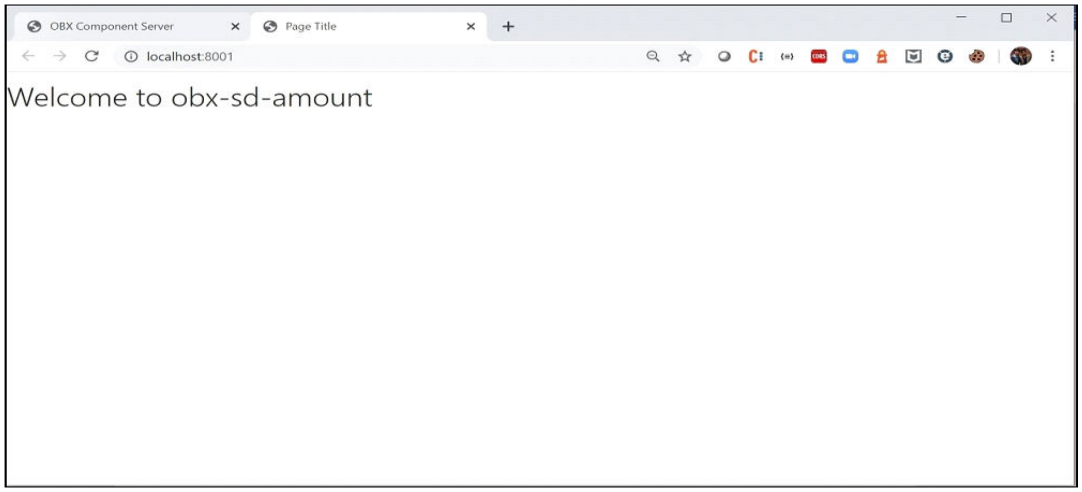
Figure 4-8 obx-sd-amount



Name	Date modified	Type
node_modules	6/8/2020 7:31 PM	File folder
reports	6/8/2020 7:31 PM	File folder
web	6/8/2020 7:31 PM	File folder
app.js	6/8/2020 7:31 PM	JS File
build.gradle	6/8/2020 7:31 PM	GRADLE File
buildExtendedComponent.sh	6/8/2020 7:31 PM	SH File
gradle.properties	6/8/2020 7:31 PM	PROPERTIES F
gulpfile.js	6/8/2020 7:31 PM	JS File
Jenkinsfile	6/8/2020 7:31 PM	File
metadata.js	6/8/2020 7:31 PM	JS File
middleware.js	6/8/2020 7:31 PM	JS File
npm-link.sh	5/7/2020 8:47 AM	SH File
package.json	6/8/2020 7:31 PM	JSON File
startCS.js	6/8/2020 7:31 PM	JS File

7. Once done come back to main tab in cmder where is waiting with question, **Please modify the Metadata.js file before proceeding. Once done press y to proceed?**
8. On completing the above process, it will automatically generate the source folder now and open a new tab on cmder where component will be running.
   Along with cmder tab it will automatically open a tab on default browser as well with component rendered on the screen.

Figure 4-9 Source folder



Welcome to obx-sd-amount

4.3 Virtual Page

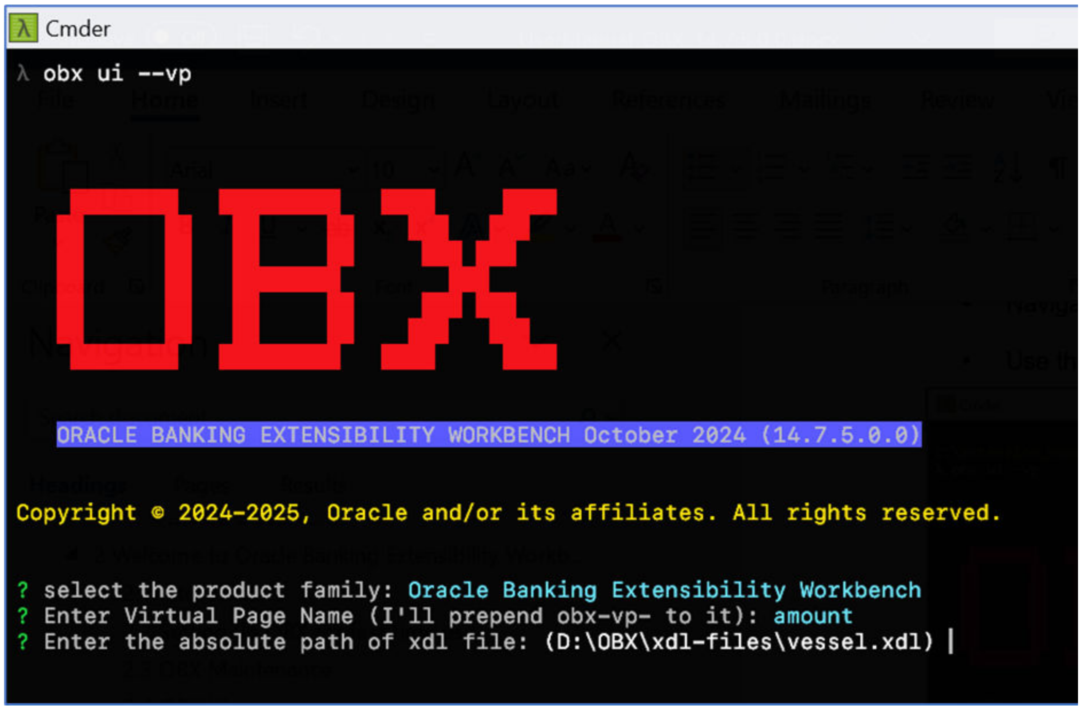
This topic describes the process of creating the virtual page component using OBX.

Following are the steps needed to be followed:

1. Navigate to **extension_home** folder from cmder.

2. Use the command **obx ui –vp.**

Figure 4-10 obx ui-vp

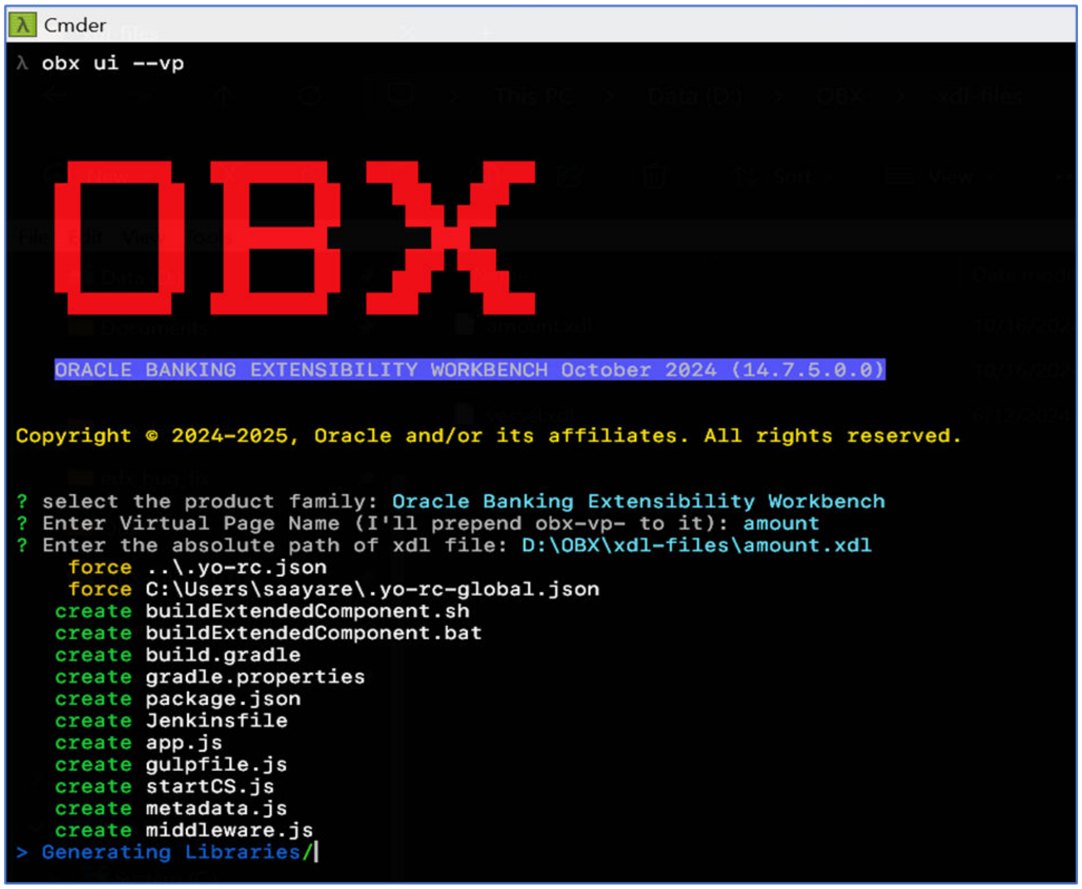


Cmder

```
λ obx ui --vp

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? select the product family: Oracle Banking Extensibility Workbench
? Enter Virtual Page Name (I'll prepend obx-vp- to it): amount
? Enter the absolute path of xdl file: (D:\OBX\xdl-files\vessel.xdl) |
```

3. Once this command is executed, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement.

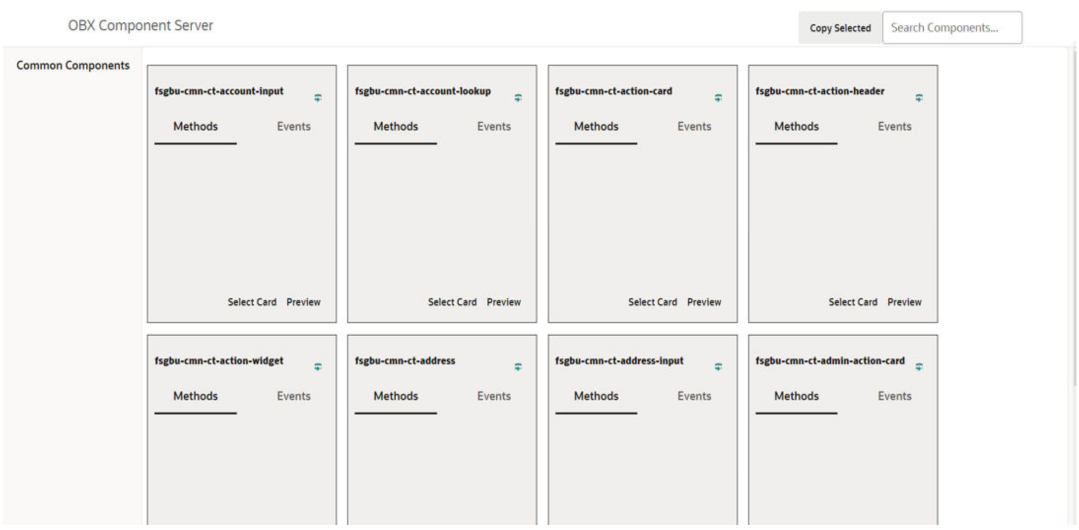
Figure 4-11 obx ui-vp



```
Cmder
λ obx ui --vp

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? select the product family: Oracle Banking Extensibility Workbench
? Enter Virtual Page Name (I'll prepend obx-vp- to it): amount
? Enter the absolute path of xdl file: D:\OBX\xdl-files\amount.xdl
    force ..\.yo-rc.json
    force C:\Users\saayare\.yo-rc-global.json
   create buildExtendedComponent.sh
   create buildExtendedComponent.bat
   create build.gradle
   create gradle.properties
   create package.json
   create Jenkinsfile
   create app.js
   create gulpfile.js
   create startCS.js
   create metadata.js
   create middleware.js
> Generating Libraries/|
```

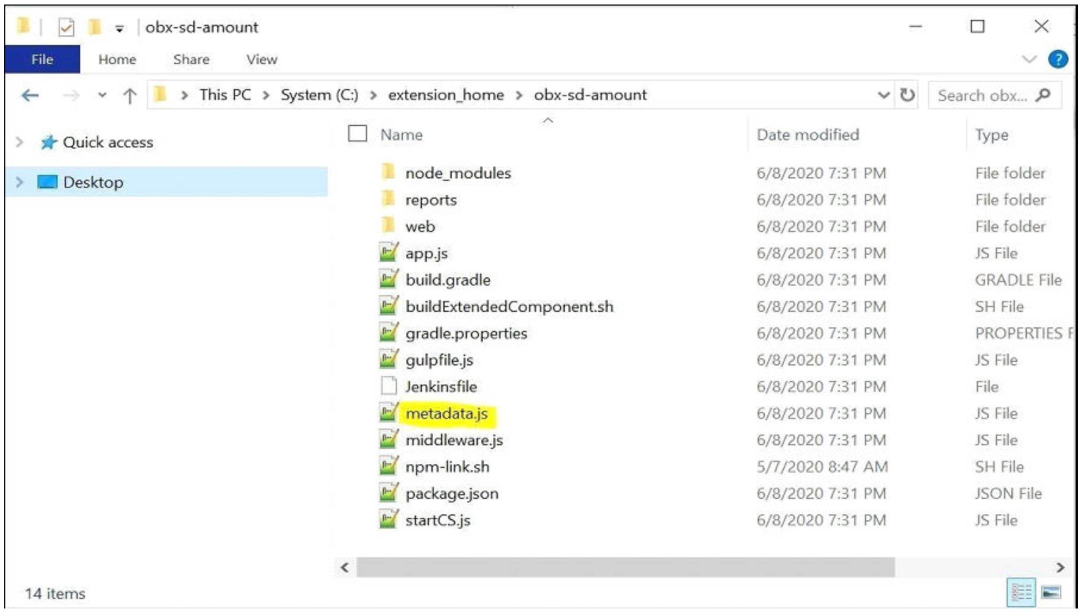
4. It will automatically generate the libraries for the component to run locally and you will be also able to see new cmder tab opened where component server is running.
5. At this point of time go to browser and navigate to http://localhost:8002. You will be able to component server home page like:

Figure 4-12 OBX Component Server



6. Select the component which you want to reuse in your extension and paste it in **module.exports = []**; inside the **metadata.js** file.

Figure 4-13   obx-sd-amount

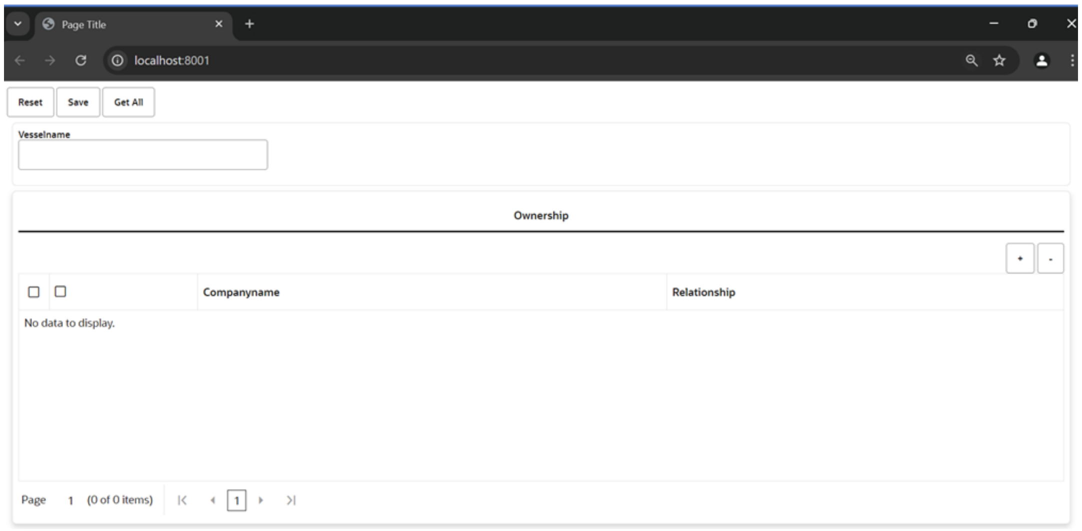


Name	Date modified	Type
node_modules	6/8/2020 7:31 PM	File folder
reports	6/8/2020 7:31 PM	File folder
web	6/8/2020 7:31 PM	File folder
app.js	6/8/2020 7:31 PM	JS File
build.gradle	6/8/2020 7:31 PM	GRADLE File
buildExtendedComponent.sh	6/8/2020 7:31 PM	SH File
gradle.properties	6/8/2020 7:31 PM	PROPERTIES F
gulpfile.js	6/8/2020 7:31 PM	JS File
Jenkinsfile	6/8/2020 7:31 PM	File
metadata.js	6/8/2020 7:31 PM	JS File
middleware.js	6/8/2020 7:31 PM	JS File
npm-link.sh	5/7/2020 8:47 AM	SH File
package.json	6/8/2020 7:31 PM	JSON File
startCS.js	6/8/2020 7:31 PM	JS File

14 items

7. Once done come back to main tab in cmder where is waiting with question: **Please modify the Metadata.js file before proceeding. Once done press y to proceed?**
8. On completing the above process, it will automatically generate the source folder now and open a new tab on cmder where component will be running.
9. Along with cmder tab it will automatically open a tab on default browser as well with component rendered on the screen.

Figure 4-14   Component



4.4 Maintenance Detail and Summary

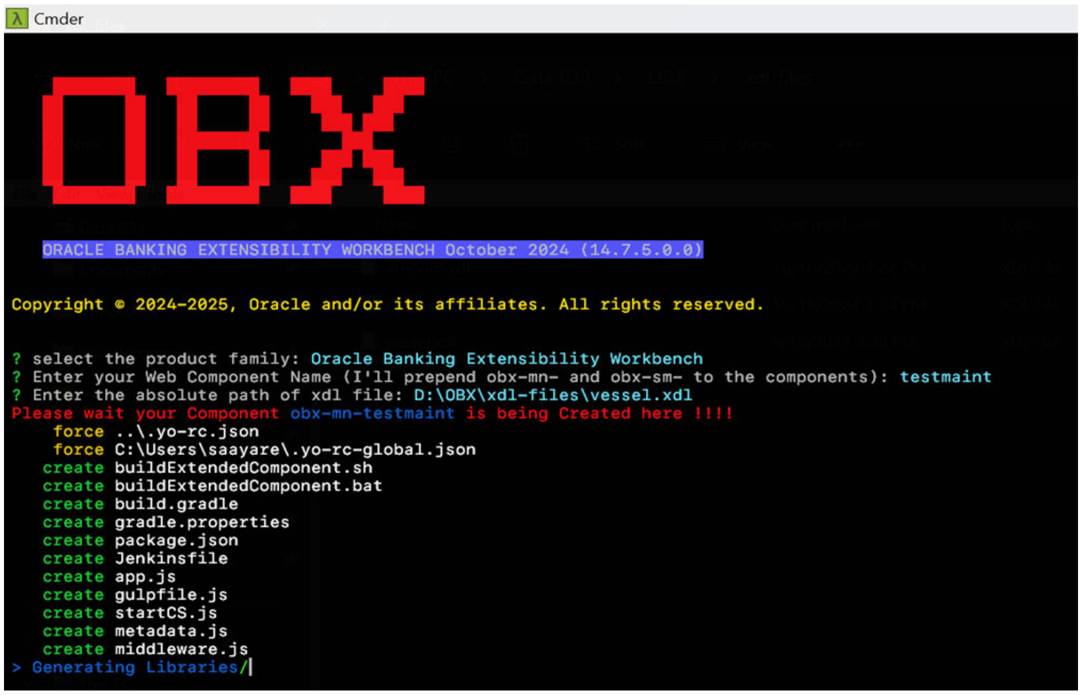
This topic describes the process of creating the Maintenance Detail and Summary component using OBX.

Here we must remember that we will be generating two web components one will be detail component and another one for summary component.

Following are the steps needed to be followed:

1. Navigate to **extension_home** folder from cmder.
2. Use the command **obx ui –mnsm**.
3. Once this command is executed, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement.

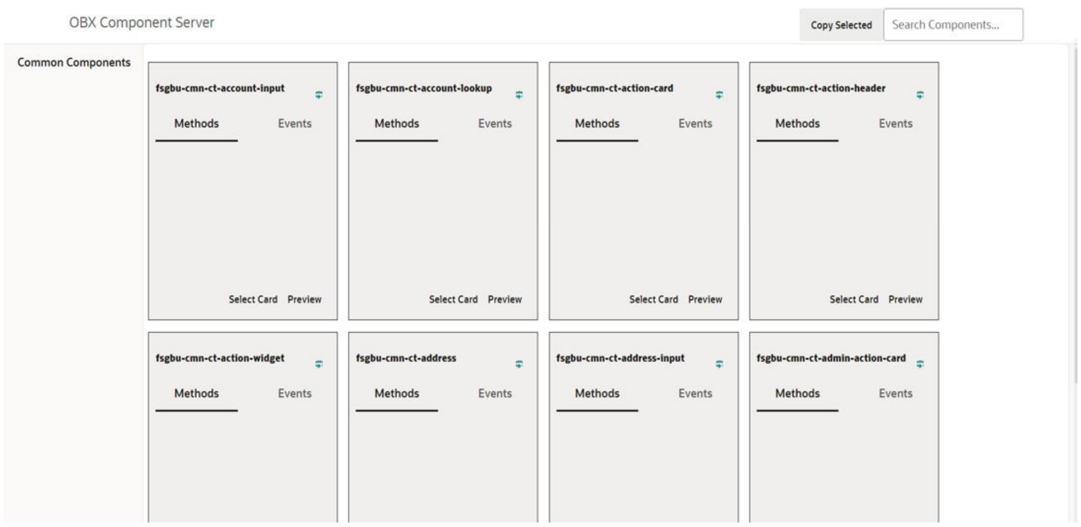
Figure 4-15 OBX UI



```
Cmder

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? select the product family: Oracle Banking Extensibility Workbench
? Enter your Web Component Name (I'll prepend obx-mn- and obx-sm- to the components): testmaint
? Enter the absolute path of xdl file: D:\OBX\xdl-files\vessel.xdl
Please wait your Component obx-mn-testmaint is being Created here !!!!
    force ..\.yo-rc.json
    force C:\Users\saayare\.yo-rc-global.json
   create buildExtendedComponent.sh
   create buildExtendedComponent.bat
   create build.gradle
   create gradle.properties
   create package.json
   create Jenkinsfile
   create app.js
   create gulpfile.js
   create startCS.js
   create metadata.js
   create middleware.js
> Generating Libraries/|
```

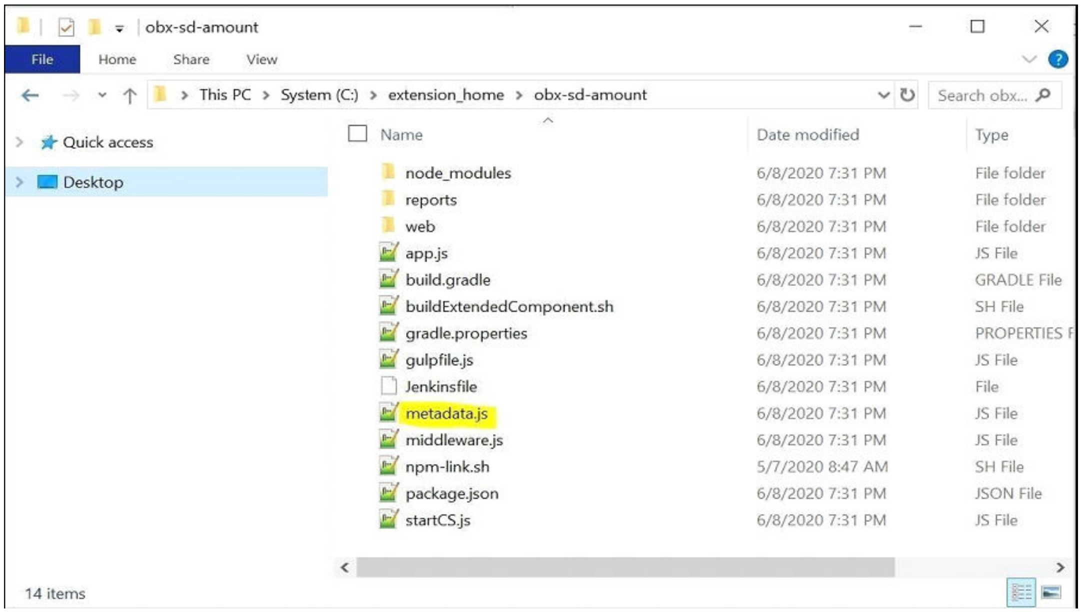
4. It will automatically generate the libraries for the components.
5. At this point of time go to browser and navigate to http://localhost:8002. You will be able to component server home page like:

Figure 4-16 OBX Component Server


OBX Component Server
Copy Selected
Search Components...
Common Components
fsgbu-cmn-ct-account-input
Methods
Events
Select Card
Preview
fsgbu-cmn-ct-account-lookup
fsgbu-cmn-ct-action-card
fsgbu-cmn-ct-action-header
fsgbu-cmn-ct-action-widget
fsgbu-cmn-ct-address
fsgbu-cmn-ct-address-input
fsgbu-cmn-ct-admin-action-card

6. Select the component which you want to reuse in your extension and paste it in **module.exports = []**; inside the **metadata.js** file.

Figure 4-17 OBX sd amount



Name	Date modified	Type
node_modules	6/8/2020 7:31 PM	File folder
reports	6/8/2020 7:31 PM	File folder
web	6/8/2020 7:31 PM	File folder
app.js	6/8/2020 7:31 PM	JS File
build.gradle	6/8/2020 7:31 PM	GRADLE File
buildExtendedComponent.sh	6/8/2020 7:31 PM	SH File
gradle.properties	6/8/2020 7:31 PM	PROPERTIES F
gulpfile.js	6/8/2020 7:31 PM	JS File
Jenkinsfile	6/8/2020 7:31 PM	File
metadata.js	6/8/2020 7:31 PM	JS File
middleware.js	6/8/2020 7:31 PM	JS File
npm-link.sh	5/7/2020 8:47 AM	SH File
package.json	6/8/2020 7:31 PM	JSON File
startCS.js	6/8/2020 7:31 PM	JS File

7. Once done come back to main tab in cmder where is waiting with question **Please modify the Metadata.js file before proceeding. Once done press y to proceed?**
8. On completing the above process, it will automatically generate the source folder for maintenance details screen and same process will followed for summary screen as well.
9. For this case we will be not able to see the component running locally as we have to 2 components generated.
10. To start the component, one needs to go inside the component are run it manually.

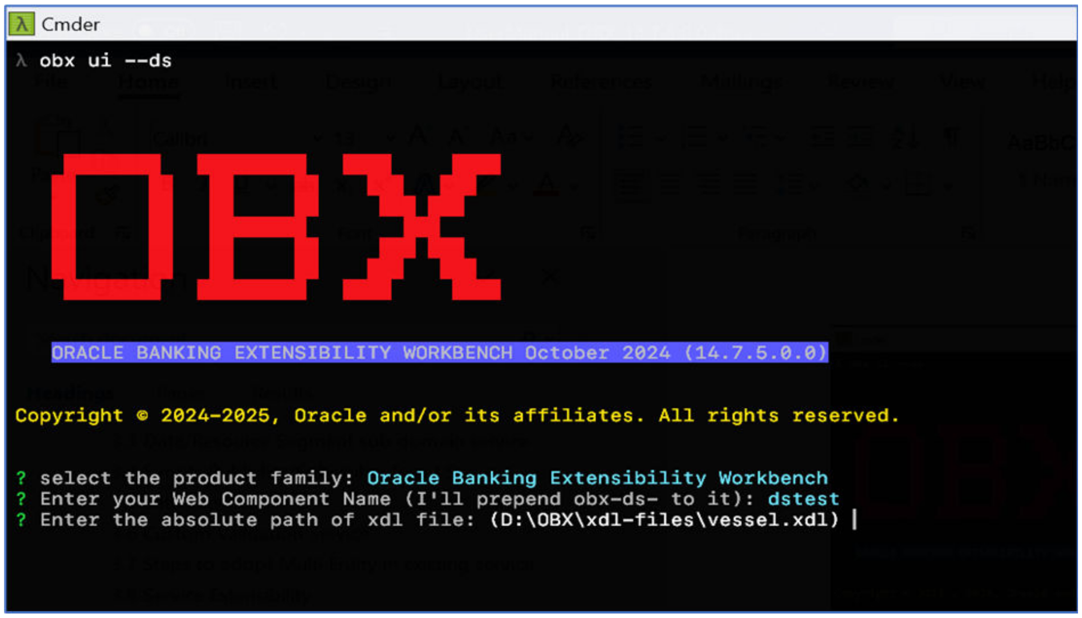
4.5 Data Segment

This topic describes the process of creating the virtual page component using OBX.

Following are the steps needed to be followed:

1. Navigate to **extension_home** folder from cmder.
2. Use the command **obx ui –ds**.
3. Once this command is executed, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement.

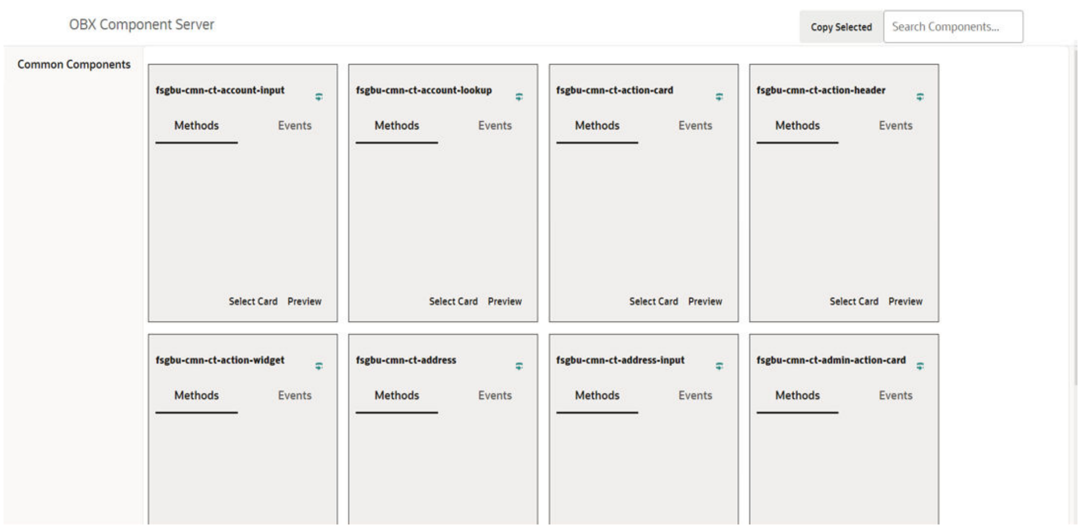
Figure 4-18 obx ui-ds



```
λ obx ui --ds

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? select the product family: Oracle Banking Extensibility Workbench
? Enter your Web Component Name (I'll prepend obx-ds- to it): dstest
? Enter the absolute path of xdl file: (D:\OBX\xdl-files\vessel.xdl) |
```

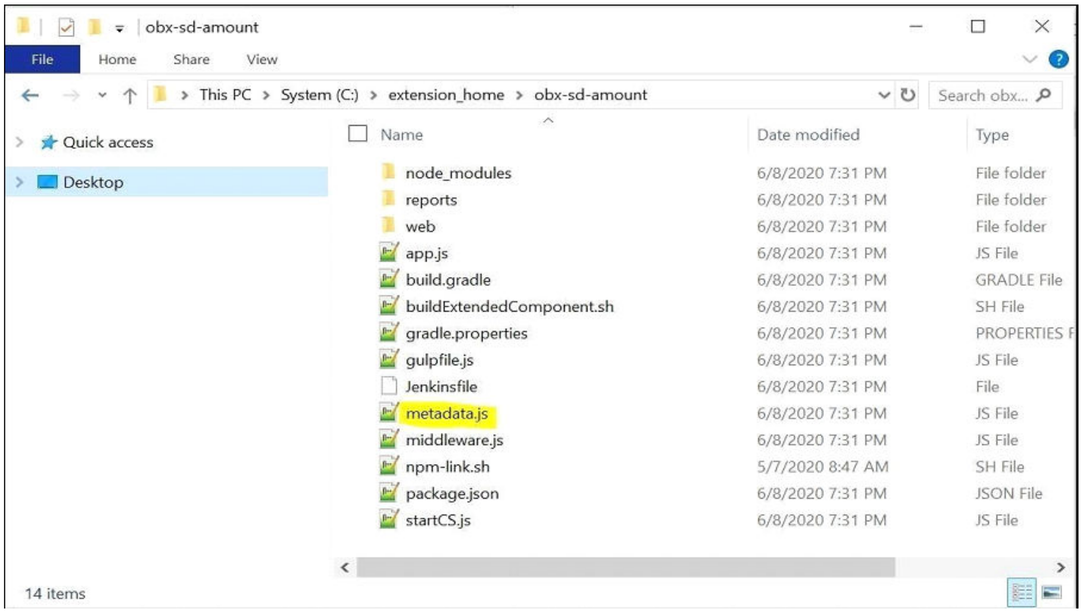
4. It will automatically generate the libraries for the component to run locally and you will be also able to see new cmder tab opened where component server is running.
5. At this point of time go to browser and navigate to http://localhost:8002. You will be able to component server home page like:

Figure 4-19 OBX Component Server


OBX Component Server
Copy Selected
Search Components...
Common Components
fsgbu-cmn-ct-account-input
fsgbu-cmn-ct-account-lookup
fsgbu-cmn-ct-action-card
fsgbu-cmn-ct-action-header
fsgbu-cmn-ct-action-widget
fsgbu-cmn-ct-address
fsgbu-cmn-ct-address-input
fsgbu-cmn-ct-admin-action-card
Methods
Events
Select Card
Preview

6. Select the component which you want to reuse in your extension and paste it in **module.exports = []**; inside the **metadata.js** file.

Figure 4-20 OBX sd amount extension home folder



Name	Date modified	Type
node_modules	6/8/2020 7:31 PM	File folder
reports	6/8/2020 7:31 PM	File folder
web	6/8/2020 7:31 PM	File folder
app.js	6/8/2020 7:31 PM	JS File
build.gradle	6/8/2020 7:31 PM	GRADLE File
buildExtendedComponent.sh	6/8/2020 7:31 PM	SH File
gradle.properties	6/8/2020 7:31 PM	PROPERTIES F
gulpfile.js	6/8/2020 7:31 PM	JS File
Jenkinsfile	6/8/2020 7:31 PM	File
metadata.js	6/8/2020 7:31 PM	JS File
middleware.js	6/8/2020 7:31 PM	JS File
npm-link.sh	5/7/2020 8:47 AM	SH File
package.json	6/8/2020 7:31 PM	JSON File
startCS.js	6/8/2020 7:31 PM	JS File

7. Once done come back to main tab in cmder where is waiting with question **Please modify the Metadata.js file before proceeding. Once done press 'y' to proceed?**
8. On completing the above process, it will automatically generate the source folder now and open a new tab on cmder where component will be running.
9. Along with cmder tab it will automatically open a tab on default browser as well with component rendered on the screen.

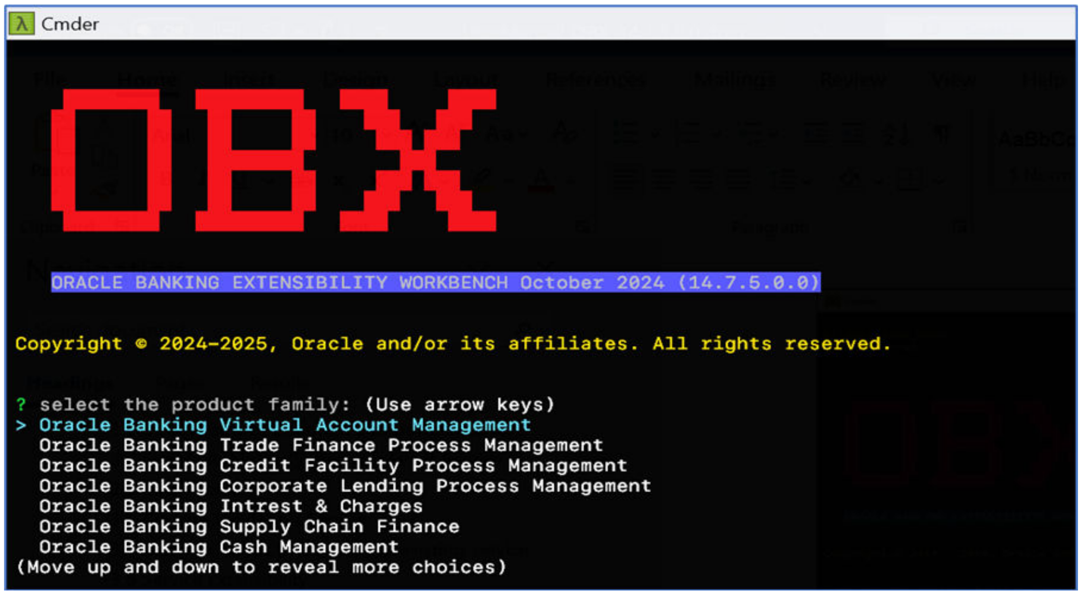
4.6 Dashboard Widget

This topic describes the process of creating the simple standalone component using OBX.

Following are the steps needed to be followed:

1. Navigate to **extension_home** folder from cmder.
2. Use the command **obx ui --wd**.

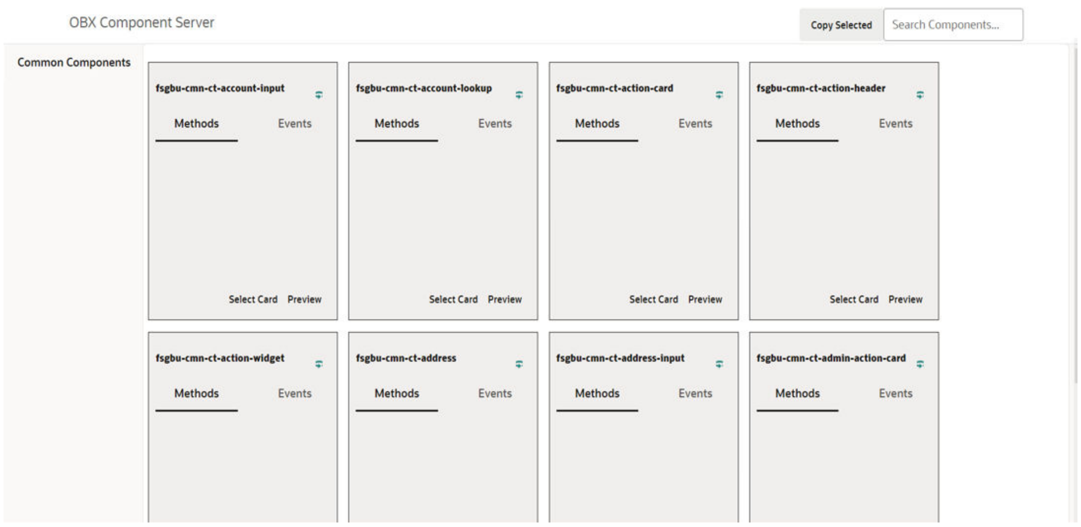
Figure 4-21 OBX UI



```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? select the product family: (Use arrow keys)
> Oracle Banking Virtual Account Management
  Oracle Banking Trade Finance Process Management
  Oracle Banking Credit Facility Process Management
  Oracle Banking Corporate Lending Process Management
  Oracle Banking Intrest & Charges
  Oracle Banking Supply Chain Finance
  Oracle Banking Cash Management
(Move up and down to reveal more choices)
```

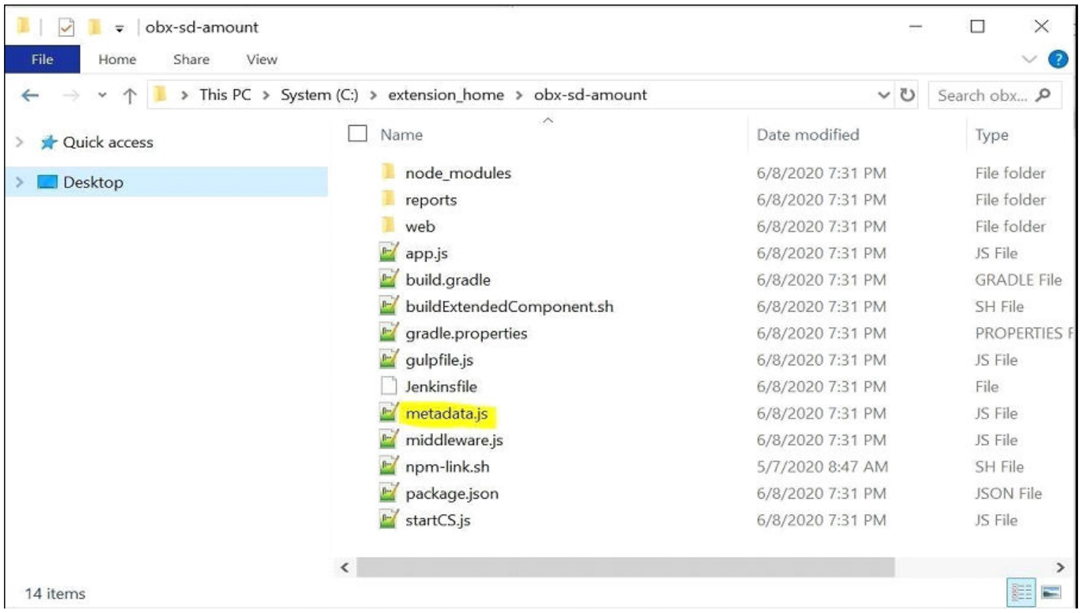
3. Once this command is executed, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement.
4. It will automatically generate the libraries for the component to run locally and you will be also able to see new cmder tab opened where component server is running.
5. At this point of time go to browser and navigate to http://localhost:8002. You will be able to see component server home page like:

Figure 4-22   OBX Component Server


OBX Component Server
Copy Selected
Search Components...
Common Components
fsgbu-cmn-ct-account-input
fsgbu-cmn-ct-account-lookup
fsgbu-cmn-ct-action-card
fsgbu-cmn-ct-action-header
fsgbu-cmn-ct-action-widget
fsgbu-cmn-ct-address
fsgbu-cmn-ct-address-input
fsgbu-cmn-ct-admin-action-card
Methods
Events
Select Card
Preview

6. Select the component which you want to reuse in your extension and paste it in **module.exports = []**; inside the **metadata.js** file.

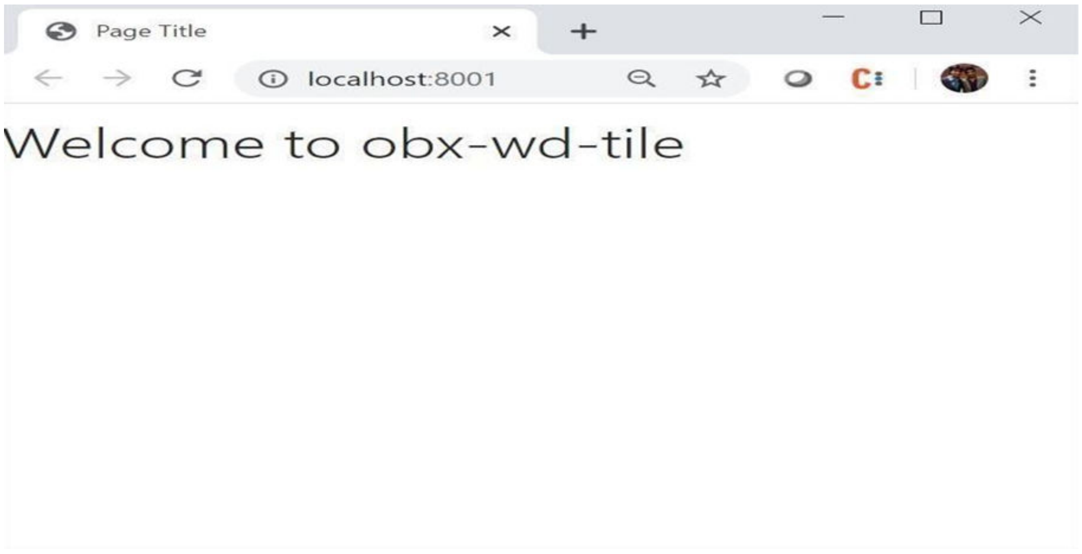
Figure 4-23   OBX sd amount extension folder



Name	Date modified	Type
node_modules	6/8/2020 7:31 PM	File folder
reports	6/8/2020 7:31 PM	File folder
web	6/8/2020 7:31 PM	File folder
app.js	6/8/2020 7:31 PM	JS File
build.gradle	6/8/2020 7:31 PM	GRADLE File
buildExtendedComponent.sh	6/8/2020 7:31 PM	SH File
gradle.properties	6/8/2020 7:31 PM	PROPERTIES F
gulpfile.js	6/8/2020 7:31 PM	JS File
Jenkinsfile	6/8/2020 7:31 PM	File
metadata.js	6/8/2020 7:31 PM	JS File
middleware.js	6/8/2020 7:31 PM	JS File
npm-link.sh	5/7/2020 8:47 AM	SH File
package.json	6/8/2020 7:31 PM	JSON File
startCS.js	6/8/2020 7:31 PM	JS File

7. Once done come back to main tab in cmder where is waiting with question **Please modify the Metadata.js file before proceeding. Once done press 'y' to proceed?**.
8. On completing the above process, it will automatically generate the source folder now and open a new tab on cmder where component will be running.
9. Along with cmder tab it will automatically open a tab on default browser as well with component rendered on the screen.

Figure 4-24 Cmder Component



Welcome to obx-wd-tile

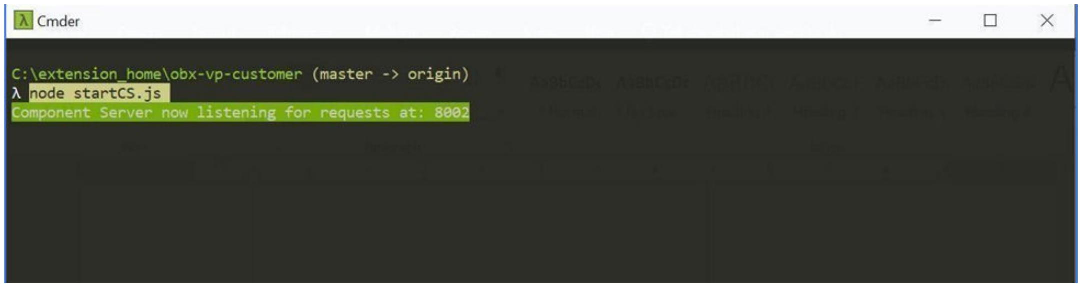
4.7 Running Component after Generation

This topic describes the steps you need to follow to re-run the component created or generated earlier.

Follow the below steps to do the same:

1. Make sure you always have the component server rightly created.
2. Open two tabs in the cmder tool.
3. Navigate to component folder in both the tabs for example **\extension_home\obx-vp-customer**.
4. From the first tab run the command **node startCS.js**.

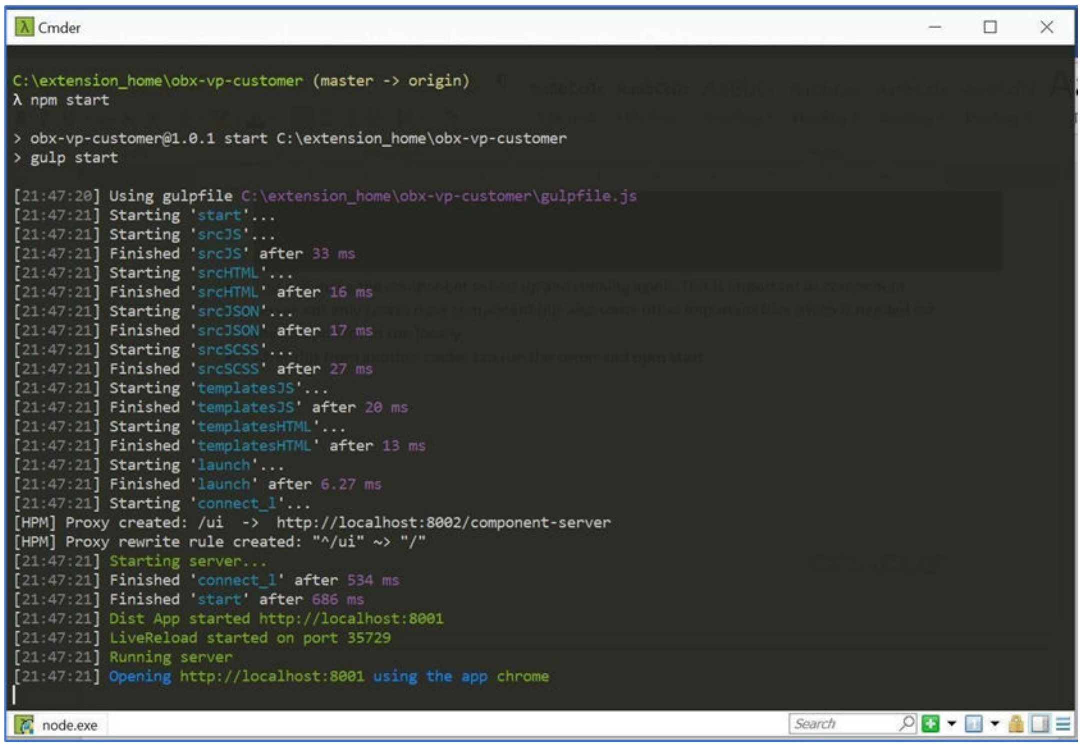
Figure 4-25 Node startCS.js



```
C:\extension_home\obx-vp-customer (master -> origin)
λ node startCS.js
Component Server now listening for requests at: 8002
```

5. This will make the component server up and running again. This is important as component server not only serves base component but also some other important files which is needed for the component to run locally.
6. After this from another cmder tab run the command **npm start**.

Figure 4-26 npm start



```
C:\extension_home\obx-vp-customer (master -> origin)
λ npm start

> obx-vp-customer@1.0.1 start C:\extension_home\obx-vp-customer
> gulp start

[21:47:20] Using gulpfile C:\extension_home\obx-vp-customer\gulpfile.js
[21:47:21] Starting 'start'...
[21:47:21] Starting 'srcJS'...
[21:47:21] Finished 'srcJS' after 33 ms
[21:47:21] Starting 'srcHTML'...
[21:47:21] Finished 'srcHTML' after 16 ms
[21:47:21] Starting 'srcJSON'...
[21:47:21] Finished 'srcJSON' after 17 ms
[21:47:21] Starting 'srcSCSS'...
[21:47:21] Finished 'srcSCSS' after 27 ms
[21:47:21] Starting 'templatesJS'...
[21:47:21] Finished 'templatesJS' after 20 ms
[21:47:21] Starting 'templatesHTML'...
[21:47:21] Finished 'templatesHTML' after 13 ms
[21:47:21] Starting 'launch'...
[21:47:21] Finished 'launch' after 6.27 ms
[21:47:21] Starting 'connect_1'...
[HPM] Proxy created: /ui  ->  http://localhost:8002/component-server
[HPM] Proxy rewrite rule created: "^/ui" ~> "/"
[21:47:21] Starting server...
[21:47:21] Finished 'connect_1' after 534 ms
[21:47:21] Finished 'start' after 686 ms
[21:47:21] Dist App started http://localhost:8001
[21:47:21] LiveReload started on port 35729
[21:47:21] Running server
[21:47:21] Opening http://localhost:8001 using the app chrome
```

7. This will make the component running again on http://localhost:8001/ and also open the default browser.

4.8 Creating final Extended Component war for Deployment

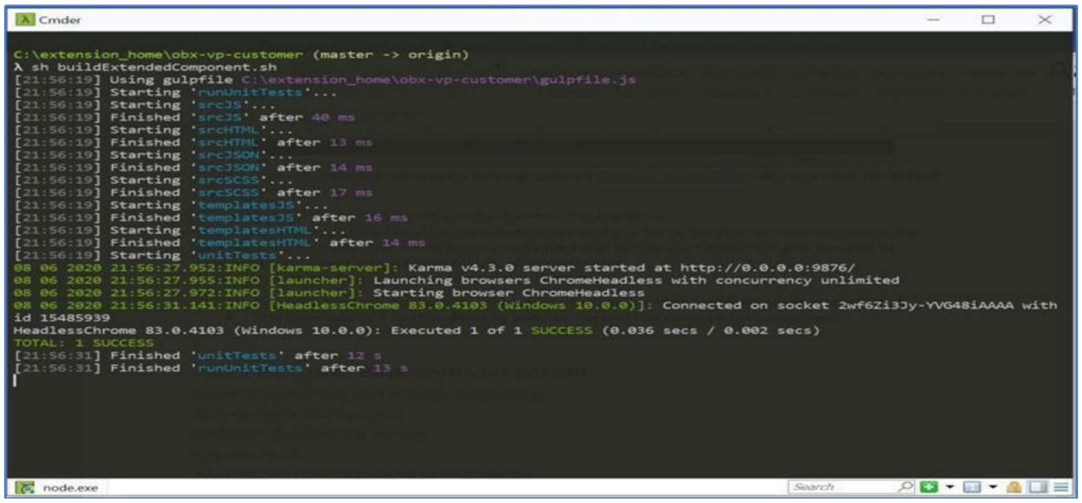
This topic describes the steps to generate Extended Component war for Deployment.

This is the final stage for generating extended-component war for all the Web components inside the **extension_home** folder. Important point to note here that before any component gets bundled to **extended-component.war**, it needs to pass all the test cases.

Perform the following steps to generate the war:

1. Go inside the individual component and run the command **sh buildExtendedComponent.sh**. This command will start performing and running unit test cases on the component.

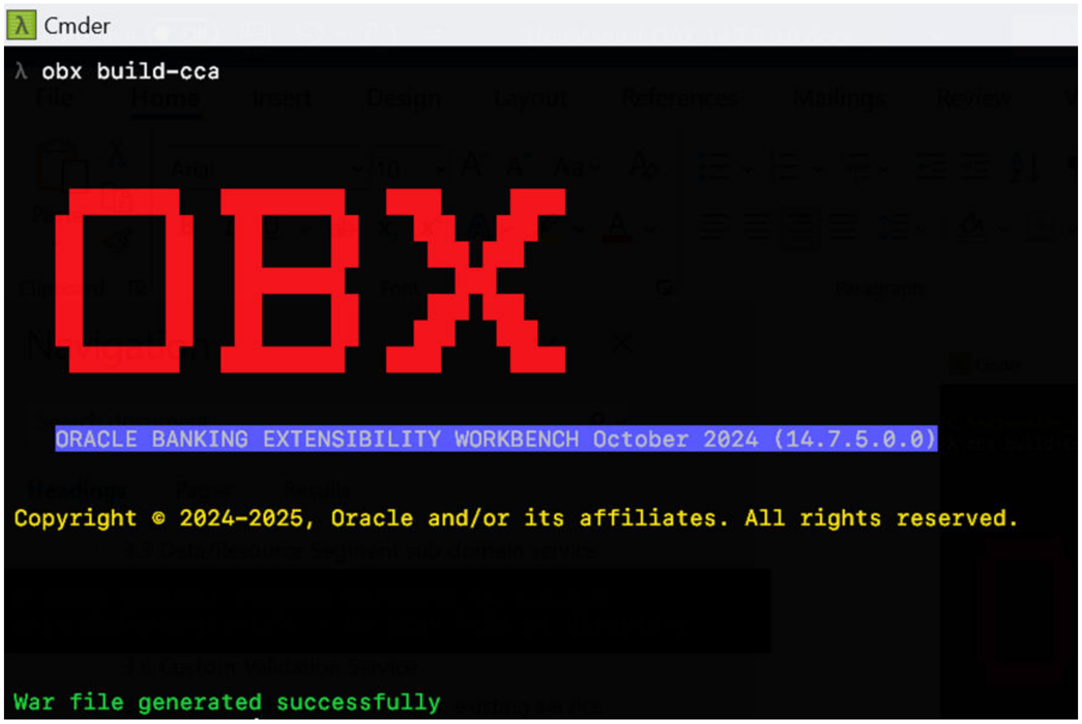
Figure 4-27   Command - sh buildExtendedComponent.sh



```
C:\extension_home\obx-vp-customer (master -> origin)
λ sh buildExtendedComponent.sh
[21:56:19] Using gulpfile C:\extension_home\obx-vp-customer\gulpfile.js
[21:56:19] Starting 'runUnitTests'...
[21:56:19] Starting 'srcJS'...
[21:56:19] Finished 'srcJS' after 40 ms
[21:56:19] Starting 'srcHTML'...
[21:56:19] Finished 'srcHTML' after 13 ms
[21:56:19] Starting 'srcJSON'...
[21:56:19] Finished 'srcJSON' after 14 ms
[21:56:19] Starting 'srcSCSS'...
[21:56:19] Finished 'srcSCSS' after 17 ms
[21:56:19] Starting 'templatesJS'...
[21:56:19] Finished 'templatesJS' after 16 ms
[21:56:19] Starting 'templatesHTML'...
[21:56:19] Finished 'templatesHTML' after 14 ms
[21:56:19] Starting 'unitTests'...
08 06 2020 21:56:27.952:INFO [karma-server]: Karma v4.3.0 server started at http://0.0.0.0:9876/
08 06 2020 21:56:27.955:INFO [launcher]: Launching browsers ChromeHeadless with concurrency unlimited
08 06 2020 21:56:27.972:INFO [launcher]: Starting browser ChromeHeadless
08 06 2020 21:56:31.141:INFO [HeadlessChrome 83.0.4103 (Windows 10.0.0)]: Connected on socket 2wf6Zi3Jy-YVG48iAAAA with
id 15485939
HeadlessChrome 83.0.4103 (Windows 10.0.0): Executed 1 of 1 SUCCESS (0.036 secs / 0.002 secs)
TOTAL: 1 SUCCESS
[21:56:31] Finished 'unitTests' after 12 s
[21:56:31] Finished 'runUnitTests' after 13 s
```

2. Once the test cases are executed successfully it will create a folder inside the **extension_home** folder named extended-components.
3. Now we have to navigate back to **extension_home** folder and run the command **obx build-cca**.

Figure 4-28   OBX UI



```
λ obx build-cca

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

War file generated successfully
```

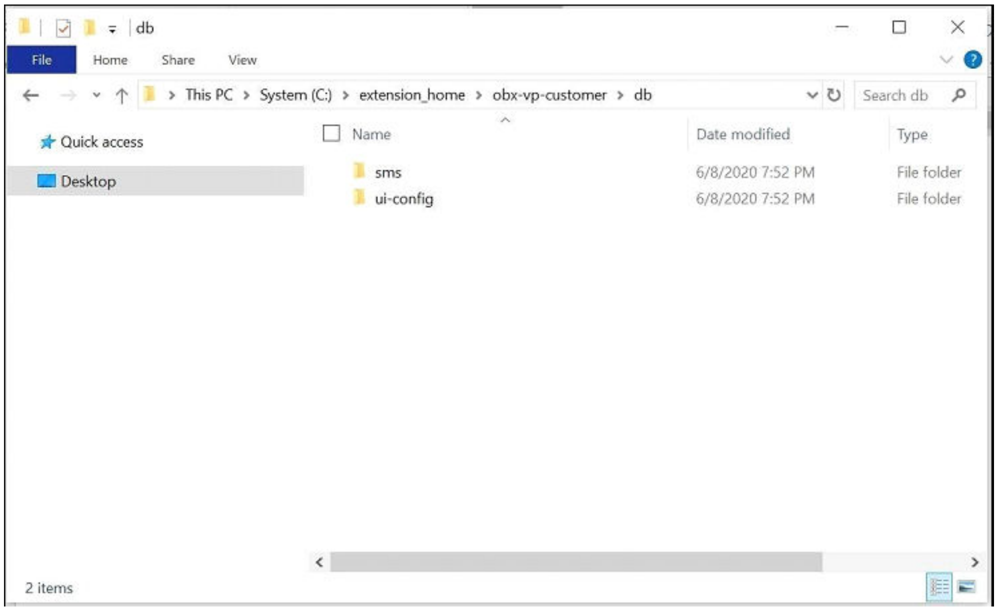
4. This **extended-component.war** should be deployed in the same domain where application shell is deployed.

4.9 Understanding DB Scripts for Web Components

This topic describes the significance of DB folder generate inside the web component folder.

This is important as without executing these scripts extension web components will not be loaded inside application shell and even these components menu will be not listed in left navigation menu.

Figure 4-29   DB Folder

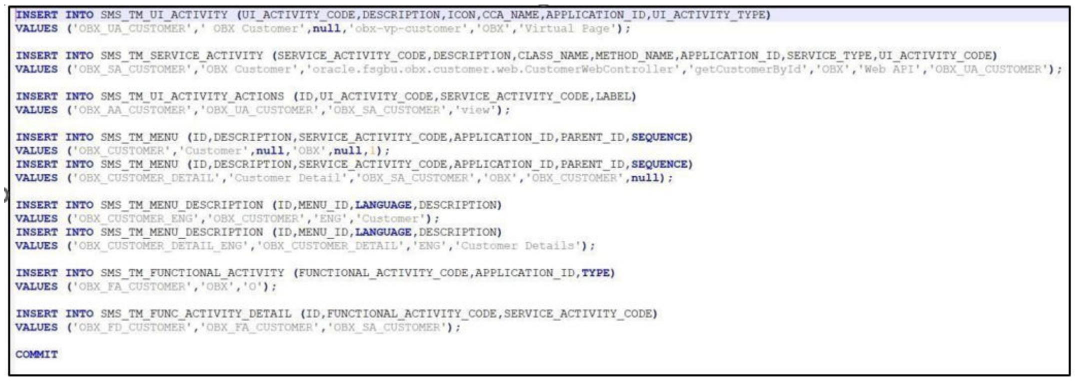

db
File
Home
Share
View
This PC > System (C:) > extension_home > obx-vp-customer > db
Search db
Quick access
Desktop
2 items

Name	Date modified	Type
sms	6/8/2020 7:52 PM	File folder
ui-config	6/8/2020 7:52 PM	File folder

DB folder inside the web component consists of two folders **sms** and **ui-config**:

- **SMS**: The sms scripts consists of all the service activity, functional activity generated all out of the box from OBX.

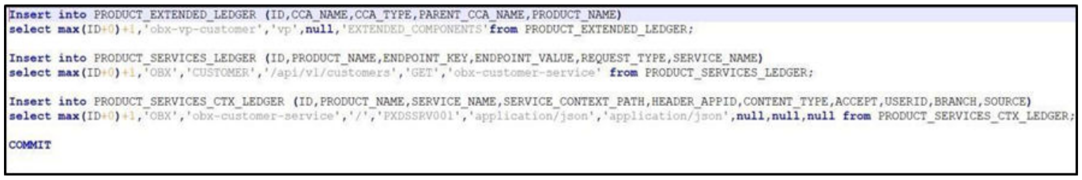
  Figure 4-30   SMS

```
INSERT INTO SMS_TM_UI_ACTIVITY (UI_ACTIVITY_CODE,DESCRIPTION,ICON,CCA_NAME,APPLICATION_ID,UI_ACTIVITY_TYPE)
VALUES ('OBX_UA_CUSTOMER',' OBX Customer',null,'obx-vp-customer','OBX','Virtual Page');

INSERT INTO SMS_TM_SERVICE_ACTIVITY (SERVICE_ACTIVITY_CODE,DESCRIPTION,CLASS_NAME,METHOD_NAME,APPLICATION_ID,SERVICE_TYPE,UI_ACTIVITY_CODE)
VALUES ('OBX_SA_CUSTOMER','OBX Customer','oracle.fsgbu.obx.customer.web.CustomerWebController','getCustomerById','OBX','Web API','OBX_UA_CUSTOMER');

INSERT INTO SMS_TM_UI_ACTIVITY_ACTIONS (ID,UI_ACTIVITY_CODE,SERVICE_ACTIVITY_CODE,LABEL)
VALUES ('OBX_AA_CUSTOMER','OBX_UA_CUSTOMER','OBX_SA_CUSTOMER','view');

INSERT INTO SMS_TM_MENU (ID,DESCRIPTION,SERVICE_ACTIVITY_CODE,APPLICATION_ID,PARENT_ID,SEQUENCE)
VALUES ('OBX_CUSTOMER','Customer',null,'OBX',null,1);
INSERT INTO SMS_TM_MENU (ID,DESCRIPTION,SERVICE_ACTIVITY_CODE,APPLICATION_ID,PARENT_ID,SEQUENCE)
VALUES ('OBX_CUSTOMER_DETAIL','Customer Detail','OBX_SA_CUSTOMER','OBX','OBX_CUSTOMER',null);

INSERT INTO SMS_TM_MENU_DESCRIPTION (ID,MENU_ID,LANGUAGE,DESCRIPTION)
VALUES ('OBX_CUSTOMER_ENG','OBX_CUSTOMER','ENG','Customer');
INSERT INTO SMS_TM_MENU_DESCRIPTION (ID,MENU_ID,LANGUAGE,DESCRIPTION)
VALUES ('OBX_CUSTOMER_DETAIL_ENG','OBX_CUSTOMER_DETAIL','ENG','Customer Details');

INSERT INTO SMS_TM_FUNCTIONAL_ACTIVITY (FUNCTIONAL_ACTIVITY_CODE,APPLICATION_ID,TYPE)
VALUES ('OBX_FA_CUSTOMER','OBX','O');

INSERT INTO SMS_TM_FUNC_ACTIVITY_DETAIL (ID,FUNCTIONAL_ACTIVITY_CODE,SERVICE_ACTIVITY_CODE)
VALUES ('OBX_FD_CUSTOMER','OBX_FA_CUSTOMER','OBX_SA_CUSTOMER');

COMMIT
```

- **UI Config**: This script should be compiled in ui-config schema. It maintains the ledger of all the extended components. App-shell uses this configuration to identify which components should be referred from extended-component war.

Figure 4-31 UI Config



```
Insert into PRODUCT_EXTENDED_LEDGER (ID,CCA_NAME,CCA_TYPE,PARENT_CCA_NAME,PRODUCT_NAME)
select max(ID+0)+1,'obx-vp-customer','vp',null,'EXTENDED_COMPONENTS'from PRODUCT_EXTENDED_LEDGER;

Insert into PRODUCT_SERVICES_LEDGER (ID,PRODUCT_NAME,ENDPOINT_KEY,ENDPOINT_VALUE,REQUEST_TYPE,SERVICE_NAME)
select max(ID+0)+1,'OBX','CUSTOMER','/api/v1/customers','GET','obx-customer-service' from PRODUCT_SERVICES_LEDGER;

Insert into PRODUCT_SERVICES_CTX_LEDGER (ID,PRODUCT_NAME,SERVICE_NAME,SERVICE_CONTEXT_PATH,HEADER_APPID,CONTENT_TYPE,ACCEPT,USERID,BRANCH,SOURCE)
select max(ID+0)+1,'OBX','obx-customer-service','/','PXDSSRV001','application/json','application/json',null,null,null from PRODUCT_SERVICES_CTX_LEDGER;

COMMIT
```

5

Modification of Base Web Component

This topic provides the systematic instructions to perform the basic operations on the selected records.

This feature of OBX enables users to create extensions which helps to modify the behavior of existing component. Modification of Base Web Component serves the one of the most common use cases from extensibility perspective. There are few important points which should be remembered before modifying the behavior of existing components.

Important Points:

- Addition of fields can be done on various locations of base screen, but this make break the CSS if not handled properly (Responsive Behavior). In such cases it is always recommended to put additional fields at the bottom of other fields.
- Wherever possible, use Data-segments to add additional field.
- In use case where you want to hide the fields from existing screen, always check whether the field is mandatory or not. If it is mandatory then it should set before making it hidden on the screen. If not done so service calls make break.
- Above point is also valid in case where you want to disable a field on the screen.

Following are the uses cases which can be achieved using modification of existing component:

- Addition of Fields
- Hiding fields from screen
- Defaulting values on screen
- Disable field
- Making Non-mandatory field Mandatory

- Steps for Modification of Base Component
  This topic provides the systematic instructions to the steps to follow in case of adding fields on the existing screen.
- Process Workbench
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- OBX Update Command
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- In-Scope DS
  This topic provides the systematic instructions to the overview of IN-Scope DS fields.
- OBX Release Command
  This topic provides information on OBX Release Command details.

5.1 Steps for Modification of Base Component

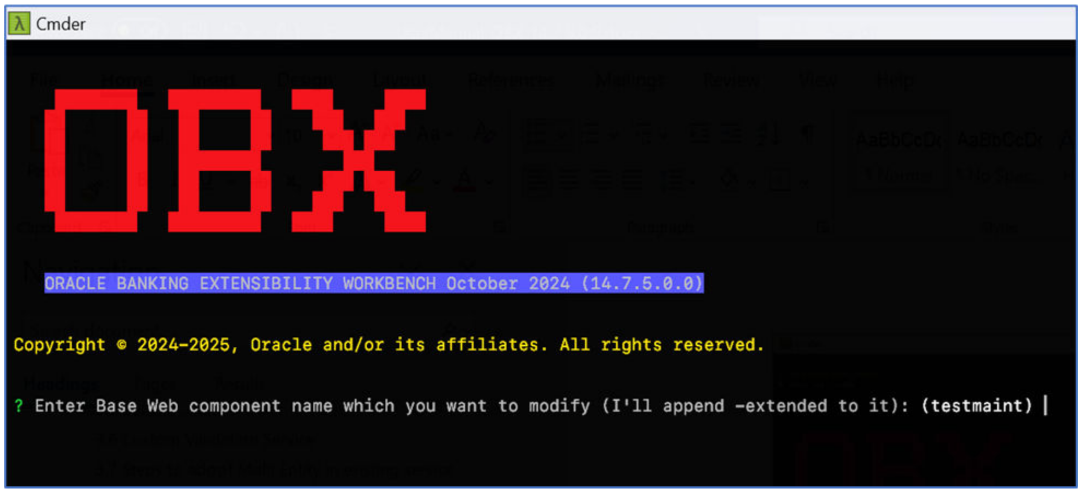
This topic provides the systematic instructions to the steps to follow in case of adding fields on the existing screen.

It is assumed that before using this command a developer knows the name of the base component in which he will be adding the additional fields.

Following are the steps needed to be followed:

1. Navigate to the **extension_home** folder from the cmder.
2. Execute the command **obx ui --mb**.

Figure 5-1 OBX UI



```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Enter Base Web component name which you want to modify (I'll append -extended to it): (testmaint) |
```

3. After above command is executed it will prompt for the name of base component. Once given it will create a folder with base component name appending - extended at the end of it.
4. Here also like above all the libraries are generated at runtime.
5. Component generated contains the boiler plate or reference code, which helps to achieve the use case.

Again, db folder contains all the relevant scripts which is needed to be executed prior to see the component live and running in main application shell.

5.2 Process Workbench

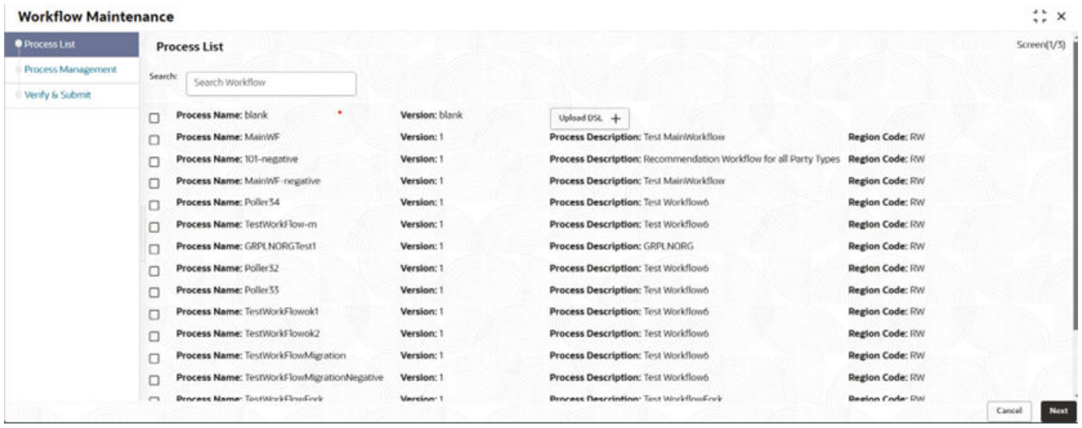
This topic provides the systematic instructions to perform the basic operations on the selected records.

The Process Workbench screen is used to create or modify processes. Users can add new stages, edit existing ones, or upload JSON-based DSLs into the system. This screen also facilitates workflow customization and allows users to download a JSON-based DSL reflecting the modifications made in the UI. Additionally, users can preview the flow diagram of a newly added or modified process. Any process changes will automatically increment the version number by 1 from the latest version.

Process Creation and Modification Screen:

1. **Screen 1** - Shows list of the processes:
   - **Displays the List of Processes:** A comprehensive list of existing processes is shown.
   - **Upload DSL Button:** Enables the upload of workflows in JSON format
   - **Blank Option (First Row):** Used to create a new process.

Figure 5-2 Workflow maintenance Process list



Workflow Maintenance

Process List | Process Management | Verify & Submit

Process List — Screen(1/3)

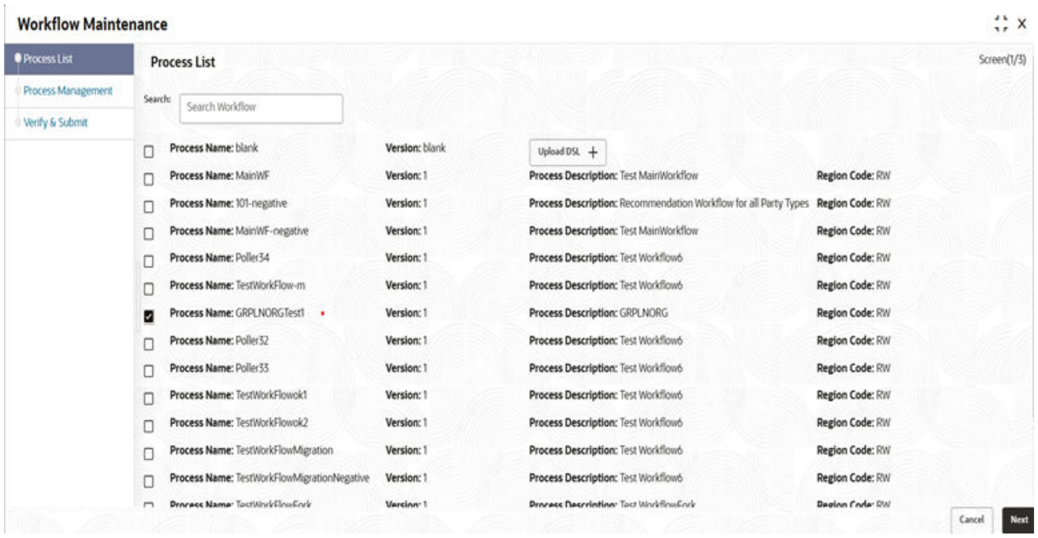
Search: Search Workflow

☐	**Process Name:** blank	**Version:** blank	Upload DSL +	
☐	**Process Name:** MainWF	**Version:** 1	**Process Description:** Test MainWorkflow	**Region Code:** RW
☐	**Process Name:** 101-negative	**Version:** 1	**Process Description:** Recommendation Workflow for all Party Types	**Region Code:** RW
☐	**Process Name:** MainWF-negative	**Version:** 1	**Process Description:** Test MainWorkflow	**Region Code:** RW
☐	**Process Name:** Poller34	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlow-m	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** GRPLNORGTest1	**Version:** 1	**Process Description:** GRPLNORG	**Region Code:** RW
☐	**Process Name:** Poller32	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** Poller33	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlowok1	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlowok2	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlowMigration	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlowMigrationNegative	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlowFork	**Version:** 1	**Process Description:** Test WorkflowFork	**Region Code:** RW

Cancel | Next

2. Select a Process.

Figure 5-3 Select a process list



Workflow Maintenance

Process List | Process Management | Verify & Submit

Process List — Screen(1/3)

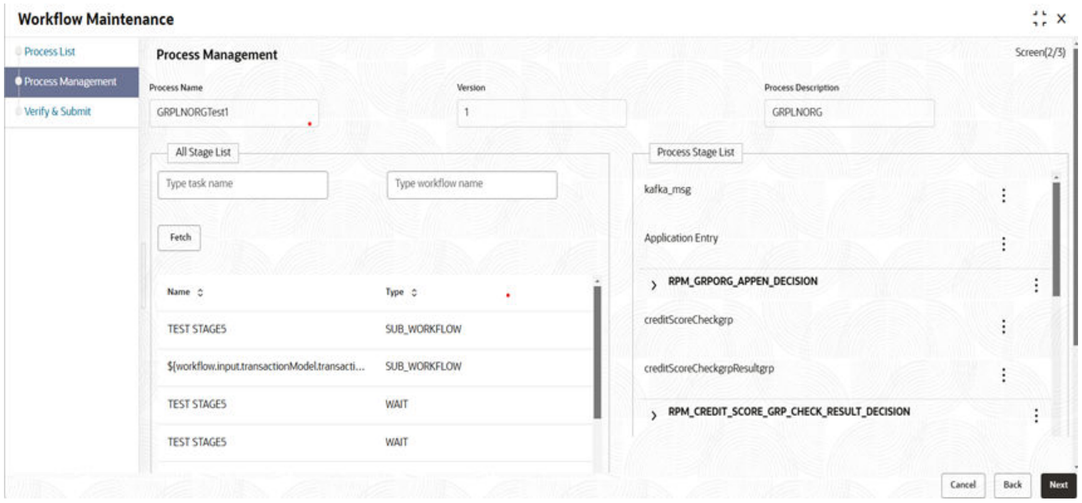
Search: Search Workflow

☐	**Process Name:** blank	**Version:** blank	Upload DSL +	
☐	**Process Name:** MainWF	**Version:** 1	**Process Description:** Test MainWorkflow	**Region Code:** RW
☐	**Process Name:** 101-negative	**Version:** 1	**Process Description:** Recommendation Workflow for all Party Types	**Region Code:** RW
☐	**Process Name:** MainWF-negative	**Version:** 1	**Process Description:** Test MainWorkflow	**Region Code:** RW
☐	**Process Name:** Poller34	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlow-m	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☑	**Process Name:** GRPLNORGTest1	**Version:** 1	**Process Description:** GRPLNORG	**Region Code:** RW
☐	**Process Name:** Poller32	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** Poller33	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlowok1	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlowok2	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlowMigration	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlowMigrationNegative	**Version:** 1	**Process Description:** Test Workflow6	**Region Code:** RW
☐	**Process Name:** TestWorkFlowFork	**Version:** 1	**Process Description:** Test WorkflowFork	**Region Code:** RW

Cancel | Next

3. Shows stages : Under the process which was selected on screen 1.

Figure 5-4   Workflow maintenance process management



Workflow Maintenance

Process List
Process Management
Verify & Submit

Process Management

Screen(2/3)

Process Name	Version	Process Description
GRPLNORGTest1	1	GRPLNORG

All Stage List

Type task name
Type workflow name
Fetch

Name	Type
TEST STAGES	SUB_WORKFLOW
${workflow.input.transactionModel.transacti...	SUB_WORKFLOW
TEST STAGES	WAIT
TEST STAGES	WAIT

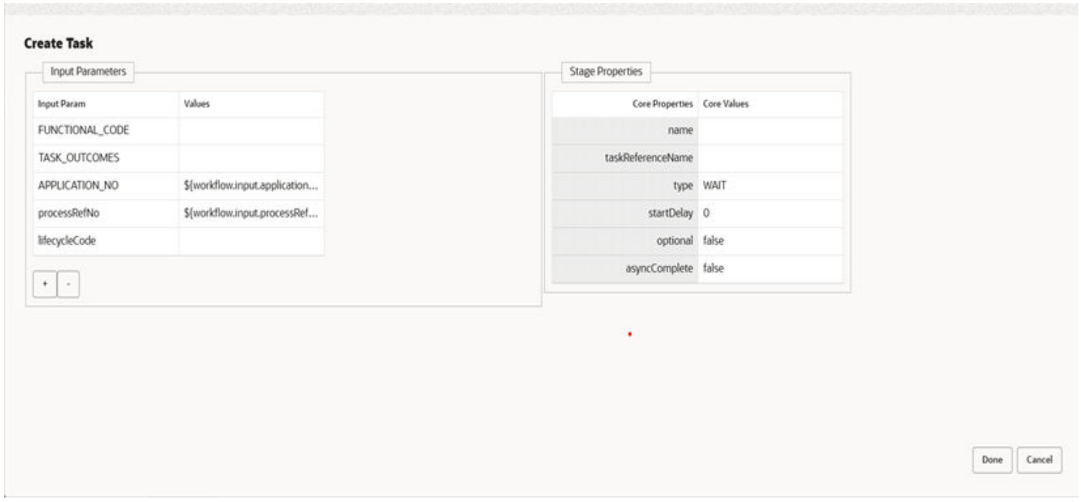
Process Stage List

kafka_msg
Application Entry
RPM_GRPORG_APPEN_DECISION
creditScoreCheckgrp
creditScoreCheckgrpResultgrp
RPM_CREDIT_SCORE_GRP_CHECK_RESULT_DECISION

Cancel
Back
Next

4. Create Stage button:
   - Used to create a new stage.
   - Dialog box for creating a new stage.

Figure 5-5   Create Task



Create Task

Input Parameters

Input Param	Values
FUNCTIONAL_CODE	
TASK_OUTCOMES	
APPLICATION_NO	${workflow.input.application...
processRefNo	${workflow.input.processRef...
lifecycleCode	

+ -

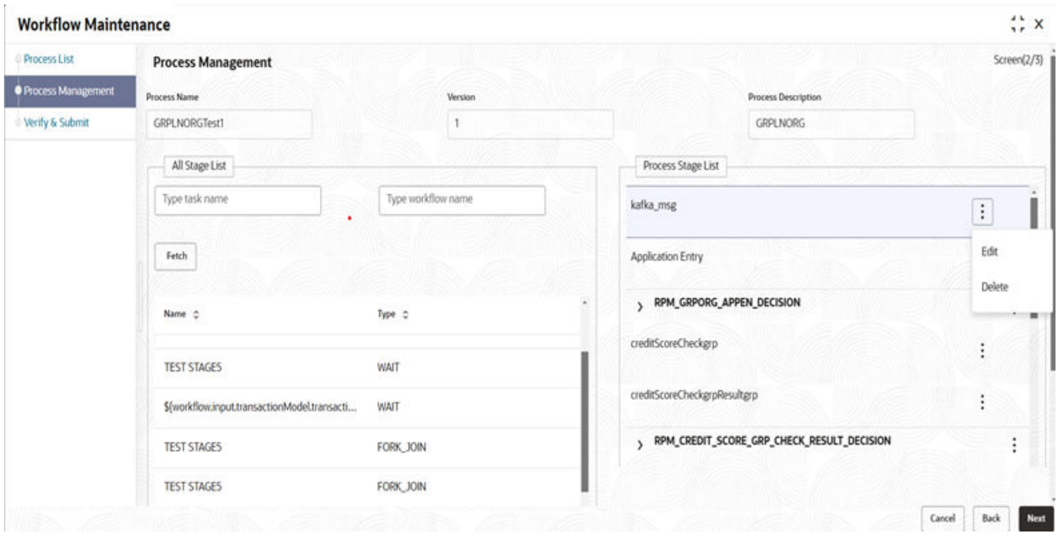
Stage Properties

Core Properties	Core Values
name	
taskReferenceName	
type	WAIT
startDelay	0
optional	false
asyncComplete	false

Done
Cancel

5. We can edit/delete a particular stage in Process Stage list.

Figure 5-6   Process management



Workflow Maintenance

Process List
Process Management
Verify & Submit

Process Management   Screen(2/3)

Process Name	Version	Process Description
GRPLNORGTest1	1	GRPLNORG

All Stage List

Type task name   Type workflow name

Fetch

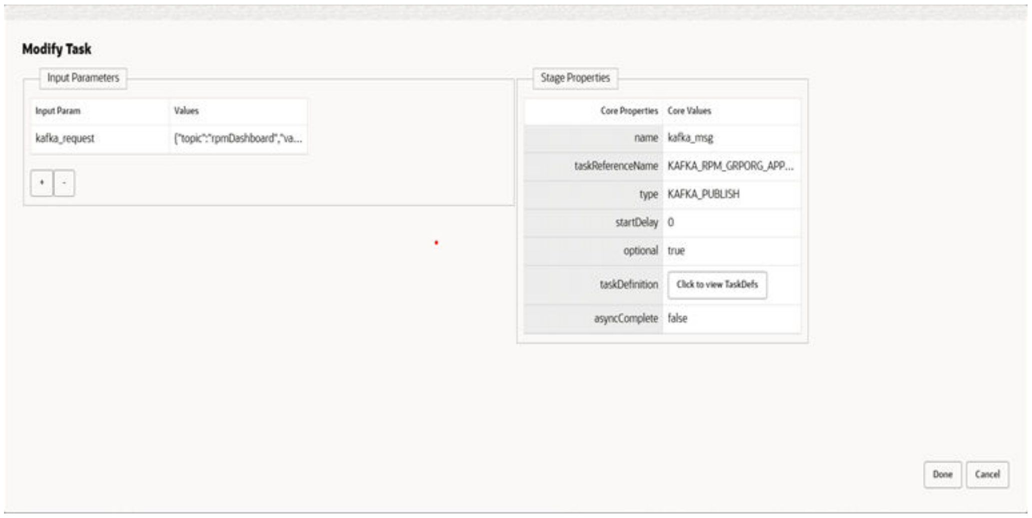
Name	Type
TEST STAGES	WAIT
${workflow.input.transactionModel.transacti...	WAIT
TEST STAGES	FORK_JOIN
TEST STAGES	FORK_JOIN

Process Stage List

kafka_msg
Application Entry
RPM_GRPORG_APPEN_DECISION
creditScoreCheckgrp
creditScoreCheckgrpResultgrp
RPM_CREDIT_SCORE_GRP_CHECK_RESULT_DECISION

6. Dialogue box which opens when we edit a particular stage.

Figure 5-7   Modify Task



Modify Task

Input Parameters

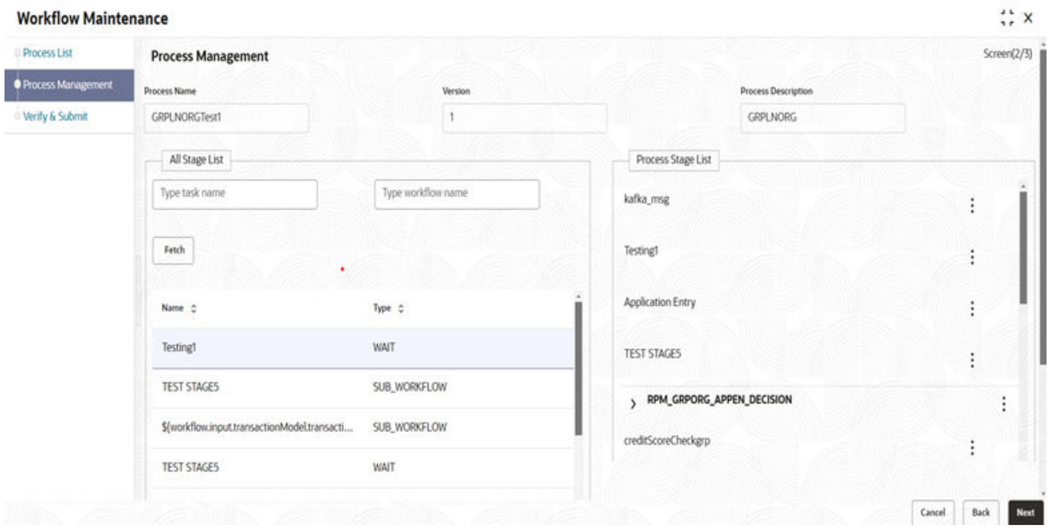
Input Param	Values
kafka_request	{"topic":"rpmDashboard","va...

Stage Properties

Core Properties	Core Values
name	kafka_msg
taskReferenceName	KAFKA_RPM_GRPORG_APP...
type	KAFKA_PUBLISH
startDelay	0
optional	true
taskDefinition	Click to view TaskDefs
asyncComplete	false

7. Drag and Drop Functionality Stage named **Testing1** from all stage list was dragged and dropped on the process stage list as shown here:

Figure 5-8 Process management Testing 1



Process Management

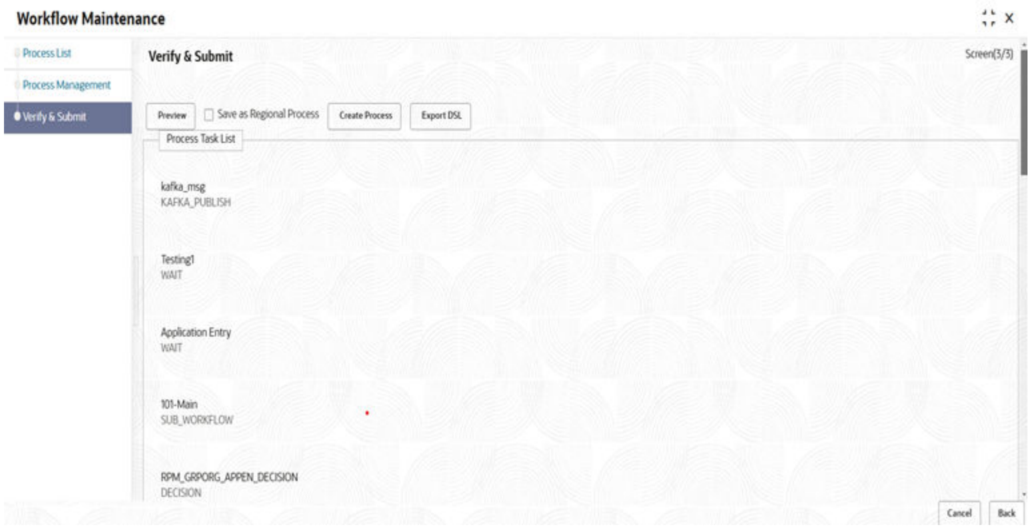
Process Name: GRPLNORGTest1
Version: 1
Process Description: GRPLNORG

Name	Type
Testing1	WAIT
TEST STAGE5	SUB_WORKFLOW
${workflow.input.transactionModel.transacti...	SUB_WORKFLOW
TEST STAGE5	WAIT

Process Stage List:
kafka_msg
Testing1
Application Entry
TEST STAGE5
RPM_GRPORG_APPEN_DECISION
creditScoreCheckgrp

8. In this process includes:
   - **Preview**: To preview flow diagram of the process selected.
   - **Create Process**: For creating a new process.
   - **Export DSL**: To Export DSL into a file in JSON format.

Figure 5-9 Workflow maintenance verify and submit



Verify & Submit

kafka_msg
KAFKA_PUBLISH

Testing1
WAIT

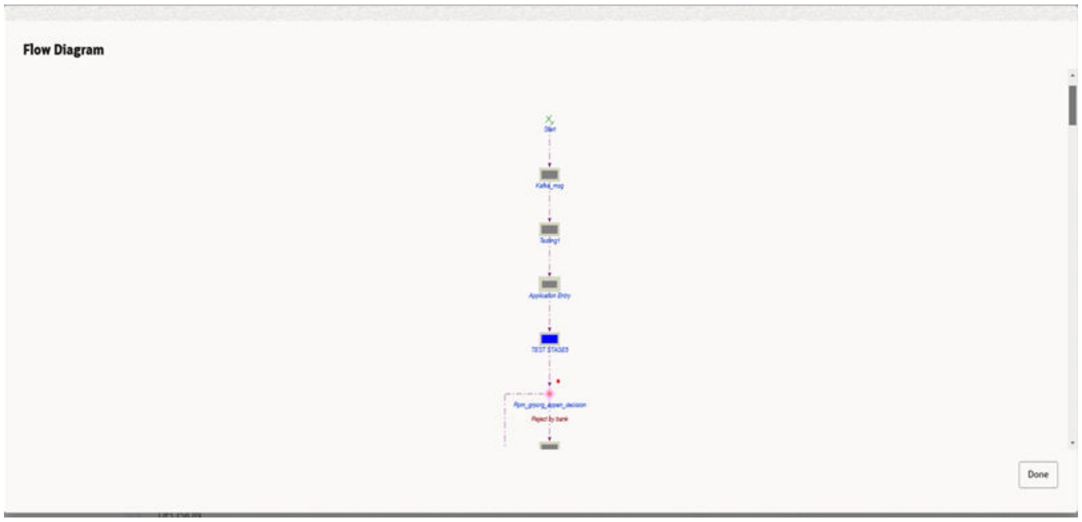
Application Entry
WAIT

101-Main
SUB_WORKFLOW

RPM_GRPORG_APPEN_DECISION
DECISION

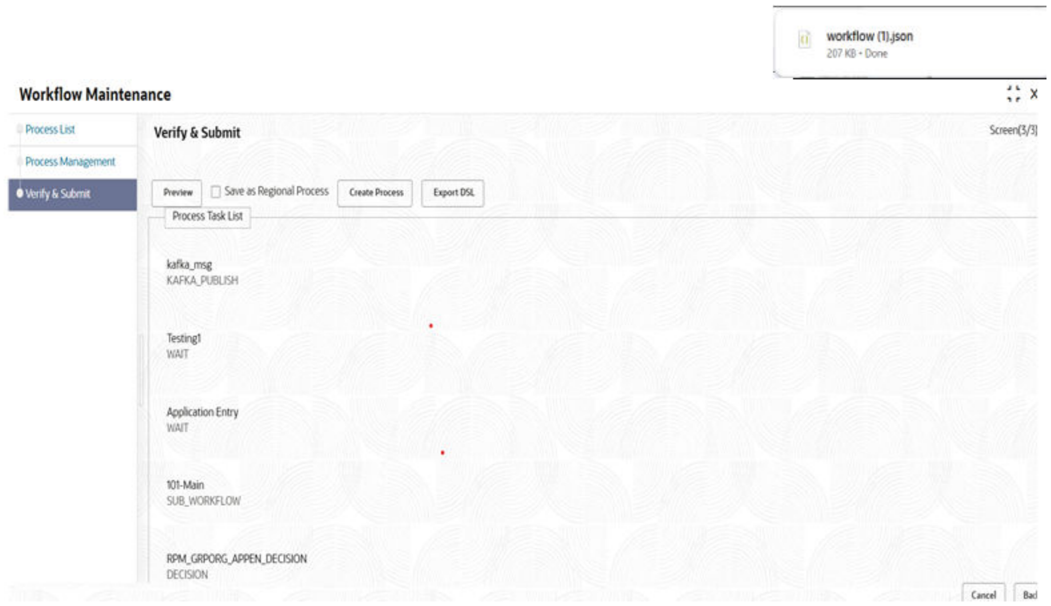
9. Flow Diagram of the modified or new added workbench process.

Figure 5-10   Flow Diagram



10. When **Export DSL** button is clicked. The DSL gets downloaded in workflow(1).json file as shown.

Figure 5-11   Export DSL



kafka_msg
KAFKA_PUBLISH

Testing1
WAIT

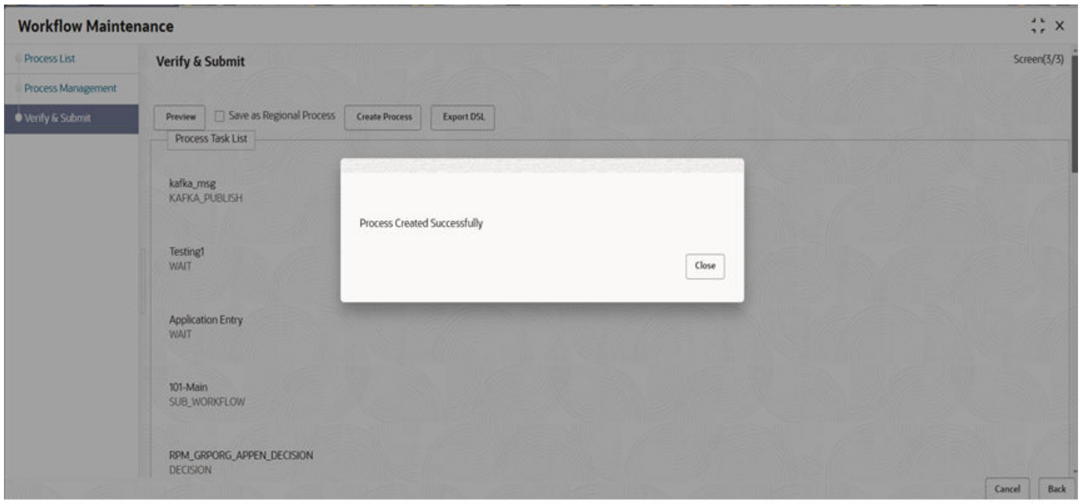
Application Entry
WAIT

101-Main
SUB_WORKFLOW

RPM_GRPORG_APPEN_DECISION
DECISION

11. When **Create Process** button is clicked. Process is Created.

Figure 5-12 Create Process

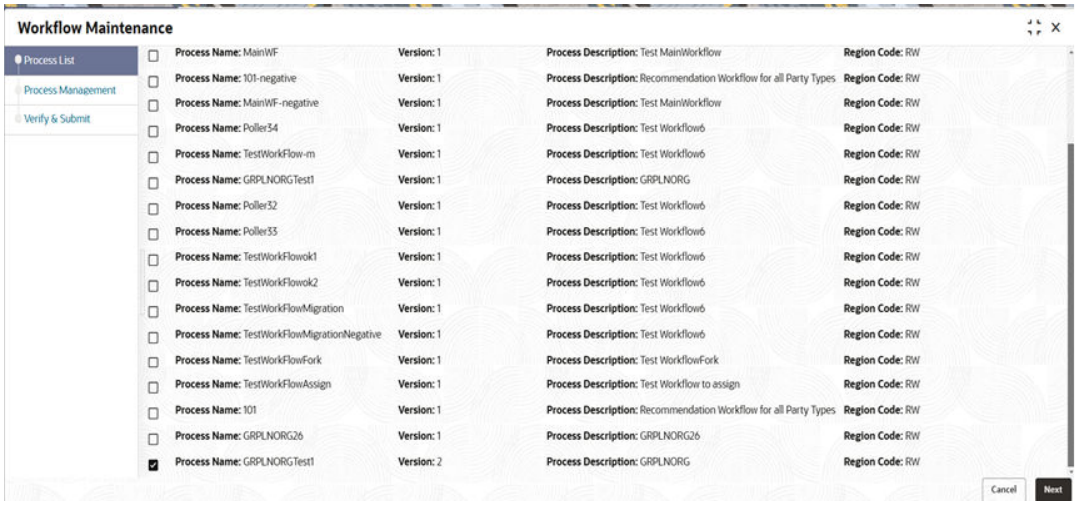


Process Created Successfully

kafka_msg	KAFKA_PUBLISH
Testing1	WAIT
Application Entry	WAIT
101-Main	SUB_WORKFLOW
RPM_GRPORG_APPEN_DECISION	DECISION

12. Version is updated when the process is created successfully.

Figure 5-13 Workflow maintenance updated version



☐	Process Name: MainWF	Version: 1	Process Description: Test MainWorkflow	Region Code: RW
☐	Process Name: 101-negative	Version: 1	Process Description: Recommendation Workflow for all Party Types	Region Code: RW
☐	Process Name: MainWF-negative	Version: 1	Process Description: Test MainWorkflow	Region Code: RW
☐	Process Name: Poller34	Version: 1	Process Description: Test Workflow6	Region Code: RW
☐	Process Name: TestWorkFlow-m	Version: 1	Process Description: Test Workflow6	Region Code: RW
☐	Process Name: GRPLNORGTest1	Version: 1	Process Description: GRPLNORG	Region Code: RW
☐	Process Name: Poller32	Version: 1	Process Description: Test Workflow6	Region Code: RW
☐	Process Name: Poller33	Version: 1	Process Description: Test Workflow6	Region Code: RW
☐	Process Name: TestWorkFlowok1	Version: 1	Process Description: Test Workflow6	Region Code: RW
☐	Process Name: TestWorkFlowok2	Version: 1	Process Description: Test Workflow6	Region Code: RW
☐	Process Name: TestWorkFlowMigration	Version: 1	Process Description: Test Workflow6	Region Code: RW
☐	Process Name: TestWorkFlowMigrationNegative	Version: 1	Process Description: Test Workflow6	Region Code: RW
☐	Process Name: TestWorkFlowFork	Version: 1	Process Description: Test WorkflowFork	Region Code: RW
☐	Process Name: TestWorkFlowAssign	Version: 1	Process Description: Test Workflow to assign	Region Code: RW
☐	Process Name: 101	Version: 1	Process Description: Recommendation Workflow for all Party Types	Region Code: RW
☐	Process Name: GRPLNORG26	Version: 1	Process Description: GRPLNORG26	Region Code: RW
✓	Process Name: GRPLNORGTest1	Version: 2	Process Description: GRPLNORG	Region Code: RW

5.3 OBX Update Command

This topic provides the systematic instructions to perform the basic operations on the selected records.

This topic helps in migrating the artifacts from previous version of OBX to latest. This is applied to both services and web components.

This topic consists of the following sub-topics:

- Service Update
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- UI Update
  This topic provides the systematic instructions to UI Update developed in OBX.

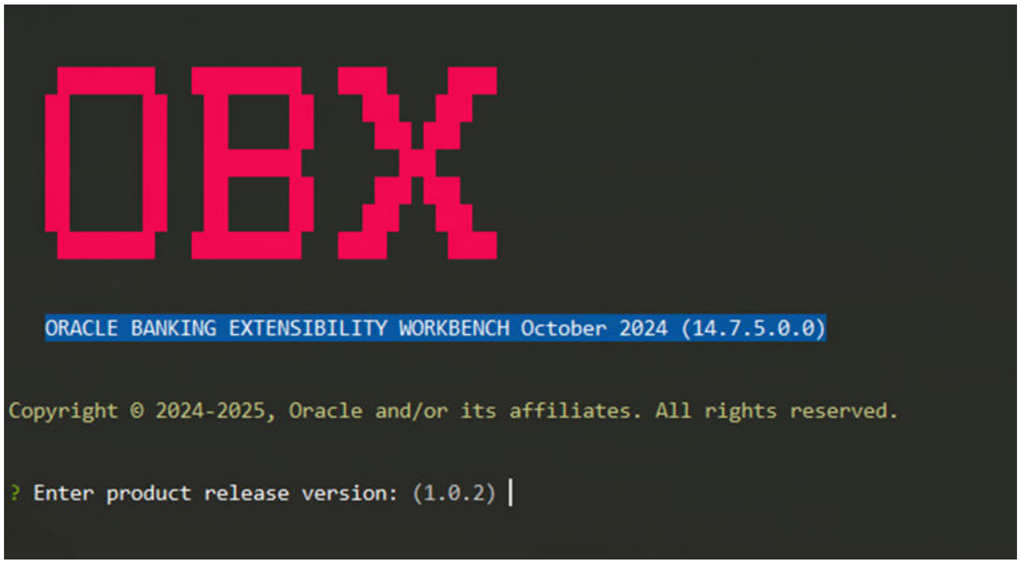
5.3.1 Service Update

This topic provides the systematic instructions to perform the basic operations on the selected records.

To migrate services developed in previous versions of OBX to latest please follow the below steps:

1. Navigate to service specific folder inside the **extension_home** directory.
2. Execute the command **obx service-update**.
3. Provide the relevant product release version number.
4. Once provided it will automatically change the build.gradle file and service is ready to be built with latest dependencies.

Figure 5-14 OBX UI-Service Update



```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Enter product release version: (1.0.2)
```

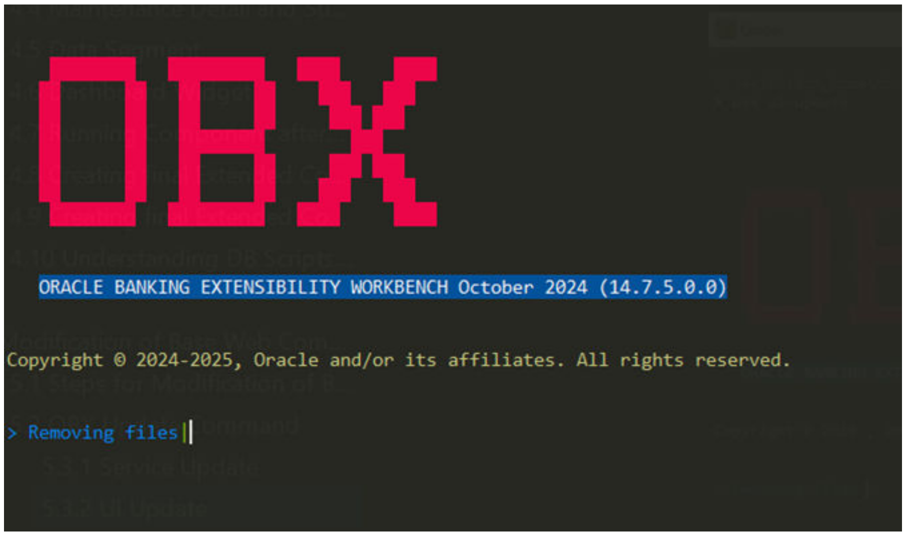
5.3.2 UI Update

This topic provides the systematic instructions to UI Update developed in OBX.

To migrate services developed in previous versions of OBX to latest please follow the below steps:

1. Navigate to UI (Web Component) specific folder inside the **extension_home** directory.
2. Execute the command **obx ui-update**.

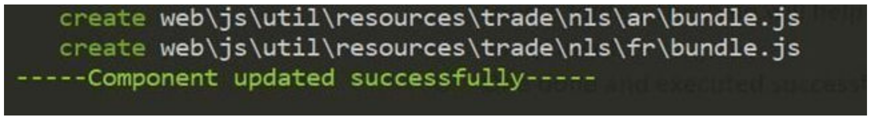
Figure 5-15 OBX UI-Update



```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

> Removing files
```

3. This command will automatically start removing old libraries without changing the source folder. This help will help you retaining the business logic already written in web component.
4. One done and executed successfully you will the below message.

Figure 5-16 Message



```
    create web\js\util\resources\trade\nls\ar\bundle.js
    create web\js\util\resources\trade\nls\fr\bundle.js
-----Component updated successfully-----
```

5. Now to run the command with new libraries run below command sequentially:
   - **sh npm-link.sh** – It will create new node module folder inside the component with latest modules and dependencies.
   - **node startCS.js** - Open a new tab in cmder and navigate to same web component directory and run command node startCS.js.
   - **npm start** – From the main tab, where we executed npm-link command run the command npm start, it will automatically run the web component with latest libraries and launch it on the browser as well.

5.4 In-Scope DS

This topic provides the systematic instructions to the overview of IN-Scope DS fields.

Following is the sequence to be followed:

- Additional of fields at any desired location in an existing data-segment is supported now.
- Data will be stored in separate custom schema.
- In-scope Data segment can be used for addition of new fields. (using jquery, at any position, we can add the field).

Example of In-Scope DS (Additional fields):

- Include the hooks required in js and html of base components accordingly.
- Run the command “obx ui --af” for adding fields in extended components.
- Include the additional field in “self.data”.

```
self.data = {
      "newField": ko.observable("")
  };
```

- Subscribe it to change handler.

```
self.data.newField.subscribe(self.changeHandler);
```

- Use jquery to insert it in the location you want to add the fields.

```
var element = context.properties.data.payload.homeBranch; $
('#homeBranch').parent().parent().parent().append($('#ui-ex-div-
newField').parent());
```

.

5.5 OBX Release Command

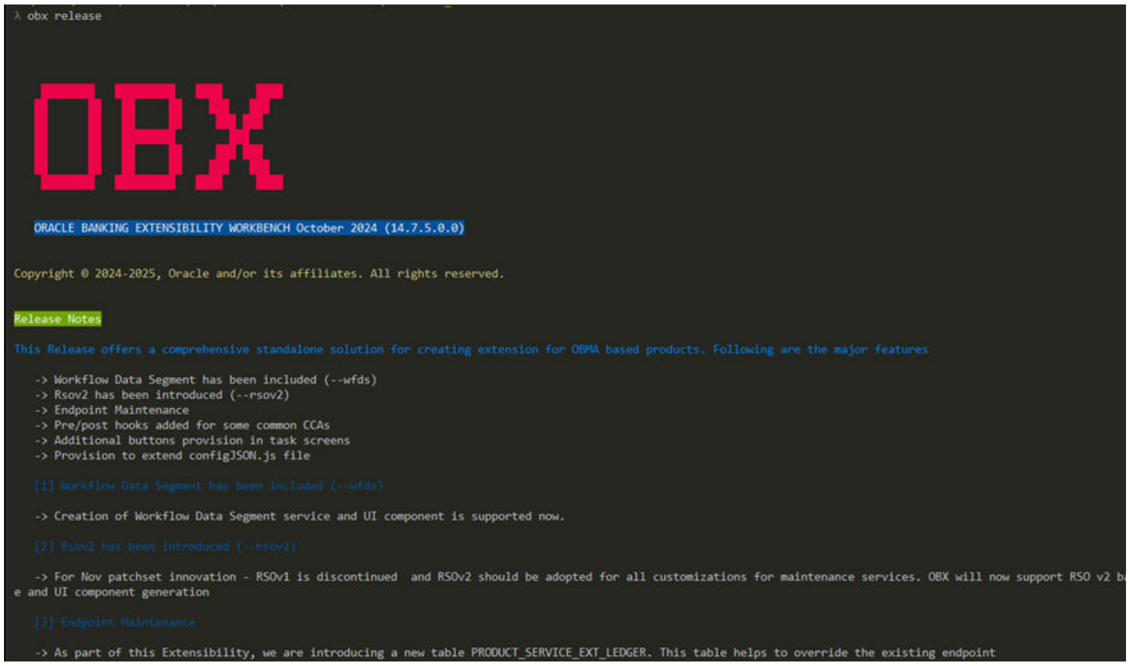
This topic provides information on OBX Release Command details.

This command is used to check all the available features bundled with OBX version installed on the machine.

To run this command,

1. Navigate to **extension_home** folder.
2. Run the command: **obx release**

Figure 5-17   OBX release



```
λ obx release

 OBX

   ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

Release Notes

This Release offers a comprehensive standalone solution for creating extension for OBMA based products. Following are the major features

   -> Workflow Data Segment has been included (--wfds)
   -> Rsov2 has been introduced (--rsov2)
   -> Endpoint Maintenance
   -> Pre/post hooks added for some common CCAs
   -> Additional buttons provision in task screens
   -> Provision to extend configJSON.js file

   [1] Workflow Data Segment has been included (--wfds)

   -> Creation of Workflow Data Segment service and UI component is supported now.

   [2] Rsov2 has been introduced (--rsov2)

   -> For Nov patchset innovation - RSOv1 is discontinued  and RSOv2 should be adopted for all customizations for maintenance services. OBX will now support RSO v2 b
e and UI component generation

   [3] Endpoint Maintenance

   -> As part of this Extensibility, we are introducing a new table PRODUCT_SERVICE_EXT_LEDGER. This table helps to override the existing endpoint
```

6 Extending Product Data Segments with Additional Fields

This topic provides the systematic instructions to perform the basic operations on the selected records.

This topic describes the following sub-topics:

- Additional Fields Maintenance
  This topic provides the systematic instructions on Additional Fields Maintenance.
- Populating Data in Corresponding Fields From UI
  This topic provides information on Populating Data in Corresponding Fields From UI.
- Fetching the Saved Values
  This topic provides information on fetching the saved values for each field during the transaction.

6.1 Additional Fields Maintenance

This topic provides the systematic instructions on Additional Fields Maintenance.

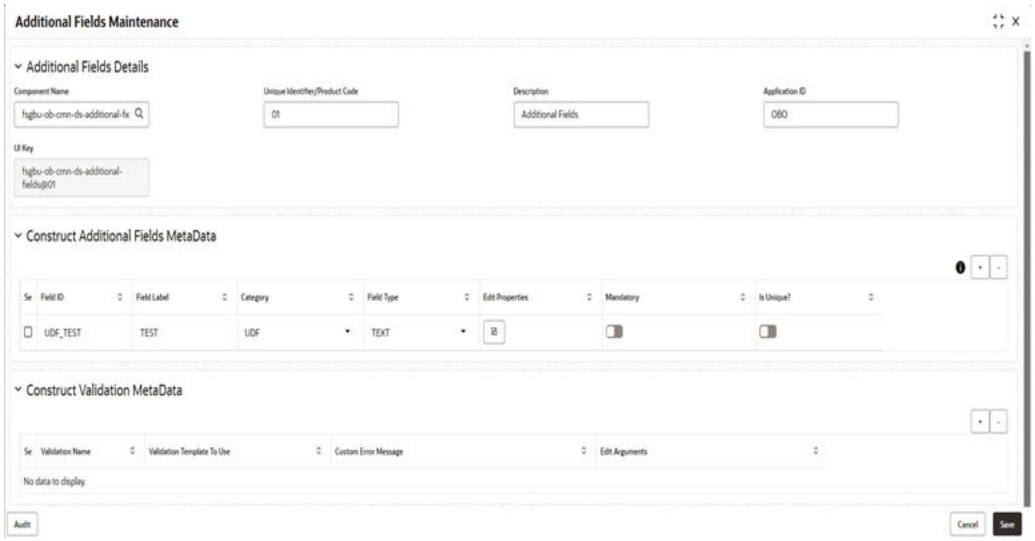
This screen is used to maintain the additional fields for a transaction screen.

To process this screen, type Additional Fields Maintenance in the Menu Item Search located at the left corner of the application toolbar and select the appropriate screen.

Follow the below steps:

1. From **Home** screen, click Core Maintenance. Under Core Maintenance, click Additional Fields Maintenance.
2. The **Additional Fields Maintenance** screen is displayed.

Figure 6-1 Additional Fields Maintenance



Additional Fields Maintenance

Additional Fields Details

Component Name	Unique Identifier/Product Code	Description	Application ID
fsgbu-ob-cmn-ds-additional-fi	01	Additional Fields	OBO

UI Key: fsgbu-ob-cmn-ds-additional-fields@01

Construct Additional Fields MetaData

Se	Field ID	Field Label	Category	Field Type	Edit Properties	Mandatory	Is Unique?
☐	UDF_TEST	TEST	UDF	TEXT			

Construct Validation MetaData

Se	Validation Name	Validation Template To Use	Custom Error Message	Edit Arguments

No data to display.

3. Specify the details in the Additional Fields Maintenance screen.
For more information on fields, refer table Field Description – Additional Field Maintenance.

Table 6-1 Additional Field Maintenance - Field Description

Field	Description
Component Name	Specify the data segment name as component name. **Note**:By default, the value **fsgbu-ob-cmndsadditional-fields** is displayed, which is the Common Core Data Segment that displays the maintained additional fields. It will fetch the corresponding maintained record for Additional Fields by querying with uiKey = DataSegmentName @ ProductCode.
Product Code	Specify the function code as product code.
Product Name	Displays the product name of the specified product code.
Description	Displays the description as **Additional Fields**.
Application ID	Displays the Application ID.
+ icon	Click this icon to add a new row.
– icon	Click this icon to delete a row, which is already added.
Construct Additional Fields MetaData	Specify the fields.
Select	Check this box to select a row.
Field ID	Specify the Field ID.
Field Label	Specify the field label.
Category	Specify the category.
Field Type	Specify the field type.
Edit	Select if a value needs to be inputted in the additional field.
Mandatory	Select if the input value is mandatory in the additional field.
Construct Validation MetaData	Specify the fields.
Select	Check this box to select a row.
Validation Name	Specify the validation name.

Table 6-1 (Cont.) Additional Field Maintenance - Field Description

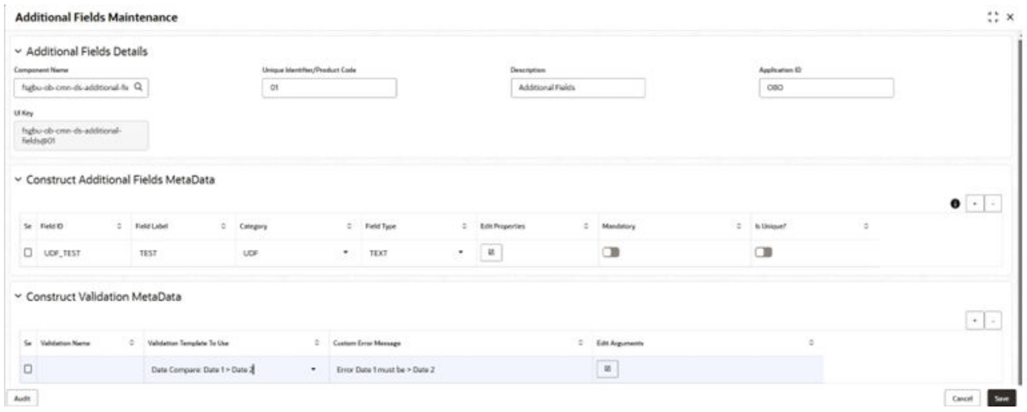
Field	Description
Validation Template to Use	Specify the template to be used for validation.
Custom Error Message	Specify the custom error message to be displayed.
Edit Arguments	Select if arguments needs to be edited in the additional field.

4. Click Save to add the additional field in the maintenance work table (CMC_TW_ADDT_ATTR_MASTER).

> **Note:**
>
> Once it is approved, the data will persist in the master table. Currently, Mobile Number and Date are added as additional fields. In addition,the validation is added for Date.

5. Sign in with different user ID since maker will not be able to approve the records with the same user ID.

Figure 6-2 Additional Fields Maintenance Records



Additional Fields Maintenance

Additional Fields Details

Component Name	Unique Identifier/Product Code	Description	Application ID
fsgbu-ob-cmn-ds-additional-fi	01	Additional Fields	OBO

UI Key: fsgbu-ob-cmn-ds-additional-fields@01

Construct Additional Fields MetaData

Se	Field ID	Field Label	Category	Field Type	Edit Properties	Mandatory	Is Unique?
☐	UDF_TEST	TEST	UDF	TEXT			

Construct Validation MetaData

Se	Validation Name	Validation Template To Use	Custom Error Message	Edit Arguments
☐		Data Compare: Date 1 > Date 2	Error Date 1 must be > Date 2	

6. Map the new data segment for the function code. Make sure that the data is present in **CMC_TM_SCREEN_DS_MAPPING**.

> **Note:**
>
> Once the additional fields are added for a particular function code, a separate data segment will be enabled in the transaction screen for Additional Fields.

7. Click Submit, to save the transaction data of additional fields to the CMC_TB_ADDT_ATTR_DATA.
   In addition, the following actions have been performed from service side:
   - Fetch record through inter-service call to additional attributes service in common transaction with record ID.

- Append the field data to the main payload for the ejlogging.

```
{
          "data": {
           "addDtls": {
             "signatureVerifyIndicator": "Y",
             "hostStatus": null,
             "hostMultiTripId": null,
             "txnBranchCcy": "GBP",
             "txnBranchDate": "2020-03-25T18:30:00.000+0000",
             "txnType": "C",
             "cashInOutIndicator": "I",
             "ejLoggingRequired": null,
             "ejTxnAmtMapping": "TO",
             "ejTxnCcyMapping": "TO",
             "adviceName": null,
             "orchestratorId": null,
             "rsp": null,
             "isReversal": "N",
             "isAdvice": "N",
             "reversalButton": "N",
             "ignoreApproval": false,
             "ignoreWarning": false,
             "isExternal": false
        },
        "txnDtls": {
             "functionCode": "1401",
             "txnBranchCode": null,
             "txnBranchCcy": null,
             "txnBranchDate": null,
             "requestStatus": "COMPLETED",
             "assignmentMode": null,
             "txnId": "f6b36a91-889d-4505-aac0-d7b98484d098",
             "txnRefNumber": "989124345493245",
             "tellerSeqNumber": null,
             "overrideConfirmFlag": null,
             "supervisorId": null,
             "onlineOfflineTxn": null,
             "userComments": null,
             "authoriserComments": null,
             "eventCode": null,
             "accountType": "UBS"
        },
        "dataPayload": {
             "datasegment": null,
             "fromAccountAmt": 100,
             "fromAccountCcy": "GBP",
             "toAccountCcy": "GBP",
             "beneficiaryName": null,
             "beneficiaryAddress1": null,
             "beneficiaryAddress2": null,
             "beneficiaryAddress3": null,
             "beneficiaryAddress4": null,
             "identificationType": null,
             "identificationNumber": null,
             "exchangeRate": 1,
"recievedAccount
 Ccy": null,
"recievedAccount
 Amt": null,
"totalCharges":
 null,
"cashAmount":
 100,
"netAccountCcy": null,
"netAccountAmt": null,
"narrative": "Cash Deposit",
"txnControllerRefNo": null,
"recordId": "f6b36a91-889d-4505-aac0-
 d7b98484d098", "cashAmtCcy": null,
"cashAmt":
 null,
"chequeDate": null,
"chequeNumber": null,
"eventCode": null,
"ejId": null,
"emailId": null,
"fromAccountBranch": "000",
"fromAccountNumber": null,
"mobileNumber": null,
"orginalExchangeRate": null,
"payee": null,
"productCode": null,
"reversalDate": null,
"stationId": null,
"toAccountBranch": "000",
"toAccountNumber": "00000008010010",
"toAccountAmt": 100,
"txnBranchCode": "000",
"functionCode": null,
"txnCustomer": null,
"tellerId": null,
"txnDate": 1585161000000,
"txnRefNumber": "9892566557744",
"txnSeqNumber": null,
"uniqueIdentifierNumber": null,
"uniqueIdentifierType": null,
"userRefNumber": null,
"valueDate": null,
"versionNumber": null,
"referenceNumber": null,
"createdBy": null,
"createdTs": null,
"updatedBy": null,
"updatedTs": null,
"demDtls": [],
"fxInDemDtls": null,
"fxOutDemDtls": null,
"prcDtls": [],
"addDtls": null,
"txnDtls": null,
"overrideDtls": null,
          "batchTableDetails": null,
          "cmcAddlFields": [
     {
          "id": "OTH_passprt",
          "label": "Passport No",
          "type": "TEXT",
          "value": "43243"
     },
     {
          "id": "UDF_aadhar",
          "label": "Aadhar",
          "type": "TEXT",
          "value": "1243"
     },
     {
          "id": "TMIS_toDate",
          "label": "To Date",
          "type": "DATE",
          "value": ""
     },
     {
          "id": "TMIS_fromDate",
          "label": "From Date",
          "type": "DATE",
          "value": ""
     }
     },
          "extDetails": null,
          "warDtls": [],
          "authoriserDtls": []
     },
          "errors": null,
          "warnings": null,
          "informations": null,
          "authorizations": null,
          "paging": ""
     }
```

6.2 Populating Data in Corresponding Fields From UI

This topic provides information on Populating Data in Corresponding Fields From UI.

Unlike the other transaction screen data-segments, the ejlogged data is not required. Instead, two GET calls that happen during screen launch fetches all the details.

To fetch the corresponding **Additional-Fields-Maintenance** screen record based on which it will display the maintained fields for this function code.

Endpoint : `CORE.GET_CMC_ADDITIONAL_ATTRIBUTES`

Request URL : http://whf00peb.in.oracle.com:8003/api-gateway/cmc-additional-attributesservices/cmcadditional-attributes-services/?uiKey=fsgbu-ob-cmn-ds-additional-fields@1006

Sample Response :

```
{
    "data": [
    {
              "keyId": "33347926-842b-4232-af31-8c1b59612244",
              "makerId": "ABHINAV",
              "makerDateStamp": null,
              "checkerId": null,
              "checkerDateStamp": null,
              "modNo": 1,
              "recordStatus": "O",
              "authStatus": "A",
              "onceAuth": null,
              "doerRemarks": null,
              "approverRemarks": null,
              "links": [
                       {
                       "rel": "self",
                       "href": "http://10.40.158.157:8005/cmc-
                        additional-attributesservices/cmcadditional-
                        attributes-services/33347926-842b-4232-
                        af318c1b59612244"
                        }
                        ],
               "description": "Additional Fields",
               "fieldMetaData":
               "[{\"id\":\"OTH_Mobile\",\"label\":\"Mobile
               Number\",\"type\":\"NUMBER\",\"required\":true},{\"id\":\"OTH
               _From\",\"label\":\"Fr om
               Date\",\"type\":\"DATE\",\"required\":true},{\"id\":\"OTH_To_D
               ate\",\"label\":\"To
               Date\",\"type\":\"DATE\",\"required\":true}]", "uiKey": "fsgbu-
               ob-cmn-ds-additional-fields@1006", "validationMetaData":
        "[{\"id\":\"\",\"validateMethod\":\"compareFromToDates\",\"type\":\"
               \",\"args\":[{\"ty
               pe\":\"FIELD\",\"value\":\"OTH_From\"},
       {\"type\":\"FIELD\",\"value\
               ":\"OTH_To_Date\"
               }],\"errorMsg\":\"Error Date 1 must be &amp;gt; Date
               2\",\"validationName\":\"Date
               Validation\"}]",
              "applicationId": "OBTFPM"
               } ],
        "paging": {
               "totalResults": 1,
               "links": {
               "next": null,
               "prev": null
                }
            }
        }
```

6.3 Fetching the Saved Values

This topic provides information on fetching the saved values for each field during the transaction.

You can fetch the values saved for each field during the transaction.

Endpoint : `CORE.GET_ADDITIONAL_ATTRIBUTES`.

Request URL : http://whf00peb.in.oracle.com:8003/api-gateway/cmc-additionalattributesservices/additionalattributes/?uiKey=fsgbu-ob-cmn-ds-additionalfields@1006&dataReferenceKey=00a01dfd- 0d6f-4400-a9c5-0f56551165e4

Samples Response :

```
{
     "ExtensibleDTO": [
     {
              "id": "1644022a-179e-429b-82c8-873761c3ac74",
              "uiKey": "fsgbu-ob-cmn-ds-additional-fields@1006",
              "dataReferenceKey": "00a01dfd-0d6f-4400-a9c5-
               0f56551165e4",
              "fieldMetaDataVersion": "1",
              "fieldData": [
                   {
                   "id": "OTH_Mobile",
                   "label": "Mobile Number",
                   "type": "NUMBER",
                   "value": “678688789”
                   },
                   {
                   "id": "OTH_From",
                   "label": "From Date",
                   "type": "DATE",
                   "value": ”678688789”
                   },
                   {
                   "id": "OTH_To_Date",
                   "label": "To Date",
                   "type": "DATE",
                   "value": null
                   }
                ],
                "applicationId": "OBREMO"
                }
            ]
        }
```

7 Action URL and Static Tag Maintenance

This topic provides the systematic instructions to perform the basic operations on Action URL and Static Tag Maintenance.

This topic consists of the following sub-topics:

- Action URL Maintenance
  This topic provides the systematic instructions of action URL maintenance.
- Static Tag Maintenance
  This topic provides the systematic instructions to static tag maintennace.

7.1 Action URL Maintenance

This topic provides the systematic instructions of action URL maintenance.

Endpoints are maintained in cmn-transaction-services for the specific transaction based on function code. The operation has to be maintained as action URL in table SRV_TB_BC_ACTIONS_URL. Action URL will be called from all the domain services based on function code and action (like OPENCHECK, CREATE, OVERRIDE, REVERSAL, PENDING_APPROVAL, or AUTHORIZE).

The database details are as follows:

Schema: BRANCHCOMMON

Table: SRV_TB_BC_ACTIONS_URL

If the action URL is not maintained for the specific operation of the particular transaction, the error message will be displayed as Action URL not maintained. Error code is maintained in ERTB_MSGS as RM-BC-UR-01.

7.2 Static Tag Maintenance

This topic provides the systematic instructions to static tag maintennace.

Static tag is maintained for accounting, till update, and debit-credit for each transaction based on the function code in table SRV_TB_TX_STATIC_TAGS.

The database details are as follows:

Schema : TRANSACTION

Table : SRV_TB_TX_STATIC_TAGS

TILL_TAGS, DRCR_TAGS and ACCOUNTING_TAGS are maintained as JSON structure. Static tags will be fetched from cmn-transaction-services based on function code. If it is not maintained for the particular function code, the transaction will be failed

8

Extensibility Use Cases for OBBRN Servicing

This topic provides the systematic instructions to perform the basic operations on Extensibility Use Cases for OBBRN Servicing.

This topic describes the following sub-topics:

- New Transaction Screen – 1499 (Exact Clone of 1401)
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- Exact Clone with Additional Fields Using Common Code
  This topic provides the systematic instructions to exact clone with additional fields using common code.
- Exact Clone with Additional Fields Using Extensible Code
  This topic provides the systematic instructions to the exact clone with additional fields using extensible code.
- Jar Deployment in Weblogic
  This topic provides the systematic instructions to the Jar Deployment in Weblogic.

8.1 New Transaction Screen – 1499 (Exact Clone of 1401)

This topic provides the systematic instructions to perform the basic operations on the selected records.

For this use case, you need to ensure data is present in the tables similar to 1401.

The below mentioned tables need to be checked in SMS schema:

- SMS_TM_MENU
- SMS_TM_MENU_Description
- SMS_TM_SERVICE_ACTIVITY
- SMS_TM_FUNCTIONAL_ACTIVITY
- SMS_TM_FUNC_ACTIVITY_DETAIL
- SMS_TM_ROLE_ACTIVITY
- SMS_TM_UI_ACTIVITY

The below mentioned tables need to be checked in common core schema:

- CMC_TM_SCREEN_CLASS
- CMC_TM_SCREEN_DS_MAPPING

The below mentioned tables need to be checked in branch common schema:

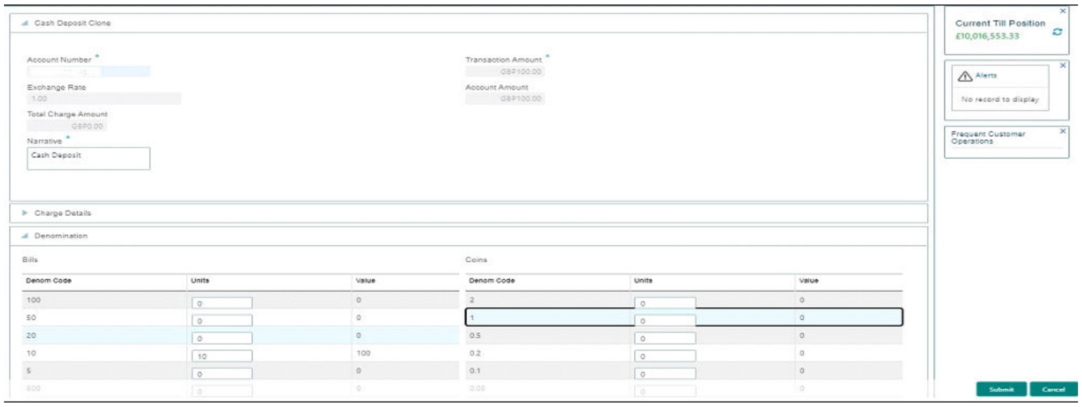
- SRV_TM_BC_FUNCTION_INDICATOR
- SRV_TM_BC_FUNCTION_CODE
- SRV_TM_BC_FUNCTION_PREF
- SRV_TM_BC_FUNCTION_PREF_DTLS

- SRV_TM_BC_BRANCH_ACCOUNTING
- SRV_TM_MENU_CONFIG
- SRV_TB_BC_ACTIONS_URL

The below mentioned tables need to be checked in transaction schema:

- SRV_TB_TX_STATIC_TAGS

Figure 8-1 Cash Deposit Clone



Cash Deposit Clone

Account Number *

Exchange Rate
1.00

Total Charge Amount
GBP0.00

Narrative *
Cash Deposit

Transaction Amount *
GBP100.00

Account Amount
GBP100.00

Charge Details

Denomination

Bills

Denom Code	Units	Value
100	0	0
50	0	0
20	0	0
10	10	100
5	0	0
500	0	0

Coins

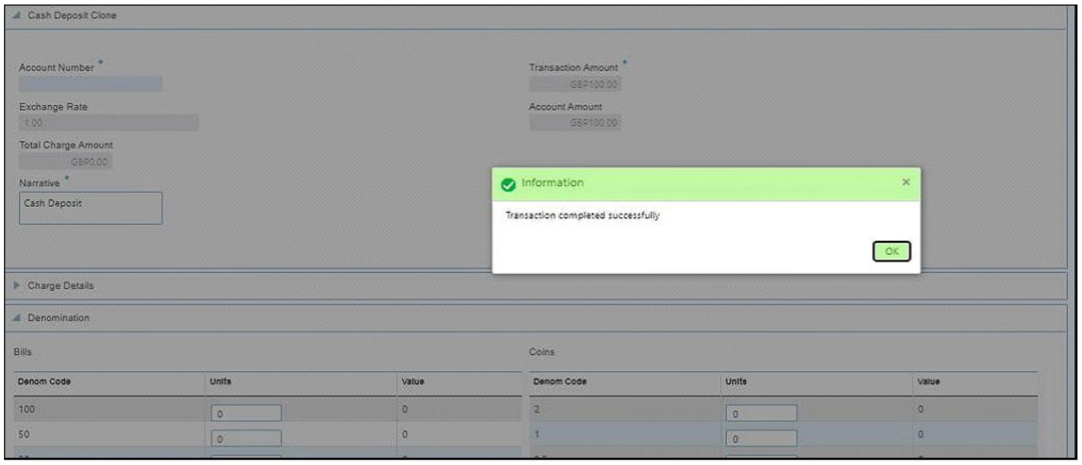
Denom Code	Units	Value
2	0	0
1	0	0
0.5	0	0
0.2	0	0
0.1	0	0
0.05	0	0

Current Till Position
£10,016,553.33

Alerts
No record to display

Frequent Customer Operations

Figure 8-2 Information Message



Cash Deposit Clone

Account Number *

Exchange Rate
1.00

Total Charge Amount
GBP0.00

Narrative *
Cash Deposit

Transaction Amount *
GBP100.00

Account Amount
GBP100.00

Information

Transaction completed successfully

Charge Details

Denomination

Bills

Denom Code	Units	Value
100	0	0
50	0	0

Coins

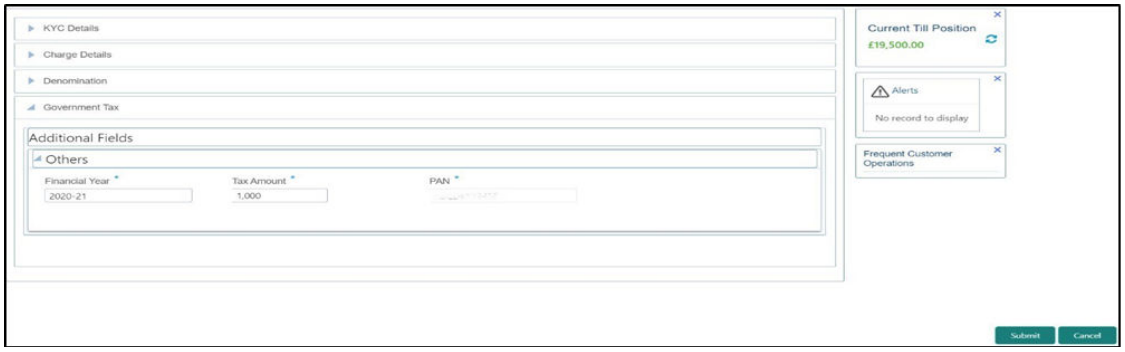
Denom Code	Units	Value
2	0	0
1	0	0

8.2 Exact Clone with Additional Fields Using Common Code

This topic provides the systematic instructions to exact clone with additional fields using common code.

A new screen is available with function code 9999. The Additional Fields is shown as 4th data segment as below:

Figure 8-3 Additional Fields Segment



KYC Details
Charge Details
Denomination
Government Tax
Additional Fields
Others
Financial Year * 2020-21
Tax Amount * 1,000
PAN *

Current Till Position
£19,500.00
Alerts
No record to display
Frequent Customer Operations

Submit Cancel

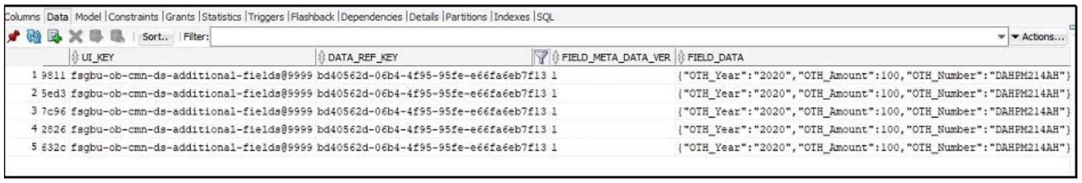
- The library reference in weblogic.xml is available for extensibility, for example, obremo-srv-ext-common-txn. A new jar obremo-srv-cmn-common-txn, which holds the most of the code of transaction service and can be a dependency in the external jar.

```
<wls:library-ref>
       <wls:library-name>obremo-srv-cmn-common-
 txn</wls:library-name> </wls:library-ref>
```

Response :

```
{
      "data": {
            "addDtls": {
            "signatureVerifyIndicator": "Y",
            "hostStatus": null,
            "hostMultiTripId": null,
            "txnBranchCcy": "GBP",
            "txnBranchDate": "2020-03-25T18:30:00.000+0000",
            "txnType": "C",
            "cashInOutIndicator": "I",
            "ejLoggingRequired": null,
            "ejTxnAmtMapping": "TO",
            "ejTxnCcyMapping": "TO",
            "adviceName": null,
            "orchestratorId": null,
            "rsp": null,
            "isReversal": "N",
            "crossCcyEnabled": null,
            "isTotChargesReq": null
        },
        "txnDtls": {
            "functionCode": "9999",
            "txnBranchCode": null,
            "txnBranchCcy": null,
            "txnBranchDate": null,
            "requestStatus": "COMPLETED",
            "assignmentMode": null,
            "txnId": "71a08a0f-ee2a-405b-a1e3-b77ca9e59b6e",
        "txnRefNumber": "0002008600007160",
        "tellerSeqNumber": null,
        "overrideConfirmFlag": "N",
        "supervisorId": null,
        "onlineOfflineTxn": null,
        "userComments": null,
        "authoriserComments": null,
        "eventCode": null,
        "accountType": "UBS"
    },
    "dataPayload": {
        "datasegment": null,
        "fromAccountAmt": 100,
        "fromAccountCcy": "GBP",
        "toAccountCcy": "GBP",
        "beneficiaryName": null,
        "beneficiaryAddress1": null,
        "beneficiaryAddress2": null,
        "beneficiaryAddress3": null,
        "beneficiaryAddress4": null,
        "identificationType": null,
        "identificationNumber": null,
        "exchangeRate": 1,
        "recievedAccount
        Ccy": null,
        "recievedAccount
        Amt": null,
        "totalCharges":
        null,
        "cashAmount":
        null,
        "netAccountCcy": null,
        "netAccountAmt": null,
        "narrative": "Cash Deposit",
        "txnControllerRefNo": null,
        "recordId": "bd40562d-06b4-4f95-95fe-
        e66fa6eb7f13", "cashAmtCcy": null,
        "cashAmt":
        null,
        "chequeDate": null,
        "chequeNumber": null,
        "eventCode": null,
        "ejId": null,
        "emailId": null,
        "fromAccountBranch": "000",
        "fromAccountNumber": null,
        "mobileNumber": null,
        "orginalExchangeRate": null,
        "payee": null,
        "productCode": null,
        "reversalDate": null,
        "stationId": null,
        "toAccountBranch": "000",
        "toAccountNumber": "00000008010010",
        "toAccountAmt": 100,
        "txnBranchCode": "000",
            "functionCode": null,
            "txnCustomer": null,
            "tellerId": null,
            "txnDate": 1585161000000,
            "txnRefNumber": "0002008600007160",
            "txnSeqNumber": null,
            "uniqueIdentifierNumber": null,
            "uniqueIdentifierType": null,
            "userRefNumber": null,
            "valueDate": null,
            "versionNumber": null,
            "referenceNumber": null,
            "createdBy": null,
            "createdTs": null,
            "updatedBy": null,
            "updatedTs": null,
            "demDtls": null,
            "fxInDemDtls": null,
            "fxOutDemDtls": null,
            "prcDtls": null,
            "addDtls": null,
            "txnDtls": null,
            "overrideDtls": null,
            "batchTableDetails": null
        },
        "extDetails": null,
        "warDtls": [],
        "authoriserDtls": []
    },
    "errors": null,
    "warnings": null,
    "informations": null,
    "authorizations": null,
    "paging": ""
}
```

Figure 8-4   Common Core Additional Attributes

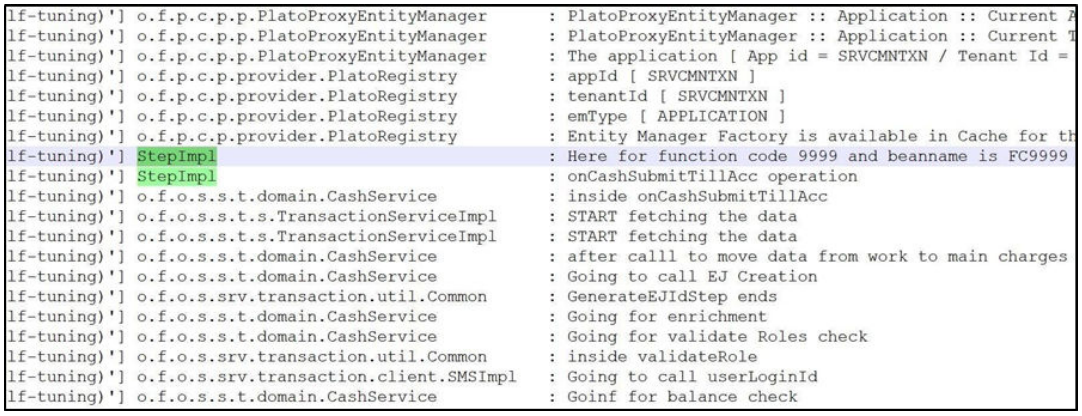


Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

	UI_KEY	DATA_REF_KEY	FIELD_META_DATA_VER	FIELD_DATA
1	9811 fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}
2	5ed3 fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}
3	7c96 fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}
4	2826 fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}
5	632c fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}

- In the debug, you can find that the common code is used, stempImpl onCashSubmitTillAcc will be called.

Figure 8-5   Common Code



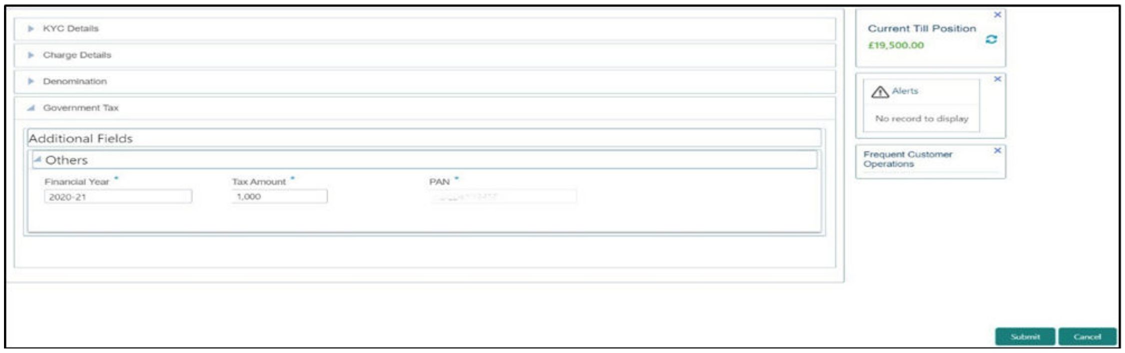
```
lf-tuning)'] o.f.p.c.p.p.PlatoProxyEntityManager      : PlatoProxyEntityManager :: Application :: Current A
lf-tuning)'] o.f.p.c.p.p.PlatoProxyEntityManager      : PlatoProxyEntityManager :: Application :: Current T
lf-tuning)'] o.f.p.c.p.p.PlatoProxyEntityManager      : The application [ App id = SRVCMNTXN / Tenant Id =
lf-tuning)'] o.f.p.c.p.provider.PlatoRegistry         : appId [ SRVCMNTXN ]
lf-tuning)'] o.f.p.c.p.provider.PlatoRegistry         : tenantId [ SRVCMNTXN ]
lf-tuning)'] o.f.p.c.p.provider.PlatoRegistry         : emType [ APPLICATION ]
lf-tuning)'] o.f.p.c.p.provider.PlatoRegistry         : Entity Manager Factory is available in Cache for th
lf-tuning)'] StepImpl                                 : Here for function code 9999 and beanname is FC9999
lf-tuning)'] StepImpl                                 : onCashSubmitTillAcc operation
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : inside onCashSubmitTillAcc
lf-tuning)'] o.f.o.s.s.t.s.TransactionServiceImpl     : START fetching the data
lf-tuning)'] o.f.o.s.s.t.s.TransactionServiceImpl     : START fetching the data
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : after calll to move data from work to main charges
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : Going to call EJ Creation
lf-tuning)'] o.f.o.s.srv.transaction.util.Common      : GenerateEJIdStep ends
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : Going for enrichment
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : Going for validate Roles check
lf-tuning)'] o.f.o.s.srv.transaction.util.Common      : inside validateRole
lf-tuning)'] o.f.o.s.srv.transaction.client.SMSImpl   : Going to call userLoginId
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : Goinf for balance check
```

8.3 Exact Clone with Additional Fields Using Extensible Code

This topic provides the systematic instructions to the exact clone with additional fields using extensible code.

A screen is created with function code 9999 and Additional Fields as 4th data segment.

Figure 8-6   Additional Fields Segment



Financial Year * 2020-21

Tax Amount * 1,000

PAN * [illegible]

£19,500.00

No record to display

- A library reference is added weblogic.xml (obremo-srv-ext-common-txn) for extensibility. A new jar obremosrvcmn-common-txn, which holds the most of the code of transaction service and can be a dependency in the external jar

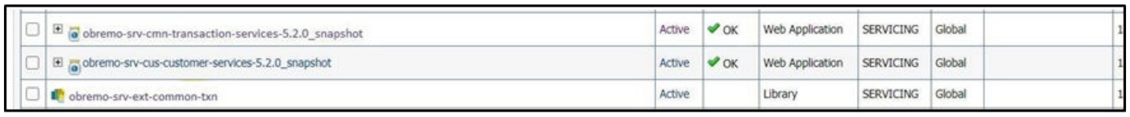
```
<wls:library-ref>
            <wls:library-name>obremo-srv-cmn-common-txn</wls:library-name>
            </wls:library-ref>
```

8.4 Jar Deployment in Weblogic

This topic provides the systematic instructions to the Jar Deployment in Weblogic.

Below screen shows the Jar Deployment in weblogic.

Figure 8-7   Jar Deployment

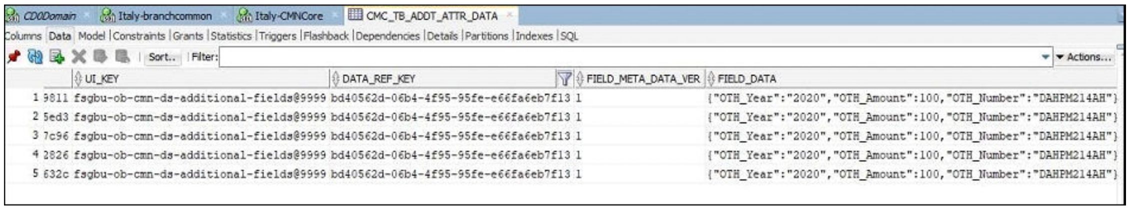


☐	obremo-srv-cmn-transaction-services-5.2.0_snapshot	Active	✓ OK	Web Application	SERVICING	Global		1
☐	obremo-srv-cus-customer-services-5.2.0_snapshot	Active	✓ OK	Web Application	SERVICING	Global		1
☐	obremo-srv-ext-common-txn	Active		Library	SERVICING	Global		1

Response:

```
{
        "data": {
           "addDtls": {
                  "signatureVerifyIndicator": "Y",
                  "hostStatus": null,
                  "hostMultiTripId": null,
                  "txnBranchCcy": "GBP",
                  "txnBranchDate": "2020-03-25T18:30:00.000+0000",
                  "txnType": "C",
                  "cashInOutIndicator": "I",
                  "ejLoggingRequired": null,
                  "ejTxnAmtMapping": "TO",
                  "ejTxnCcyMapping": "TO",
                  "adviceName": null,
                  "orchestratorId": null,
                  "rsp": null,
                  "isReversal": "N",
                  "crossCcyEnabled": null,
                  "isTotChargesReq": null
              },
              "txnDtls": {
                  "functionCode": "9999",
                  "txnBranchCode": null,
                  "txnBranchCcy": null,
                  "txnBranchDate": null,
                  "requestStatus": "COMPLETED",
                  "assignmentMode": null,
                  "txnId": "71a08a0f-ee2a-405b-a1e3-b77ca9e59b6e",
                  "txnRefNumber": "0002008600007160",
                  "tellerSeqNumber": null,
                  "overrideConfirmFlag": "N",
                  "supervisorId": null,
                  "onlineOfflineTxn": null,
                  "userComments": null,
                  "authoriserComments": null,
                  "eventCode": null,
                  "accountType": "UBS"
              },
              "dataPayload": {
                  "datasegment": null,
                  "fromAccountAmt": 100,
                  "fromAccountCcy": "GBP",
                  "toAccountCcy": "GBP",
                  "beneficiaryName": null,
                  "beneficiaryAddress1": null,
"beneficiaryAddress2": null,
"beneficiaryAddress3": null,
"beneficiaryAddress4": null,
"identificationType": null,
"identificationNumber": null,
"exchangeRate": 1,
"recievedAccountCcy": null,
"recievedAccountAmt": null,
"totalCha
rges":
null,
"cashAm
ount":
null,
"netAccountCcy": null,
"netAccountAmt": null,
"narrative": "Cash Deposit",
"txnControllerRefNo": null,
"recordId": "bd40562d-06b4-4f95-95fe-
e66fa6eb7f13", "cashAmtCcy": null,
"cashAmt":
null,
"chequeDate": null,
"chequeNumber": null,
"eventCode": null,
"ejId": null,
"emailId": null,
"fromAccountBranch": "000",
"fromAccountNumber": null,
"mobileNumber": null,
"orginalExchangeRate": null,
"payee": null,
"productCode": null,
"reversalDate": null,
"stationId": null,
"toAccountBranch": "000",
"toAccountNumber": "00000008010010",
"toAccountAmt": 100,
"txnBranchCode": "000",
"functionCode": null,
"txnCustomer": null,
"tellerId": null,
"txnDate": 1585161000000,
"txnRefNumber": "0002008600007160",
"txnSeqNumber": null,
"uniqueIdentifierNumber": null,
"uniqueIdentifierType": null,
"userRefNumber": null,
"valueDate": null,
"versionNumber": null,
"referenceNumber": null,
"createdBy": null,
"createdTs": null,
"updatedBy": null,
"updatedTs": null,
"demDtls": null,
                    "fxInDemDtls": null,
                    "fxOutDemDtls": null,
                    "prcDtls": null,
                    "addDtls": null,
                    "txnDtls": null,
                    "overrideDtls": null,
                    "batchTableDetails": null
          },
                    "extDetails": null,
                    "warDtls": [],
                    "authoriserDtls": []
          },
                    "errors": null,
                    "warnings": null,
                    "informations": null,
                    "authorizations": null,
                    "paging": ""
          }
```

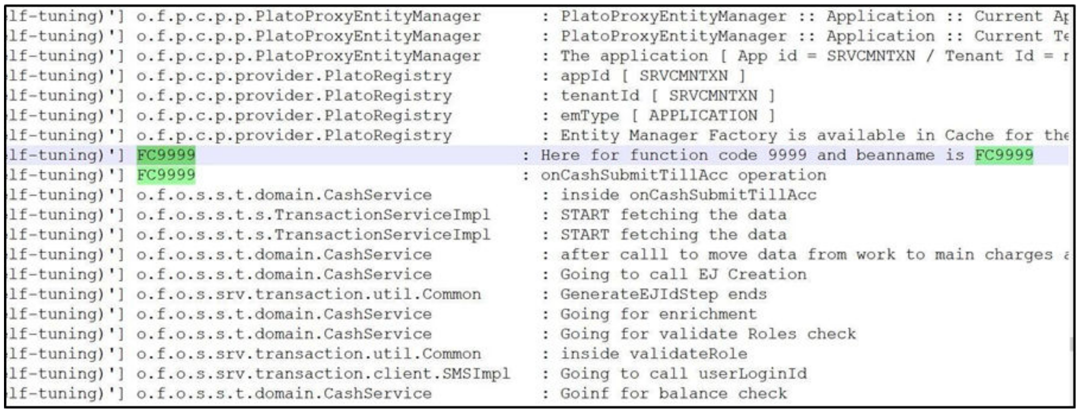
Figure 8-8 Common Core Additional Attributes


CD0Domain
Italy-branchcommon
Italy-CMNCore
CMC_TB_ADDT_ATTR_DATA
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:
Actions...

	UI_KEY	DATA_REF_KEY	FIELD_META_DATA_VER	FIELD_DATA
1	9811 fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}
2	5ed3 fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}
3	7c96 fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}
4	2826 fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}
5	632c fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}

- In the debug, the extensible code is used, which is present in the extension jar (obremo-srv-ext-commontxn.jar). Instead stempImpl onCashSubmitTillAcc, FC9999 onCashSubmitTillAcc will be called, where you can add code that is required for the new dataSegment added or to achieve different functionality of charging, accounting, till updates, etc

Figure 8-9 Debug Codes



```
lf-tuning)'] o.f.p.c.p.p.PlatoProxyEntityManager      : PlatoProxyEntityManager :: Application :: Current Ap
lf-tuning)'] o.f.p.c.p.p.PlatoProxyEntityManager      : PlatoProxyEntityManager :: Application :: Current Te
lf-tuning)'] o.f.p.c.p.p.PlatoProxyEntityManager      : The application [ App id = SRVCMNTXN / Tenant Id = r
lf-tuning)'] o.f.p.c.p.provider.PlatoRegistry         : appId [ SRVCMNTXN ]
lf-tuning)'] o.f.p.c.p.provider.PlatoRegistry         : tenantId [ SRVCMNTXN ]
lf-tuning)'] o.f.p.c.p.provider.PlatoRegistry         : emType [ APPLICATION ]
lf-tuning)'] o.f.p.c.p.provider.PlatoRegistry         : Entity Manager Factory is available in Cache for the
lf-tuning)'] FC9999                                 : Here for function code 9999 and beanname is FC9999
lf-tuning)'] FC9999                                 : onCashSubmitTillAcc operation
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : inside onCashSubmitTillAcc
lf-tuning)'] o.f.o.s.s.t.s.TransactionServiceImpl     : START fetching the data
lf-tuning)'] o.f.o.s.s.t.s.TransactionServiceImpl     : START fetching the data
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : after calll to move data from work to main charges a
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : Going to call EJ Creation
lf-tuning)'] o.f.o.s.srv.transaction.util.Common      : GenerateEJIdStep ends
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : Going for enrichment
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : Going for validate Roles check
lf-tuning)'] o.f.o.s.srv.transaction.util.Common      : inside validateRole
lf-tuning)'] o.f.o.s.srv.transaction.client.SMSImpl   : Going to call userLoginId
lf-tuning)'] o.f.o.s.s.t.domain.CashService           : Goinf for balance check
```

9

Extensibility Use Cases for OBX

This topic provides the systematic instructions to perform the basic operations on the Extensibility Use Cases for OBX.

This topic describes the following sub-topics:

- New Transaction screen – 1499 (Clone of 1401)
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- New Data Segment in Existing 1401 Screen
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- HTML Changes
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- JS Changes
  This topic provides the systematic instructions to JS fields.
- JSON Changes
  This topic describes the changes JSON fields across all the screens.
- Model Changes
  This topic provides the systematic instructions to Model Changes.
- Database Changes
  This topic provides the systematic instructions to Database Changes.
- Service Component
  This topic provides the systematic instructions to the Service Component.
- New Field in Existing Base Data Segment
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- HTML Changes (Extended Components)
  This topic describes the changes Extended Component HTML fields across all the screens.
- HTML Changes (Base Component)
  This topic describes the bade components HTML fields changes for all the screens.
- JS Changes (Base Component)
  This topic describes the base components JS fields changes for all the screens.
- JS Changes (Extended Component)
  This topic describes the extended components JS fields changes for all the screens.
- JSON Changes (Extended Component)
  This topic describes the extended components JSON fields changes for all the screens.
- JSON Changes (Base Component)
  This topic describes the base components JSON fields changes for all the screens.

- DB Changes
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- Add New Columns in Base Component Table
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- Steps for adding extra column in task grid
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- Steps to use Additional Buttons provision in Task Screen
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- Steps to create common-extended folder for extending configJSON.js file
  This topic provides the systematic instructions to perform the basic operations on common-extended folder for extending configJSON.js file.
- Customizing Existing LOV Fetch Result
  This topic provides the systematic instructions to perform the basic operations on the Customizing Existing LOV Fetch Result.
- Steps for adding Pre/post methods in extended components
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- ENDPOINT Overrides
  This topic describes the endpoint overrides.
- Steps to create util-extended folder
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- Dynamic Data Configuration (DDC)
  This topic provides the systematic instructions to perform the basic operations on the selected records.
- Task Screen Custom Config
  This topic provides the systematic instructions to perform the basic operations on the selected records.

9.1 New Transaction screen – 1499 (Clone of 1401)

This topic provides the systematic instructions to perform the basic operations on the selected records.

For this use case, make sure that the data is present in the below tables similar to 1401. The below mentioned tables need to be checked in SMS schema:

- SMS_TM_MENU
- SMS_TM_MENU_Description
- SMS_TM_SERVICE_ACTIVITY
- SMS_TM_FUNCTIONAL_ACTIVITY
- SMS_TM_FUNC_ACTIVITY_DETAIL
- SMS_TM_ROLE_ACTIVITY
- SMS_TM_UI_ACTIVITY

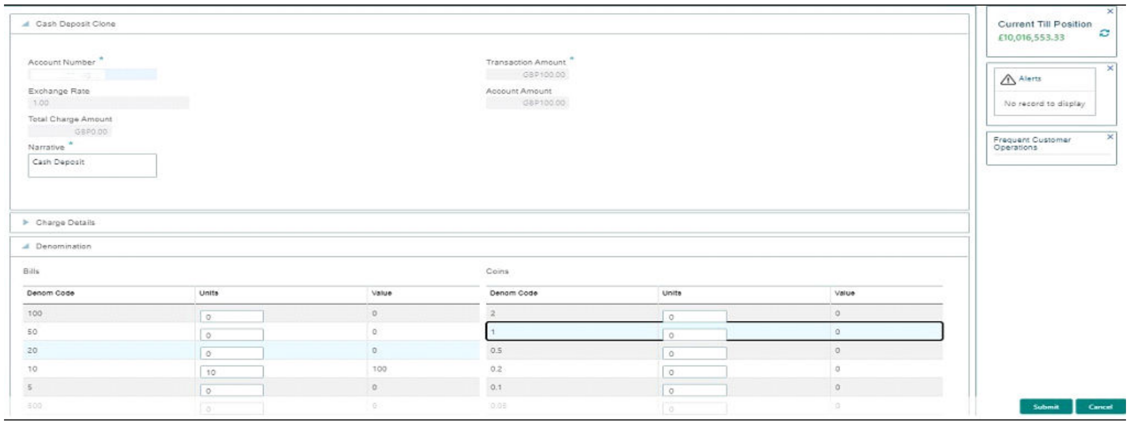
The below mentioned tables need to be checked in Common Core schema:

- CMC_TM_SCREEN_CLASS
- CMC_TM_SCREEN_DS_MAPPING

The below mentioned tables need to be checked in branch Common schema:

- SRV_TM_BC_FUNCTION_INDICATOR
- SRV_TM_BC_FUNCTION_CODE
- SRV_TM_BC_FUNCTION_PREF
- SRV_TM_BC_FUNCTION_PREF_DTLS
- SRV_TM_BC_BRANCH_ACCOUNTING
- SRV_TM_MENU_CONFIG

Figure 9-1 Cash Deposit Clone



Cash Deposit Clone

Account Number *

Exchange Rate: 1.00

Total Charge Amount: GBP0.00

Narrative *: Cash Deposit

Transaction Amount *: GBP100.00

Account Amount: GBP100.00

Charge Details

Denomination

Bills

Denom Code	Units	Value
100	0	0
50	0	0
20	0	0
10	10	100
5	0	0
500	0	0

Coins

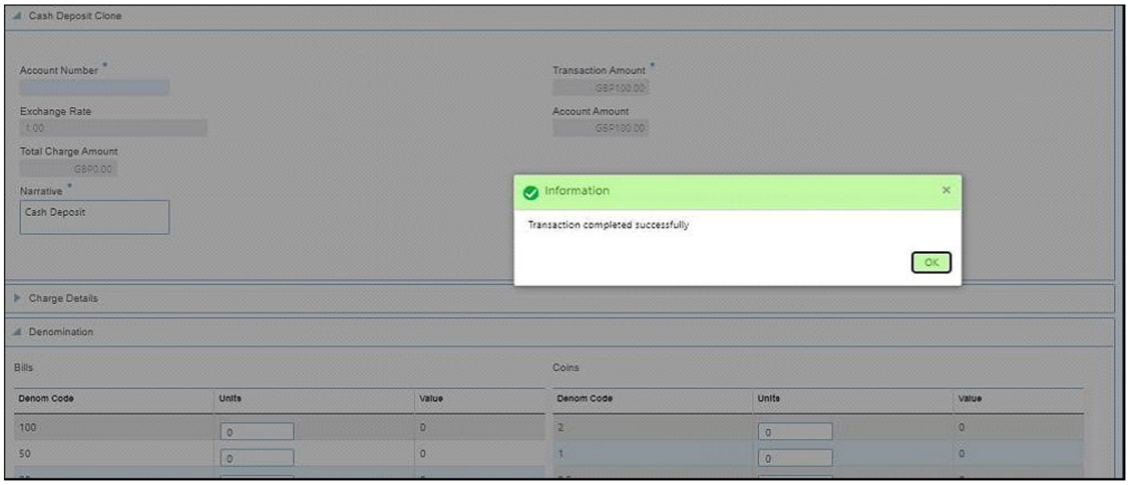
Denom Code	Units	Value
2	0	0
1	0	0
0.5	0	0
0.2	0	0
0.1	0	0
0.05	0	0

Current Till Position
£10,016,553.33

Alerts
No record to display

Frequent Customer Operations

Figure 9-2 Information Message



Cash Deposit Clone

Account Number *

Exchange Rate: 1.00

Total Charge Amount: GBP0.00

Narrative *: Cash Deposit

Transaction Amount *: GBP100.00

Account Amount: GBP100.00

Information

Transaction completed successfully

Charge Details

Denomination

Bills

Denom Code	Units	Value
100	0	0
50	0	0

Coins

Denom Code	Units	Value
2	0	0
1	0	0

9.2 New Data Segment in Existing 1401 Screen

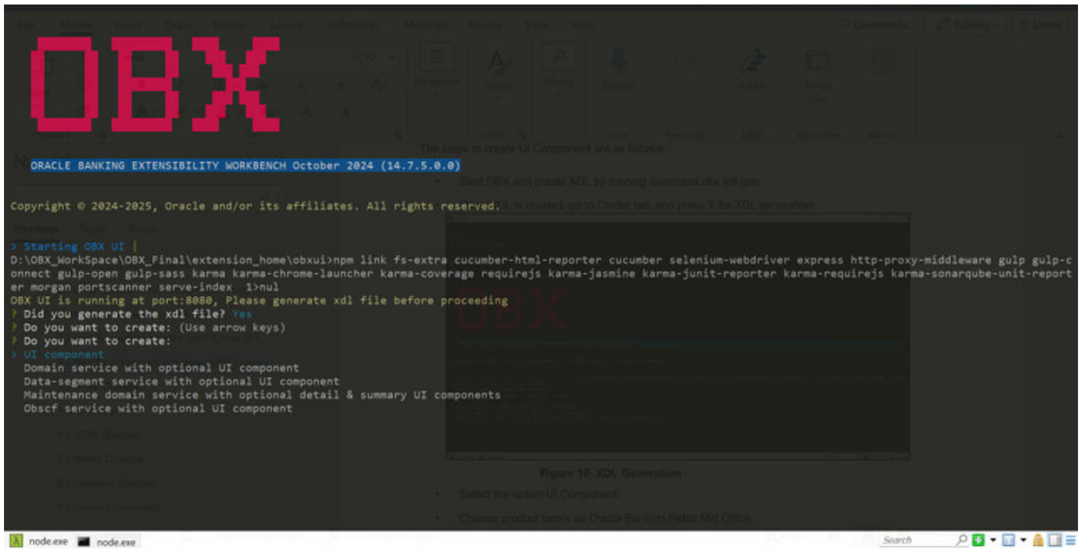
This topic provides the systematic instructions to perform the basic operations on the selected records.

For this use case, it is needed to implement UI Component and Service side to persist data.

The steps to create UI Component are as follows:

1. Start OBX and create XDL by running command **obx xdl-gen**.
2. Once XDL is created, go to Cmder tab, and press Y for XDL generation.

Figure 9-3 OBX XDL generation



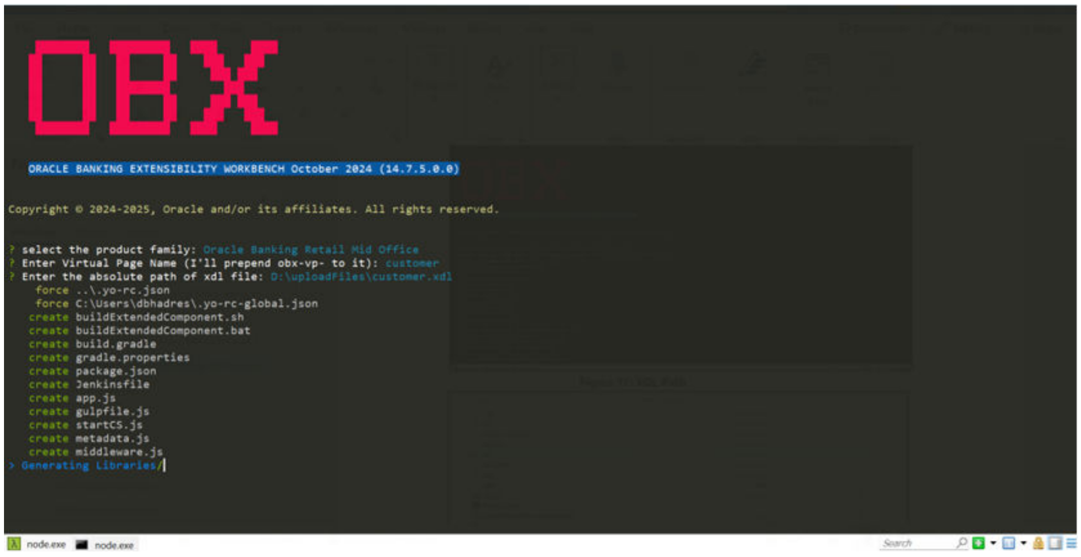
```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

> Starting OBX UI |
D:\OBX_WorkSpace\OBX_Final\extension_home\obxui>npm link fs-extra cucumber-html-reporter cucumber selenium-webdriver express http-proxy-middleware gulp gulp-c
onnect gulp-open gulp-sass karma karma-chrome-launcher karma-coverage requirejs karma-jasmine karma-junit-reporter karma-requirejs karma-sonarqube-unit-report
er morgan portscanner serve-index  1>nul
OBX UI is running at port:8080, Please generate xdl file before proceeding
? Did you generate the xdl file? Yes
? Do you want to create: (Use arrow keys)
? Do you want to create:
> UI component
  Domain service with optional UI component
  Data-segment service with optional UI component
  Maintenance domain service with optional detail & summary UI components
  Obscf service with optional UI component
```

3. Select the option UI Component.
4. Select product family as Oracle Banking Retail Mid Office.
5. Specify the name of virtual page/data-segment/stand-alone component to be created.
6. Specify absolute path of the XDL generated. (XDL is generated inside extension_home folder).

Note:

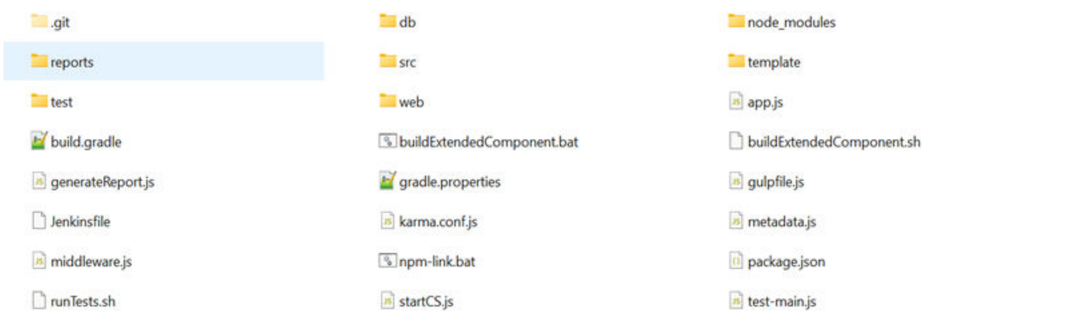
A new UI Component will be created in extension_home folder with prefix obx-vp/obx-ds. In the Cmder tab, OBX will prompt to modify Metadata.js file of the newly created component. In addition, the component-server will start running at port 8002.

Figure 9-4 XDL Path



```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? select the product family: Oracle Banking Retail Mid Office
? Enter Virtual Page Name (I'll prepend obx-vp- to it): customer
? Enter the absolute path of xdl file: D:\uploadFiles\customer.xdl
    force ..\.yo-rc.json
    force C:\Users\dbhadres\.yo-rc-global.json
   create buildExtendedComponent.sh
   create buildExtendedComponent.bat
   create build.gradle
   create gradle.properties
   create package.json
   create Jenkinsfile
   create app.js
   create gulpfile.js
   create startCS.js
   create metadata.js
   create middleware.js
> Generating Libraries/
```

Figure 9-5 Extension Home Folder



.git	db	node_modules
reports	src	template
test	web	app.js
build.gradle	buildExtendedComponent.bat	buildExtendedComponent.sh
generateReport.js	gradle.properties	gulpfile.js
Jenkinsfile	karma.conf.js	metadata.js
middleware.js	npm-link.bat	package.json
runTests.sh	startCS.js	test-main.js

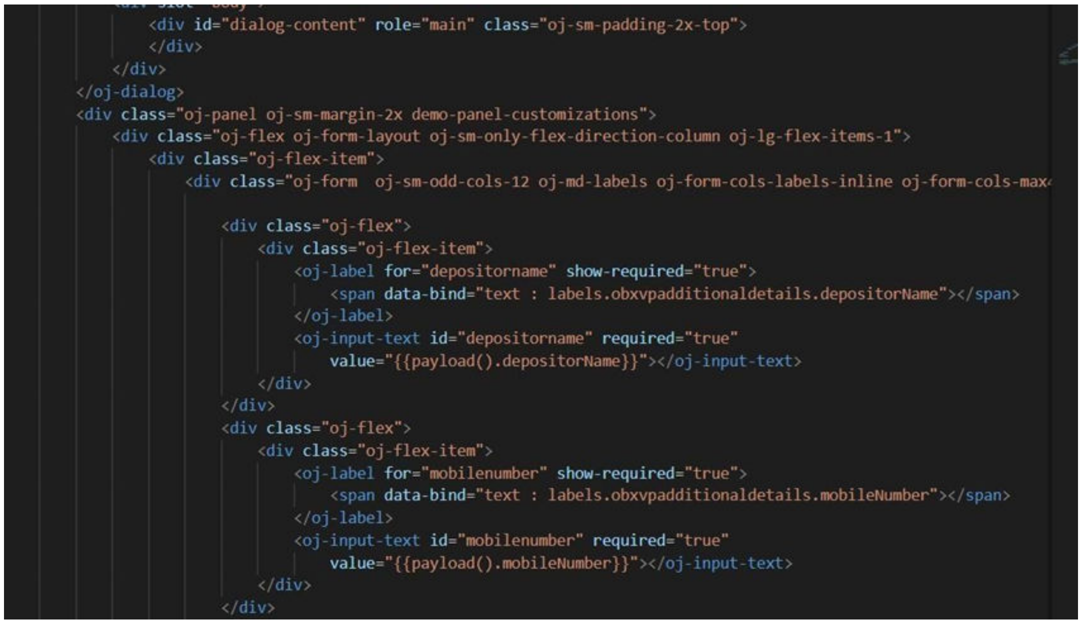
7. The generated UI component contains boiler plate code to do the common operations of Save, Get, Get All etc. Changes needed in the newly created component from OBX tool from UI side.

9.3 HTML Changes

This topic provides the systematic instructions to perform the basic operations on the selected records.

- According to the screen design, one can change the HTML values like payload() and mobileNumber. If mobileNumber field is entered by the user, value of mobileNumber will directly update the JS payload that will be going as a part of save call.

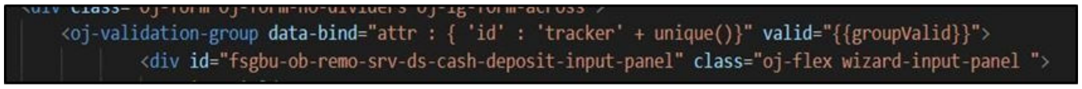
Figure 9-6   HTML Changes



```
        <div id="dialog-content" role="main" class="oj-sm-padding-2x-top">
        </div>
      </div>
    </oj-dialog>
    <div class="oj-panel oj-sm-margin-2x demo-panel-customizations">
      <div class="oj-flex oj-form-layout oj-sm-only-flex-direction-column oj-lg-flex-items-1">
        <div class="oj-flex-item">
          <div class="oj-form  oj-sm-odd-cols-12 oj-md-labels oj-form-cols-labels-inline oj-form-cols-max

            <div class="oj-flex">
              <div class="oj-flex-item">
                <oj-label for="depositorname" show-required="true">
                  <span data-bind="text : labels.obxvpadditionaldetails.depositorName"></span>
                </oj-label>
                <oj-input-text id="depositorname" required="true"
                  value="{{payload().depositorName}}"></oj-input-text>
              </div>
            </div>
            <div class="oj-flex">
              <div class="oj-flex-item">
                <oj-label for="mobilenumber" show-required="true">
                  <span data-bind="text : labels.obxvpadditionaldetails.mobileNumber"></span>
                </oj-label>
                <oj-input-text id="mobilenumber" required="true"
                  value="{{payload().mobileNumber}}"></oj-input-text>
              </div>
            </div>
```

- The oj-validation-group is required for configuring the HTML as part of validation.

Figure 9-7   Validation



```
  <oj-validation-group data-bind="attr : { 'id' : 'tracker' + unique()}" valid="{{groupValid}}">
      <div id="fsgbu-ob-remo-srv-ds-cash-deposit-input-panel" class="oj-flex wizard-input-panel ">
```

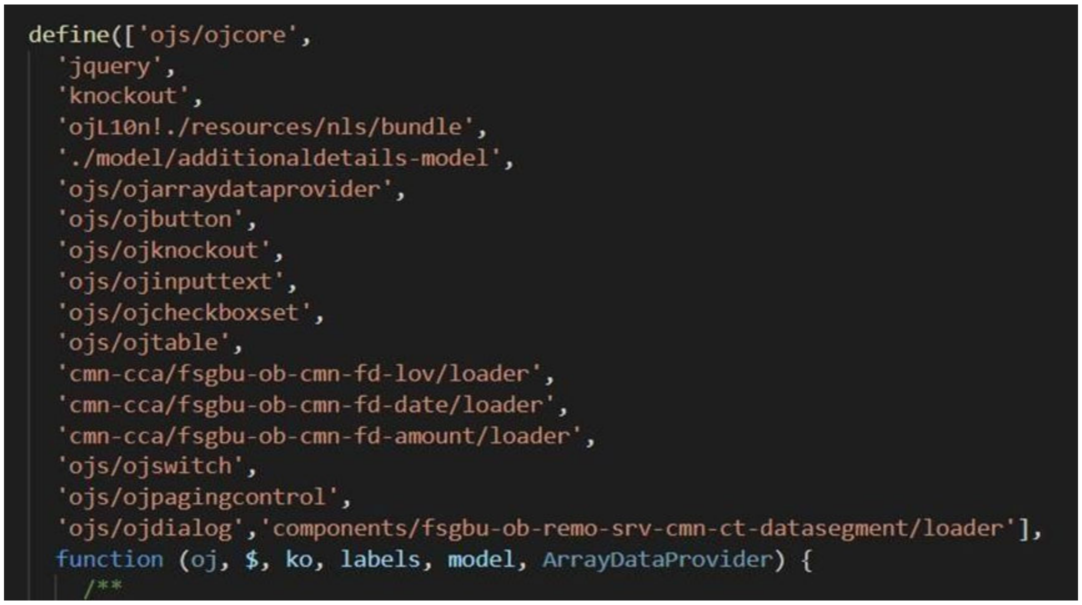
9.4 JS Changes

This topic provides the systematic instructions to JS fields.

Perform the following steps to implement JS changes:

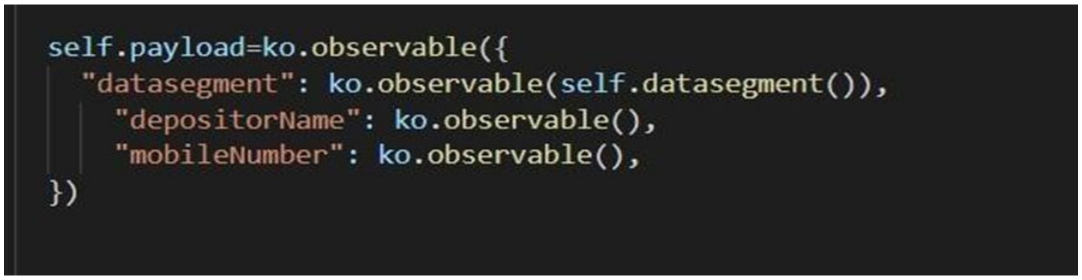
1. Add all the dependencies in define block.
2. The JS **self.payload** is an observable, which will hold all the info inputted from the HTML. All keys in **self.payload** is directly linked with HTML.

Figure 9-8   JS Changes



```
define(['ojs/ojcore',
  'jquery',
  'knockout',
  'ojL10n!./resources/nls/bundle',
  './model/additionaldetails-model',
  'ojs/ojarraydataprovider',
  'ojs/ojbutton',
  'ojs/ojknockout',
  'ojs/ojinputtext',
  'ojs/ojcheckboxset',
  'ojs/ojtable',
  'cmn-cca/fsgbu-ob-cmn-fd-lov/loader',
  'cmn-cca/fsgbu-ob-cmn-fd-date/loader',
  'cmn-cca/fsgbu-ob-cmn-fd-amount/loader',
  'ojs/ojswitch',
  'ojs/ojpagingcontrol',
  'ojs/ojdialog','components/fsgbu-ob-remo-srv-cmn-ct-datasegment/loader'],
  function (oj, $, ko, labels, model, ArrayDataProvider) {
    /**
```

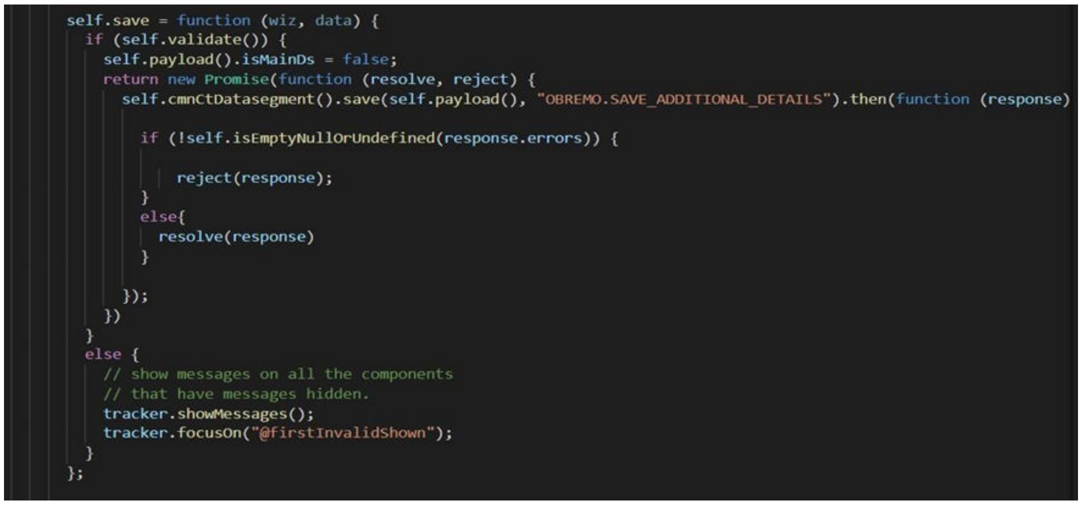
Figure 9-9   JS Self Payload



```
self.payload=ko.observable({
  "datasegment": ko.observable(self.datasegment()),
    "depositorName": ko.observable(),
    "mobileNumber": ko.observable(),
})
```

3. Save method implementation will look like in below figure. In the next line, it is making a promise and calling the save function of cmn-ct-datasegment providing the payload and endpoint as parameters. If save is success, it will resolve and for failures it will come to reject.

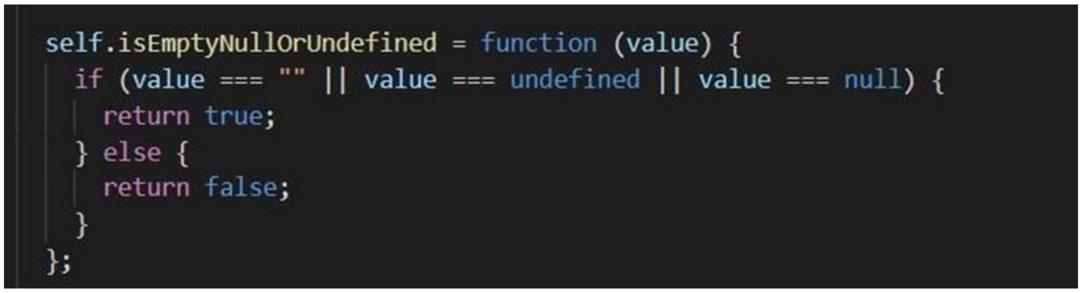
Figure 9-10   Save Method



```
self.save = function (wiz, data) {
  if (self.validate()) {
    self.payload().isMainDs = false;
    return new Promise(function (resolve, reject) {
      self.cmnCtDatasegment().save(self.payload(), "OBREMO.SAVE_ADDITIONAL_DETAILS").then(function (response)

        if (!self.isEmptyNullOrUndefined(response.errors)) {

          reject(response);
        }
        else{
          resolve(response)
        }

      });
    })
  }
  else {
    // show messages on all the components
    // that have messages hidden.
    tracker.showMessages();
    tracker.focusOn("@firstInvalidShown");
  }
};
```

4. The function null check is as shown below:

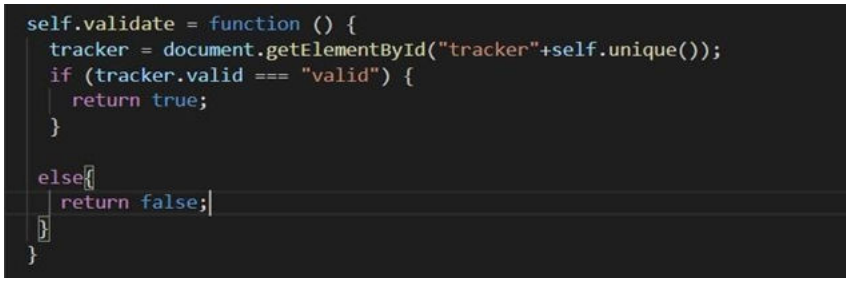
Figure 9-11   Function Null Check



```
self.isEmptyNullOrUndefined = function (value) {
  if (value === "" || value === undefined || value === null) {
    return true;
  } else {
    return false;
  }
};
```

5. The validate function is shown in the below mentioned validate function screen, which will check all mandatory fields during save.

Figure 9-12   Validate Function



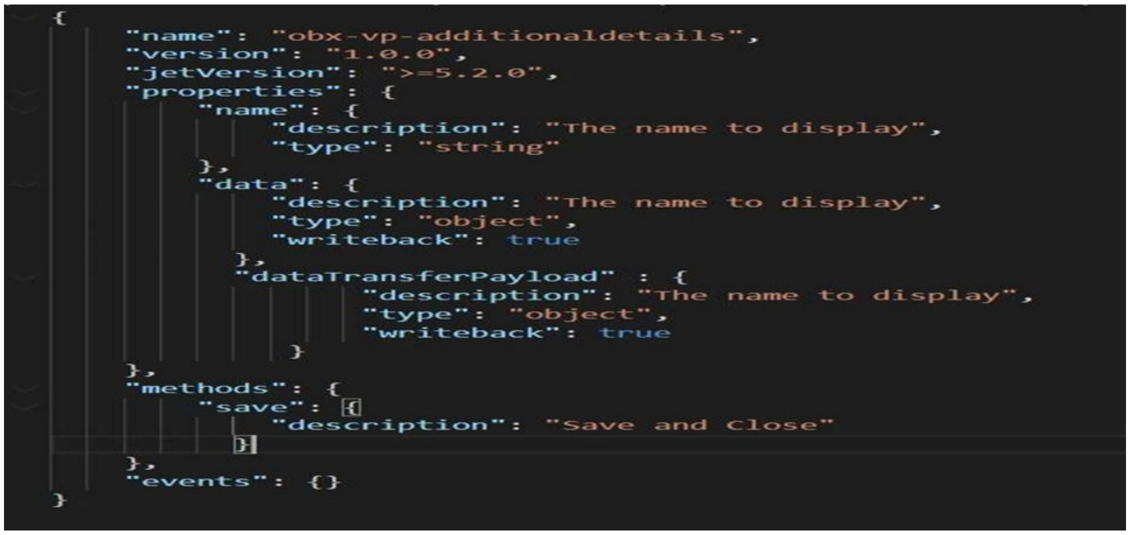
```
self.validate = function () {
  tracker = document.getElementById("tracker"+self.unique());
  if (tracker.valid === "valid") {
    return true;
  }

 else{
  return false;
 }
}
```

9.5 JSON Changes

This topic describes the changes JSON fields across all the screens.

The data and datatransferPayload properties need to be exposed from JSON. The data property is used to take the information of transaction specific and the datatransferPayload property is used to share data between data segments.

Figure 9-13 JSON Changes



```
{
    "name": "obx-vp-additionaldetails",
    "version": "1.0.0",
    "jetVersion": ">=5.2.0",
    "properties": {
        "name": {
            "description": "The name to display",
            "type": "string"
        },
        "data": {
            "description": "The name to display",
            "type": "object",
            "writeback": true
          },
          "dataTransferPayload" : {
               "description": "The name to display",
               "type": "object",
               "writeback": true
             }
    },
    "methods": {
        "save": {
            "description": "Save and Close"
          }
    },
    "events": {}
}
```

9.6 Model Changes

This topic provides the systematic instructions to Model Changes.

There will be no methods in the model. All the REST calls needs to go through cmn-ct-datasegment similar to Save.

Perform the following steps to make model changes:

1. Run the DB Scripts present in this component.

> **Note:**
>
> he OBX generates SQL script with default HEADER_APPID as PXDSSRV001 for all components. This script can be changed and used

2. Create extended war for the component and deploy.

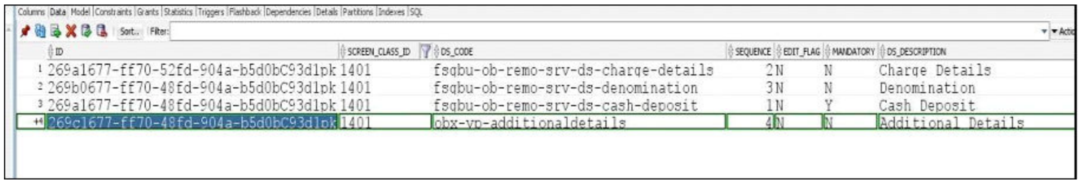
9.7 Database Changes

This topic provides the systematic instructions to Database Changes.

To add database changes to do the following:

1. Add the newly created data segment name in the **PRODUCT_EXTENDED_LEDGER** table (this will be done when DB script from UI component is run).
2. Make a fourth Data Segment entry for function code 1401 in **CMC_TM_SCREEN_DS_MAPPING** table of **CMNCORE**. The **DS_CODE** should be the name of the UI Component created. The entry is as shown in the Data Segment Entry.

Figure 9-14 Data Segment Entry



Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

	ID	SCREEN_CLASS_ID	DS_CODE	SEQUENCE	EDIT_FLAG	MANDATORY	DS_DESCRIPTION
1	269a1677-ff70-52fd-904a-b5d0bC93d1pk	1401	fsgbu-ob-remo-srv-ds-charge-details	2	N	N	Charge Details
2	269b0677-ff70-48fd-904a-b5d0bC93d1pk	1401	fsgbu-ob-remo-srv-ds-denomination	3	N	N	Denomination
3	269a1677-ff70-48fd-904a-b5d0bC93d1pk	1401	fsgbu-ob-remo-srv-ds-cash-deposit	1	N	Y	Cash Deposit
+4	269c1677-ff70-48fd-904a-b5d0bC93d1pk	1401	obx-vp-additionaldetails	4	N	N	Additional Details

3. If the service is created separately than UI Component, change the endpoint URL in SQL script for table PRODUCT_SERVICES_LEDGER accordingly.

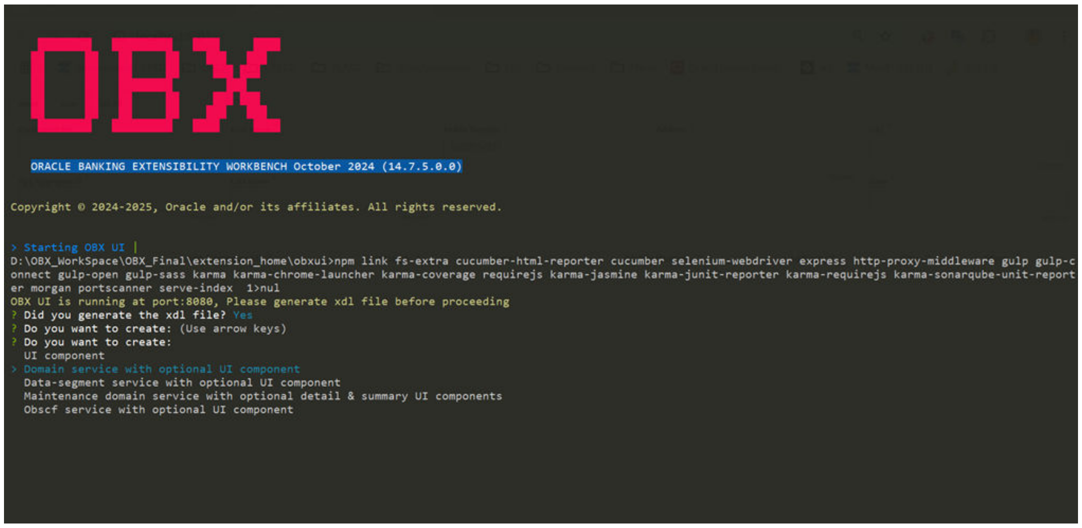
9.8 Service Component

This topic provides the systematic instructions to the Service Component.

To create a service component do the following:

1. Start OBX and use the **XDL file** that is already generated.
2. Select the domain service with optional UI component.

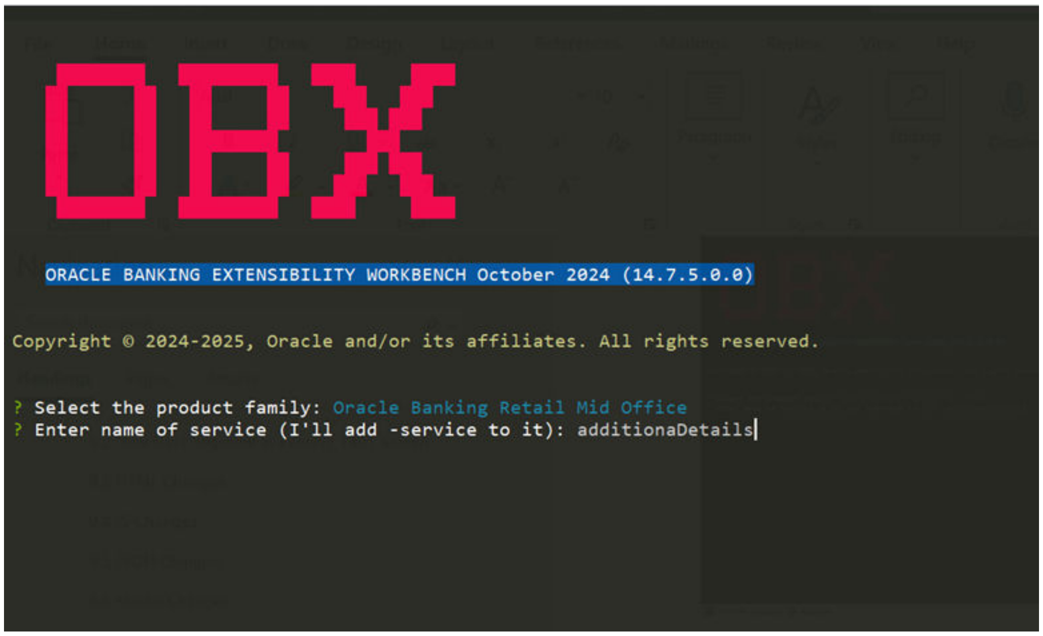
Figure 9-15 Domain Service



```
OBX

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

> Starting OBX UI |
D:\OBX_WorkSpace\OBX_Final\extension_home\obxui>npm link fs-extra cucumber-html-reporter cucumber selenium-webdriver express http-proxy-middleware gulp gulp-c
onnect gulp-open gulp-sass karma karma-chrome-launcher karma-coverage requirejs karma-jasmine karma-junit-reporter karma-requirejs karma-sonarqube-unit-report
er morgan portscanner serve-index  1>nul
OBX UI is running at port:8080, Please generate xdl file before proceeding
? Did you generate the xdl file? Yes
? Do you want to create: (Use arrow keys)
? Do you want to create:
  UI component
> Domain service with optional UI component
  Data-segment service with optional UI component
  Maintenance domain service with optional detail & summary UI components
  Obscf service with optional UI component
```

3. Select product family as **Oracle Banking Retail Mid Office**.

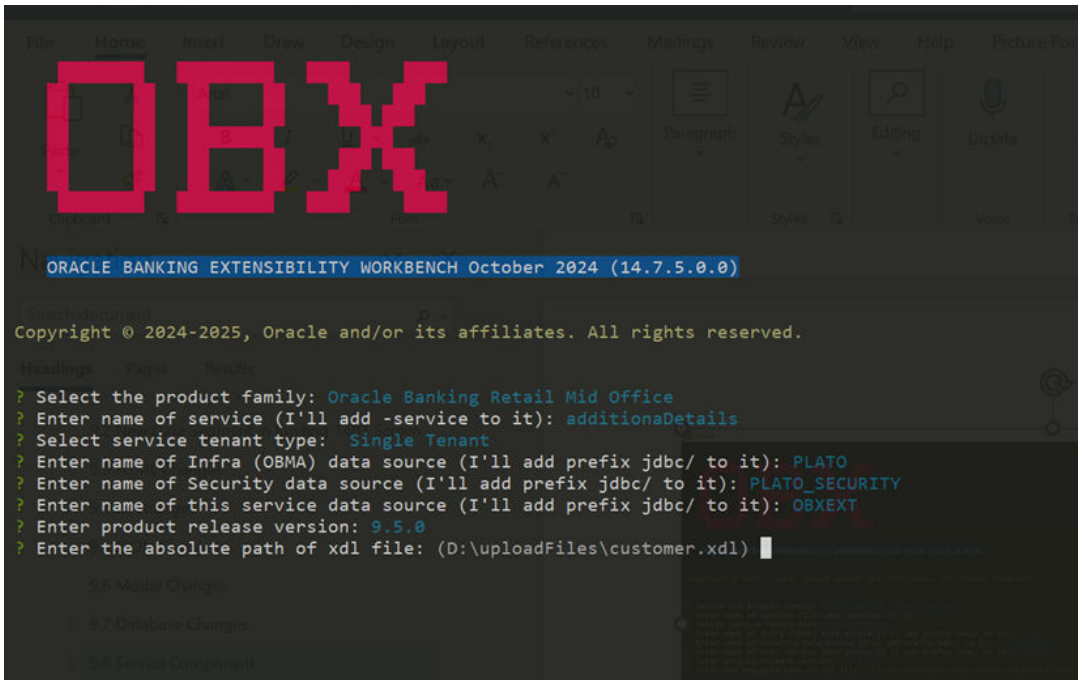
Figure 9-16 Product Family



```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Select the product family: Oracle Banking Retail Mid Office
? Enter name of service (I'll add -service to it): additionaDetails
```

4. Specify the service name as additional Details and all the remaining details as mentioned in the service name screen.

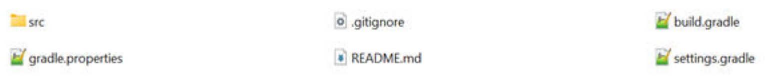
Figure 9-17 Service Name



```
ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Select the product family: Oracle Banking Retail Mid Office
? Enter name of service (I'll add -service to it): additionaDetails
? Select service tenant type:  Single Tenant
? Enter name of Infra (OBMA) data source (I'll add prefix jdbc/ to it): PLATO
? Enter name of Security data source (I'll add prefix jdbc/ to it): PLATO_SECURITY
? Enter name of this service data source (I'll add prefix jdbc/ to it): OBXEXT
? Enter product release version: 9.5.0
? Enter the absolute path of xdl file: (D:\uploadFiles\customer.xdl)
```

5. A new service is generated in **extension_home** folder with prefix **obremo-additionadetails-service**.

Figure 9-18 Extension Home Folder



6. Run the DB scripts present in this service.

> **Note:**
>
> It will create a new table to persist data of new data segment. For example, a table is created as ADDITIONALDETAILS. This table can be created in existing schema or in a new schema.

7. If you need to create a new schema, mention that in table. PRODUCT_SERVICES_CTX_LEDGER while running UI Component Script.
8. Restart plato servers once this change is completed.
9. If required, make appropriate changes in the service, build it, and deploy.

> **Note:**
>
> After deploying extended war and additional details service along with proper DB entry, you can see a new data segment in the appshell screen.

10. Fill the necessary details and click Submit, the data for new DS will be saved in new table.

Figure 9-19 Additional Details Segment

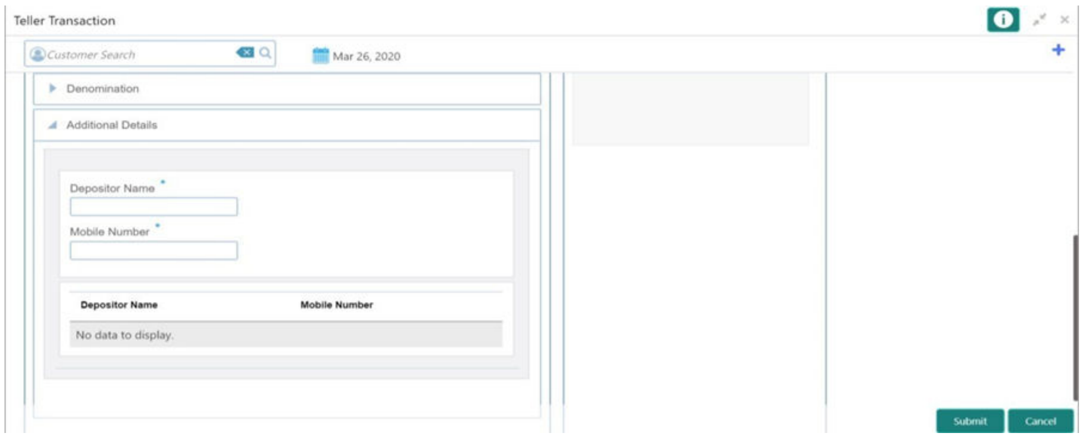
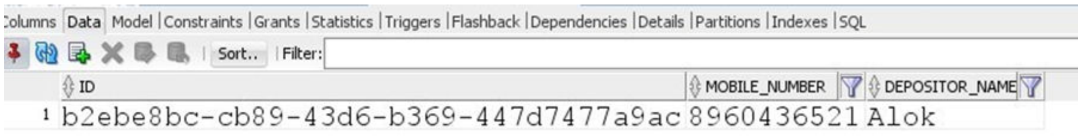


Figure 9-20   Updated Data in New Table


Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

	ID	MOBILE_NUMBER	DEPOSITOR_NAME
1	b2ebe8bc-cb89-43d6-b369-447d7477a9ac	8960436521	Alok

9.9 New Field in Existing Base Data Segment

This topic provides the systematic instructions to perform the basic operations on the selected records.

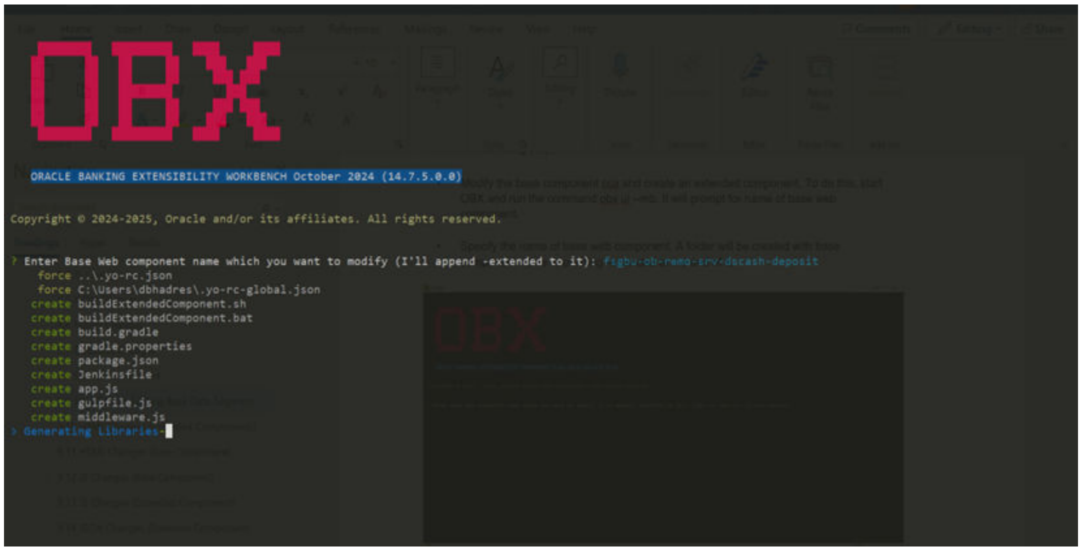
This use case defines a new field in the existing base data segment (fsgbu-ob-remo-srv-ds-cash-deposit) in 1401 screen class.

For this use case, you need to create an extended UI Component, make changes in the existing UI appshell, and make changes in the service.

Perform the following steps:

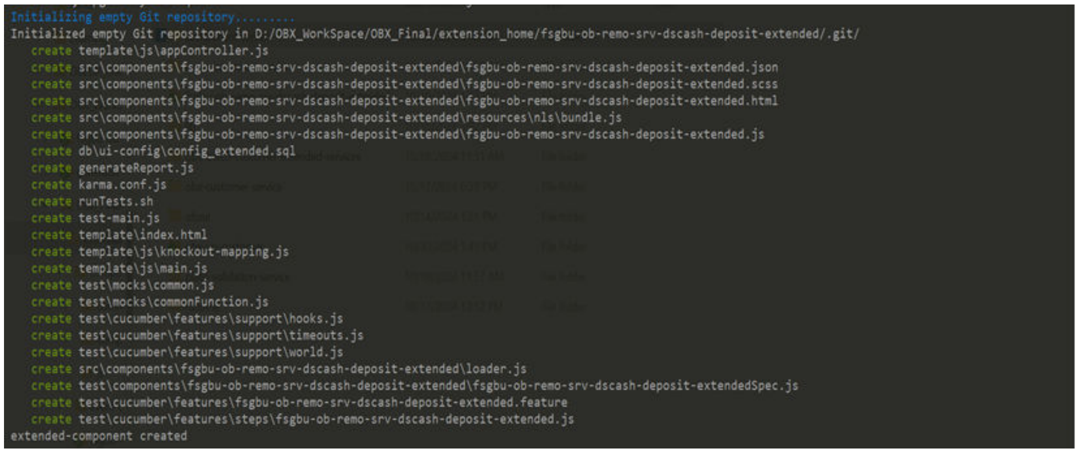
1. Modify the base component cca and create an extended component. To do this, start OBX and run the command obx ui --mb. It will prompt for name of base web component.
2. Specify the name of base web component. A folder will be created with base component name appending -extended at the end of it.

Figure 9-21   Base Web Component



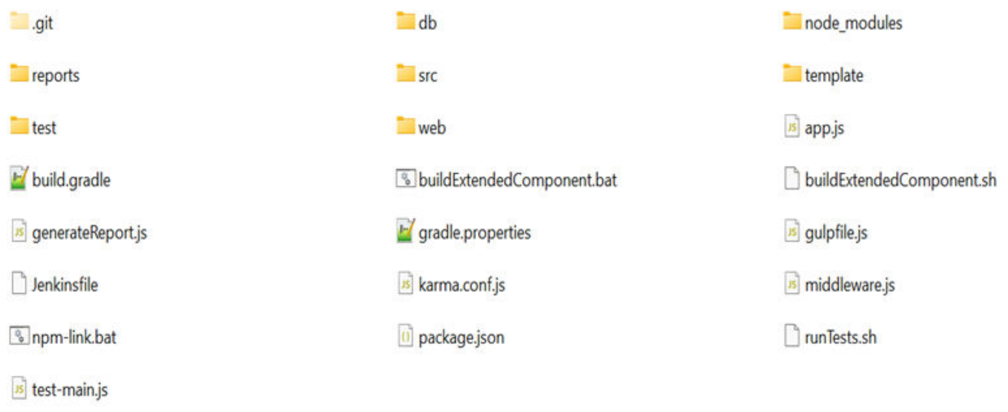
```
OBX

ORACLE BANKING EXTENSIBILITY WORKBENCH October 2024 (14.7.5.0.0)

Copyright © 2024-2025, Oracle and/or its affiliates. All rights reserved.

? Enter Base Web component name which you want to modify (I'll append -extended to it): fsgbu-ob-remo-srv-dscash-deposit
   force ..\.yo-rc.json
   force C:\Users\dbhadres\.yo-rc-global.json
  create buildExtendedComponent.sh
  create buildExtendedComponent.bat
  create build.gradle
  create gradle.properties
  create package.json
  create Jenkinsfile
  create app.js
  create gulpfile.js
  create middleware.js
> Generating Libraries-
```

Figure 9-22   Base Web Component



```
Initializing empty Git repository.........
Initialized empty Git repository in D:/OBX_WorkSpace/OBX_Final/extension_home/fsgbu-ob-remo-srv-dscash-deposit-extended/.git/
   create template\js\appController.js
   create src\components\fsgbu-ob-remo-srv-dscash-deposit-extended\fsgbu-ob-remo-srv-dscash-deposit-extended.json
   create src\components\fsgbu-ob-remo-srv-dscash-deposit-extended\fsgbu-ob-remo-srv-dscash-deposit-extended.scss
   create src\components\fsgbu-ob-remo-srv-dscash-deposit-extended\fsgbu-ob-remo-srv-dscash-deposit-extended.html
   create src\components\fsgbu-ob-remo-srv-dscash-deposit-extended\resources\nls\bundle.js
   create src\components\fsgbu-ob-remo-srv-dscash-deposit-extended\fsgbu-ob-remo-srv-dscash-deposit-extended.js
   create db\ui-config\config_extended.sql
   create generateReport.js
   create karma.conf.js
   create runTests.sh
   create test-main.js
   create template\index.html
   create template\js\knockout-mapping.js
   create template\js\main.js
   create test\mocks\common.js
   create test\mocks\commonFunction.js
   create test\cucumber\features\support\hooks.js
   create test\cucumber\features\support\timeouts.js
   create test\cucumber\features\support\world.js
   create src\components\fsgbu-ob-remo-srv-dscash-deposit-extended\loader.js
   create test\components\fsgbu-ob-remo-srv-dscash-deposit-extended\fsgbu-ob-remo-srv-dscash-deposit-extendedSpec.js
   create test\cucumber\features\fsgbu-ob-remo-srv-dscash-deposit-extended.feature
   create test\cucumber\features\steps\fsgbu-ob-remo-srv-dscash-deposit-extended.js
extended-component created
```

Figure 9-23   Extended Folder



.git	db	node_modules
reports	src	template
test	web	app.js
build.gradle	buildExtendedComponent.bat	buildExtendedComponent.sh
generateReport.js	gradle.properties	gulpfile.js
Jenkinsfile	karma.conf.js	middleware.js
npm-link.bat	package.json	runTests.sh
test-main.js		

Note:

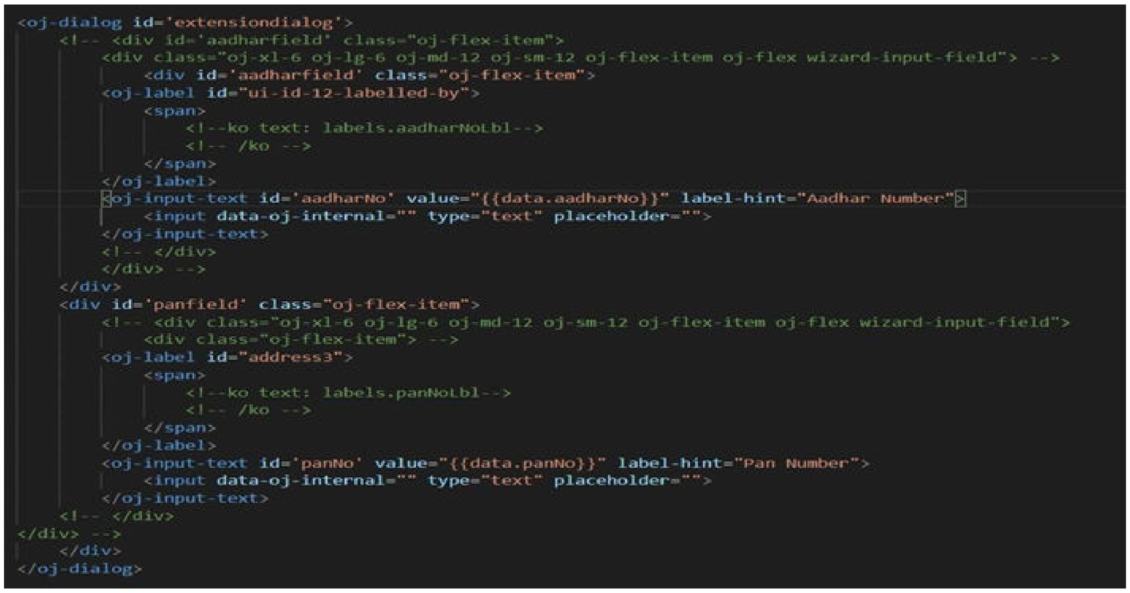
The changes are required in the extended component from the UI side.

9.10 HTML Changes (Extended Components)

This topic describes the changes Extended Component HTML fields across all the screens.

The extended component contains the boiler plate codes, in which you need to make the changes as shown in the below HTML Changes (Extended Component) screen. After you make the necessary changes, the additional fields will be added after the existing fields in the base component.

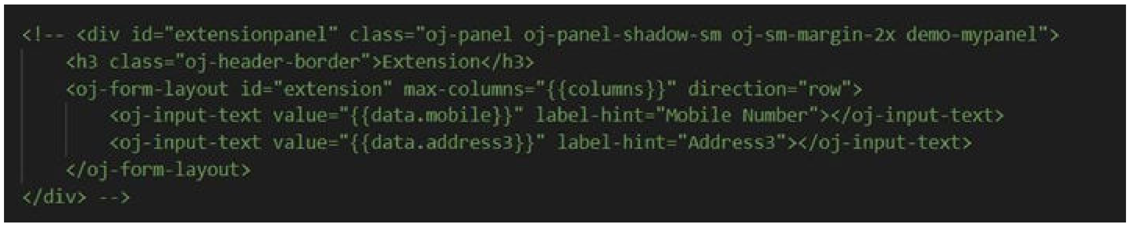
Figure 9-24   HTML Changes (Extended Component)



```
<oj-dialog id='extensiondialog'>
    <!-- <div id='aadharfield' class="oj-flex-item">
        <div class="oj-xl-6 oj-lg-6 oj-md-12 oj-sm-12 oj-flex-item oj-flex wizard-input-field"> -->
            <div id='aadharfield' class="oj-flex-item">
        <oj-label id="ui-id-12-labelled-by">
            <span>
                <!--ko text: labels.aadharNoLbl-->
                <!-- /ko -->
            </span>
        </oj-label>
        <oj-input-text id='aadharNo' value="{{data.aadharNo}}" label-hint="Aadhar Number">
            <input data-oj-internal="" type="text" placeholder="">
        </oj-input-text>
        <!-- </div>
        </div> -->
    </div>
    <div id='panfield' class="oj-flex-item">
        <!-- <div class="oj-xl-6 oj-lg-6 oj-md-12 oj-sm-12 oj-flex-item oj-flex wizard-input-field">
            <div class="oj-flex-item"> -->
        <oj-label id="address3">
            <span>
                <!--ko text: labels.panNoLbl-->
                <!-- /ko -->
            </span>
        </oj-label>
        <oj-input-text id='panNo' value="{{data.panNo}}" label-hint="Pan Number">
            <input data-oj-internal="" type="text" placeholder="">
        </oj-input-text>
    <!-- </div>
</div> -->
    </div>
</oj-dialog>
```

The following changes are required only if you need to add the additional field at the end of the base component and in a separate extension panel. You can choose to add the additional fields in the existing base component or in the extension panel as per the requirement.

Figure 9-25   Extension Panel



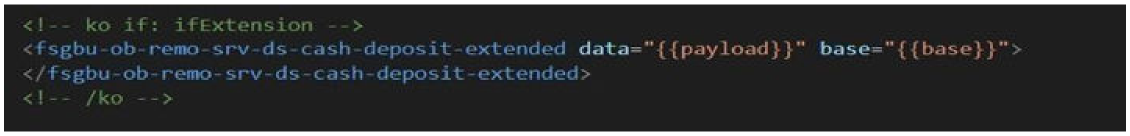
```
<!-- <div id="extensionpanel" class="oj-panel oj-panel-shadow-sm oj-sm-margin-2x demo-mypanel">
    <h3 class="oj-header-border">Extension</h3>
    <oj-form-layout id="extension" max-columns="{{columns}}" direction="row">
        <oj-input-text value="{{data.mobile}}" label-hint="Mobile Number"></oj-input-text>
        <oj-input-text value="{{data.address3}}" label-hint="Address3"></oj-input-text>
    </oj-form-layout>
</div> -->
```

9.11 HTML Changes (Base Component)

This topic describes the bade components HTML fields changes for all the screens.

Perform the HTML changes in the base component.

Figure 9-26   HTML Changes (Base Component)



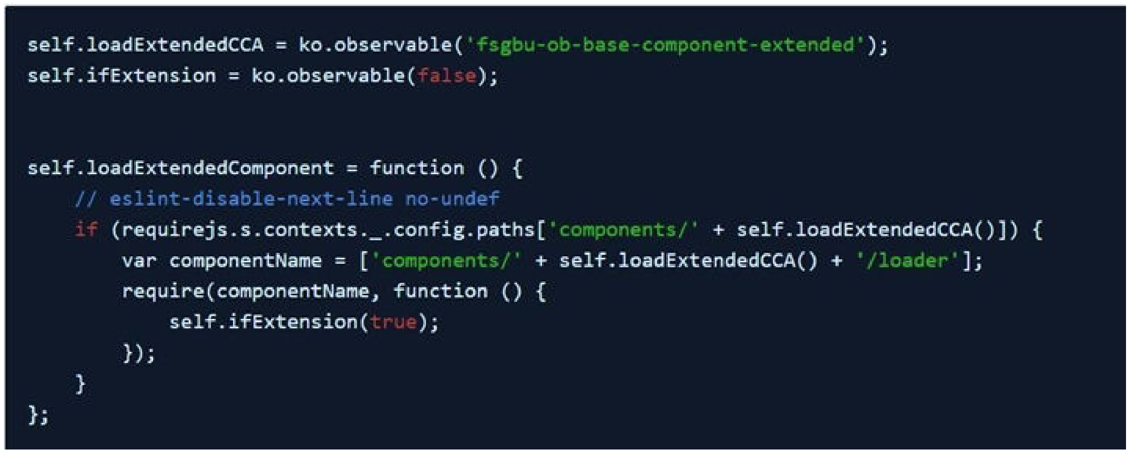
```
<!-- ko if: ifExtension -->
<fsgbu-ob-remo-srv-ds-cash-deposit-extended data="{{payload}}" base="{{base}}">
</fsgbu-ob-remo-srv-ds-cash-deposit-extended>
<!-- /ko -->
```

9.12 JS Changes (Base Component)

This topic describes the base components JS fields changes for all the screens.

Perform the JS changes in the base component as shown in the JS Changes (Base Component) screen.

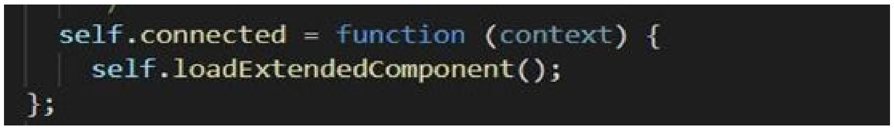
Figure 9-27   JS Changes (Base Component)



```
self.loadExtendedCCA = ko.observable('fsgbu-ob-base-component-extended');
self.ifExtension = ko.observable(false);


self.loadExtendedComponent = function () {
    // eslint-disable-next-line no-undef
    if (requirejs.s.contexts._.config.paths['components/' + self.loadExtendedCCA()]) {
        var componentName = ['components/' + self.loadExtendedCCA() + '/loader'];
        require(componentName, function () {
            self.ifExtension(true);
        });
    }
};
```

The part of code shown below is present in JS or view model file. From the self.connected method, you need to call self.loadExtendedComponent method.

Figure 9-28   Self Connected Method



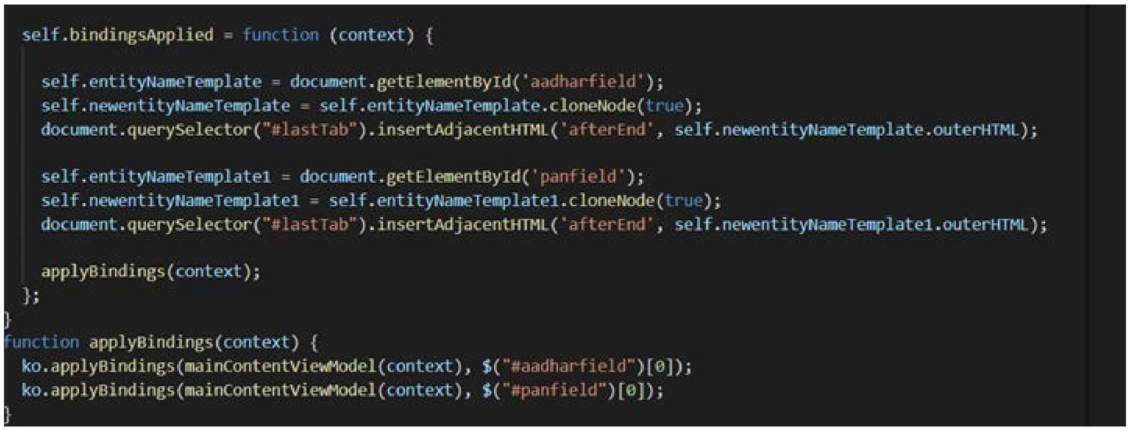
```
  self.connected = function (context) {
    self.loadExtendedComponent();
};
```

9.13 JS Changes (Extended Component)

This topic describes the extended components JS fields changes for all the screens.

In the bindings applied, it will take the ID of the fields and add the additional fields after the field base component. Both additional fields will be added after the field of base component for which the ID is **lastTab**.

Figure 9-29 JS Changes (Extended Component)



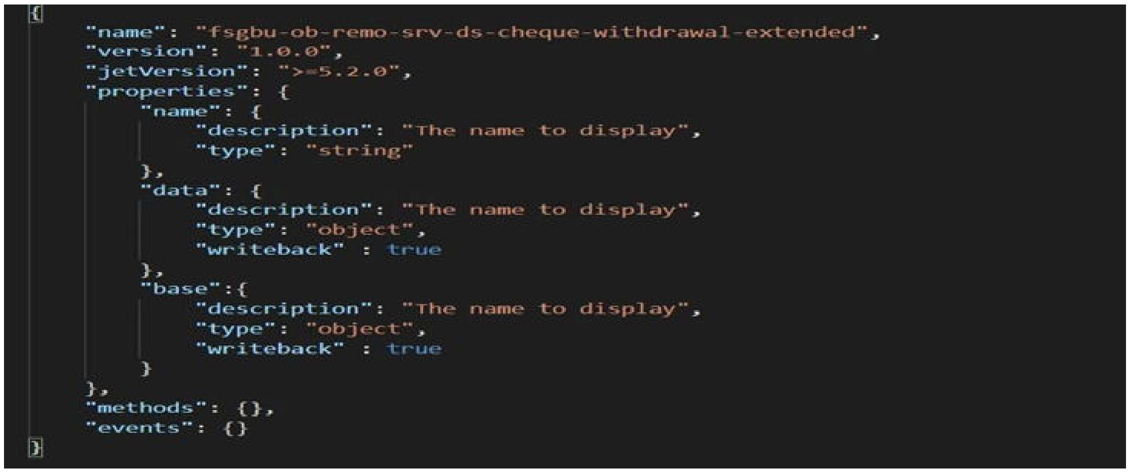
```
self.bindingsApplied = function (context) {

  self.entityNameTemplate = document.getElementById('aadharfield');
  self.newentityNameTemplate = self.entityNameTemplate.cloneNode(true);
  document.querySelector("#lastTab").insertAdjacentHTML('afterEnd', self.newentityNameTemplate.outerHTML);

  self.entityNameTemplate1 = document.getElementById('panfield');
  self.newentityNameTemplate1 = self.entityNameTemplate1.cloneNode(true);
  document.querySelector("#lastTab").insertAdjacentHTML('afterEnd', self.newentityNameTemplate1.outerHTML);

  applyBindings(context);
};
}
function applyBindings(context) {
  ko.applyBindings(mainContentViewModel(context), $("#aadharfield")[0]);
  ko.applyBindings(mainContentViewModel(context), $("#panfield")[0]);
}
```

9.14 JSON Changes (Extended Component)

This topic describes the extended components JSON fields changes for all the screens.

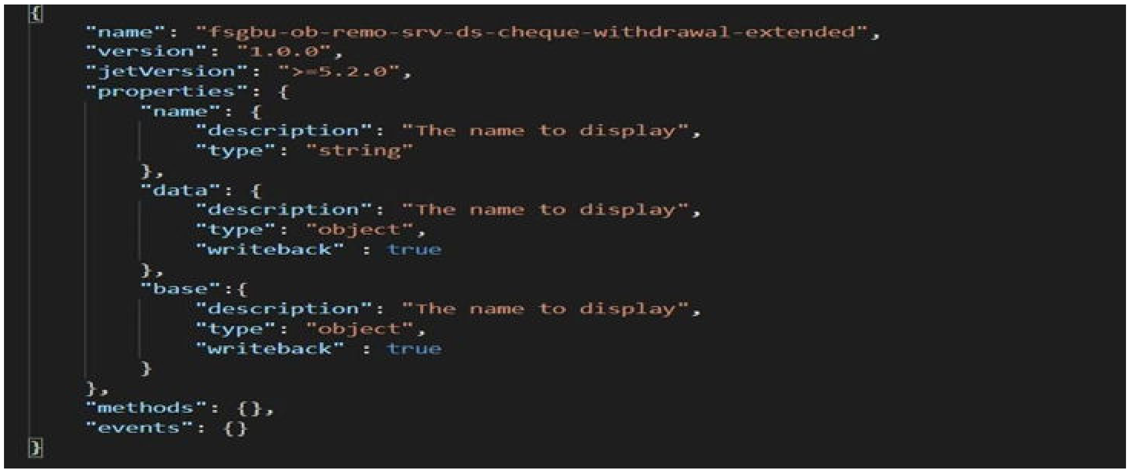
Perform the HTML changes to add data and base property for extended component.

Figure 9-30 Json Changes (Extended Component)



```
{
    "name": "fsgbu-ob-remo-srv-ds-cheque-withdrawal-extended",
    "version": "1.0.0",
    "jetVersion": ">=5.2.0",
    "properties": {
        "name": {
            "description": "The name to display",
            "type": "string"
        },
        "data": {
            "description": "The name to display",
            "type": "object",
            "writeback" : true
        },
        "base":{
            "description": "The name to display",
            "type": "object",
            "writeback" : true
        }
    },
    "methods": {},
    "events": {}
}
```

Figure 9-31 Json Changes (Extended Component)



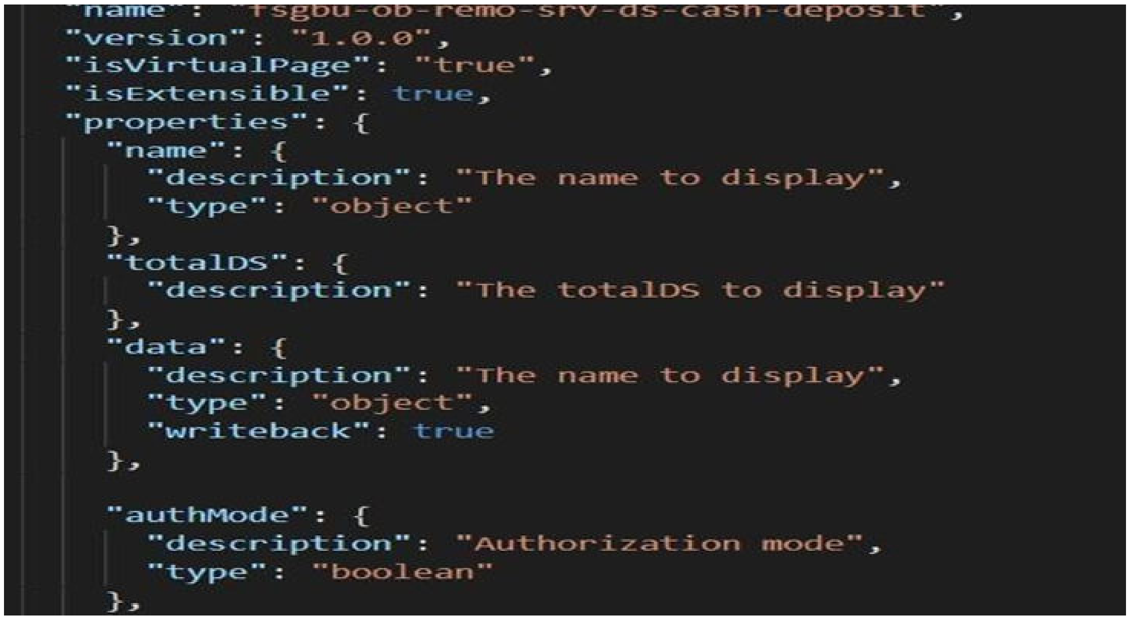
```
{
    "name": "fsgbu-ob-remo-srv-ds-cheque-withdrawal-extended",
    "version": "1.0.0",
    "jetVersion": ">=5.2.0",
    "properties": {
        "name": {
            "description": "The name to display",
            "type": "string"
        },
        "data": {
            "description": "The name to display",
            "type": "object",
            "writeback" : true
        },
        "base":{
            "description": "The name to display",
            "type": "object",
            "writeback" : true
        }
    },
    "methods": {},
    "events": {}
}
```

9.15 JSON Changes (Base Component)

This topic describes the base components JSON fields changes for all the screens.

In base component JSON file, the properties is Extensible and authMode are present. You need to make changes in the existing appshell UI component so that it reads the extended component. In addition, it will contain DB scripts which need to be run.

Figure 9-32 JSON Changes (Base Component)



```
    "name": "fsgbu-ob-remo-srv-ds-cash-deposit",
    "version": "1.0.0",
    "isVirtualPage": "true",
    "isExtensible": true,
    "properties": {
      "name": {
        "description": "The name to display",
        "type": "object"
      },
      "totalDS": {
        "description": "The totalDS to display"
      },
      "data": {
        "description": "The name to display",
        "type": "object",
        "writeback": true
      },

      "authMode": {
        "description": "Authorization mode",
        "type": "boolean"
      },
```

9.16 DB Changes

This topic provides the systematic instructions to perform the basic operations on the selected records.

Add the newly created data segment name in the PRODUCT_EXTENDED_LEDGER table.

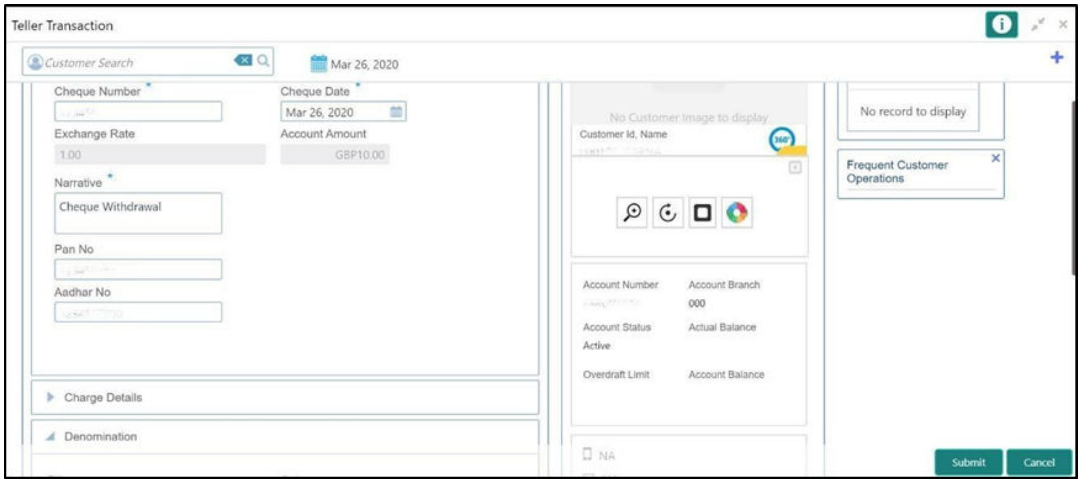
Perform the following steps to make the service level change:

1. Add a new field named additionalFields with data type String in work and main table entity classes of the respective service. The corresponding setters and getters should also be added in these classes.
   @Column(name = “ADDITIONAL_FIELDS”) private String additionalFields.
2. Add a column with the name **ADDITIONAL_FIELDS** in the main and work tables of the DB with CLOB data type.
3. For persistence of data in main table, add **additionalFields** with data type String in model class.
4. Deploy the changed service, extended war component, and changed appshell.

> **Note:**
>
> After deployment, the two additional fields named **Pan Number** and **Aadhaar Number** will be added in existing data segment.

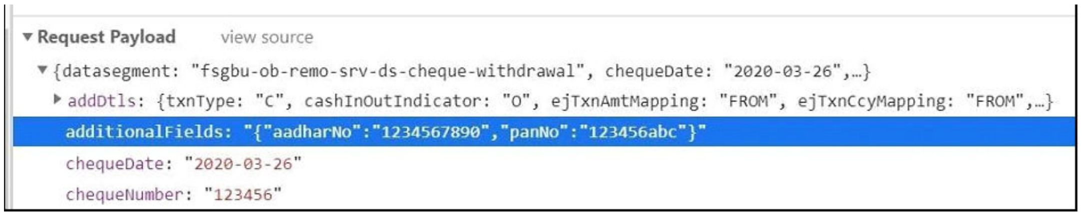
5. Specify the necessary details and click **Submit**. The additional fields will be saved in respective work and main table in an additional column **ADDITIONAL_FIELDS**.

Figure 9-33 Data Segment with Additional Fields



6. In the request payload from UI to backend, the values appear as follows:

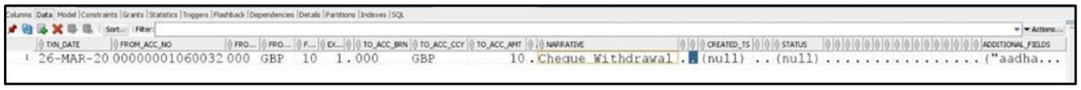
Figure 9-34 Request Payload



```
▼Request Payload     view source
 ▼{datasegment: "fsgbu-ob-remo-srv-ds-cheque-withdrawal", chequeDate: "2020-03-26",…}
  ▶addDtls: {txnType: "C", cashInOutIndicator: "O", ejTxnAmtMapping: "FROM", ejTxnCcyMapping: "FROM",…}
   additionalFields: "{"aadharNo":"1234567890","panNo":"123456abc"}"
   chequeDate: "2020-03-26"
   chequeNumber: "123456"
```

7. The data will get saved in newly added column Additional Fields in the respective table.

Figure 9-35 SRV_TB_CH_CASH_TXN Table



Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL

	TXN_DATE	FROM_ACC_NO	FRO...	FRO...	F...	EX...	TO_ACC_BRN	TO_ACC_CCY	TO_ACC_AMT	NARRATIVE	CREATED_TS	STATUS	ADDITIONAL_FIELDS
1	26-MAR-20	00000001060032	000	GBP	10	1.000		GBP	10.	Cheque Withdrawal	(null)	(null)	{"aadha...

9.17 Add New Columns in Base Component Table

This topic provides the systematic instructions to perform the basic operations on the selected records.

For adding new columns in base component table to do the following.

1. Create an extended component for the base cca by making these changes in the base accordingly.
2. Changes in base
   In HTML

```
<!-- ko if: ifExtension -->
<componentName-extended data="{{base}}">
</componentName-extended>
<!-- /ko -->
```

   In JS

```
       self.base
= this;
      self.ifExtension = ko.observable(false);
      self.connected = function () {         if
(requirejs.s.contexts._.config.paths['components/componentName-
extended']) {
          require(['components/componentName-extended/loader'], function
() {
self.ifExtension(true);
           });
         }
       }
```

   .
3. Changes in extended

```
       self.bindingsApplied = function (context) {
context.props.then(function (properties) {
console.log(properties.data.columnArray);
           properties.data.columnArrray.splice(columnIndex, 0,
{
headerText: "Manager Id",                                field: "ManagerId"
         });
tableId.refresh(properties.data.columnArray);
        });
```

4. Changes needed at service level.
   For data inside table, custom projection service had to be written, custom events needs to be raised while custom fields persistence. For base fields, a call can be made from projection service to base service to fetch data and persisting the same over projection schema.

9.18 Steps for adding extra column in task grid

This topic provides the systematic instructions to perform the basic operations on the selected records.

For adding extra column in task grid to do the following:

1. Clone the respective Free/My/Hold Task components.
2. Then the additional column can be added using the following example code snippet.

```
self.additionalColumns = [{
           dataIndex: 'customerName',
           dataType: 'string',
           displayType: 'text',
           width: '60px',
           sortable: true,
           resizable: true,
           accessTo: ['AVAILABLE', 'HOLD', 'ACQUIRED']
           }];
```

   The above code needs to be added in js file of the cloned components.

   While calling **fsgbu-ob-cmn-fd-work-list** from the html of the cloned components please make a call like this (which also sends additional columns as a property).

   Example:

```
<fsgbu-ob-cmn-fd-work-list id='completedTaskGridCCA' dashboard-
id='STANDARD' dashboard-
queue-name='ACQUIRED'
  process-code={{processCode}} dashboard-queue-type='L' worklist-
columns='{{columnArray}}'
  additional-columns='{{additionalColumns}}' page-size=20 dependent-
vm="{{dialogParameters}}"></fsgbu-ob-cmn-fd-work-list>
```

3. Making these changes would display the extra column in the task screens.

9.19 Steps to use Additional Buttons provision in Task Screen

This topic provides the systematic instructions to perform the basic operations on the selected records.

In the custom component (example - fsgbu-ob-slp0-vp-wl-locked-task-extended) from where you will be calling **fsgbu-ob-cmn-fd-work-list**, make the following changes:

1. In the js file you can declare an array of the buttons you want to include like this-

```
self.extraButtons = [{       label: 'Extraa',       icons: {
start: 'oj-ux-ico-refresh' },       display: 'all',
accessTo: ['L', 'F', 'H', 'C', 'S', 'A', 'O', 'T', 'WFCC']
  },   {       label: 'Extrab',
icons: { start: 'oj-ux-ico-refresh' },
display: 'all',       accessTo: ['L', 'F']
  }
   ]
```

2. And also the method which needs to be executed on the button click.

```
self.extraa = function(data){
console.log("it got called");
     }
```

Note:

The function name should be same as label of the button (in lower case).

3. In the HTML file, additional buttons attribute needs to be included like this:

```
 <fsgbu-ob-cmn-fd-work-list id='completedTaskGridCCA' dashboard-
id='STANDARD'
dashboard-queue-name='ACQUIRED' dashboard-queue-type='L' worklist-
columns='{{columnArray}}'  additional-columns='{{additionalColumns}}'
additional-buttons='{{extraButtons}}' page-size=20>
</fsgbu-ob-cmn-fd-work-list>
```

4. In the json file, the methods which would be implemented on the custom button click needs to be exposed.

```
    "methods": {
      "extraa": {
        "description": "Would be implemented on Extraa button click"
      },
       "extrab": {
         "description": "Would be implemented on Extrab button click"
      }
     }
```

9.20 Steps to create common-extended folder for extending configJSON.js file

This topic provides the systematic instructions to perform the basic operations on common-extended folder for extending configJSON.js file.

For creating common-extended folder for extending configJSON.js file to do the following:

1. Create a folder inside **extended-components\js\components**.
2. Folder structure **\common-extended\js\util**.
3. Next we will add a file configJSON.js in the created folder.
4. The code inside this configJSON.js would be like-

```
define(['cmn-util/configJSON'], function (baseobj) {
baseobj.applicationObject.entityIdByProcessCode['CUSTOM'] = {'ccName':
'fsgbu-
ob-remo-deposit-ct-process-flow', 'Name': 'RD Amount Block', 'shortName':
'RD
Amount Block'};
});
```

5. Some understanding of the code: -
   - Including the base object by giving the path of configJSON.js base file.
   - Then for example adding the entry for custom process as shown above.
   - The extended configJSON file would be loaded from base commonFunction.js
6. Insertion of the below script into PRODUCT_EXTENDED_LEDGER table

```
Insert into PRODUCT_EXTENDED_LEDGER
(ID,CCA_NAME,CCA_TYPE,PARENT_CCA_NAME,PRODUCT_NAME)  select nvl(new_uuid
,'common-extended','config',null,'EXTENDED_COMPONENTS'from
PRODUCT_EXTENDED_LEDGER;
```

9.21 Customizing Existing LOV Fetch Result

This topic provides the systematic instructions to perform the basic operations on the Customizing Existing LOV Fetch Result.

Modifying the retrieval output of an existing LOV to meet specific requirements.

- Ins cope Data segment can be used for addition of new fields. (using jquery, at any position, we can add the field).
- Service Extensibility to be used for overriding the base method, OBX tool will generate the base service jar from base service war and this jar should be used to override the base service method and implement the custom changes.
- From UI, call will go to custom service , from custom service, call will go to base service for base field persistence as Java to Java call, then custom functionality to be implemented for persistence of custom fields as part of REST call to another custom service.
- For LOV data, custom projection service to be written. Custom Event needs to be raised while custom fields persistence. For base fields, a call can be made from projection service to base service to fetch data and persisting over the projection schema.

9.22 Steps for adding Pre/post methods in extended components

This topic provides the systematic instructions to perform the basic operations on the selected records.

Suppose here we consider that we want to persist custom fields on postnext call (which means first ‘self.next’ method of base would get called and then the control will come in postnext method written in extended component).

1. Write postnext method in .js file of the extended component – wherein you can call the custom Api for persisting the custom fields.
2. Expose this method in the .json file of the extended component.
3. Similarly we can add prenext method as well.(it would get executed before ‘self.next’ method of base executes).

> **Note:**
>
> The hooks for these methods to work should be a part of common infrastructure components in appshell.

4. Below is the list of CCAs and the common methods which has pre and post hooks:

Table 9-1 List of CCAs - Field Description

CCA Name	Common method name	Pre hook present	Post hook present
fsgbu-ob-cmn-ct-authorization	compare	No	Yes
	approve	No	Yes
fsgbu-ob-cmn-ct-act-summary- template	delete	No	Yes
	reopen	No	Yes
	close	No	Yes
fsgbu-ob-cmn-ct-maintenance	save	Yes	Yes
fsgbu-ob-cmn-ct-wizard	next	Yes	Yes
	previous	Yes	Yes
	saveClose	Yes	Yes
	cancel	Yes	Yes
	hold	Yes	Yes
	Applicable for custom footer buttons as well	Yes	Yes
fsgbu-ob-cmn-ct-rs-authorization	approve	No	Yes
fsgbu-ob-cmn-ct-summary-template	delete	No	Yes
	open	No	Yes
	close	No	Yes

9.23 ENDPOINT Overrides

This topic describes the endpoint overrides.

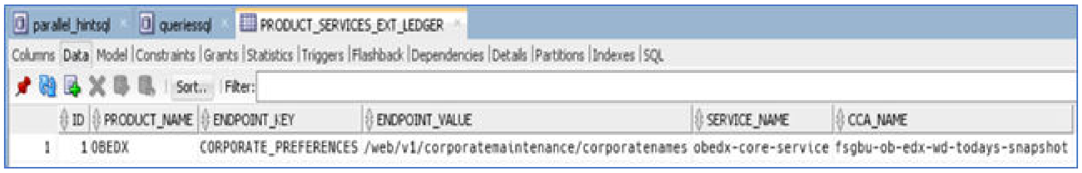
To enhance the endpoint override extensibility, we've added a new column, `CCA_NAME`, to the **PRODUCT_SERVICE_EXT_LEDGER** table.

This column provides an extensibility for overriding the existing endpoint behaviour for specific UI components.

How to configure:

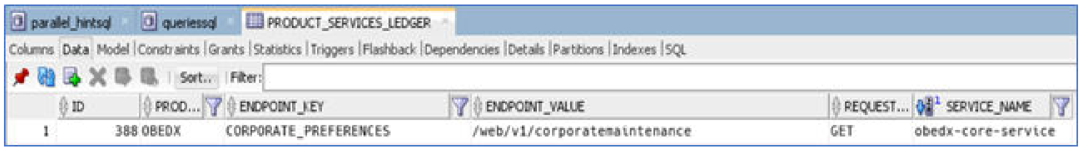
1. Determine the component for which you want to override the endpoint.
2. Enter the component's name in the **CCA_NAME** column of the **PRODUCT_SERVICE_EXT_LEDGER** table.
3. **PRODUCT_NAME** & **ENDPOINT_KEY** must be same as endpoint we are extending.
4. The **ENDPOINT_VALUE** field should be populated with the new endpoint URI, while the **SERVICE_NAME** field should specify the corresponding service to which this endpoint belongs.
5. An entry of extension service should also be present in **PRODUCT_SERVICE_CTX_LEDGER** to pick up the new APPID or other properties.
6. If **CCA_NAME** column contains **NULL** value, then endpoint override will be applicable across all components subscribed to respective **ENDPOINT_KEY**.

Figure 9-36 Endpoint 1


parallel_hintsql
queriessql
PRODUCT_SERVICES_EXT_LEDGER
Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL
Sort..
Filter:

	ID	PRODUCT_NAME	ENDPOINT_KEY	ENDPOINT_VALUE	SERVICE_NAME	CCA_NAME
1	1	OBEDX	CORPORATE_PREFERENCES	/web/v1/corporatemaintenance/corporatenames	obedx-core-service	fsgbu-ob-edx-wd-todays-snapshot

Figure 9-37 Endpoint 2


parallel_hintsql
queriessql
PRODUCT_SERVICES_LEDGER
Columns Data Model Constraints Grants Statistics Triggers Flashback Dependencies Details Partitions Indexes SQL
Sort...
Filter:

	ID	PROD...	ENDPOINT_KEY	ENDPOINT_VALUE	REQUEST...	SERVICE_NAME
1	388	OBEDX	CORPORATE_PREFERENCES	/web/v1/corporatemaintenance	GET	obedx-core-service

How it works:

When a request is made for the component, the ext orchestrator service will now consult the **CCA_NAME** column. If a matching entry exists, the endpoint specified in the ext orchestrator service (PRODUCT_SERVICE_EXT_LEDGER) will take precedence over the existing endpoint of base product.

This new approach offers several advantages:

- Any endpoint can be extended using this approach.
- The **PRODUCT_SERVICE_EXT_LEDGER** table is independent of product-related flyway updates, ensuring that future changes won't impact existing overrides.

- This extensibility allows for specific endpoint overrides, other components are unaffected with their original endpoints.

9.24 Steps to create util-extended folder

This topic provides the systematic instructions to perform the basic operations on the selected records.

1. Create a folder inside extended-components\js\components in app-shell for component you want to make label-changes.
2. Folder structure: <%componentName%>-util-extended\resources\<component-name>\nls. **Example** : for sms it would look like: sms-util-extended\resources\sms\nls.
3. Add the file bundle.js in the created folder.
4. The code inside bundle.js would be like-

```
define(['ojL10n!' + window.location.origin + '/<%componentName%>-component-
server/js/components/resources/<%componentName%>/nls/bundle.js'],
 function (baseLabels) {
baseLabels.fsgbuobsmsmnusers.lblhomeBranch = "Foreig111n Branch"
baseLabels.fsgbuobsmsmnusers.lblstatusChangedOn = "Yogesh"   return
{
     'root': baseLabels
   };
});
```

5. Some understanding for the code: -
   - Including the base labels by giving the path of bundle.js of main component.
   - Then changing the labels accordingly like in the example above -> Home Branch is replaced with “Foreign111n Branch”.
   - Returning the labels (including the changes).
6. Insertion of the below script into **PRODUCT_EXTENDED_LEDGER** table.

```
Insert into PRODUCT_EXTENDED_LEDGER
ID,CCA_NAME,CCA_TYPE,PARENT_CCA_NAME,PRODUCT_NAME)
select nvl(new_uuid   ,'<%=componentName%>-util-
extended','util',null,'EXTENDED_COMPONENTS'from PRODUCT_EXTENDED_LEDGER;
```

9.25 Dynamic Data Configuration (DDC)

This topic provides the systematic instructions to perform the basic operations on the selected records.

DDC is an infrastructure component comprising a user interface and a service.

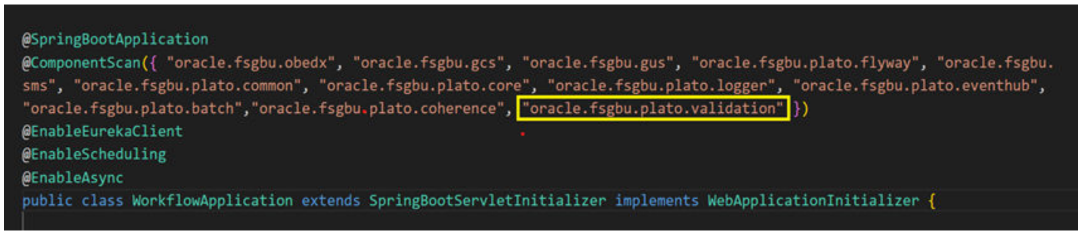
It empowers developers to define prepared statements for dynamic data retrieval. The Dynamic Data Configuration (DDC) service's response is utilized by UI components or invoking services to render List of Values (LOV) results.

Dynamic Data Configuration infra can be utilized with OBX code to call endpoint and bind the response.

Prerequisites:

- For domain services to perform dynamic data queries on the domain schema, the @ComponentScan annotation must include the "oracle.fsgbu.plato.validation" where domain services reside.

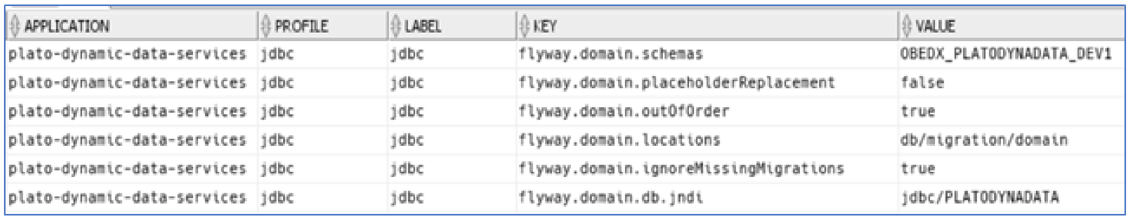
Figure 9-38 Plato Validation



```
@SpringBootApplication
@ComponentScan({ "oracle.fsgbu.obedx", "oracle.fsgbu.gcs", "oracle.fsgbu.gus", "oracle.fsgbu.plato.flyway", "oracle.fsgbu.
sms", "oracle.fsgbu.plato.common", "oracle.fsgbu.plato.core", "oracle.fsgbu.plato.logger", "oracle.fsgbu.plato.eventhub",
"oracle.fsgbu.plato.batch","oracle.fsgbu.plato.coherence", "oracle.fsgbu.plato.validation" })
@EnableEurekaClient
@EnableScheduling
@EnableAsync
public class WorkflowApplication extends SpringBootServletInitializer implements WebApplicationInitializer {
```

- A database schema created for the Dynamic Data Configuration service.
- A configured JDBC data source named **jdbc/PLATODYNADATA** on the server.
- Configure newly created schema name in **PROPERTIES** table of **PLATO** schema.

Figure 9-39 PLATO schema



APPLICATION	PROFILE	LABEL	KEY	VALUE
plato-dynamic-data-services	jdbc	jdbc	flyway.domain.schemas	OBEDX_PLATODYNADATA_DEV1
plato-dynamic-data-services	jdbc	jdbc	flyway.domain.placeholderReplacement	false
plato-dynamic-data-services	jdbc	jdbc	flyway.domain.outOfOrder	true
plato-dynamic-data-services	jdbc	jdbc	flyway.domain.locations	db/migration/domain
plato-dynamic-data-services	jdbc	jdbc	flyway.domain.ignoreMissingMigrations	true
plato-dynamic-data-services	jdbc	jdbc	flyway.domain.db.jndi	jdbc/PLATODYNADATA

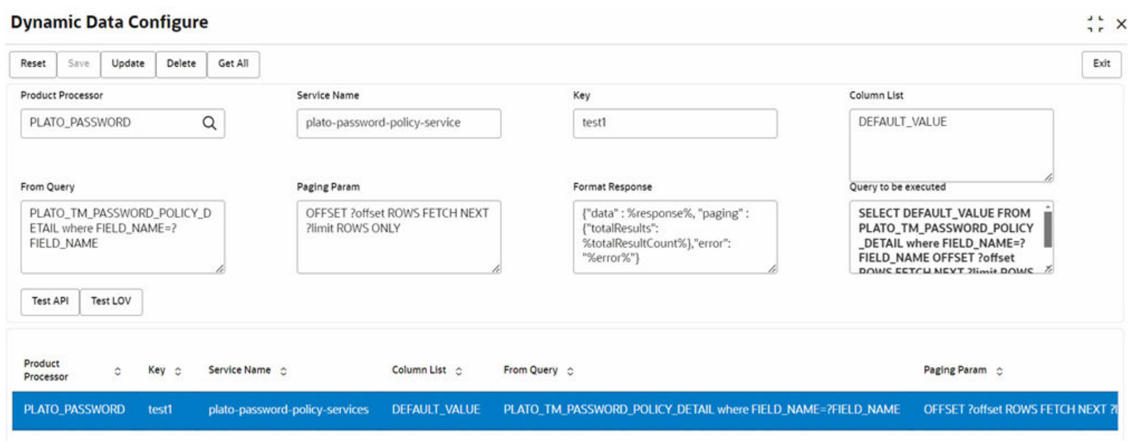
Deployment Steps:

1. Deploy the Dynamic Data Configuration service to the server.
2. Once deployed, the Dynamic Data Configuration user interface should be accessible.

Configuration steps:

1. Select the desired **product processor**.
2. Specify the **service name**.
3. Define the **unique key** for the data.
4. List the required **columns**.
5. Provide the **from query** to retrieve data.
6. Set the **paging parameters** (if applicable).
7. Determine the desired **response format**.

Figure 9-40 Dynamic Data Configure



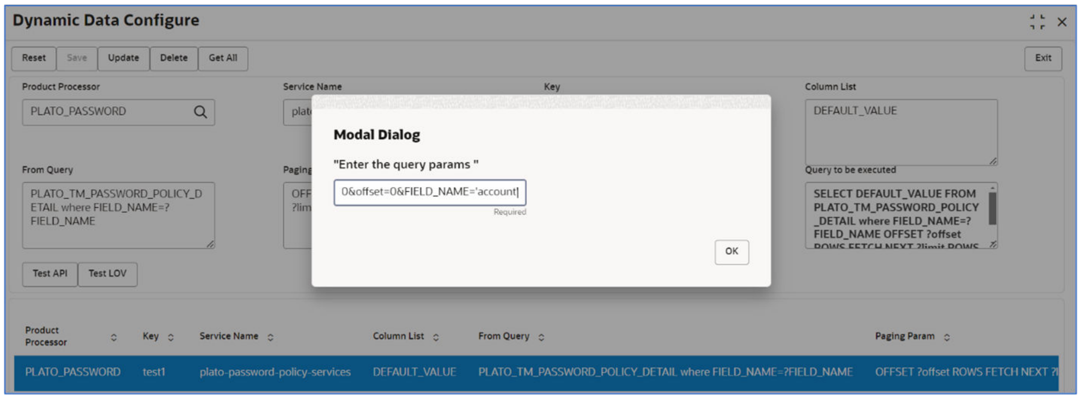
Product Processor: PLATO_PASSWORD
Service Name: plato-password-policy-service
Key: test1
Column List: DEFAULT_VALUE
From Query: PLATO_TM_PASSWORD_POLICY_DETAIL where FIELD_NAME=?FIELD_NAME
Paging Param: OFFSET ?offset ROWS FETCH NEXT ?limit ROWS ONLY
Format Response: {"data" : %response%, "paging" : {"totalResults": %totalResultCount%},"error": "%error%"}
Query to be executed: SELECT DEFAULT_VALUE FROM PLATO_TM_PASSWORD_POLICY_DETAIL where FIELD_NAME=?FIELD_NAME OFFSET ?offset ROWS FETCH NEXT ?limit ROWS

Product Processor	Key	Service Name	Column List	From Query	Paging Param
PLATO_PASSWORD	test1	plato-password-policy-services	DEFAULT_VALUE	PLATO_TM_PASSWORD_POLICY_DETAIL where FIELD_NAME=?FIELD_NAME	OFFSET ?offset ROWS FETCH NEXT ?l

Test Query:

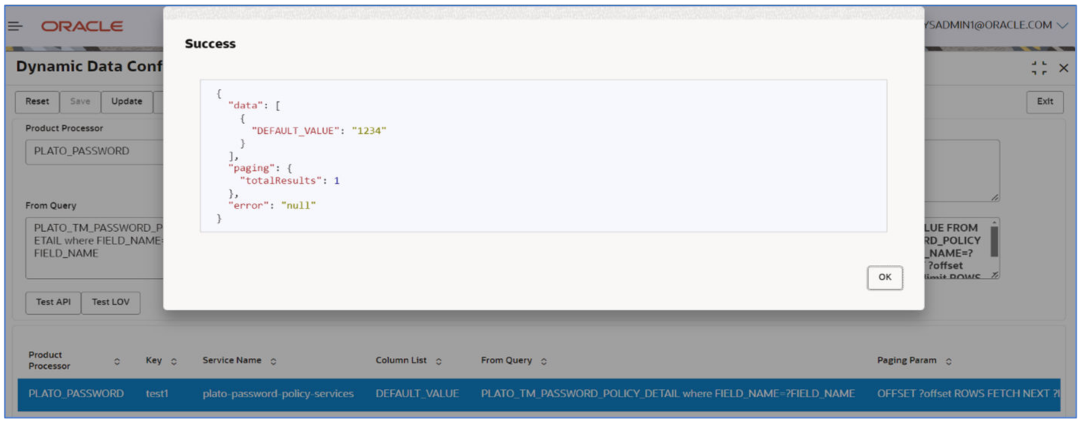
- **Test API**: Use the test API to execute the query. Provide any necessary query parameters and click "OK." The results will be displayed based on the query.

Figure 9-41 Modal Dialog



Modal Dialog
"Enter the query params "
0&offset=0&FIELD_NAME='account'
Required

Figure 9-42 Success

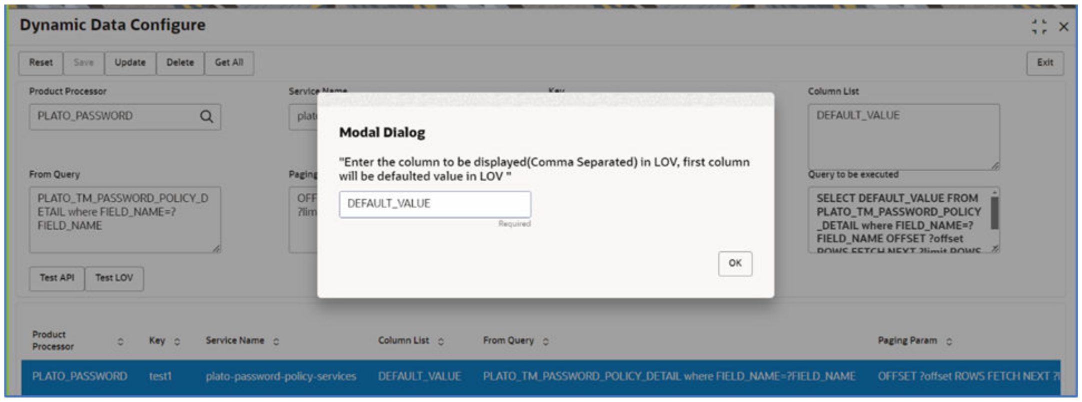


Success

```
{
  "data": [
    {
      "DEFAULT_VALUE": "1234"
    }
  ],
  "paging": {
    "totalResults": 1
  },
  "error": "null"
}
```

- **Test LOV**: If applicable, use the test List of Values (LOV) to test the query.

Figure 9-43 Modal Dialog



Product Processor: PLATO_PASSWORD

From Query: PLATO_TM_PASSWORD_POLICY_DETAIL where FIELD_NAME=?FIELD_NAME

Column List: DEFAULT_VALUE

Query to be executed: SELECT DEFAULT_VALUE FROM PLATO_TM_PASSWORD_POLICY_DETAIL where FIELD_NAME=?FIELD_NAME OFFSET ?offset ROWS FETCH NEXT ?limit ROWS

Modal Dialog

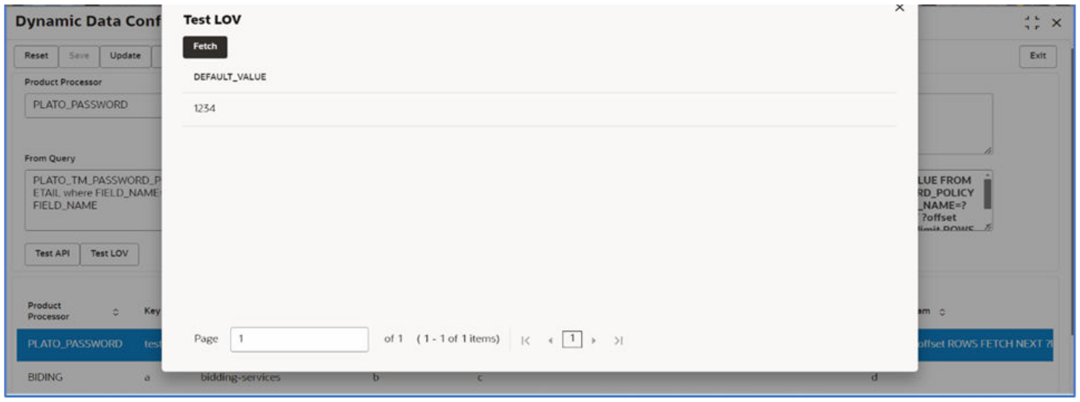
"Enter the column to be displayed(Comma Separated) in LOV, first column will be defaulted value in LOV "

DEFAULT_VALUE

Required

Product Processor	Key	Service Name	Column List	From Query	Paging Param
PLATO_PASSWORD	test1	plato-password-policy-services	DEFAULT_VALUE	PLATO_TM_PASSWORD_POLICY_DETAIL where FIELD_NAME=?FIELD_NAME	OFFSET ?offset ROWS FETCH NEXT ?l

Figure 9-44 Test LOV



Test LOV

DEFAULT_VALUE
1234

Page 1 of 1 (1 - 1 of 1 items)

Product Processor	Key	Service Name	Column List	From Query	Paging Param
PLATO_PASSWORD	tes				offset ROWS FETCH NEXT ?
BIDING	a	bidding-services	b	c	d

Once satisfied with the results, save the dynamic data query.

9.26 Task Screen Custom Config

This topic provides the systematic instructions to perform the basic operations on the selected records.

This document outlines how to customize the task screen using a **CUSTOM_CONFIG** table, you can show or hide existing columns, and even add additional filters to the task search screen for specific fields.

Prerequisites:

Ensure that all columns on the task screen are listed in the **CUSTOM_CONFIG** table present in **PLATO_ORCH** schema. All the default columns would already be present in this table.

- For all the default columns in Task screen, **TASK_SCREEN_VIEW** column value will be set to YES by default. If consulting wishes to hide any default column, they can set it to **NO**.
- Also, if they wish to add new custom column, they need to add the key (key in which we will get the value of custom field in response of task screen **plato-orch-service/api/v1/**

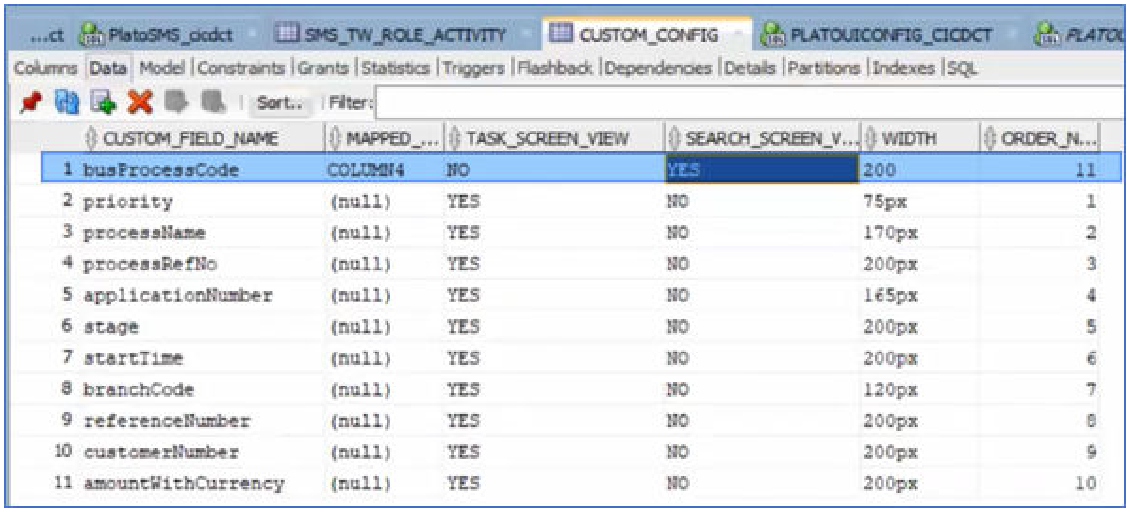
extn/tasks api) of that column in **CUSTOM_FIELD_NAME** column in CUSTOM_CONFIG table.

Adding Custom Filters

1. Determine the custom field for which you want to add a filter.
2. Update **CUSTOM_CONFIG** table:
   - Add the field name (custom field key) to the **CUSTOM_FIELD_NAME** column.
   - Set the **SEARCH_SCREEN_VIEW** column to YES for this field.

Once these changes are made, the additional filter will be displayed on the search screen's UI.

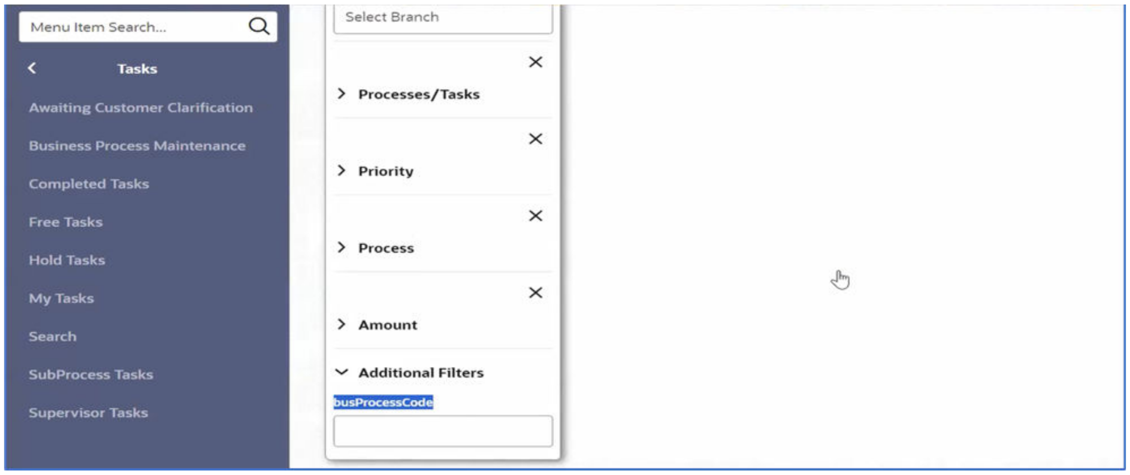
Figure 9-45 Custom Fields 1



...ct
PlatoSMS_cicdct
SMS_TW_ROLE_ACTIVITY
CUSTOM_CONFIG
PLATOUICONFIG_CICDCT
PLATO
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

	CUSTOM_FIELD_NAME	MAPPED_...	TASK_SCREEN_VIEW	SEARCH_SCREEN_V...	WIDTH	ORDER_N...
1	busProcessCode	COLUMN4	NO	YES	200	11
2	priority	(null)	YES	NO	75px	1
3	processName	(null)	YES	NO	170px	2
4	processRefNo	(null)	YES	NO	200px	3
5	applicationNumber	(null)	YES	NO	165px	4
6	stage	(null)	YES	NO	200px	5
7	startTime	(null)	YES	NO	200px	6
8	branchCode	(null)	YES	NO	120px	7
9	referenceNumber	(null)	YES	NO	200px	8
10	customerNumber	(null)	YES	NO	200px	9
11	amountWithCurrency	(null)	YES	NO	200px	10

Figure 9-46 Screen UI

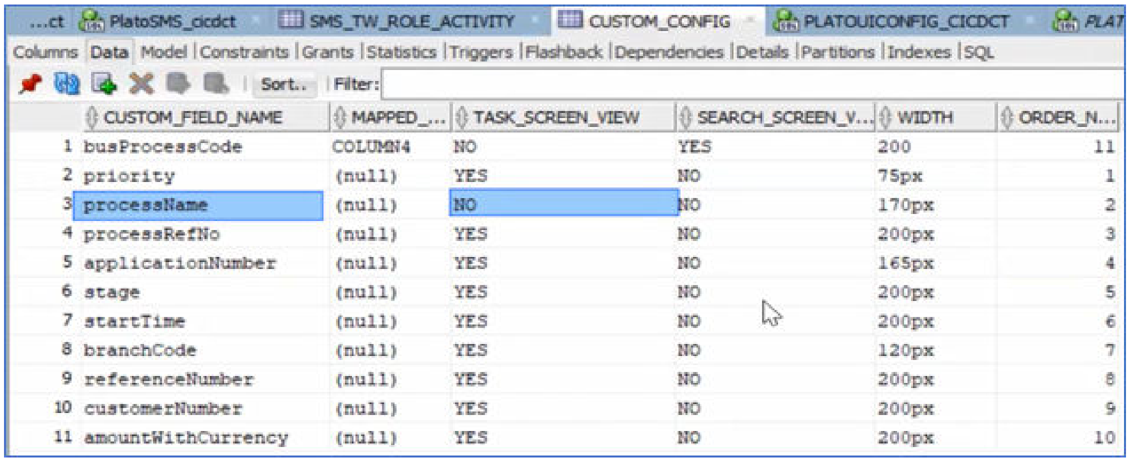


Hiding/Adding Columns on the Task Screen:

1. Identify the column you want to hide.
2. Update **CUSTOM_CONFIG** table:
   Set the **TASK_SCREEN_VIEW** column to **NO** for that column.

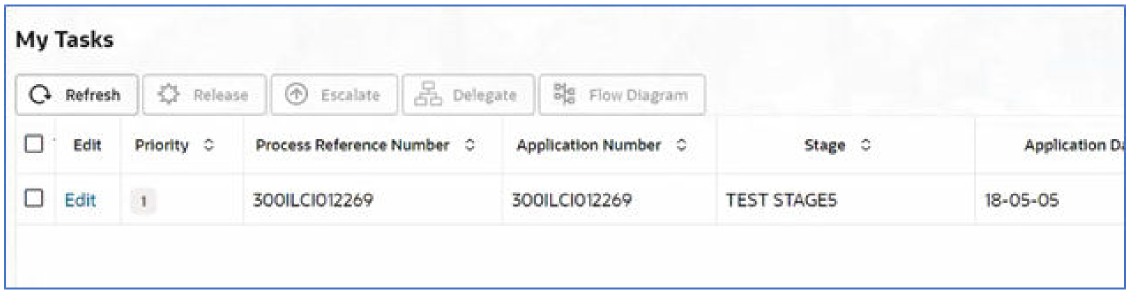
After updating the configuration, the column will no longer be visible on the task screen.

Figure 9-47 Custom Fields 2


...ct
PlatoSMS_cicdct
SMS_TW_ROLE_ACTIVITY
CUSTOM_CONFIG
PLATOUICONFIG_CICDCT
PLAT
Columns | Data | Model | Constraints | Grants | Statistics | Triggers | Flashback | Dependencies | Details | Partitions | Indexes | SQL
Sort..
Filter:

	CUSTOM_FIELD_NAME	MAPPED_...	TASK_SCREEN_VIEW	SEARCH_SCREEN_V...	WIDTH	ORDER_N...
1	busProcessCode	COLUMN4	NO	YES	200	11
2	priority	(null)	YES	NO	75px	1
3	processName	(null)	NO	NO	170px	2
4	processRefNo	(null)	YES	NO	200px	3
5	applicationNumber	(null)	YES	NO	165px	4
6	stage	(null)	YES	NO	200px	5
7	startTime	(null)	YES	NO	200px	6
8	branchCode	(null)	YES	NO	120px	7
9	referenceNumber	(null)	YES	NO	200px	8
10	customerNumber	(null)	YES	NO	200px	9
11	amountWithCurrency	(null)	YES	NO	200px	10

Figure 9-48 Task List



	Edit	Priority	Process Reference Number	Application Number	Stage	Application D
☐	Edit	1	300ILCI012269	300ILCI012269	TEST STAGE5	18-05-05

Similarly , we can even add new custom column in Task screens. For this they need to add the custom field name (key in which we will get the value of custom field in response of task screen **plato-orch-service/api/v1/extn/tasks** api) of that column in **CUSTOM_FIELD_NAME** column in **CUSTOM_CONFIG** table.

Configurations needed from backend:

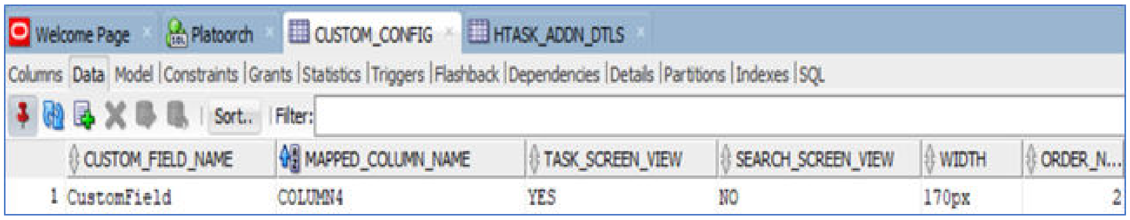
Configurations needed from backend side to get the custom field in **plato-orch-service/api/v1/extn/tasks** response –

During workflow initiation, the customer provides key-value pairs for specific columns. In the **CUSTOM_CONFIG** table, columns are mapped under the **MAPPED_COLUMN_NAME** field. For instance, COLUMN4 is mapped to a custom_field_name, such as CustomField.

Here's how it works: In the **CUSTOM_CONFIG** table, COLUMN4 is mapped to the field CustomField. During workflow initiation, the customer provides the value for COLUMN4, such as COLUMN4 = CF_1. The system uses this mapping to interpret the value as follows: CustomField (from the CUSTOM_CONFIG mapping) will get the value CF_1 for that task , provided by the customer during initiation. This allows the customer to input COLUMN4 = CF_1 during workflow initiation, and it will be mapped with **CUSTOM_FIELD_NAME** based on the mapping defined in the **CUSTOM_CONFIG** table This way, you can map any internal column to a custom field name that suits your specific use case.

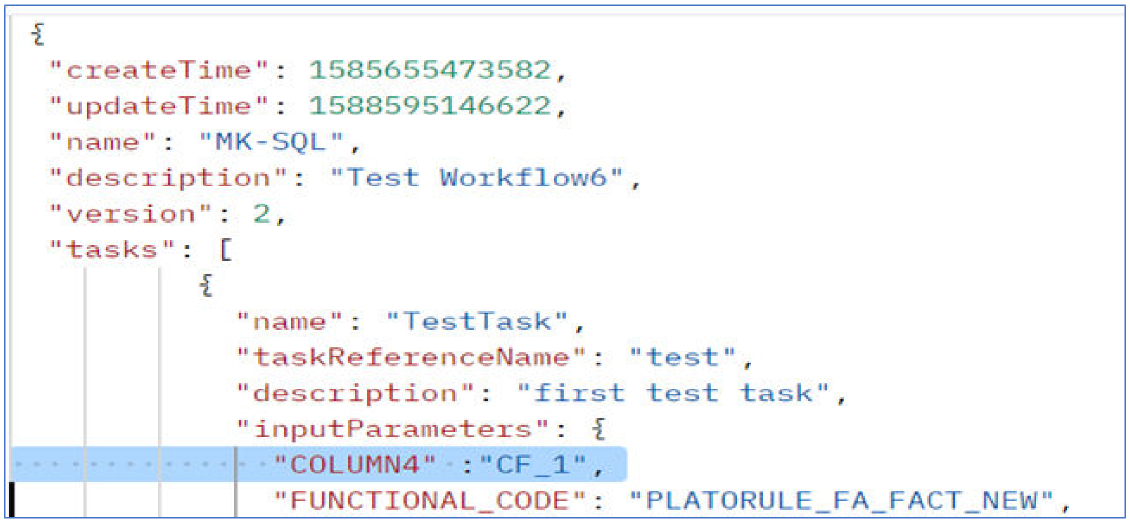
Additionally, the columns that can be used for such mappings currently range from COLUMN1 to COLUMN20, providing flexibility to define up to 20 custom fields.

Figure 9-49   Custom Fields 3



	CUSTOM_FIELD_NAME	MAPPED_COLUMN_NAME	TASK_SCREEN_VIEW	SEARCH_SCREEN_VIEW	WIDTH	ORDER_N...
1	CustomField	COLUMN4	YES	NO	170px	2

Figure 9-50   Test Workflow



```
{
  "createTime": 1585655473582,
  "updateTime": 1588595146622,
  "name": "MK-SQL",
  "description": "Test Workflow6",
  "version": 2,
  "tasks": [
          {
            "name": "TestTask",
            "taskReferenceName": "test",
            "description": "first test task",
            "inputParameters": {
              "COLUMN4" :"CF_1",
              "FUNCTIONAL_CODE": "PLATORULE_FA_FACT_NEW",
```

10

Reference and Feedback

This section describes following topics:

- Reference
- Documentation Accessibility
- Feedback and Support

10.1 Reference

For more information on any related features, you can refer to the following documents:

- **Oracle Banking Extensibility Workbench Installation Guide**

10.2 Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/us/corporate/accessibility/index.html.

10.3 Feedback and Support

Oracle welcomes customers' comments and suggestions on the quality and usefulness of the document. Your feedback is important to us. If you have a query that is not covered in this user guide.

Index