Oracle Banking Extensibility Workbench User Guide

Release 14.5.0.0.0

Part No. F44378-01

May 2021

Oracle Banking Extensibility Workbench
User Manual
Oracle Financial Services Software Limited

Oracle Park
Off Western Express Highway
Goregaon (East)
Mumbai, Maharashtra 400 063
India
Worldwide Inquiries: Phone: +91 22 6718 3000
Fax: +91 22 6718 3001 www.oracle.com/financialservices/

Table of Contents

Welcome to Oracle Banking Extensibility Workbench

Welcome to the Oracle Banking Extensibility Workbench (OBX) user manual. 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.

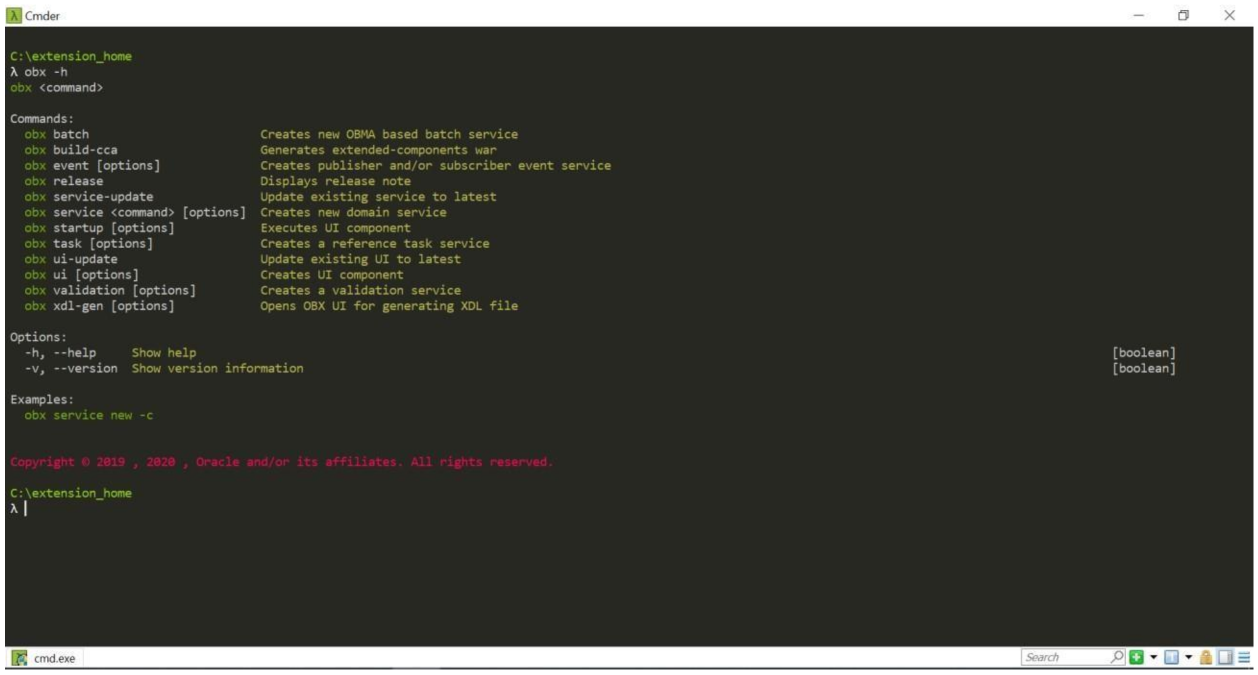
1. Introduction

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

- Service Extensions ○ Simple sub domain service ○ Maintenance sub domain service ○ Data/Resource Segment sub domain service ○ Simple Publisher/Subscriber Event Service ○ Custom Validation Service
- UI Extensions – Web Component ○ Simple Standalone ○ Virtual Page
    - Maintenance Detail and Summary ○ Data Segment ○ Dashboard Widget
- 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 Mandatory

Setting up OBX for first time use

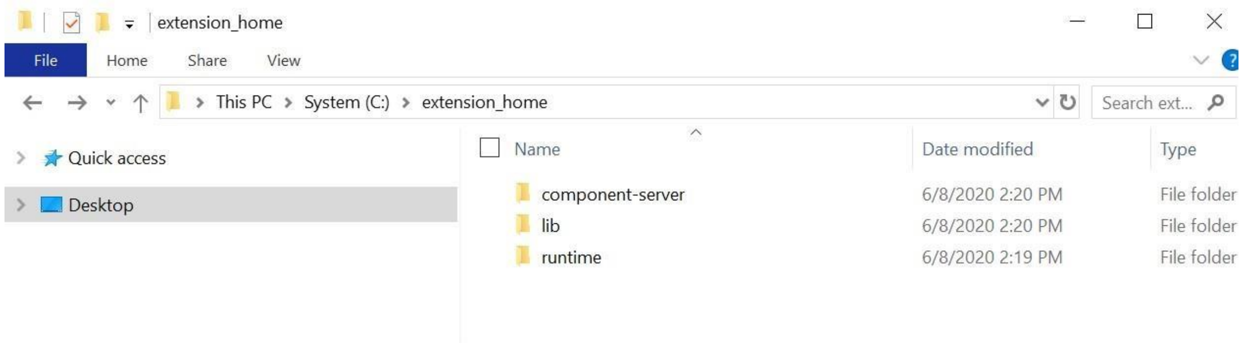
It is assumed that before setting up OBX for generating the first artifact, all the installation process is completed till extension_home folder creation and you are able to see the help menu like below:



```
C:\extension_home
λ obx -h
obx <command>

Commands:
  obx batch                         Creates new OBMA based batch service
  obx build-cca                     Generates extended-components 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 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 © 2019 , 2020 , Oracle and/or its affiliates. All rights reserved.

C:\extension_home
λ |
```

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

- Create a folder **component-server** inside extension_home directory
- Use 7zip or other similar tool to open **app-shell.war** from base product to copy the folders **common** and **components** and paste it inside **component-server** folder inside extension_home
- Create a folder **lib** inside extension_home directory
- Again, using 7zip or other similar tool open any service war like **cmc-datasegmentservices5.1.0.war,** navigate inside WEB-INF\lib folder and copy all the jars and put it inside the **lib** folder of extension_home
- Create a folder **runtime** inside extension_home directory
- From the **gradle** folder which comes inside the **obx.zip**, navigate inside the lib folder and copy **extra_jars** which are compile time dependencies for services, and paste it inside **runtime** folder extension_home
- After all the above process extension_home folder looks like below



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

- Once all of the above process is done we cannot now generate the artifact

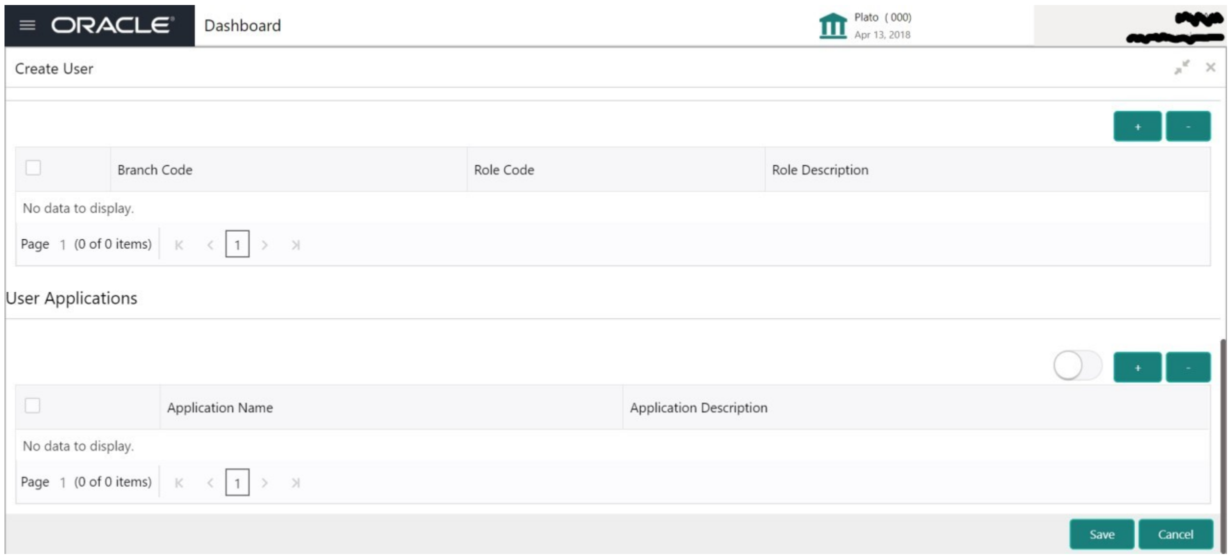
OBX Maintenance

Before generating the artifact please verify the below items from the base installation

- In the **plato-ui-config** schema, verify if the table '**PRODUCT_EXTENDED_LEDGER**' is present or not. If not available please execute the below script:

```
--------------------------------------------------------
--  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")
```

- Please maintain the product name '**OBX'** in the table '**SMS_TM_APPLICATION**' inside SMS schema
- Please grant user '**OBX**' application access through '**SMS_TM_USER_APPLICATION**' or preferred use the UI



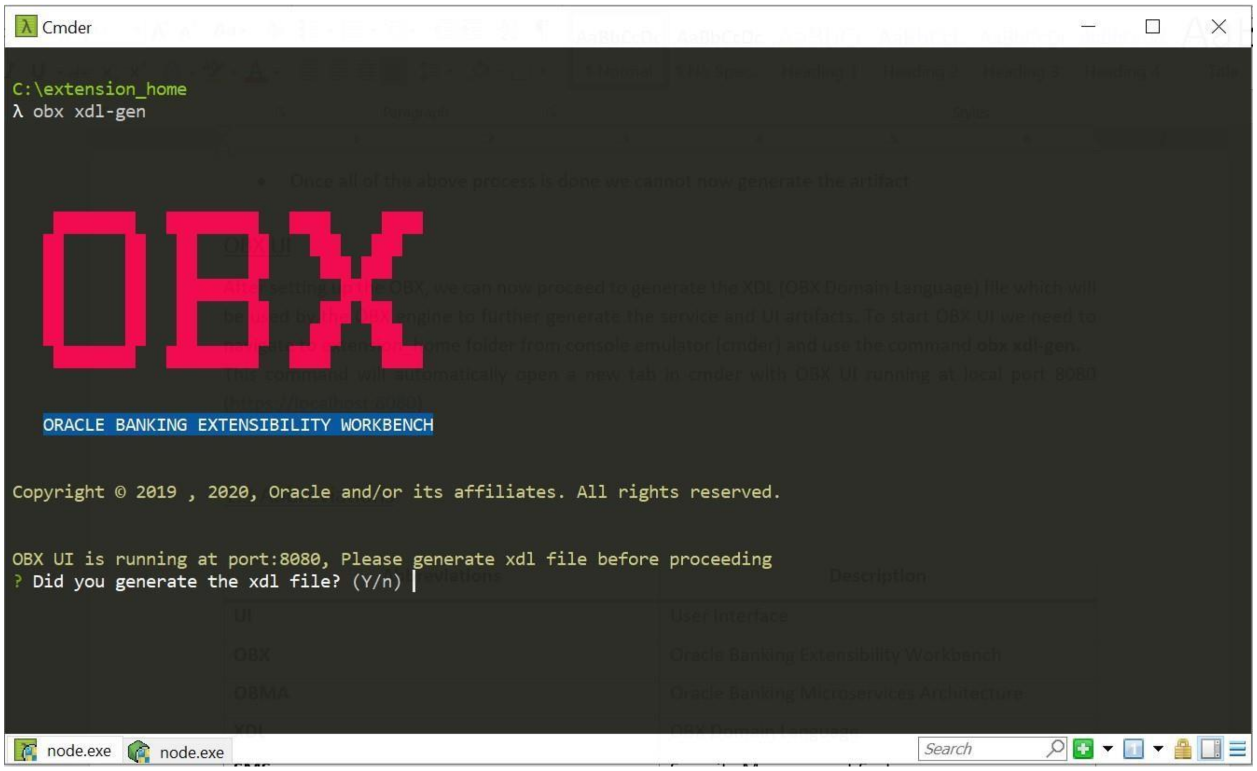
	Branch Code	Role Code	Role Description
No data to display.			

	Application Name	Application Description
No data to display.		

OBX UI

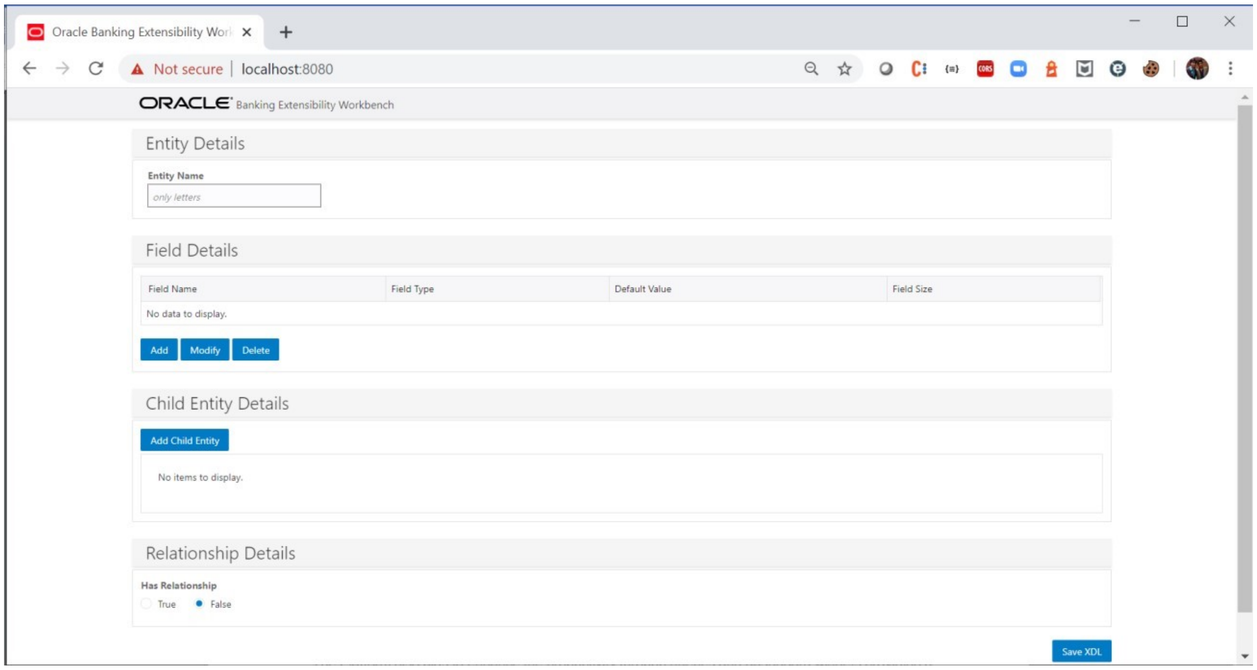
After setting up the OBX, we can now proceed to generate the XDL (OBX Domain Language) file which will be used by the OBX engine to further generate the service and UI artifacts. To start OBX UI we need to navigate to extension_home folder from console emulator (cmder) and use the command **obx xdl-gen.** 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 any running on port number 8080, you may need to stop that to make obx ui up and running.



```
C:\extension_home
λ obx xdl-gen

OBX

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.

OBX UI is running at port:8080, Please generate xdl file before proceeding
? Did you generate the xdl file? (Y/n)
```

Please open browser once obx UI is up and running and navigate to http://localhost:8080



Following are blocks present on the OBX UI-

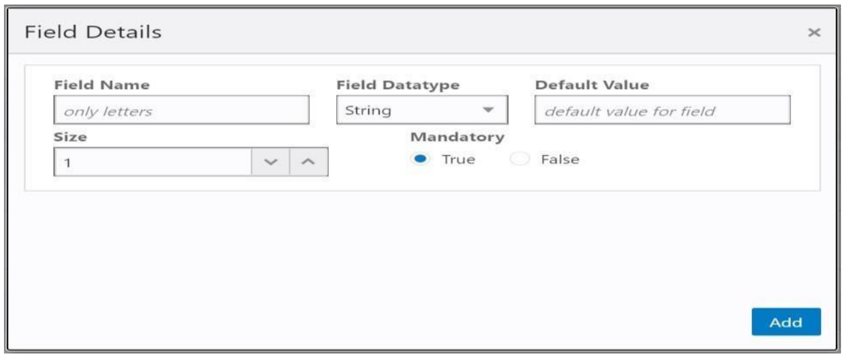
- Entity Details
- Field Details
- Child Entity Details
- Relationship Details

Entity Details

In this section you will capture the entity name. As the Domain Entity pattern "an object is primarily defined by its identity is called an Entity."

Field Details

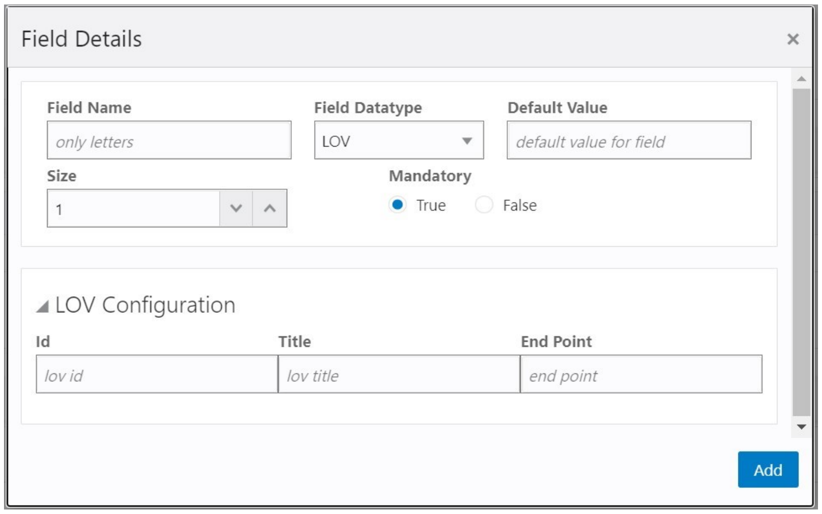
For the main entity you need to define the fields in this section. For doing that click on the Add button and provide the field details


Field Details

Field Name: only letters
Field Datatype: String
Default Value: default value for field
Size: 1
Mandatory: [✓ True] False

Following are the different types of field types supported in OBX:

- **String**: This is inbuild field type of OBX, it gets translated to varchar for sql scripts, string type in java files and normal text field in UI component
- **Integer**: This is inbuild field type of OBX, it gets translated to number for sql scripts, integer type in java files and normal text field in UI component
- **Float:** This is inbuild field type of OBX, it gets translated to number for sql scripts, float type in java files and normal text field in UI component
- **LOV:** This field type is inherited from the base product and has its own configuration as below


Field Details

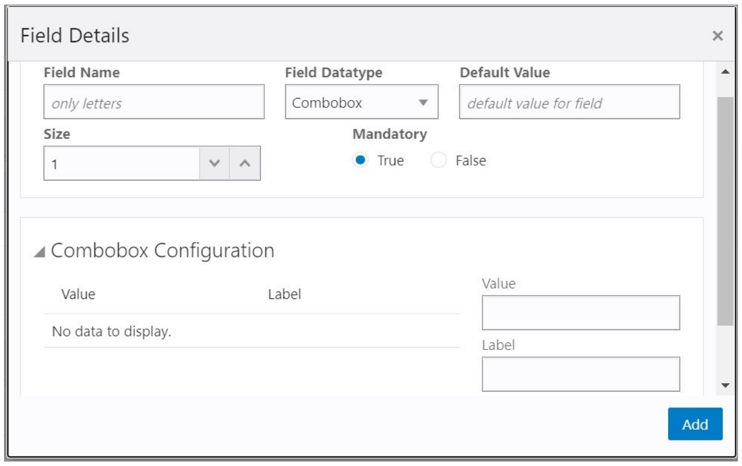
Field Name: only letters
Field Datatype: LOV
Default Value: default value for field
Size: 1
Mandatory: [✓ True] False

LOV Configuration

Id: lov id
Title: lov title
End Point: end point

Here, ID is the specific id given to this LOV component, Title is displayed on the LOV dialog box and End -point is the service end-point which this field connects to for fetching values

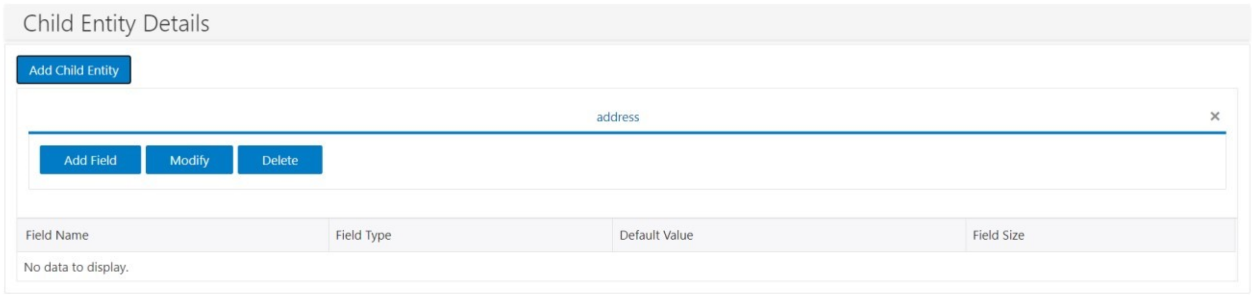
- **Date:** This field is also inherited from the base product and add date component on the screen
- **Amount:** This field is also inherited from the base product and add the amount field on the screen. This field also captures currency along with the amount
- **Combo box:** This field is taken from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



- **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

Child Entity Details

Use this block for adding the child entities. Once clicked on 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

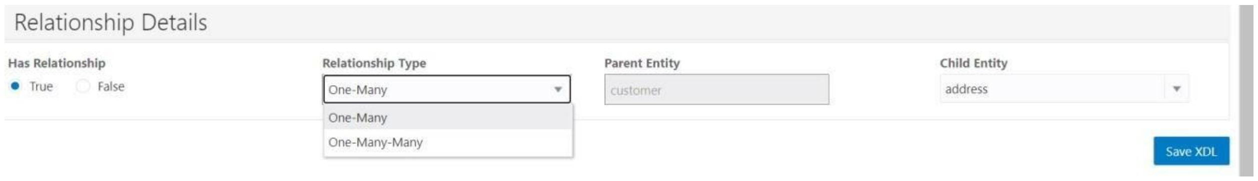


Please add the child entity field details in a similar way like we added for main entity

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



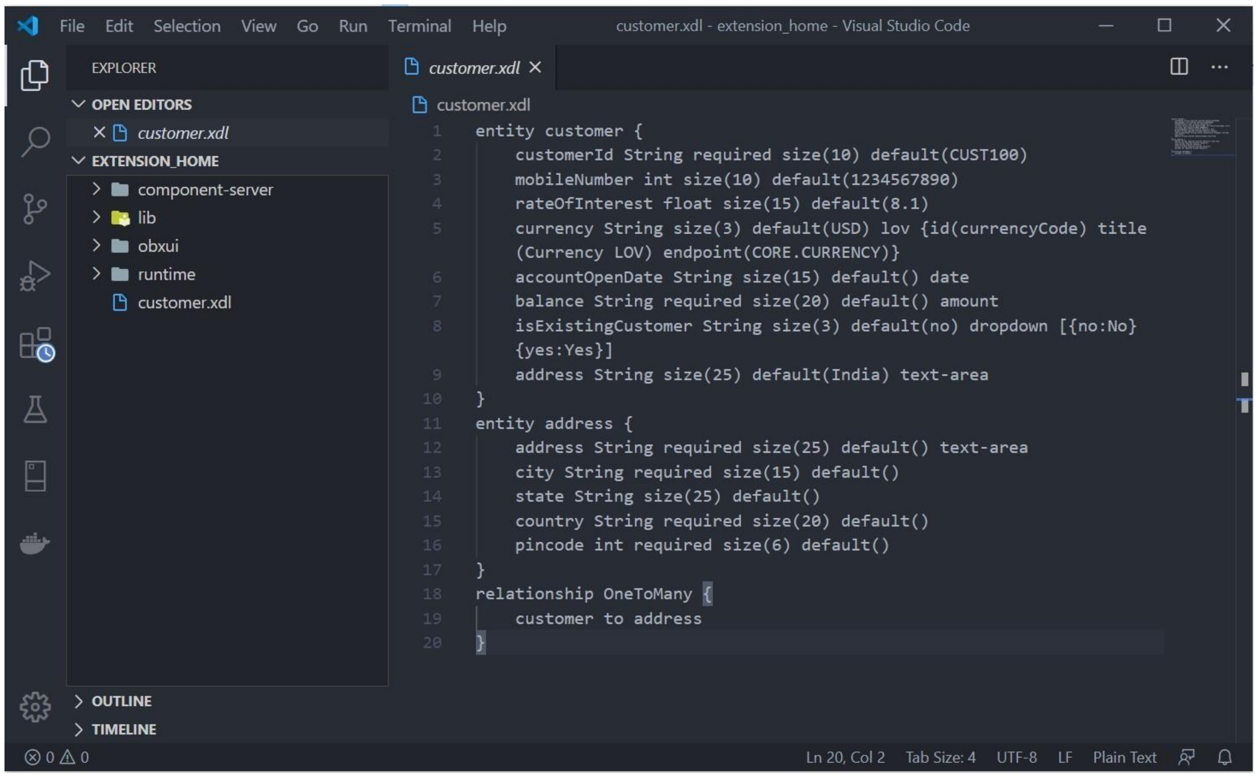
Relationship Details

Has Relationship	Relationship Type	Parent Entity	Child Entity
● True ○ False	One-Many	customer	address

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)

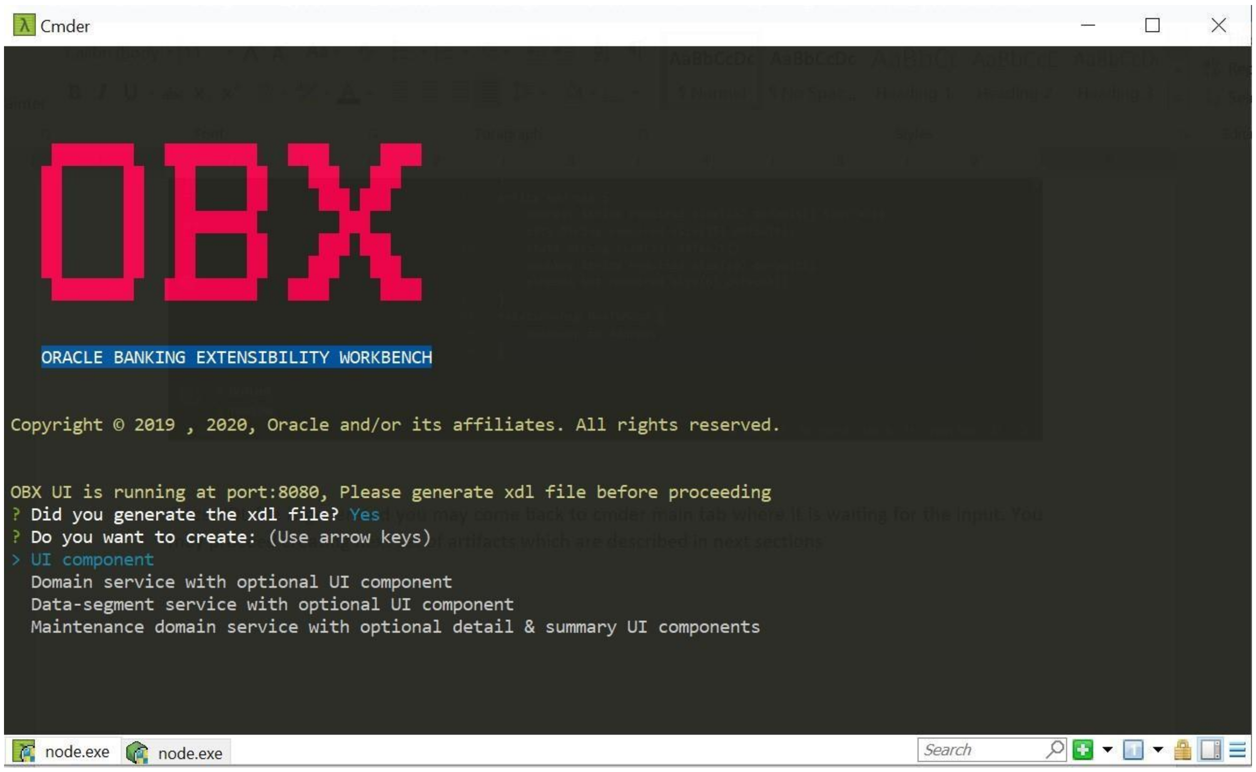
The final XDL file looks like this:



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



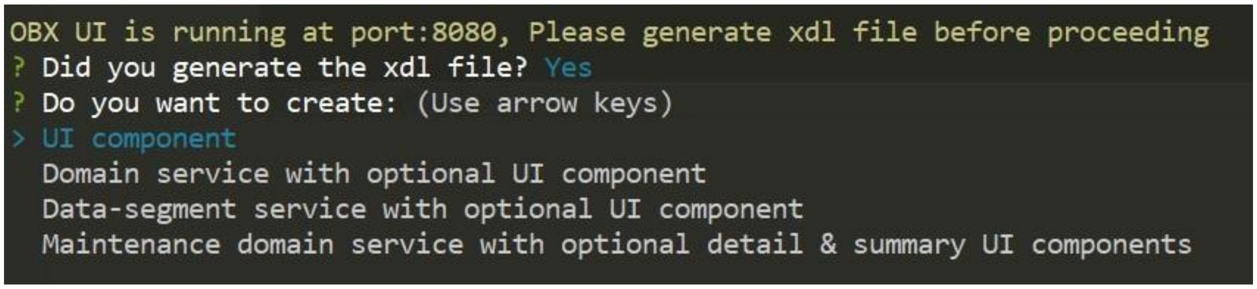
```
ORACLE BANKING EXTENSIBILITY WORKBENCH


Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.


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
```

2. Service Extensions

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.

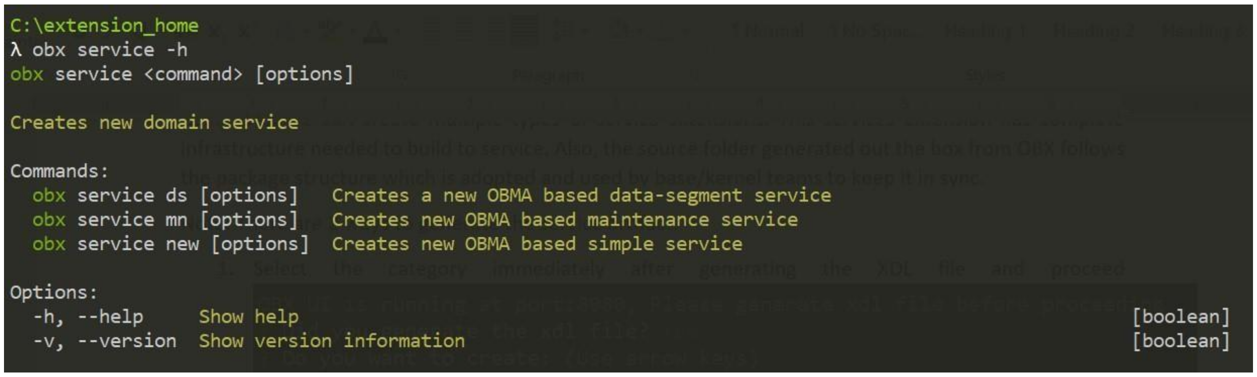
Note: There are 2 ways to generate the service artifact:

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



```
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
```

2. Use the service specific command to generate different types



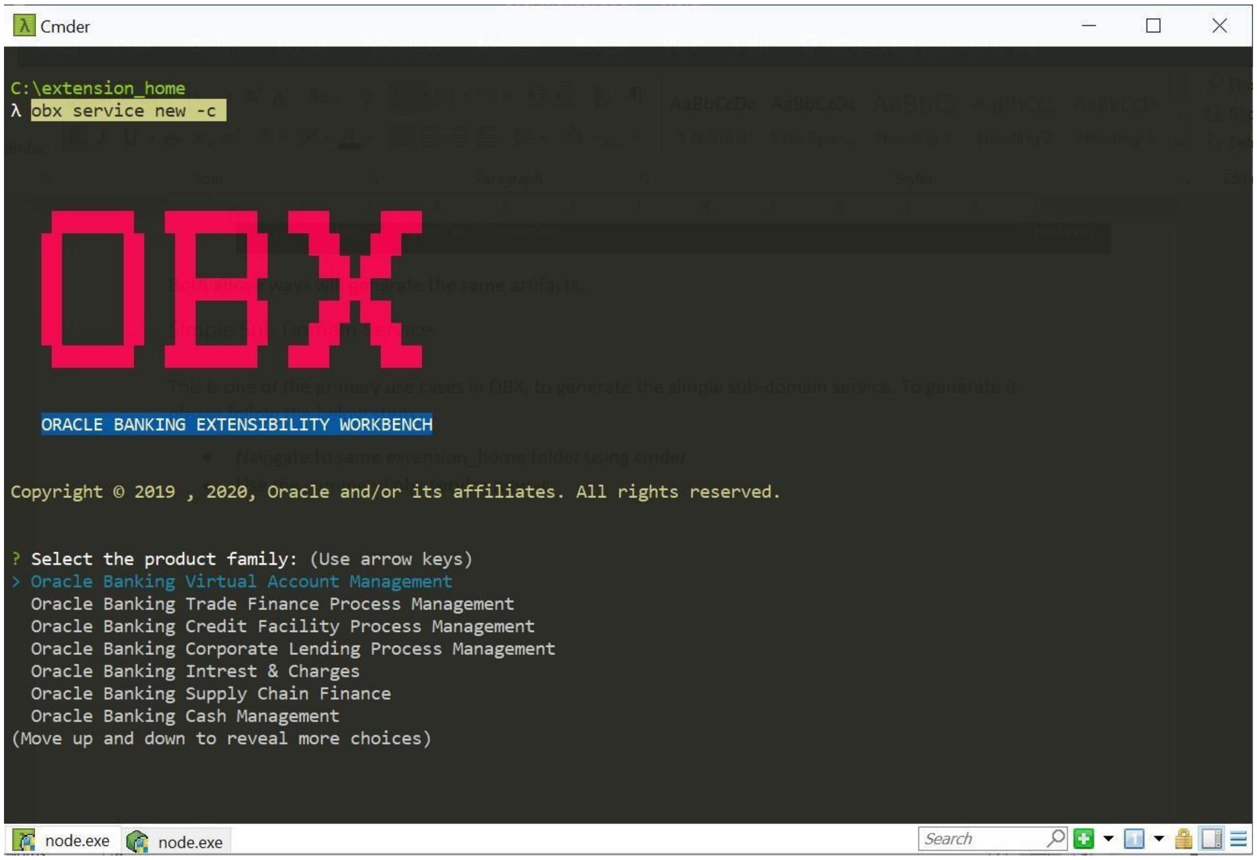
```
C:\extension_home
λ obx service -h
obx service <command> [options]

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

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

Both above ways will generate the same artifacts.

Simple Sub Domain Service

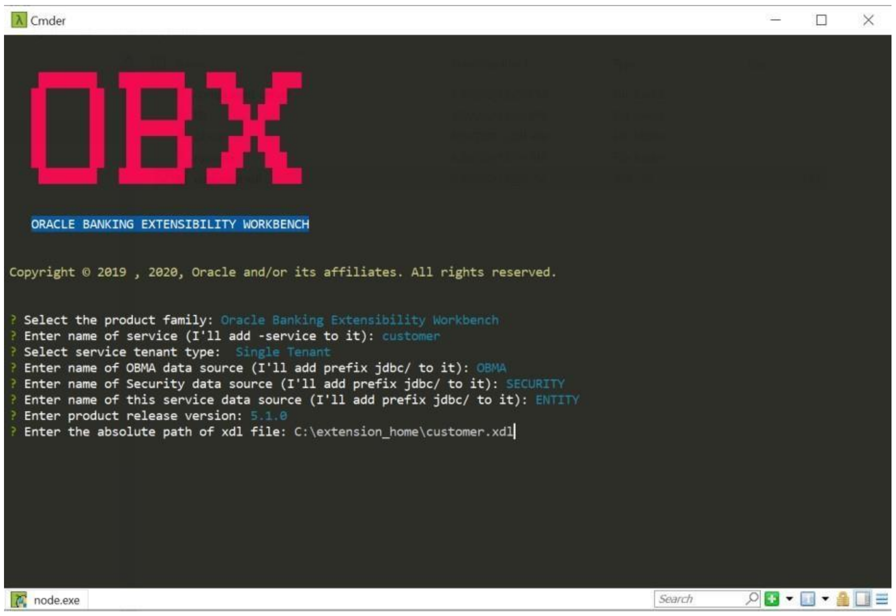
This is one of the primary use cases in OBX, to generate the simple sub-domain service. To generate it please follow the below steps:

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



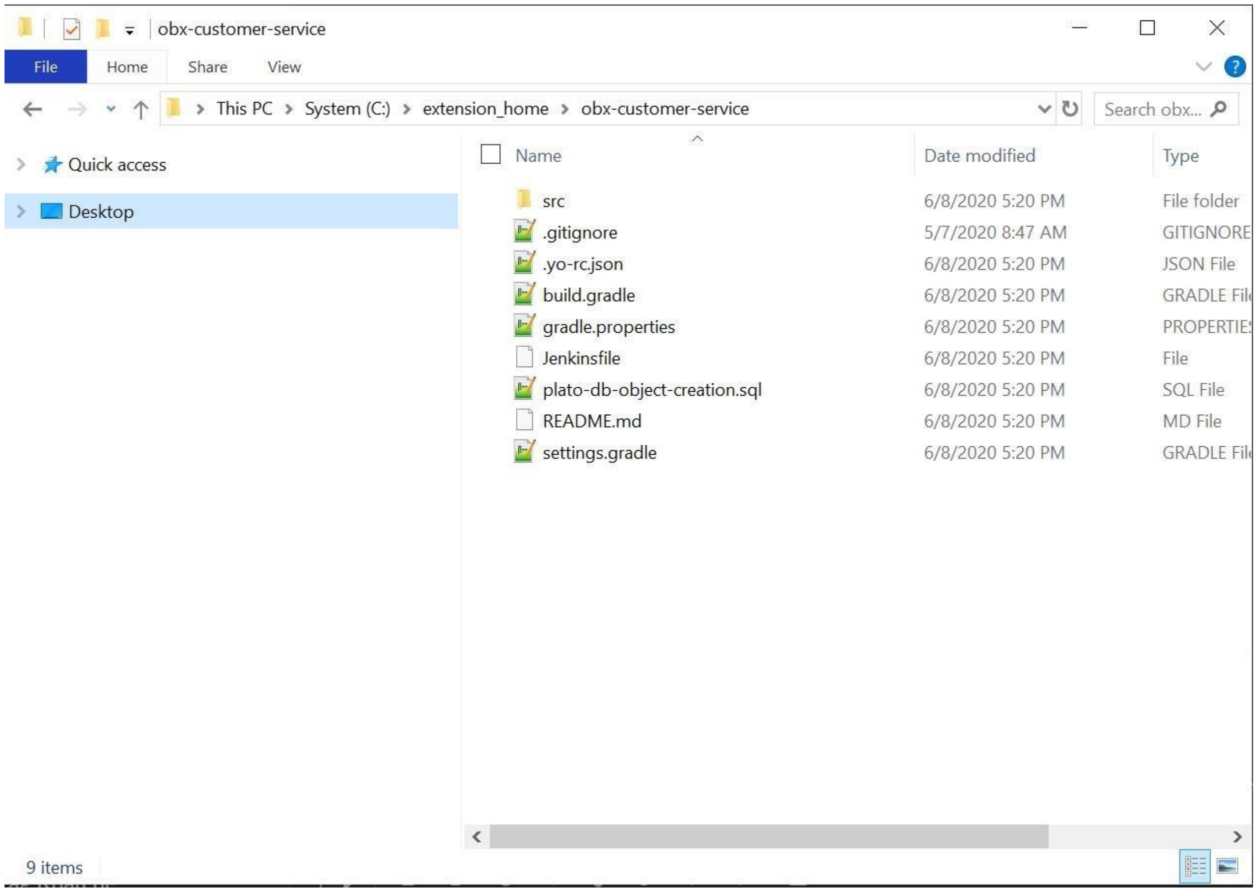
```
C:\extension_home
λ obx service new -c

OBX

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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)
```

- 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



```
ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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 OBMA data source (I'll add prefix jdbc/ to it): OBMA
? Enter name of Security data source (I'll add prefix jdbc/ to it): SECURITY
? Enter name of this service data source (I'll add prefix jdbc/ to it): ENTITY
? Enter product release version: 5.1.0
? Enter the absolute path of xdl file: C:\extension_home\customer.xdl
```

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



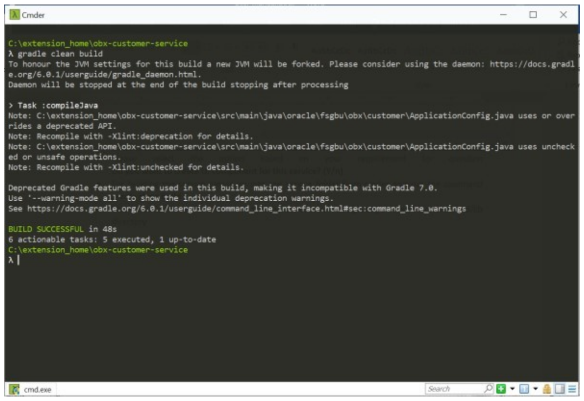
This PC > System (C:) > extension_home > obx-customer-service

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

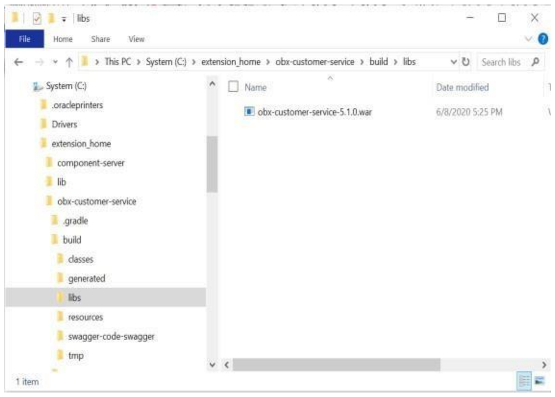
9 items

- Please select the option based on your requirement for question **Do you want to create UI component for this service? (Y/n)**
- For building the service please go into the service folder from cmder and run the command **gradle clean build**

- This will build the service and we can find the war of the service getting created inside the build/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
```



- Use this service and deploy it in your environment

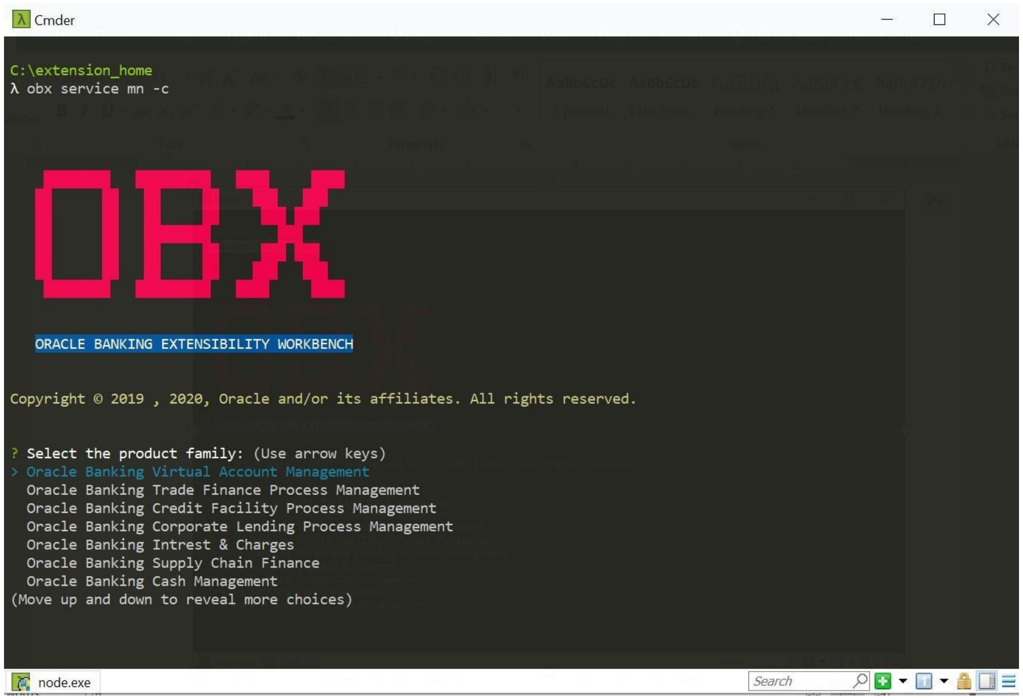
Notes:

- DB scripts for the service will be generated inside the folder **\extension_home\obxcustomerservice\src\main\resources\db**
- Please 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, please verify **PLATO_TABLE_SCRIPT**.sql before executing it in the schema it may contain some dummy values

Maintenance sub domain service

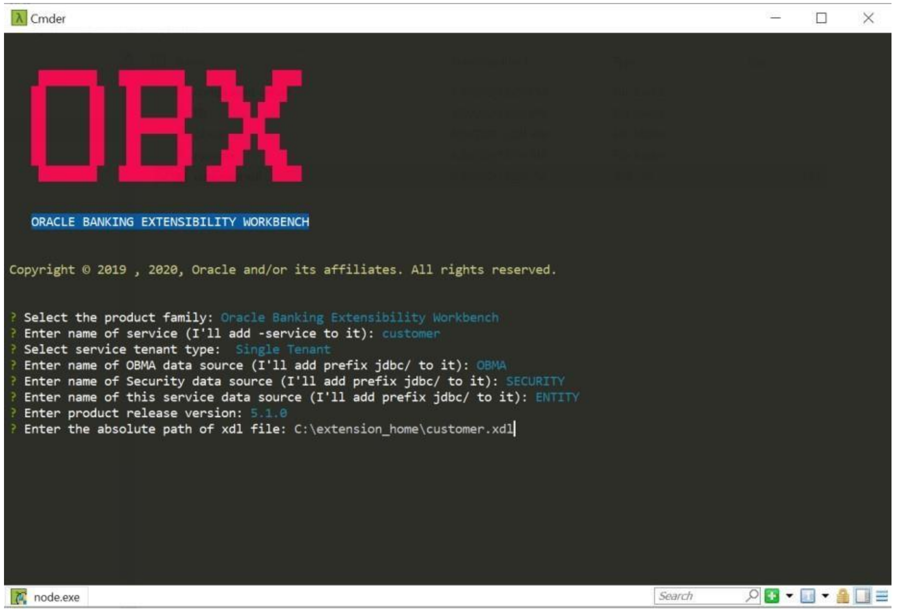
This section describes the process to generate the maintenance type of service. Maintenance service generally has concept of main and work table. This allows enables functionality where all the Authorized records goes to main table and all the unauthorized records goes to work table. Also, with this type of service we attach audit details to payload. To generate it please follow the below steps:

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



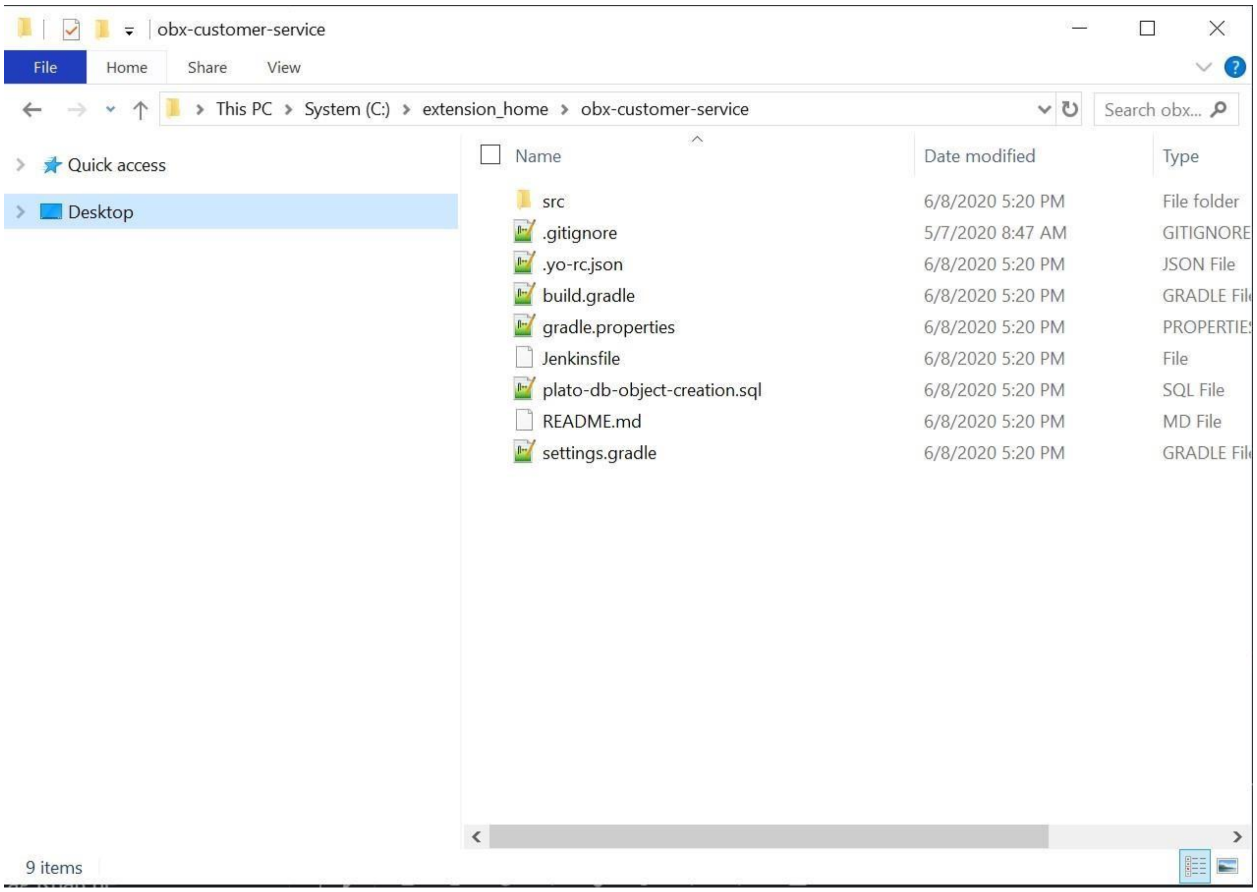
```
C:\extension_home
λ obx service mn -c

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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)
```

- 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



```
ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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 OBMA data source (I'll add prefix jdbc/ to it): OBMA
? Enter name of Security data source (I'll add prefix jdbc/ to it): SECURITY
? Enter name of this service data source (I'll add prefix jdbc/ to it): ENTITY
? Enter product release version: 5.1.0
? Enter the absolute path of xdl file: C:\extension_home\customer.xdl
```

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

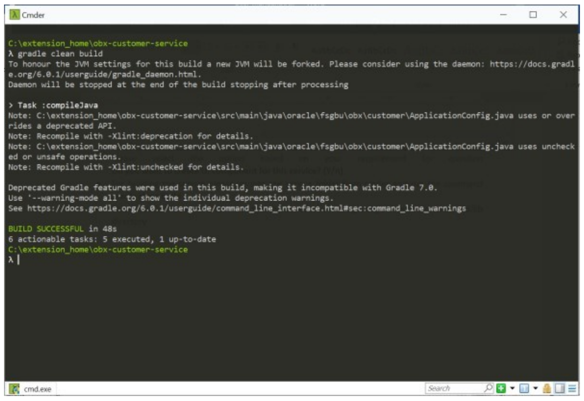


This PC > System (C:) > extension_home > obx-customer-service

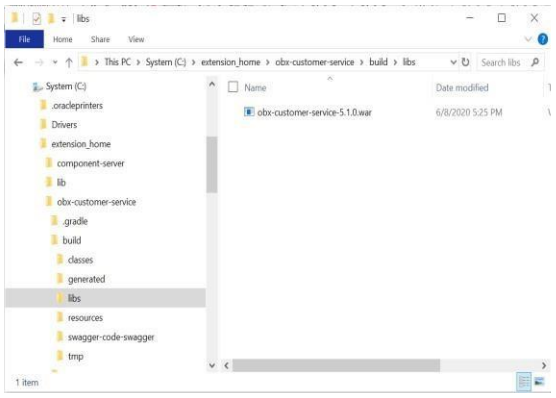
Name	Date modified	Type
src	6/8/2020 5:20 PM	File folder
.gitignore	5/7/2020 8:47 AM	GITIGNORE
.yo-rc.json	6/8/2020 5:20 PM	JSON File
build.gradle	6/8/2020 5:20 PM	GRADLE Fil
gradle.properties	6/8/2020 5:20 PM	PROPERTIE
Jenkinsfile	6/8/2020 5:20 PM	File
plato-db-object-creation.sql	6/8/2020 5:20 PM	SQL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

9 items

- Please select the option based on your requirement for question **Do you want to create a Maintenance and Summary Components for this service? (Y/n)**
- For building the service please go into the service folder from cmder and run the command **gradle clean build**
- This will build the service and we can find the war of the service getting created inside the build/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
```



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

- Use this service and deploy it in your environment

Notes:

- DB scripts for the service will be generated inside the folder **\extension_home\obxcustomerservice\src\main\resources\db**
- Please 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 SMS (Security Management System) 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

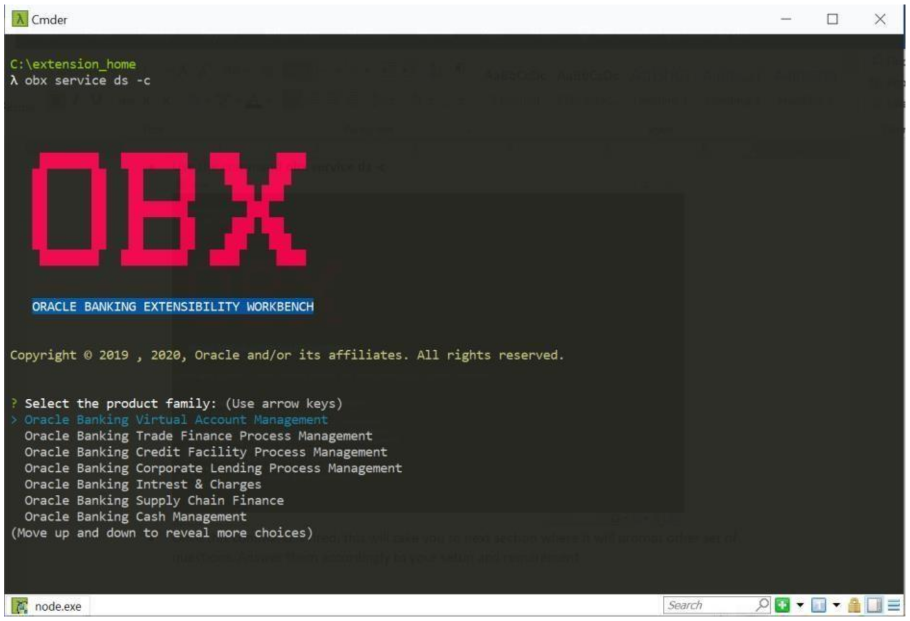
Data/Resource Segment sub domain service

This section 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

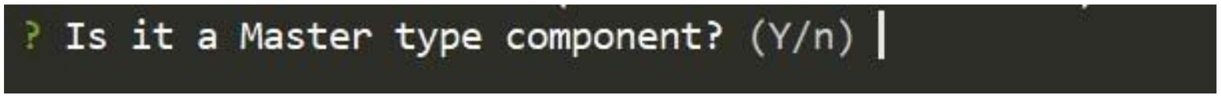
Child Type: This is primarily used when user wants to attach a single data-segment in the existing flow/process. Generally, this existing flow/process is available in the base product. We use the same screen class code from base and attach our data segment to it To generate it please follow the below steps:

- Navigate to same extension_home folder using cmder
- Use the command **obx service ds -c**



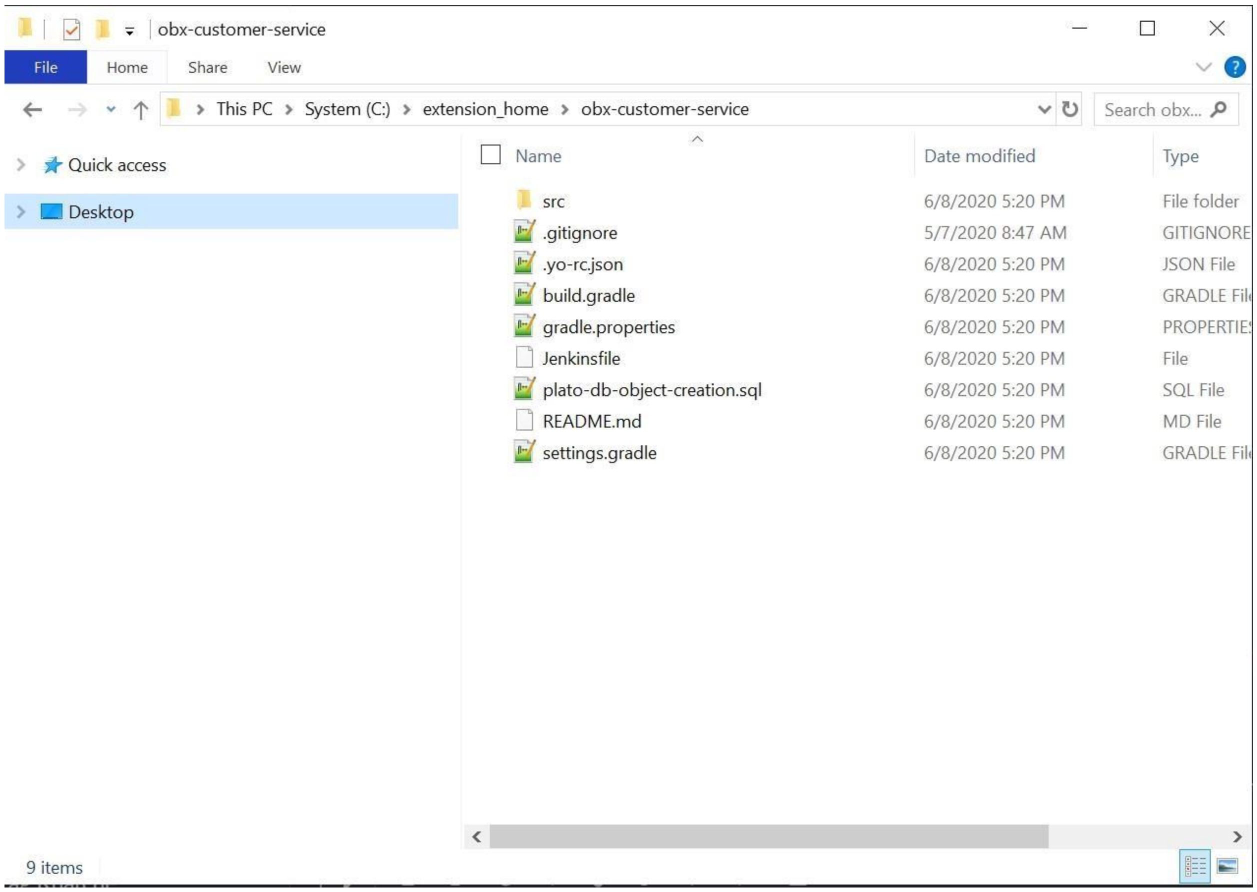
```
C:\extension_home
λ obx service ds -c

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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)
```

- 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
- Select the type of component according to your requirement



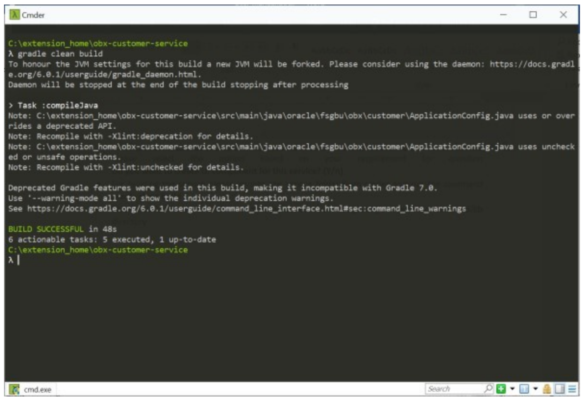
```
? Is it a Master type component? (Y/n)
```

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

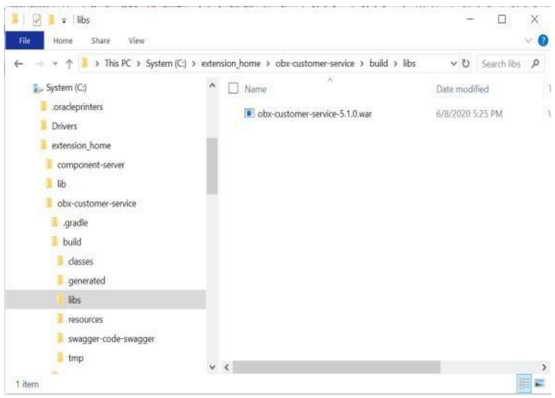


Name	Date modified	Type
src	6/8/2020 5:20 PM	File folder
.gitignore	5/7/2020 8:47 AM	GITIGNORE
.yo-rc.json	6/8/2020 5:20 PM	JSON File
build.gradle	6/8/2020 5:20 PM	GRADLE Fil
gradle.properties	6/8/2020 5:20 PM	PROPERTIE
Jenkinsfile	6/8/2020 5:20 PM	File
plato-db-object-creation.sql	6/8/2020 5:20 PM	SQL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

- Please select the option based on your requirement for question **Do you want to create a Data Segment for this service? (Y/n)**
- For building the service please go into the service folder from cmder and run the command **gradle clean build**
- This will build the service and we can find the war of the service getting created inside the build/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 |
```



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

- Use this service and deploy it in your environment

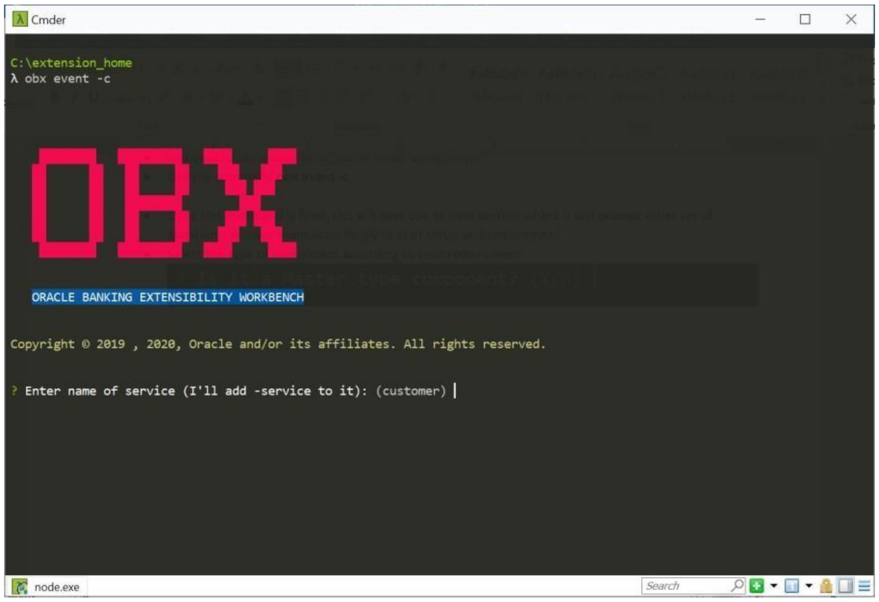
Notes:

- DB scripts for the service will be generated inside the folder **\extension_home\obxcustomerservice\src\main\resources\db**
- Please 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 SMS (Security Management System) 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**
- Please execute them in the CMC schema.
- **Screen Class and Data Segment** has to be maintained from the UI which is present under common core

Simple Publisher/Subscriber Event Service

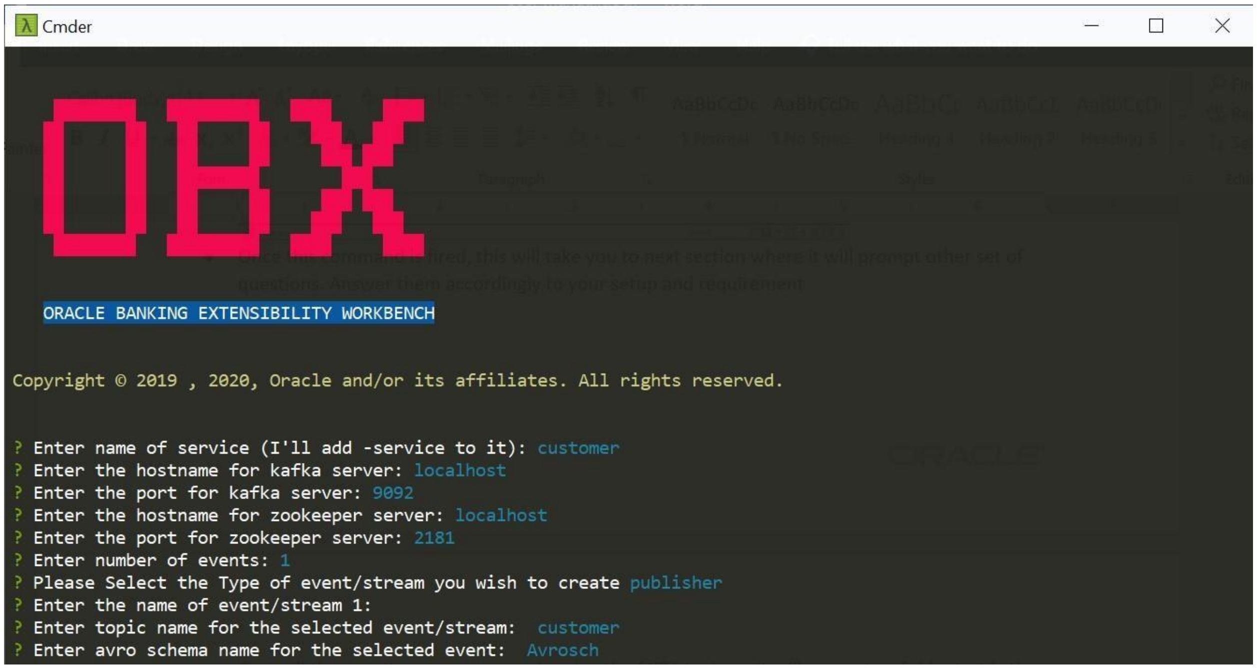
This section describes the process to generate simple publisher/subscriber event service. To generate it please follow the below steps:

- Navigate to same extension_home folder using cmder
- Use the command **obx event -c**



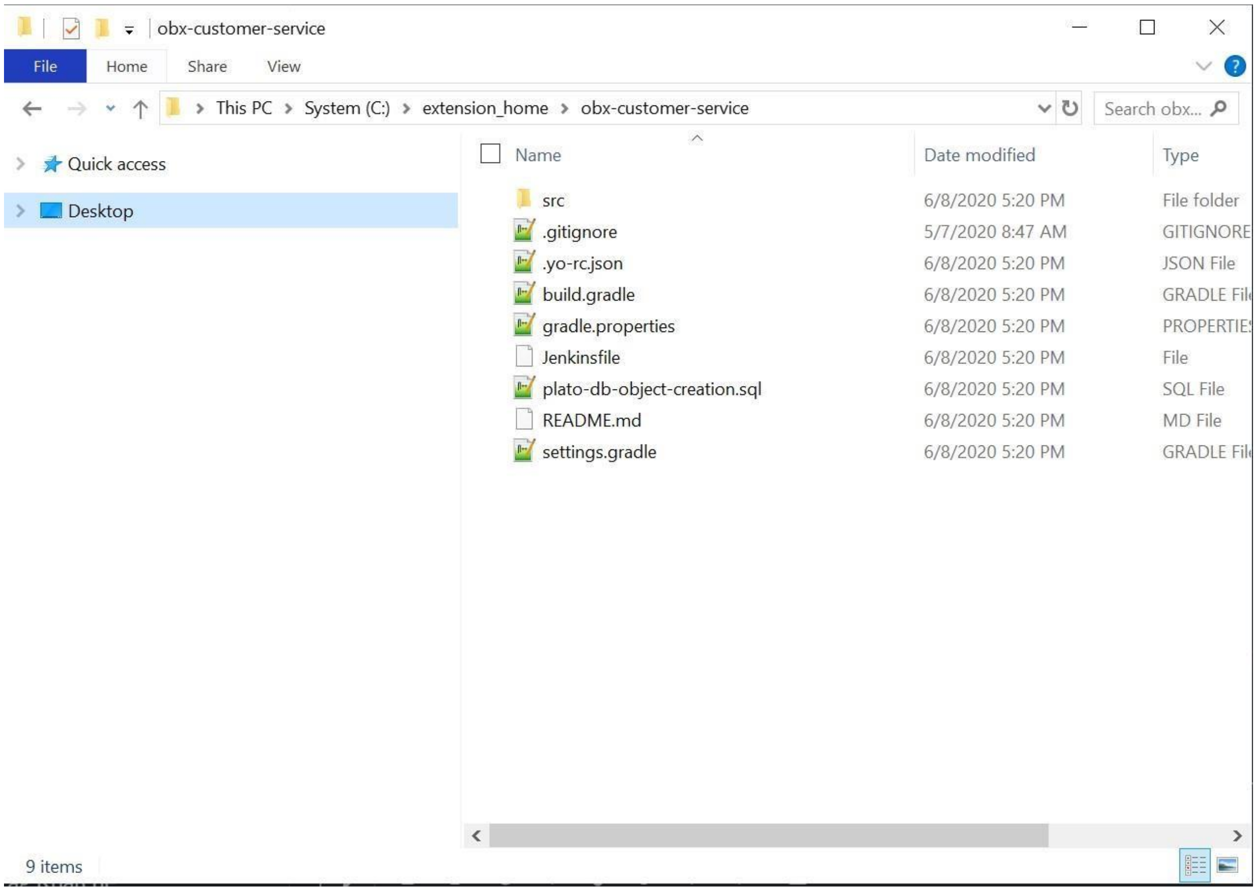
```
C:\extension_home
λ obx event -c

OBX

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.

? Enter name of service (I'll add -service to it): (customer)
```

- 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



```
ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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
```

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



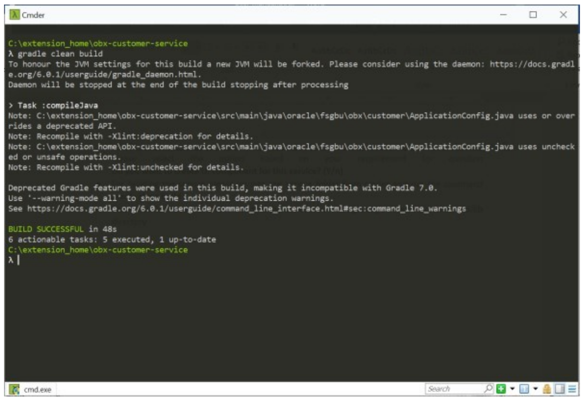
This PC > System (C:) > extension_home > obx-customer-service

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

- For building the service please go into the service folder from cmder and run the command **gradle clean build**

- This will build the service and we can find the war of the service getting created inside the build/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 |
```

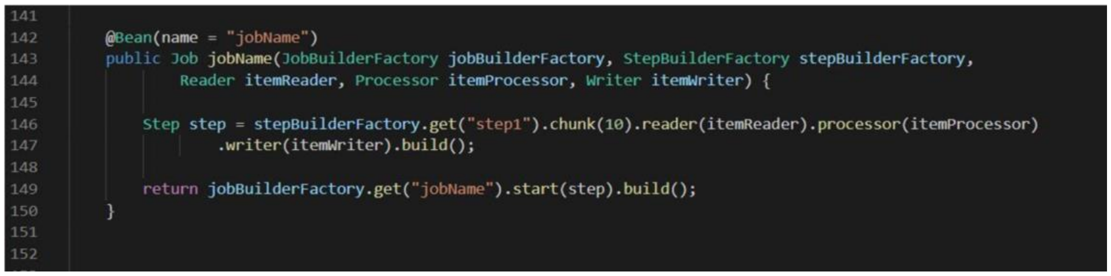
- Use this service and deploy it in your environment

Batch Service

This section describes the process to generate 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 it please follow the below steps:

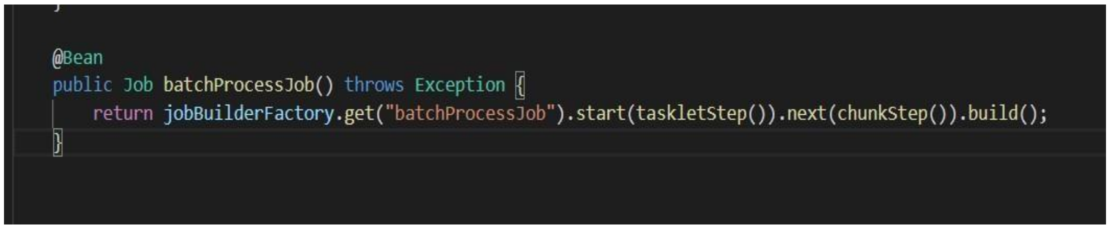
- Navigate to same extension_home folder using cmder
- Use the command **obx batch -c**
- Inputs to be given after the command ○ Select the product family ○ Enter name of the service(I'll construct it as <productFamilyName>-batch<serviceName>-extended-services):
    - Enter product release version

- Upon successful creation of batch service, user will find a folder generated with <productFamilyName>-batch-<serviceName>-extended-services having the sample service code generated
- The generated code has two types of batch job template inside. ○ **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.



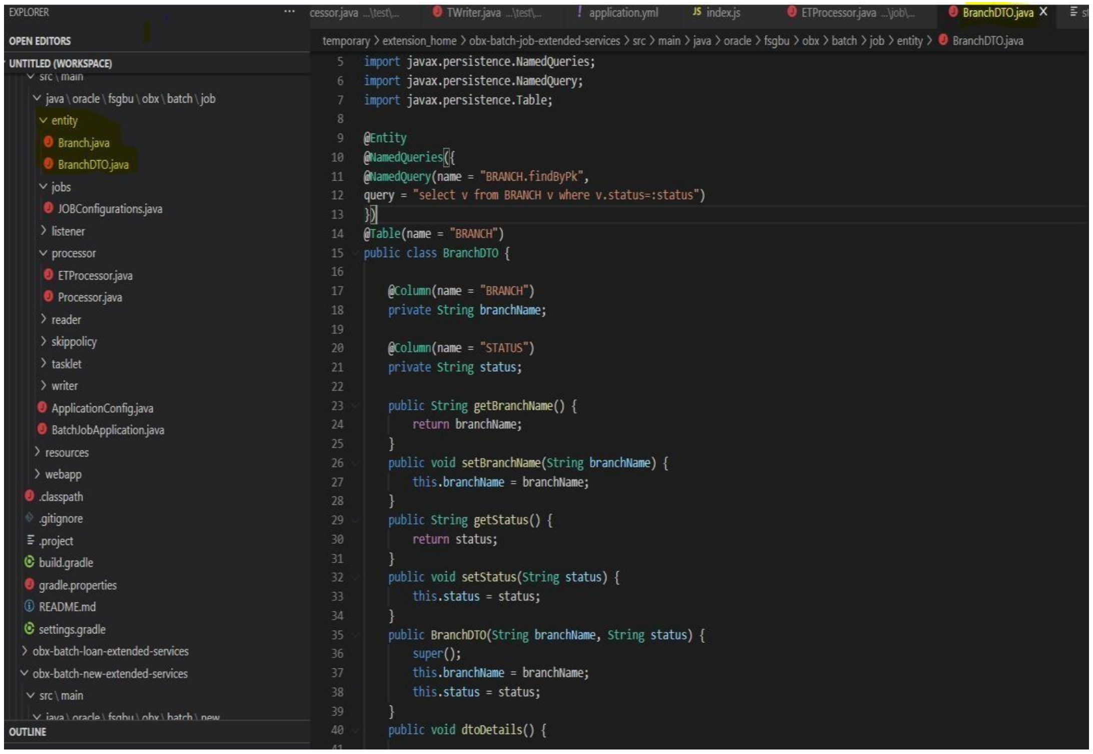
```
    @Bean(name = "jobName")
    public Job jobName(JobBuilderFactory jobBuilderFactory, StepBuilderFactory stepBuilderFactory,
            Reader itemReader, Processor itemProcessor, Writer itemWriter) {

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

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

○ **Plato batch type** job creation by keeping plato batch into consideration. 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.



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

- 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.



```java
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() {
```

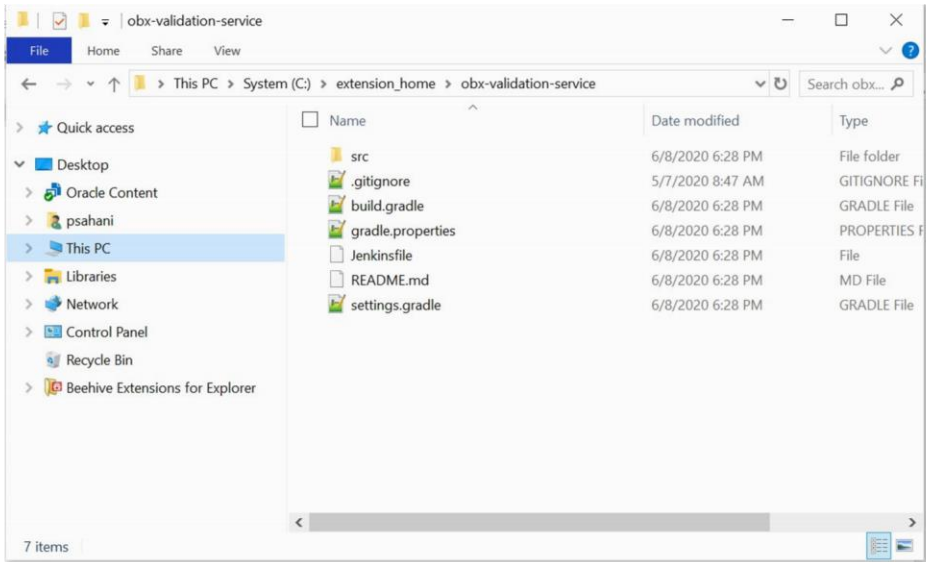
  - One needs to write methods for reader, writer and processor accordingly.

- To build the service ○ Navigate to the service.
  - Fire the command gradle clean build.
  - This will create the war file of the service in the folder structure build/libs/ productFamilyName>-batch-<serviceName>-extended-services.war
  -

Custom Validation Service

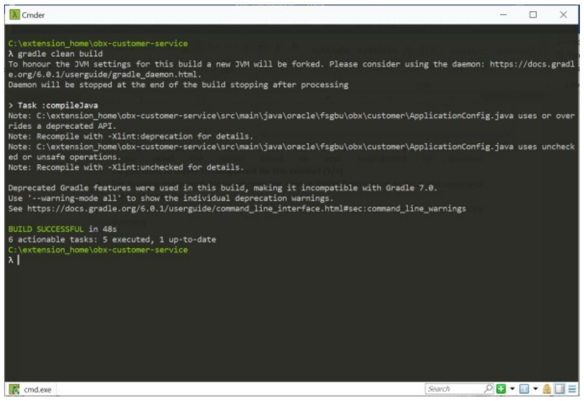
This section describes the process to generate validation service. The purpose of this service is to perform custom validations on the base service. It is important to remember that we will be only able to perform the validation and never modify the payload to change the value. To generate it please follow the below steps:

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

- For building the service please go into the service folder from cmder and run the command **gradle clean build**
- This will build the service and we can find the war of the service getting created inside the build/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 |
```

- Use this service and deploy it in your environment

Steps to adopt Multi Entity 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 logbac
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

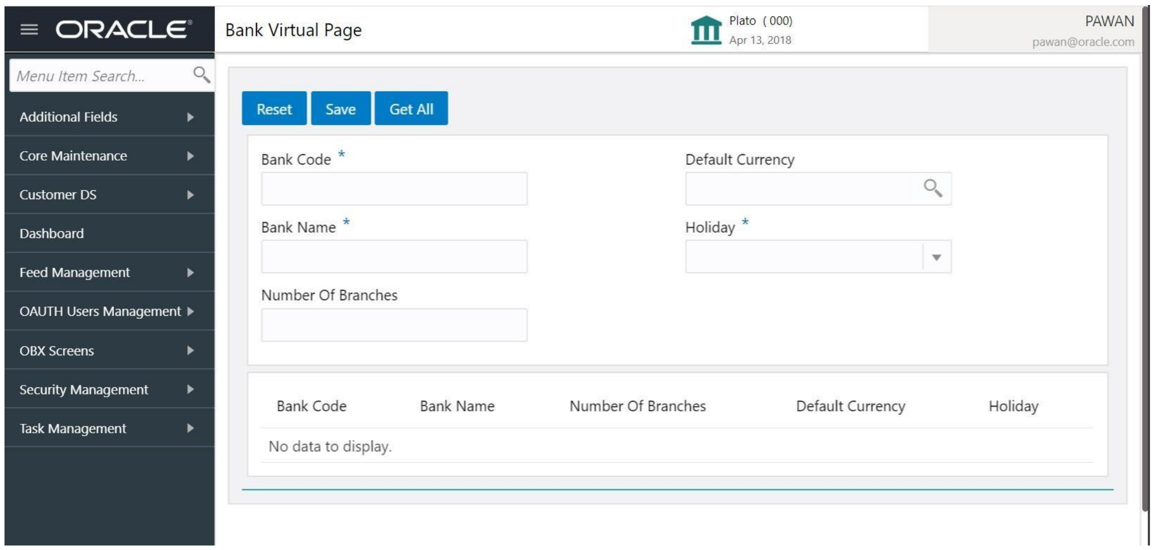
```
appId: <<appId>>
```

3. UI Extensions – Web Component

This section 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.

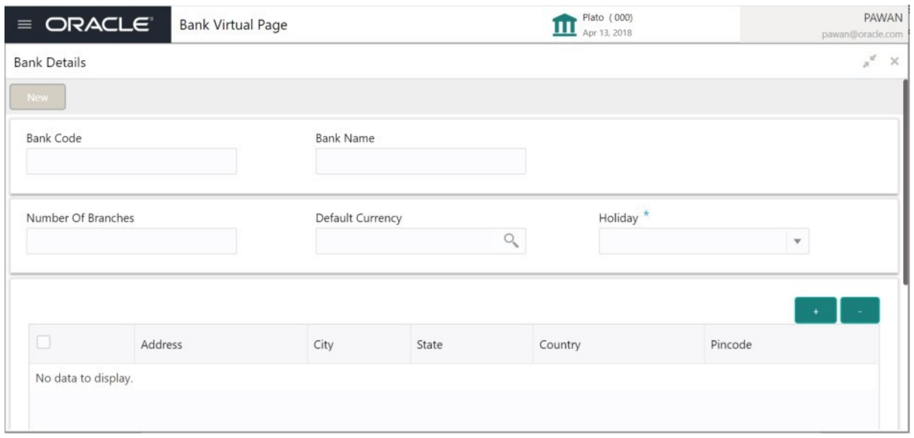
Standalone Component: A standalone component can be thought of as a smallest reusable UI component. They are generally the building blocks of main screens. Components like amount, text fields, lov etc. are part of standalone components

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



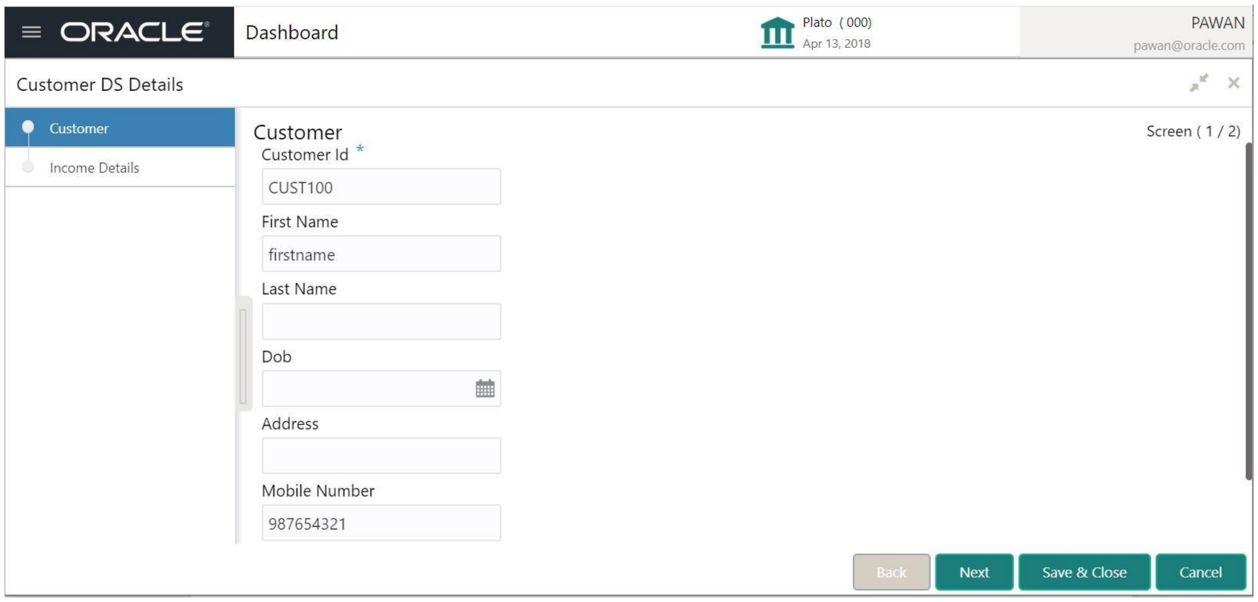
Bank Code	Bank Name	Number Of Branches	Default Currency	Holiday
No data to display.				

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



	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



Customer

Customer Id *: CUST100

First Name: firstname

Last Name:

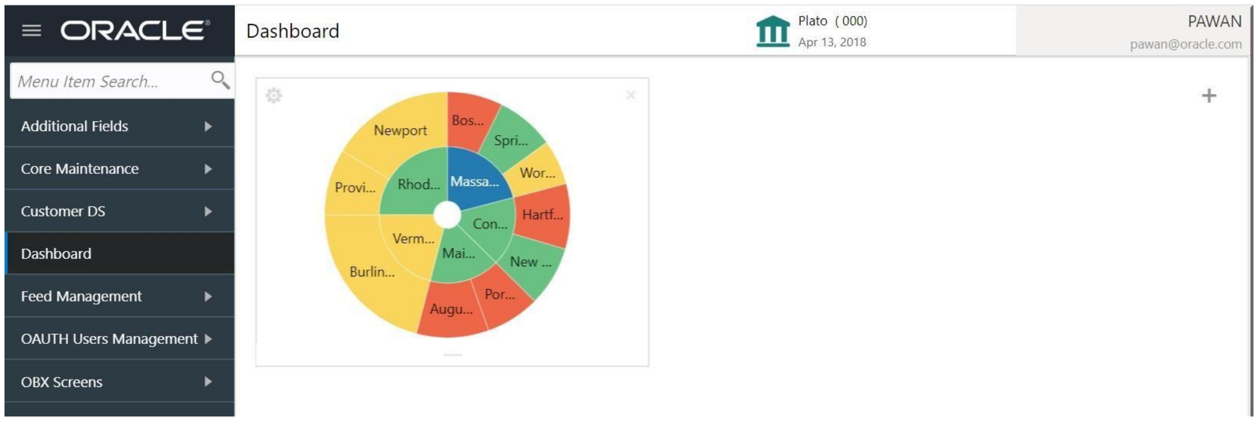
Dob:

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



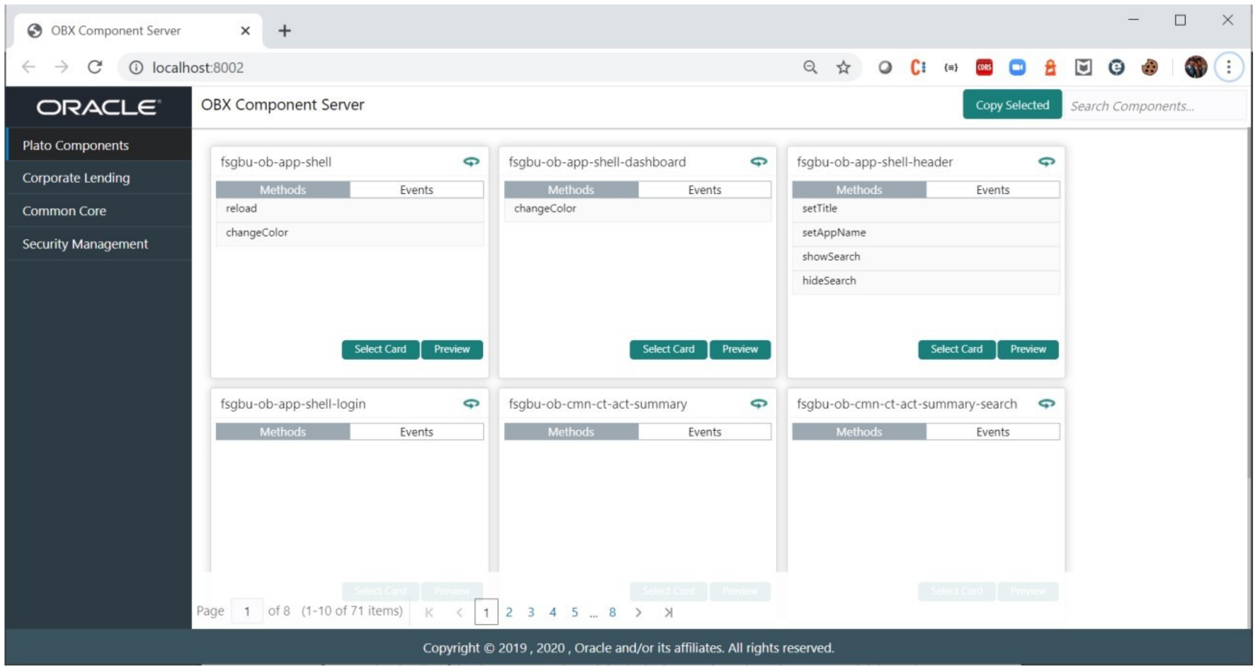
Note:

- 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

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 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.



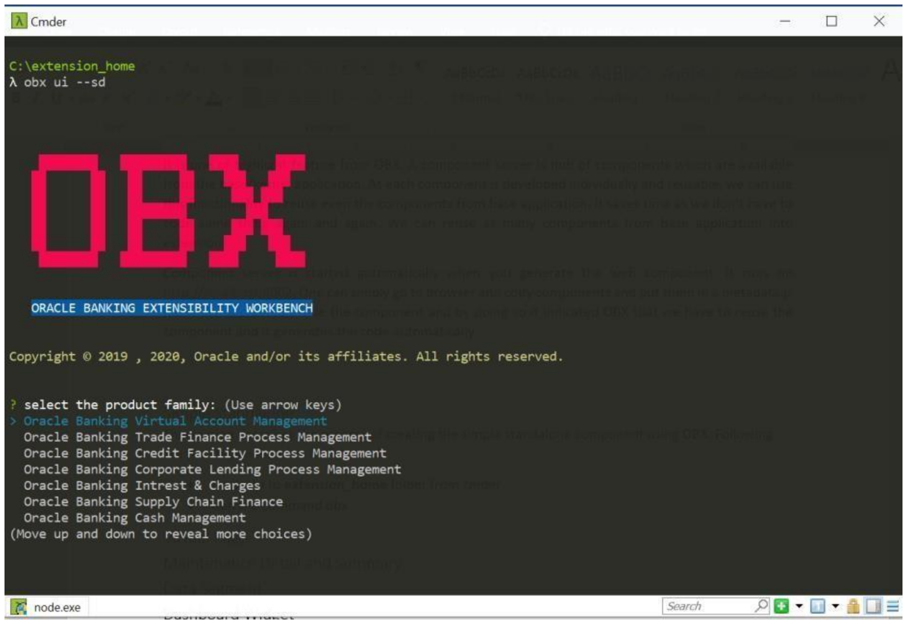
Simple Standalone

This section describes the process of creating the simple standalone component using OBX. Following are the steps needed to be followed:

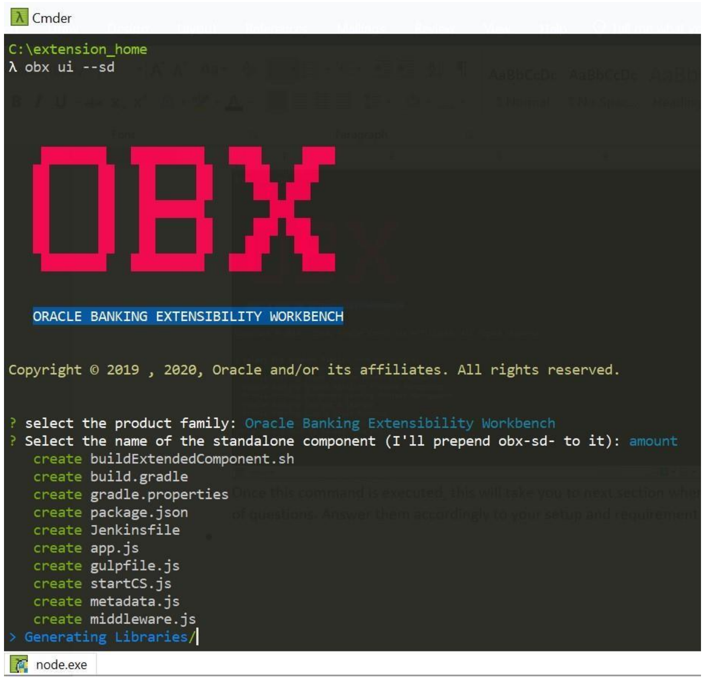
- Navigate to **extension_home** folder from cmder

•

Use the command **obx ui –sd**

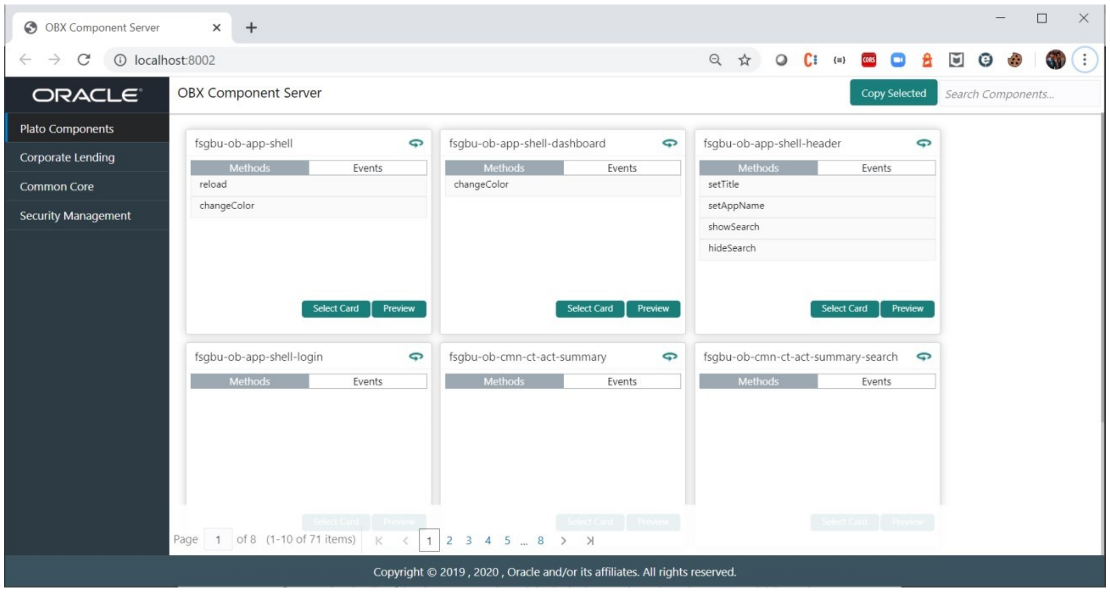

```
C:\extension_home
λ obx ui --sd

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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)
```

- 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

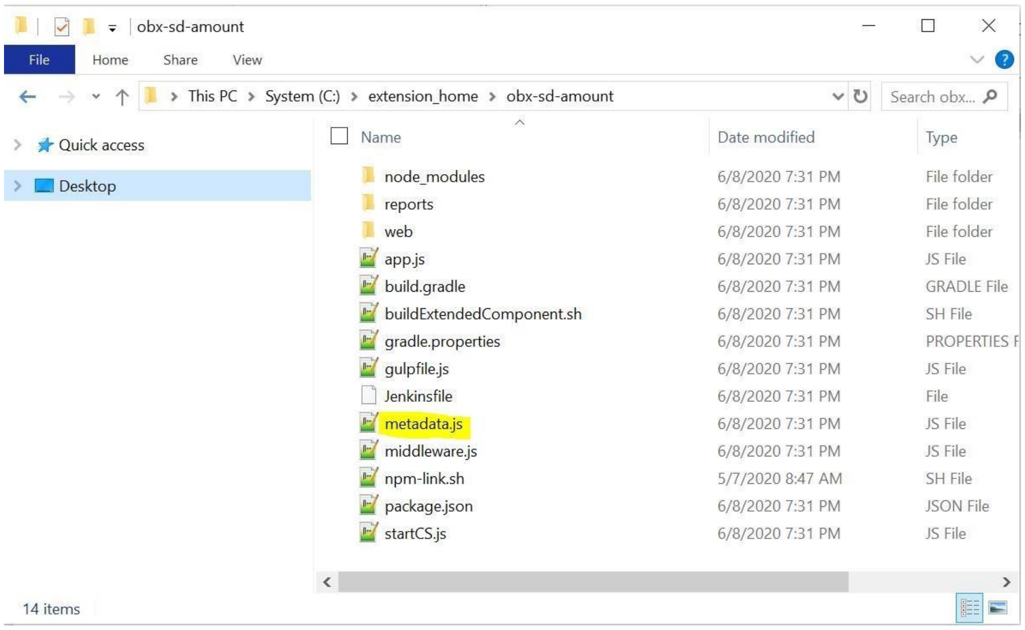

```
C:\extension_home
λ obx ui --sd

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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): amount
   create buildExtendedComponent.sh
   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/
```

- 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.
  At this point of time go to browser and navigate to http://localhost:8002. You will be able to component server home page like

•



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

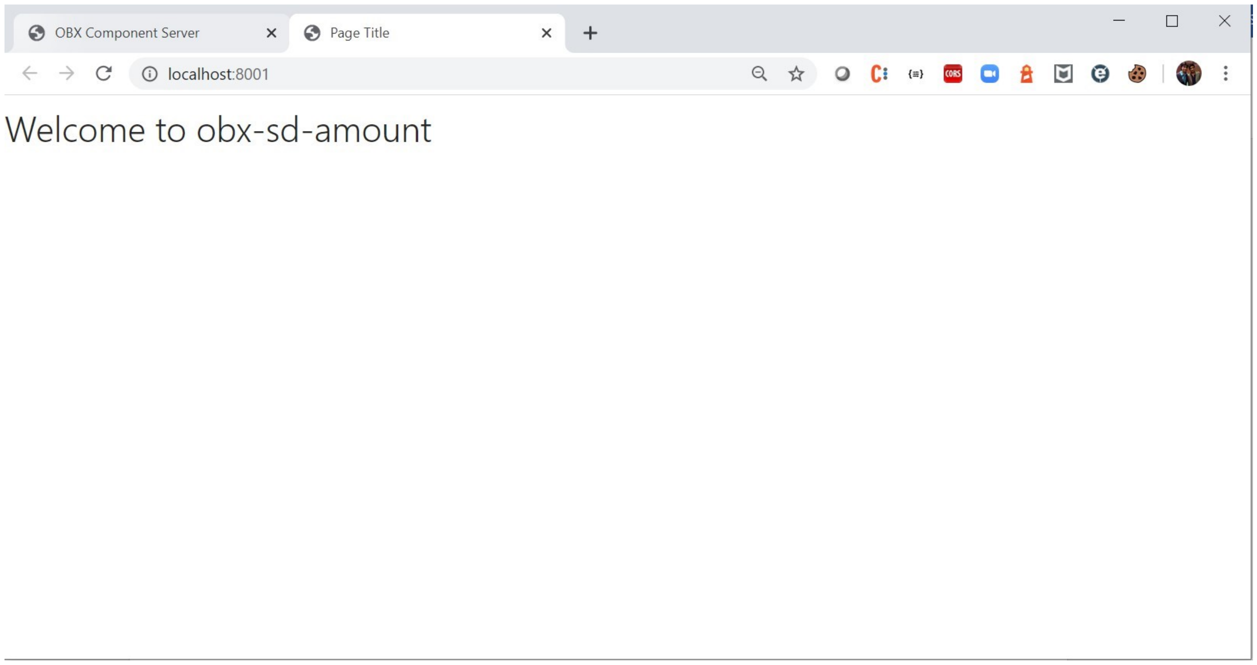


Name	Date modified	Type
node_modules	6/8/2020 7:31 PM	File folder
reports	6/8/2020 7:31 PM	File folder
web	6/8/2020 7:31 PM	File folder
app.js	6/8/2020 7:31 PM	JS File
build.gradle	6/8/2020 7:31 PM	GRADLE File
buildExtendedComponent.sh	6/8/2020 7:31 PM	SH File
gradle.properties	6/8/2020 7:31 PM	PROPERTIES 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

• 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?**

• 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

•

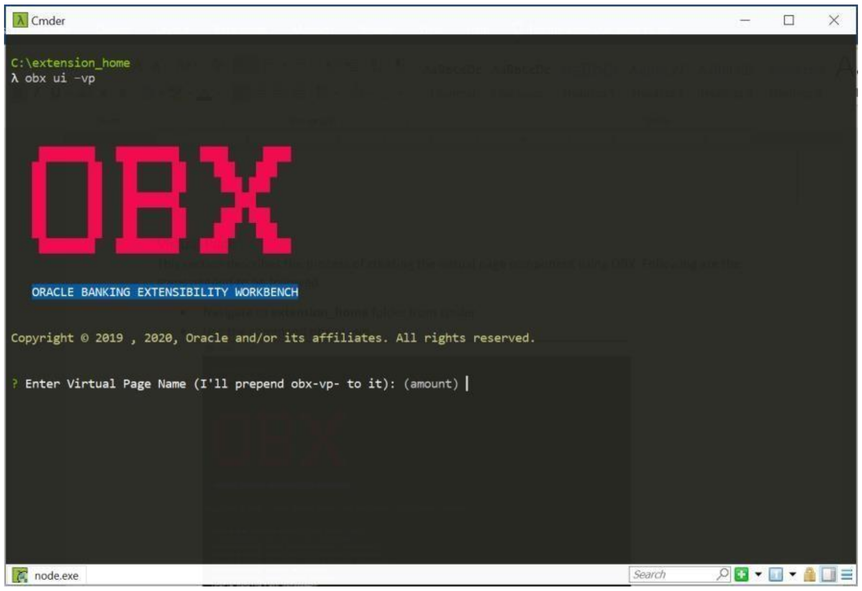


Welcome to obx-sd-amount

Virtual Page

This section describes the process of creating the virtual page component using OBX. Following are the steps needed to be followed:

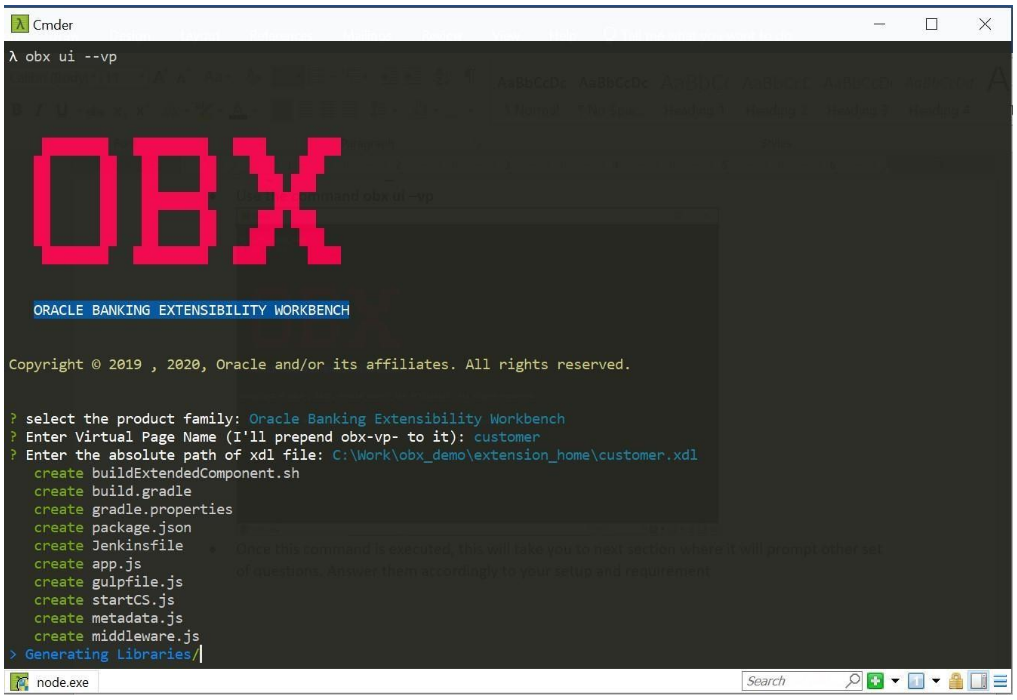
- Navigate to **extension_home** folder from cmder
- Use the command **obx ui –vp**



```
C:\extension_home
λ obx ui -vp

OBX

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.

? Enter Virtual Page Name (I'll prepend obx-vp- to it): (amount)
```

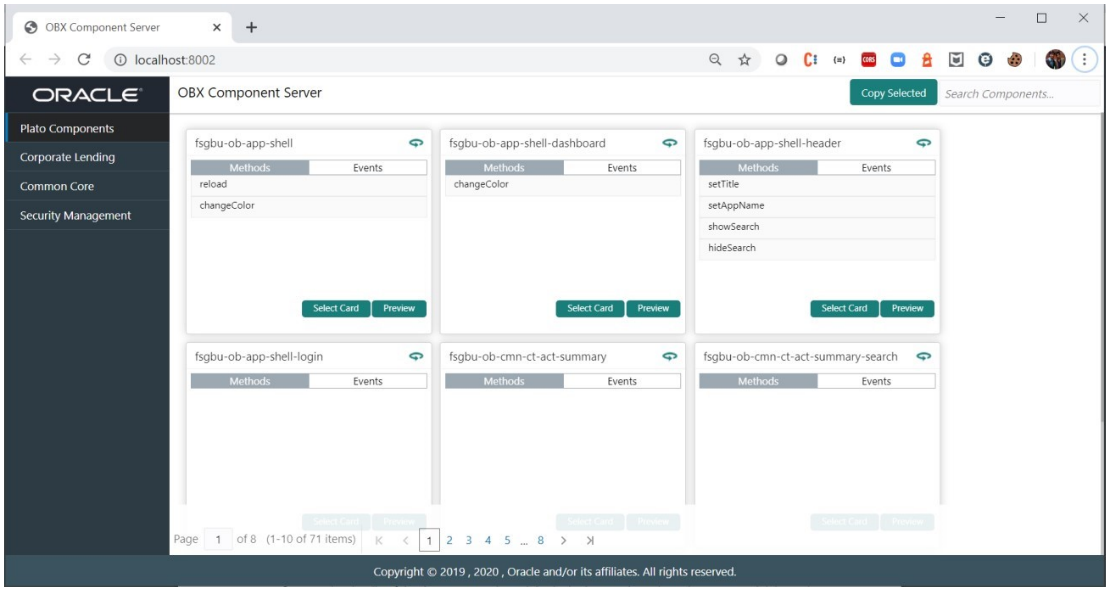
-

  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



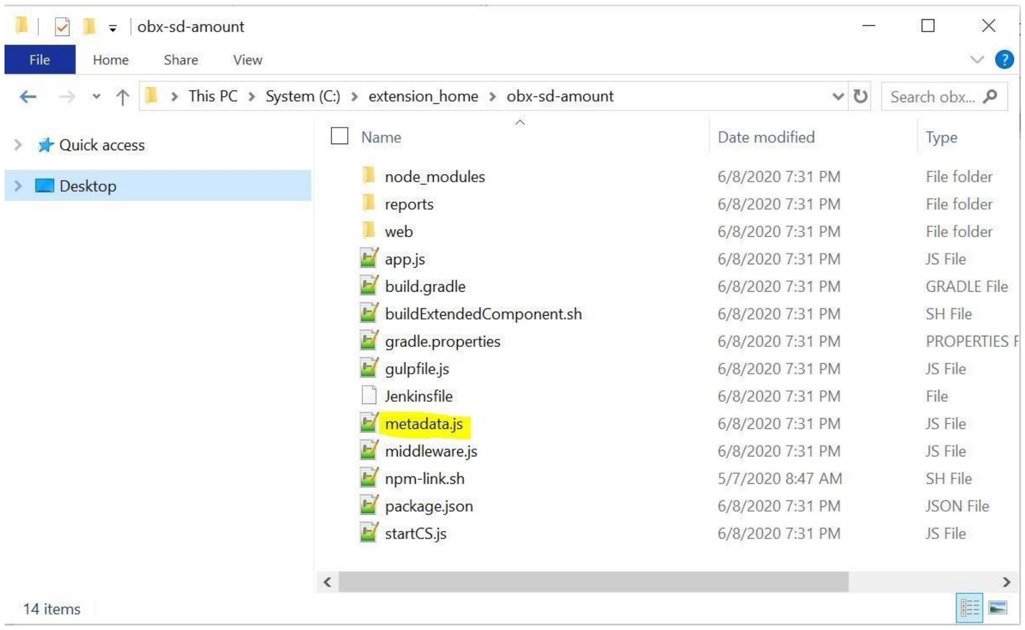
```
λ obx ui --vp

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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): customer
? Enter the absolute path of xdl file: C:\Work\obx_demo\extension_home\customer.xdl
   create buildExtendedComponent.sh
   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/
```

- 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.
- At this point of time go to browser and navigate to http://localhost:8002. You will be able to component server home page like



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

•

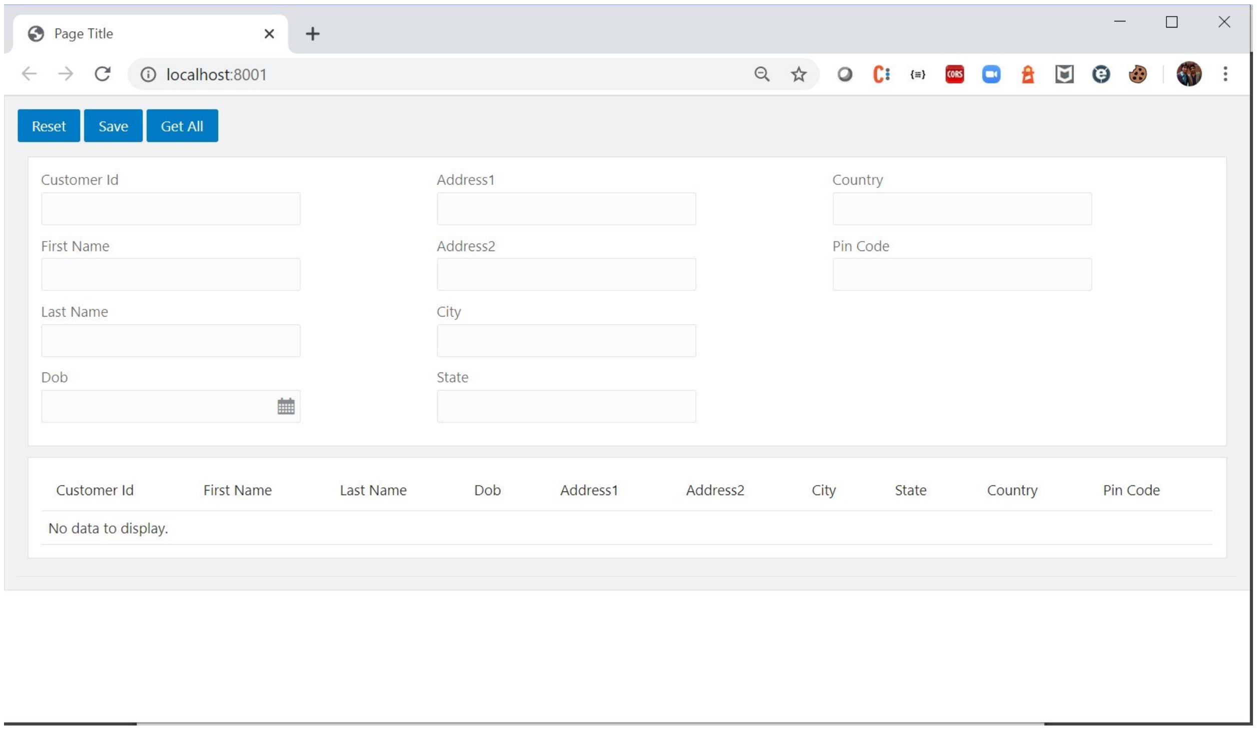

obx-sd-amount

This PC > System (C:) > extension_home > 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

- 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?**
- 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

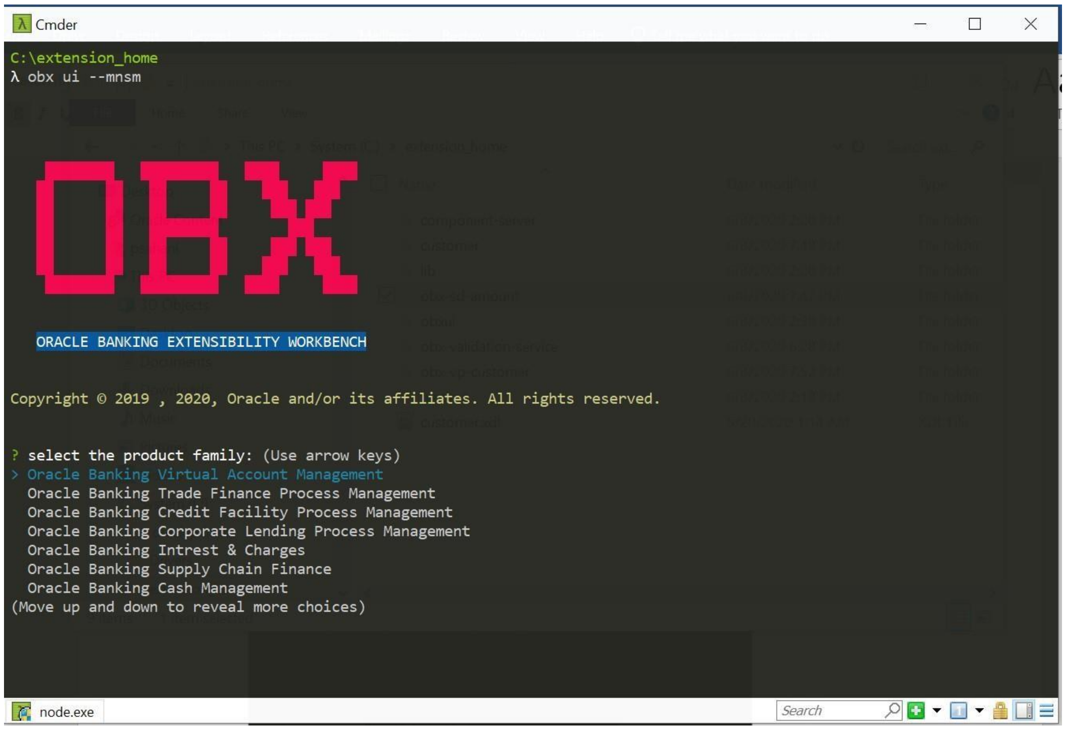


Customer Id	First Name	Last Name	Dob	Address1	Address2	City	State	Country	Pin Code
No data to display.									

Maintenance Detail and Summary

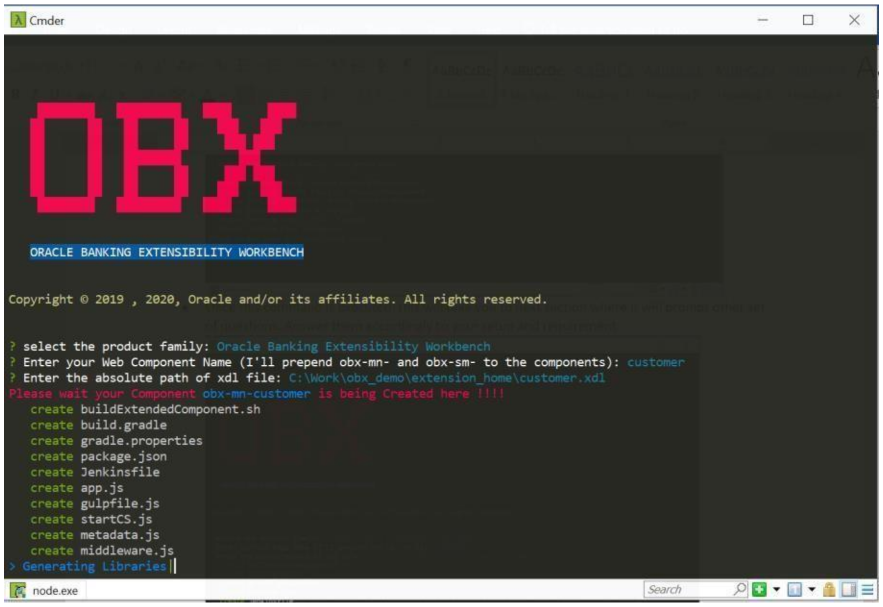
This section describes the process of creating the Maintenance Detail and Summary component using OBX. Here we have to remember that we will be generating two web components one will be detail component and another one for summary component. Following are the steps needed to be followed:

- Navigate to **extension_home** folder from cmder
- Use the command **obx ui –mnsm**



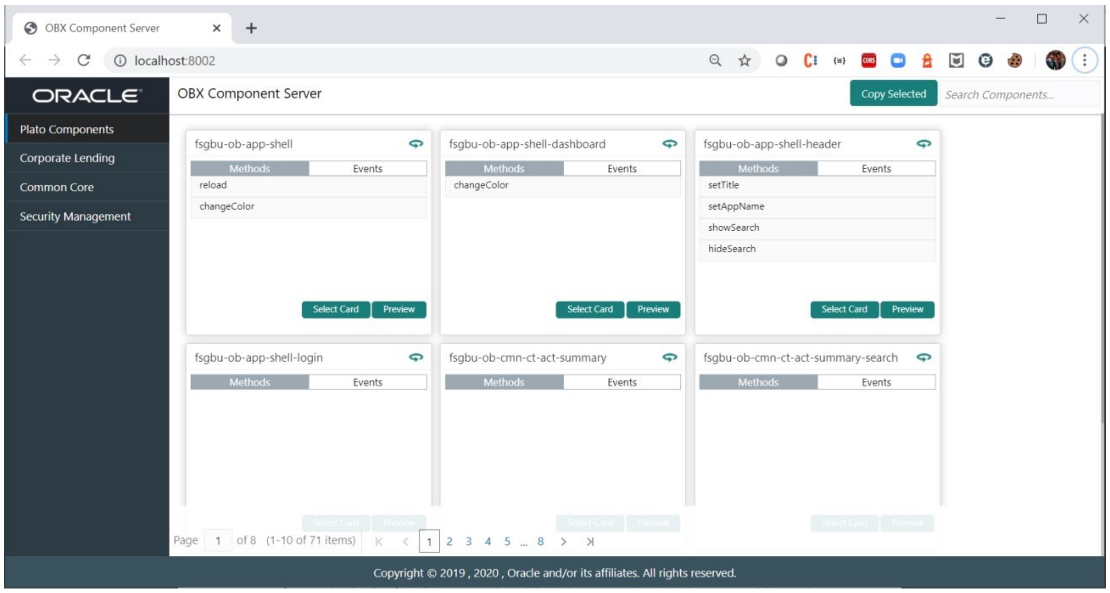
```
C:\extension_home
λ obx ui --mnsm

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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)
```

- 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

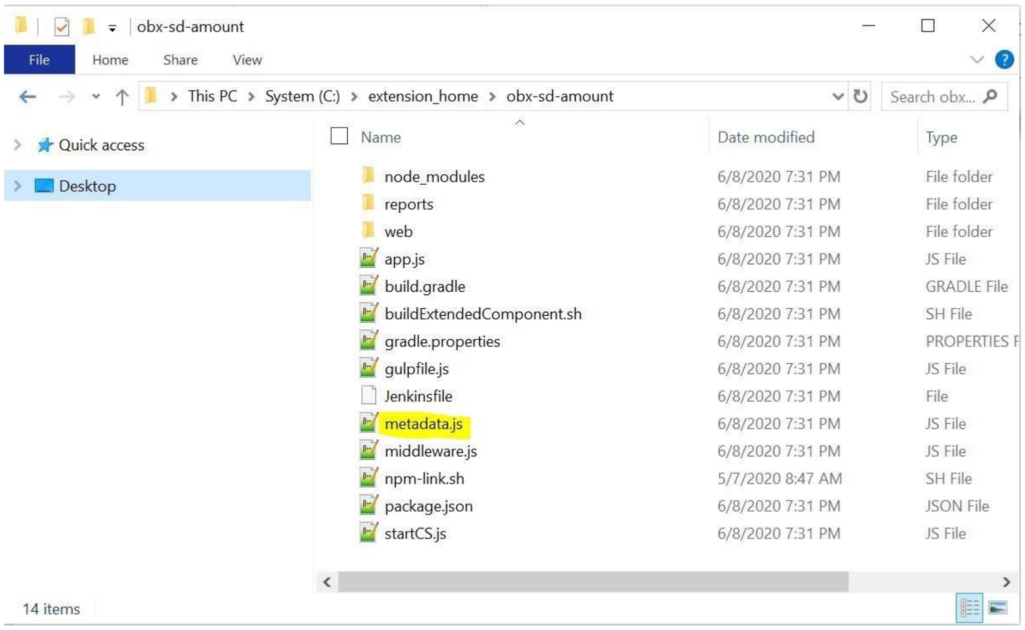


```
ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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): customer
? Enter the absolute path of xdl file: C:\Work\obx_demo\extension_home\customer.xdl
Please wait your Component obx-mn-customer is being Created here !!!!
   create buildExtendedComponent.sh
   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
```

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



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

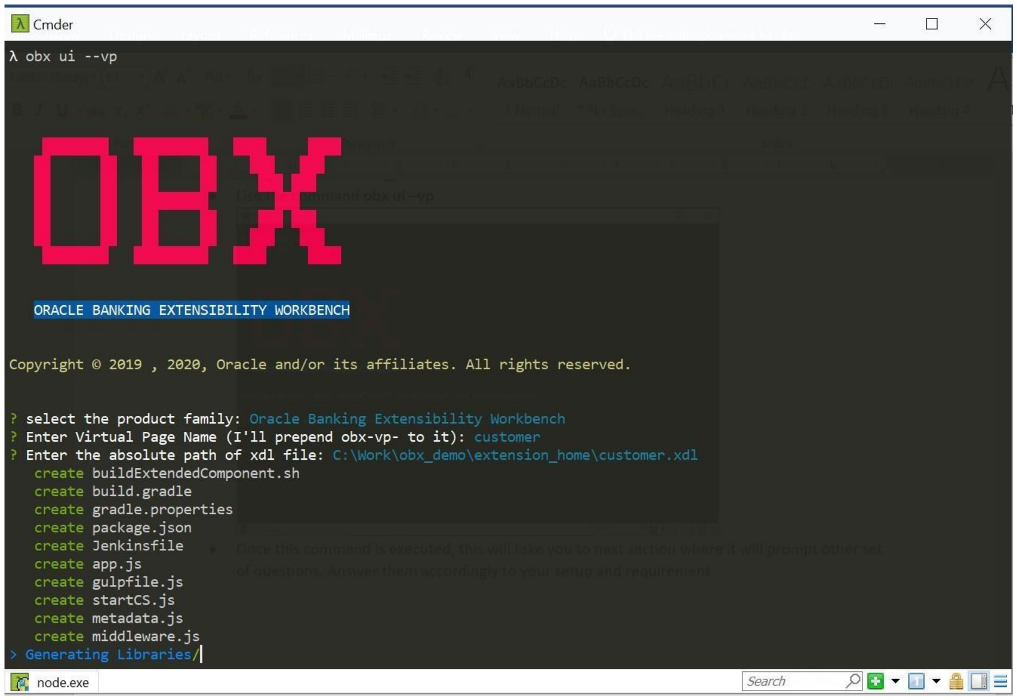


Name	Date modified	Type
node_modules	6/8/2020 7:31 PM	File folder
reports	6/8/2020 7:31 PM	File folder
web	6/8/2020 7:31 PM	File folder
app.js	6/8/2020 7:31 PM	JS File
build.gradle	6/8/2020 7:31 PM	GRADLE File
buildExtendedComponent.sh	6/8/2020 7:31 PM	SH File
gradle.properties	6/8/2020 7:31 PM	PROPERTIES 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

- 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?**
- 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.
- For this case we will be not able to see the component running locally as we have to 2 components generated.
- To start the component, one needs to go inside the component are run it manually

Data Segment
This section describes the process of creating the virtual page component using OBX. Following are the steps needed to be followed:

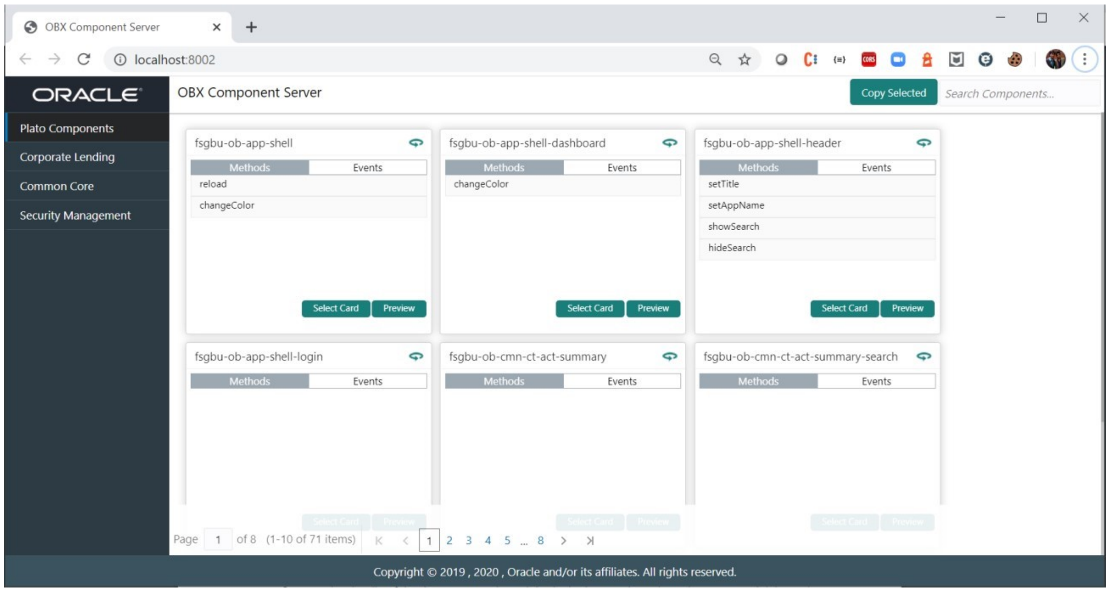
- Navigate to **extension_home** folder from cmder
- Use the command **obx ui –ds**
- Once this command is executed, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement



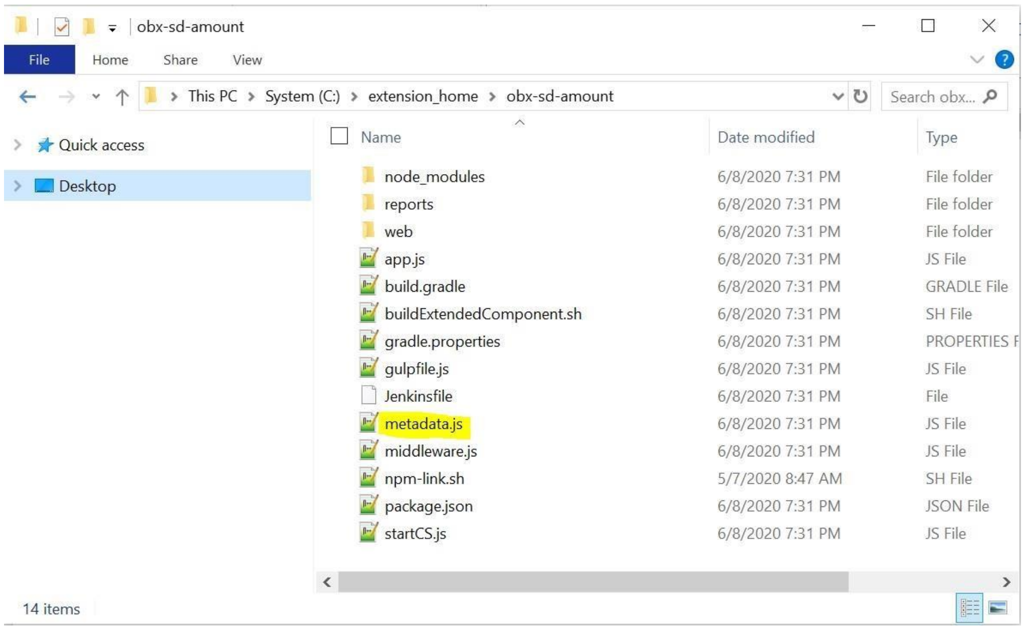
```
λ obx ui --vp

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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): customer
? Enter the absolute path of xdl file: C:\Work\obx_demo\extension_home\customer.xdl
   create buildExtendedComponent.sh
   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/
```

- 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.
- At this point of time go to browser and navigate to http://localhost:8002. You will be able to component server home page like

•



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



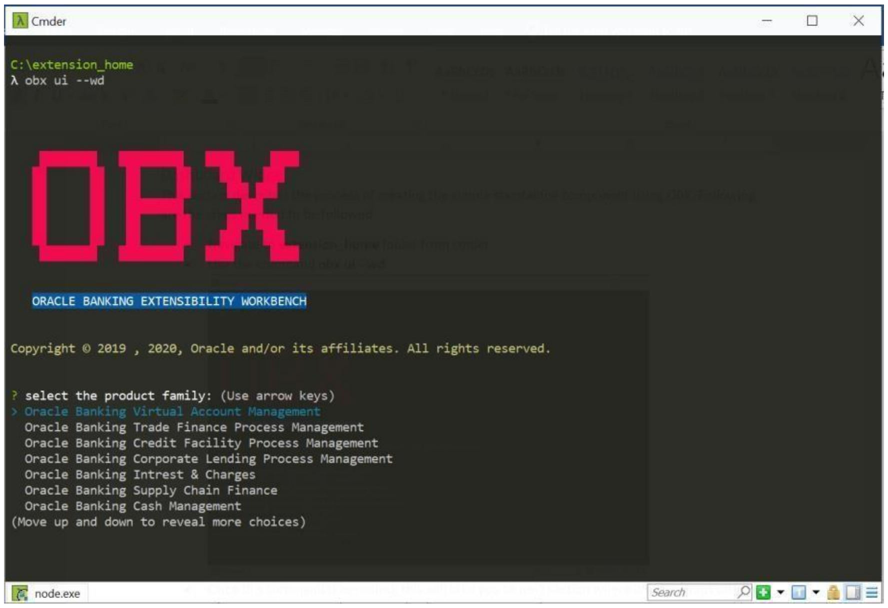
Name	Date modified	Type
node_modules	6/8/2020 7:31 PM	File folder
reports	6/8/2020 7:31 PM	File folder
web	6/8/2020 7:31 PM	File folder
app.js	6/8/2020 7:31 PM	JS File
build.gradle	6/8/2020 7:31 PM	GRADLE File
buildExtendedComponent.sh	6/8/2020 7:31 PM	SH File
gradle.properties	6/8/2020 7:31 PM	PROPERTIES 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

- 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?**
- 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

Dashboard Widget

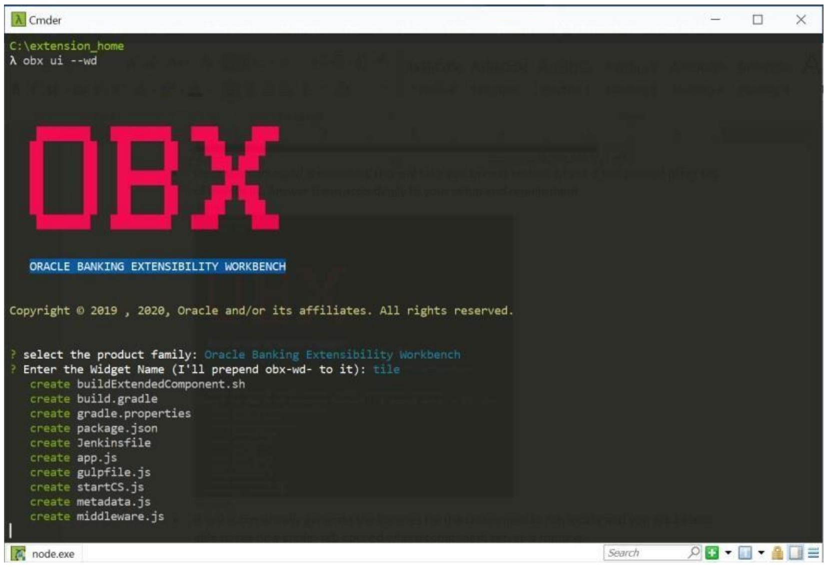
This section describes the process of creating the simple standalone component using OBX. Following are the steps needed to be followed:

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



```
C:\extension_home
λ obx ui --wd

OBX

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, 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)
```

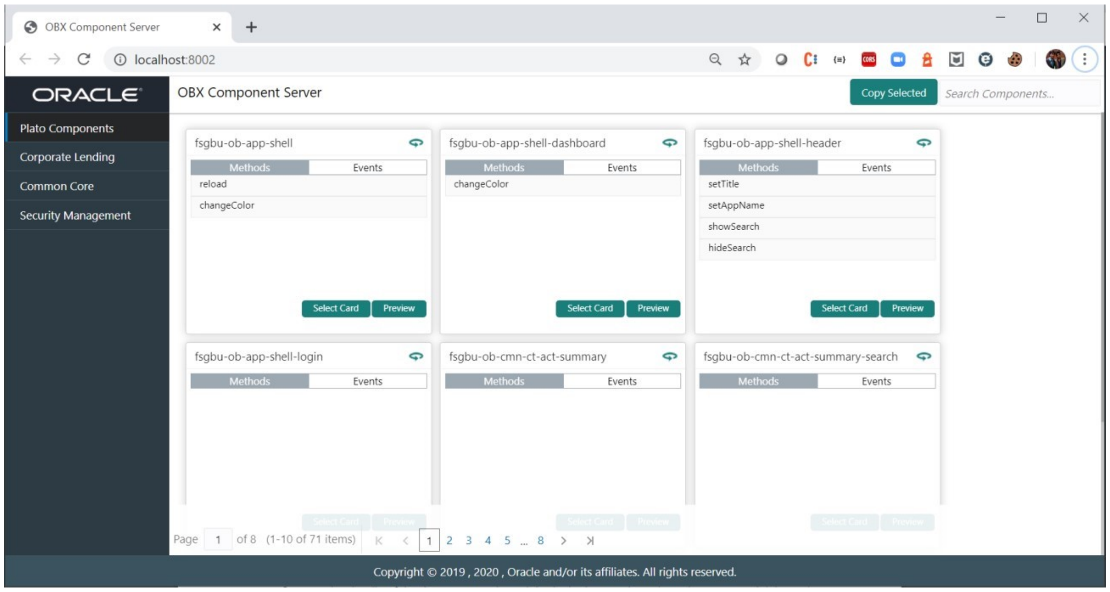
- Once this command is executed, this will take you to next section where it will prompt other set of questions. Answer them accordingly to your setup and requirement



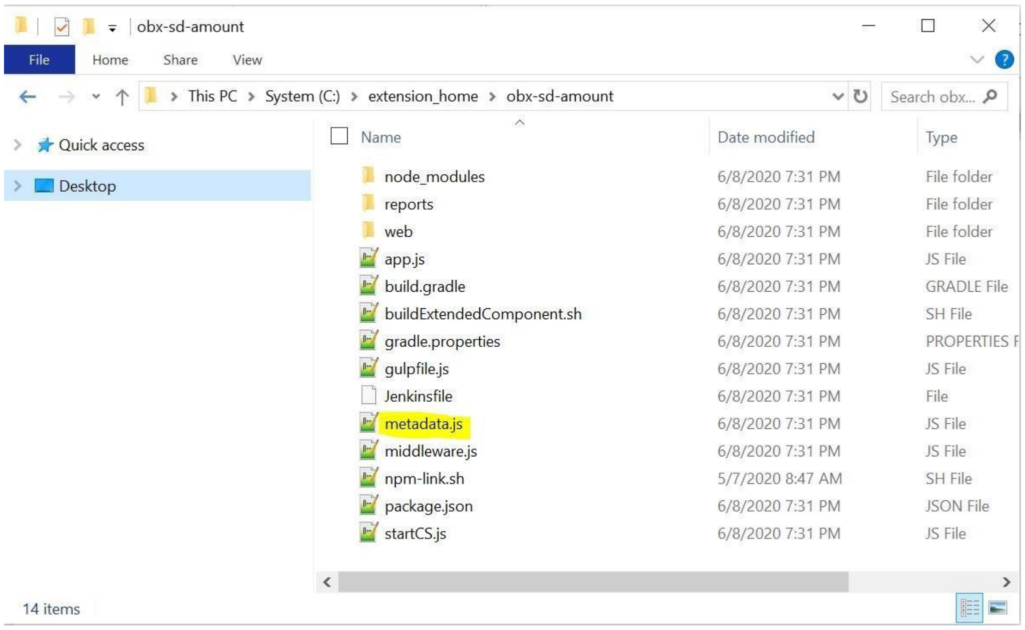
```
C:\extension_home
λ obx ui --wd

OBX

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.

? select the product family: Oracle Banking Extensibility Workbench
? Enter the Widget Name (I'll prepend obx-wd- to it): tile
   create buildExtendedComponent.sh
   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
```

- 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.
  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

•

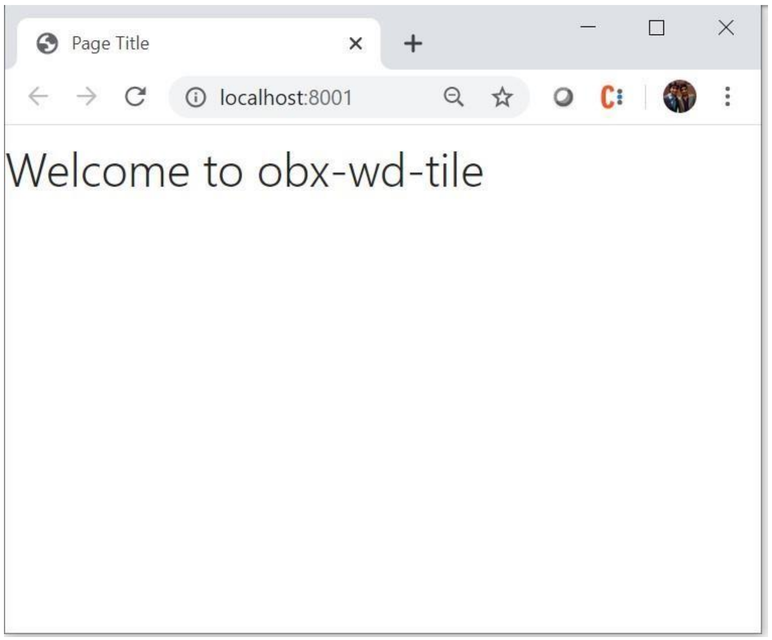


- Select the component which you want to reuse in your extension and paste it in **module.exports = [];** inside the **metadata.js** file



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

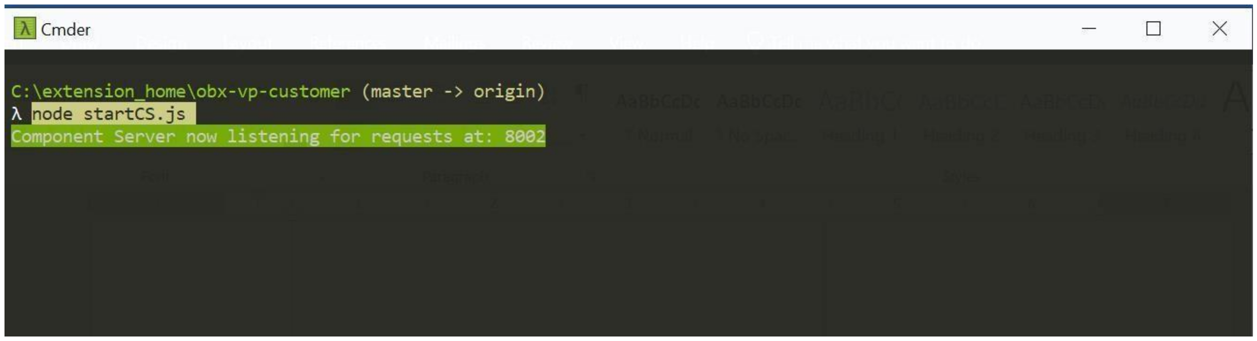
- Once done come back to main tab in cmder where is waiting with question **Please modify the Metadata.js file before proceeding. Once done press 'y' to proceed?**
- On completing the above process, it will automatically generate the source folder now and open a new tab on cmder where component will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



Running Component after Generation

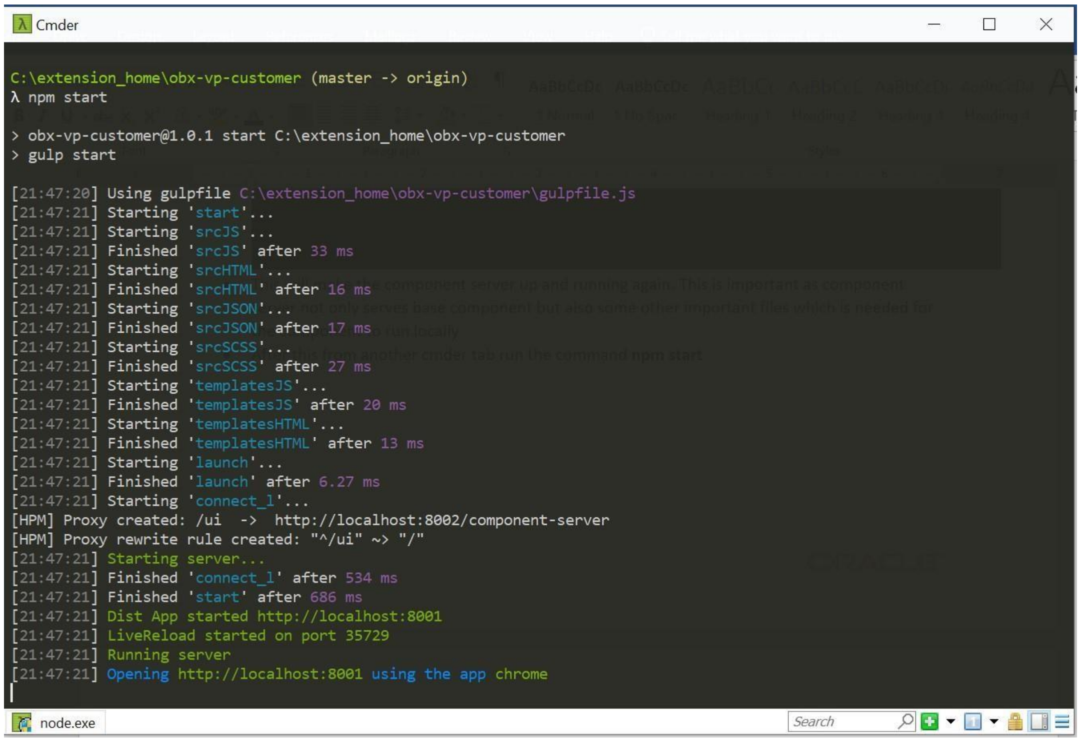
This section describes the steps you need to follow to re-run the component created or generated earlier. Please follow the below steps to do the same:

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



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

- 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
- After this from another cmder tab run the command **npm start**



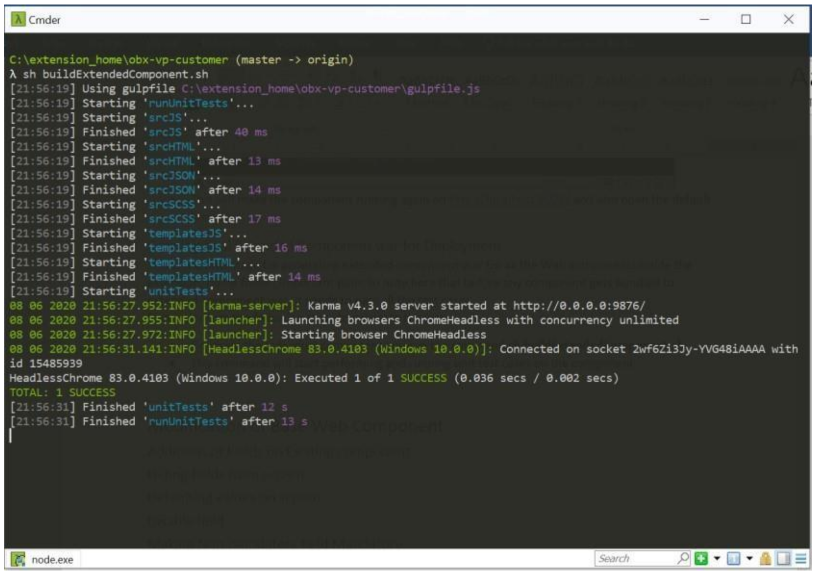
```
C:\extension_home\obx-vp-customer (master -> origin)
λ npm start

> 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_l'...
[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_l' 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
```

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

Creating final Extended Component war for Deployment

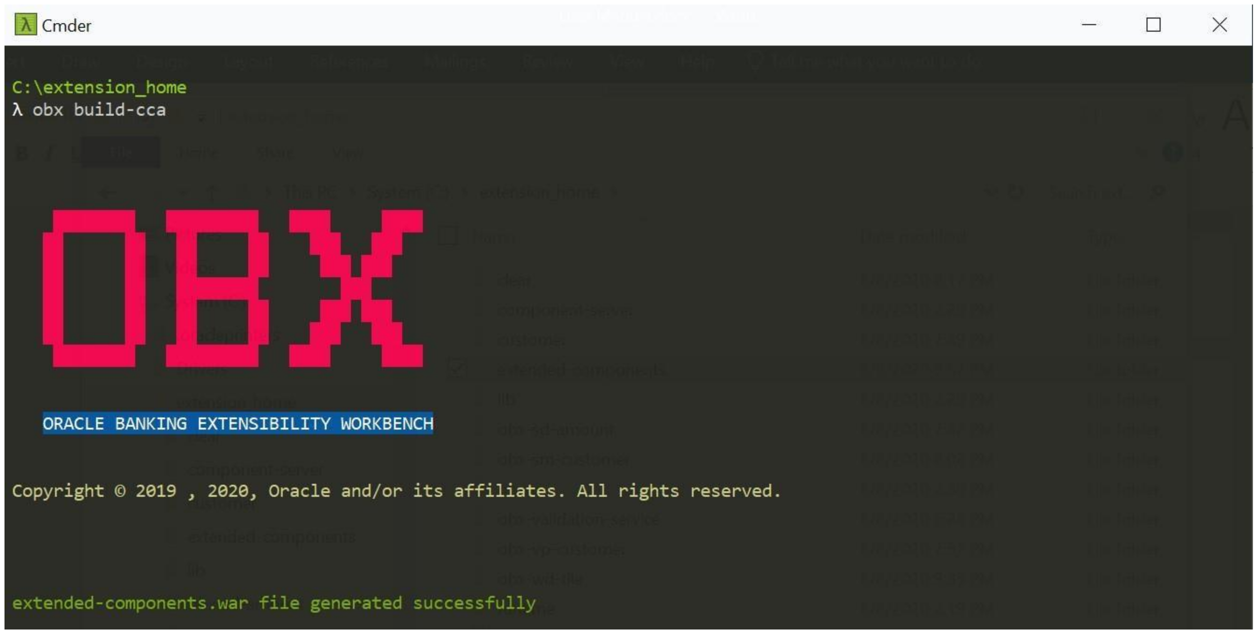
This is the final stage for generating extended-component war for all the Web components inside the extension_home folder. Important point to note here that before any component gets bundled to extended-component.war, it needs to pass all the test cases. Please perform the following steps to generate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

- 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



```
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
```

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

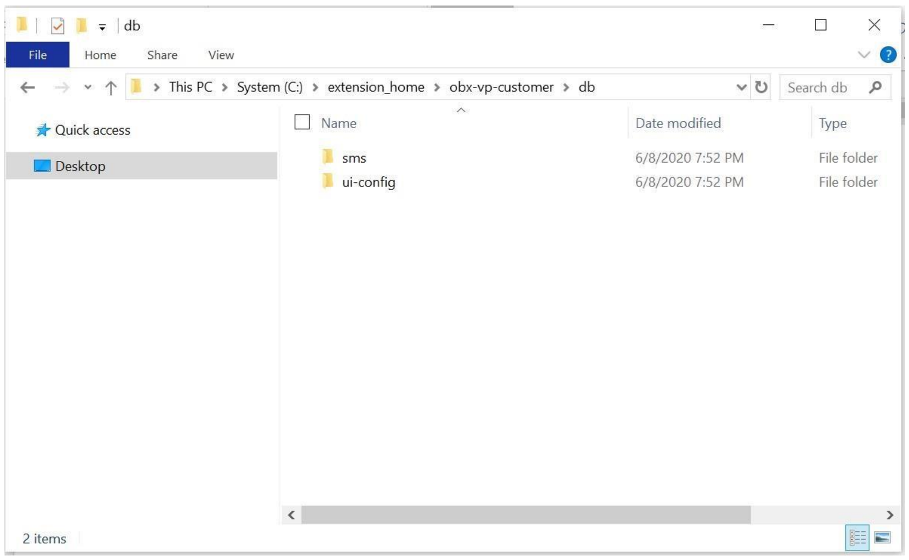


```
C:\extension_home
λ obx build-cca

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.

extended-components.war file generated successfully
```

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

Understanding DB Scripts for Web Components

This section 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.



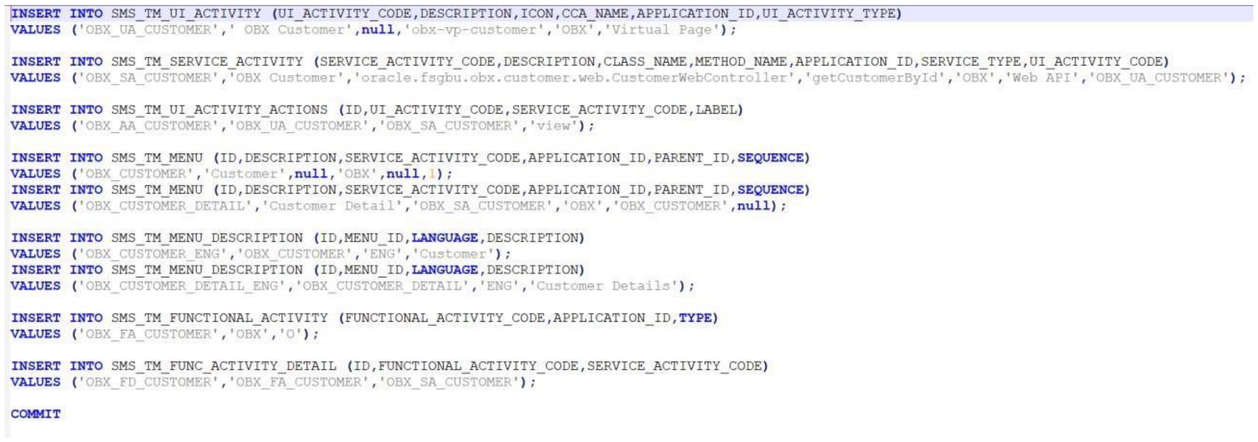
This PC > System (C:) > extension_home > obx-vp-customer > db

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

2 items

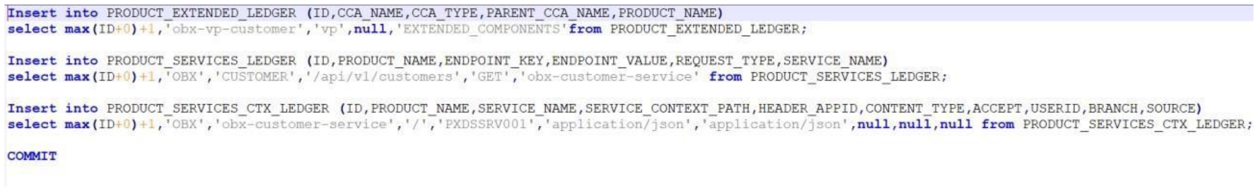
DB folder inside the web component consists of two folders sms and ui-config:

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



```sql
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



```sql
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
```

4. Modification of Base Web Component

This feature of OBX enables users to create extensions which helps to modify the behavior of existing component. It 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*

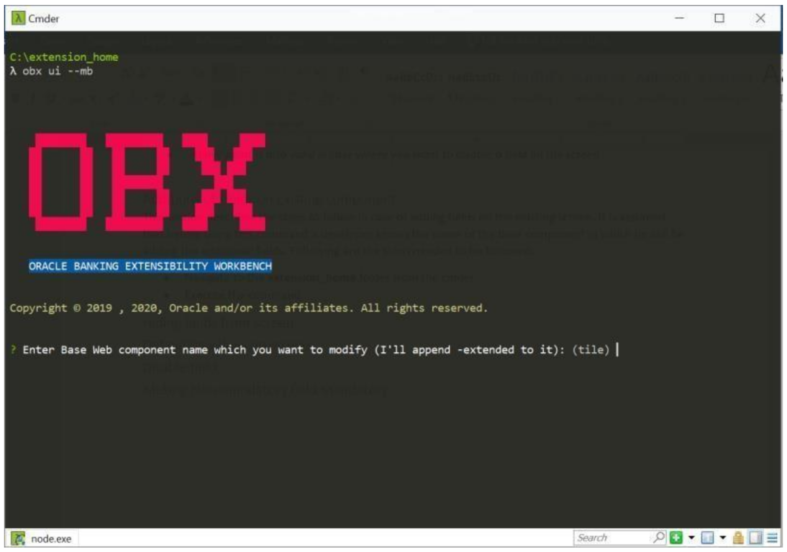
Following are the uses cases which can be achieved using modification of existing component

- 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 section describes 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:

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



```
C:\extension_home
λ obx ui --mb

ORACLE BANKING EXTENSIBILITY WORKBENCH

Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.

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

- 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
- Here also like above all the libraries are generated at runtime
- 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

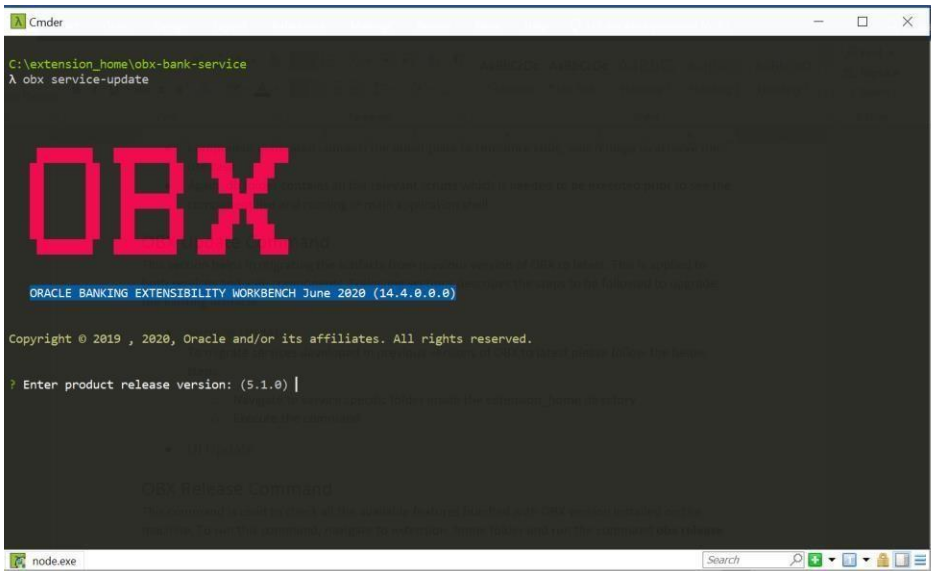
OBX Update Command

This section helps in migrating the artifacts from previous version of OBX to latest. This is applied to both services and web components. Following sections describes the steps to be followed to upgrade the existing artifacts:

Service Update

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

- Navigate to service specific folder inside the extension_home directory ○ Execute the command **obx service-update**



```
C:\extension_home\obx-bank-service
λ obx service-update

ORACLE BANKING EXTENSIBILITY WORKBENCH June 2020 (14.4.0.0.0)

Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.

? Enter product release version: (5.1.0) |
```

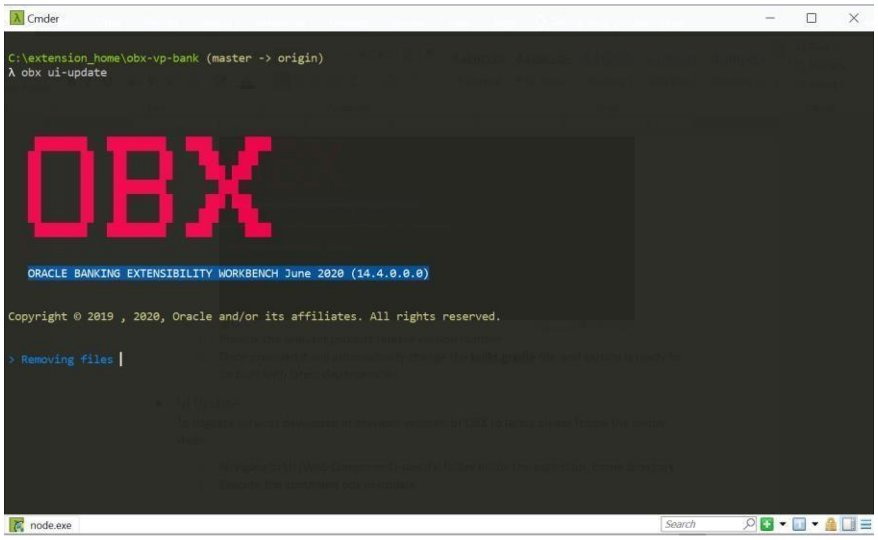
○ Provide the relevant product release version number

- ○ Once provided it will automatically change the **build.gradle** file and service is ready to be built with latest dependencies

UI Update

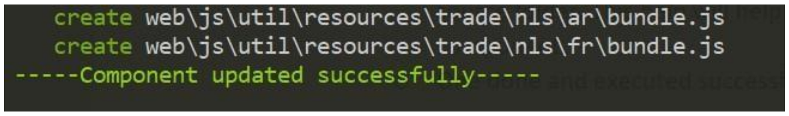
To migrate services developed in previous versions of OBX to latest please follow the below steps:

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



```
C:\extension_home\obx-vp-bank (master -> origin)
λ obx ui-update

ORACLE BANKING EXTENSIBILITY WORKBENCH June 2020 (14.4.0.0.0)

Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.

> Removing files |
```

○ 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 ○ One done and executed successfully you will the below message

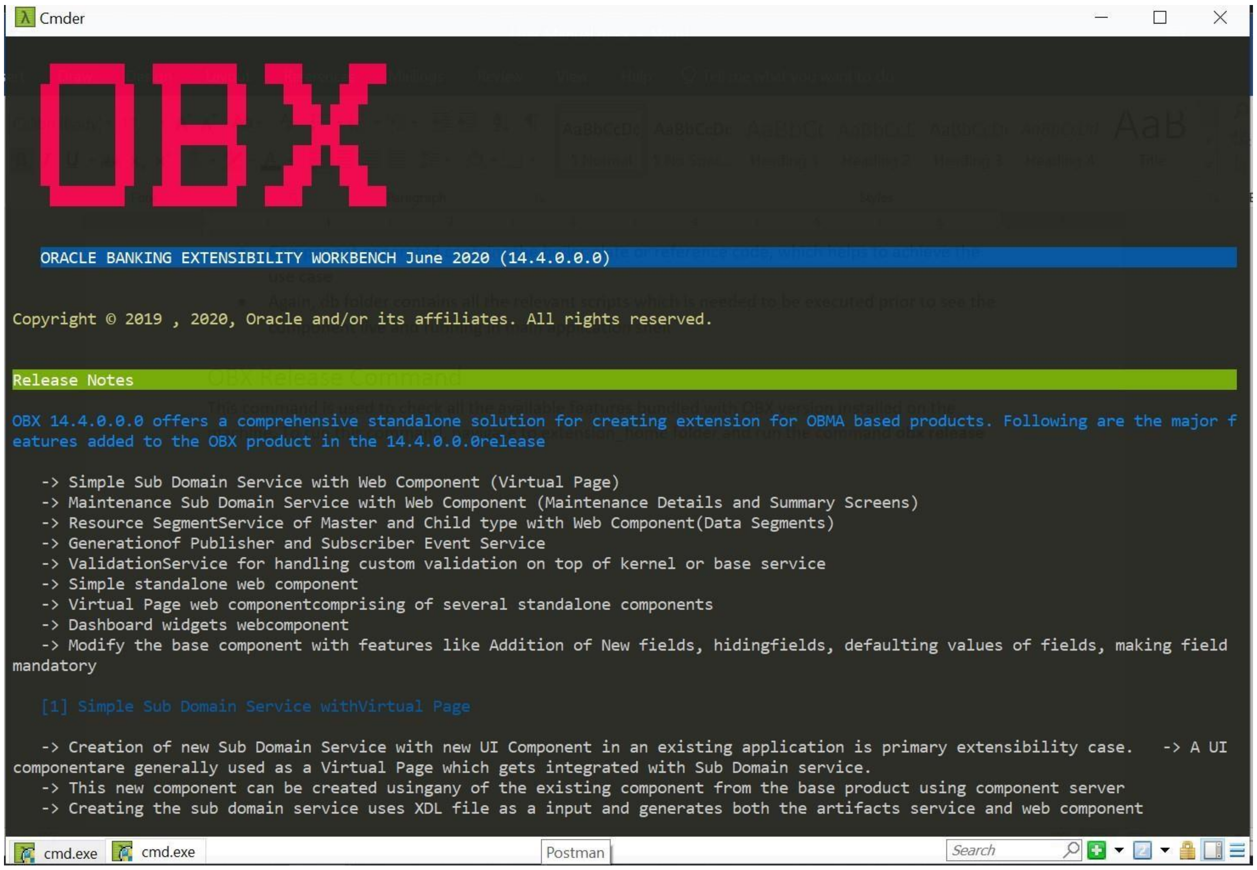


```
    create web\js\util\resources\trade\nls\ar\bundle.js
    create web\js\util\resources\trade\nls\fr\bundle.js
-----Component updated successfully-----
```

- 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

OBX Release Command

This command is used to check all the available features bundled with OBX version installed on the machine. To run this command, navigate to extension_home folder and run the command **obx release**



```
ORACLE BANKING EXTENSIBILITY WORKBENCH June 2020 (14.4.0.0.0)

Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.

Release Notes

OBX 14.4.0.0.0 offers a comprehensive standalone solution for creating extension for OBMA based products. Following are the major f
eatures added to the OBX product in the 14.4.0.0.0release

   -> Simple Sub Domain Service with Web Component (Virtual Page)
   -> Maintenance Sub Domain Service with Web Component (Maintenance Details and Summary Screens)
   -> Resource SegmentService of Master and Child type with Web Component(Data Segments)
   -> Generationof Publisher and Subscriber Event Service
   -> ValidationService for handling custom validation on top of kernel or base service
   -> Simple standalone web component
   -> Virtual Page web componentcomprising of several standalone components
   -> Dashboard widgets webcomponent
   -> Modify the base component with features like Addition of New fields, hidingfields, defaulting values of fields, making field
mandatory

   [1] Simple Sub Domain Service withVirtual Page

   -> Creation of new Sub Domain Service with new UI Component in an existing application is primary extensibility case.   -> A UI
componentare generally used as a Virtual Page which gets integrated with Sub Domain service.
   -> This new component can be created usingany of the existing component from the base product using component server
   -> Creating the sub domain service uses XDL file as a input and generates both the artifacts service and web component
```

5. Extending Product Data Segments with Additional Fields

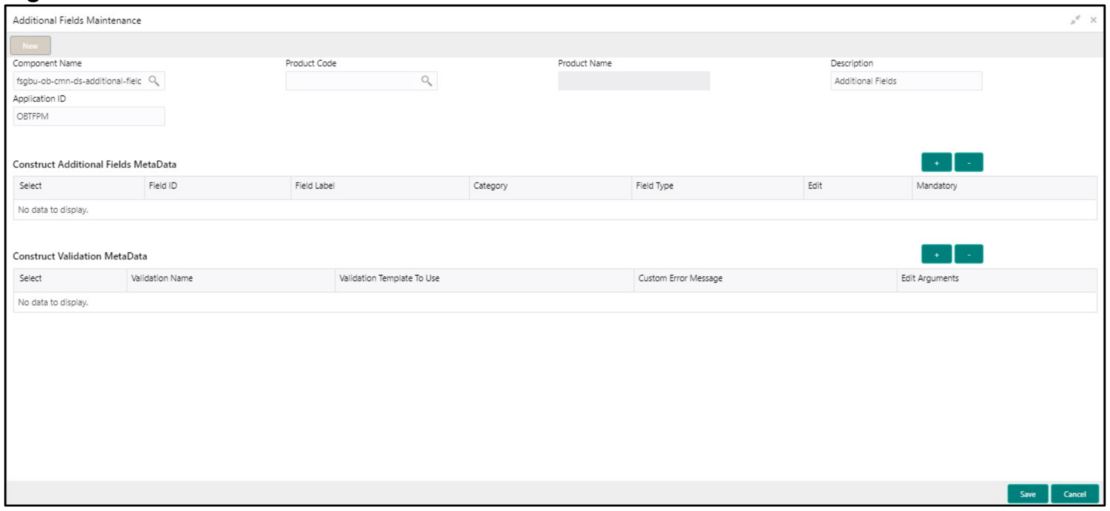
Additional Fields Maintenance

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 (or) do the following steps:

1. From **Home screen**, click **Core Maintenance**. Under **Core Maintenance**, click **Additional Fields Maintenance**.

    The **Additional Fields Maintenance** screen is displayed.

Figure 1: Additional Fields Maintenance



Additional Fields Maintenance

Component Name: fsgbu-ob-cmn-ds-additional-fielc

Product Code:

Product Name:

Description: Additional Fields

Application ID: OBTFPM

Construct Additional Fields MetaData

Select	Field ID	Field Label	Category	Field Type	Edit	Mandatory
No data to display.						

Construct Validation MetaData

Select	Validation Name	Validation Template To Use	Custom Error Message	Edit Arguments
No data to display.				

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

Field Description – Additional Field Maintenance

Field	Description
Component Name	Specify the data segment name as component name. **NOTE:** By default, the value **fsgbu-ob-cmn-dsadditional-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.

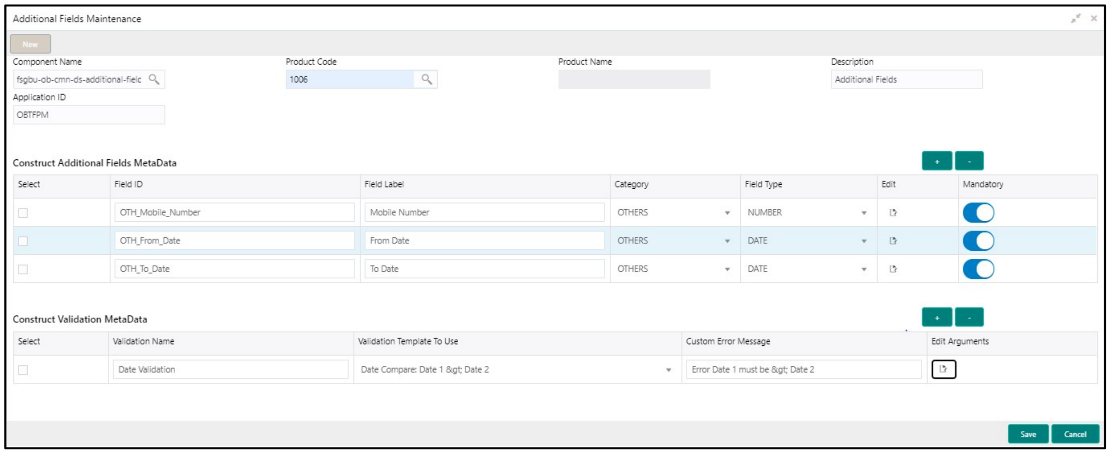
Field	Description

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.
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.

3. 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**.

   Figure 2: MetaData Examples


Additional Fields Maintenance

Component Name	Product Code	Product Name	Description
fsgbu-ob-cmn-ds-additional-fielc	1006		Additional Fields
Application ID			
OBTFPM			

Construct Additional Fields MetaData

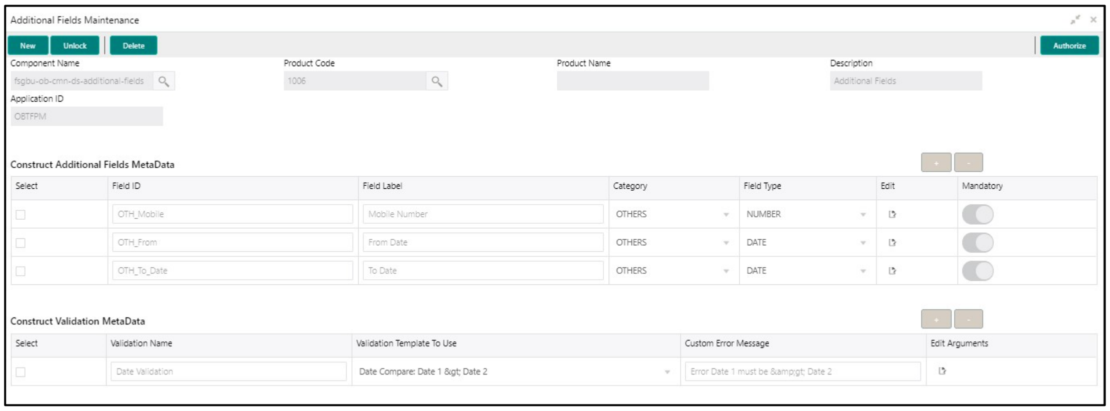
Select	Field ID	Field Label	Category	Field Type	Edit	Mandatory
	OTH_Mobile_Number	Mobile Number	OTHERS	NUMBER		
	OTH_From_Date	From Date	OTHERS	DATE		
	OTH_To_Date	To Date	OTHERS	DATE		

Construct Validation MetaData

Select	Validation Name	Validation Template To Use	Custom Error Message	Edit Arguments
	Date Validation	Date Compare: Date 1 &gt; Date 2	Error Date 1 must be &gt; Date 2	

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

Figure 3: Additional Field Maintenance Record


Additional Fields Maintenance

Component Name	Product Code	Product Name	Description
fsgbu-ob-cmn-ds-additional-fields	1006		Additional Fields
Application ID			
OBTFPM			

Construct Additional Fields MetaData

Select	Field ID	Field Label	Category	Field Type	Edit	Mandatory
	OTH_Mobile	Mobile Number	OTHERS	NUMBER		
	OTH_From	From Date	OTHERS	DATE		
	OTH_To_Date	To Date	OTHERS	DATE		

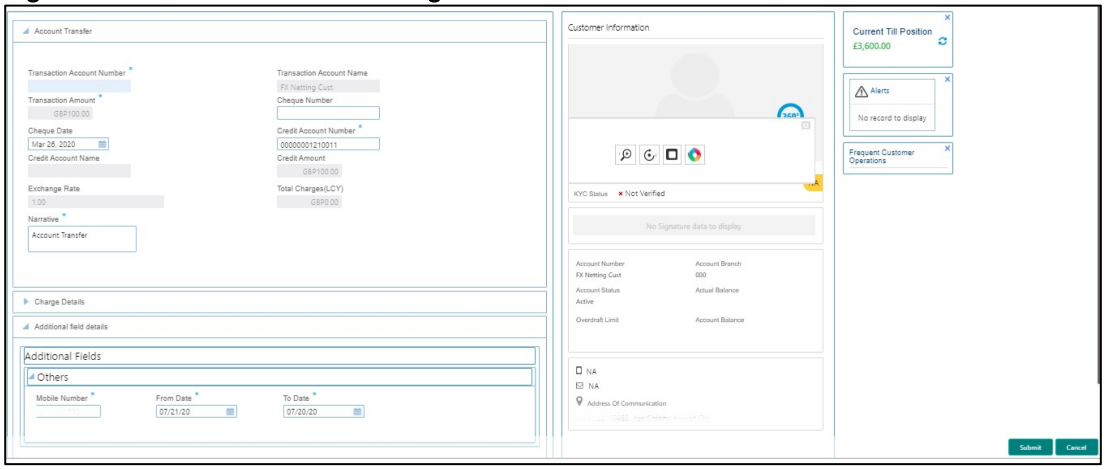
Construct Validation MetaData

Select	Validation Name	Validation Template To Use	Custom Error Message	Edit Arguments
	Date Validation	Date Compare: Date 1 &gt; Date 2	Error Date 1 must be &gt; Date 2	

5. 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**.

Figure 4: Additional Field Data Segment


Account Transfer

Transaction Account Number *	Transaction Account Name
	FX Netting Cust
Transaction Amount *	Cheque Number
GBP100.00	
Cheque Date	Credit Account Number *
Mar 26, 2020	00000001210011
Credit Account Name	Credit Amount
	GBP100.00
Exchange Rate	Total Charges(LCY)
1.00	GBP0.00
Narrative *	
Account Transfer	

Charge Details

Additional field details

Additional Fields

Others

Mobile Number *	From Date *	To Date *
	07/21/20	07/20/20

Customer Information

KYC Status × Not Verified

No Signature data to display

Account Number	Account Branch
FX Netting Cust	000
Account Status	Actual Balance
Active	
Overdraft Limit	Account Balance

NA
NA
Address Of Communication

Current Till Position
£3,600.00

Alerts
No record to display

Frequent Customer Operations

6. 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,
            "recievedAccountCcy": null,
            "recievedAccountAmt": 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": ""
    }
```

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-attributes-services/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-attributes-
                          services/cmcadditional-attributes-services/33347926-842b-4232-af31-
                          8c1b59612244"
                          }
                          ],
                "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_Date\",\"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
                }
        }
}
```

Fetching the Saved Values

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-additional-attributesservices/additionalattributes/?uiKey=fsgbu-ob-cmn-ds-additional-fields@1006&dataReferenceKey=00a01dfd-0d6f-4400-a9c5-0f56551165e4 **Sample 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"
                    }
    ]
}
```

Action URL and Static Tag Maintenance

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.

Static Tag Maintenance

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.

Extensibility Use Cases for OBBRN Servicing

New Transaction Screen – 1499 (Exact Clone of 1401)

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

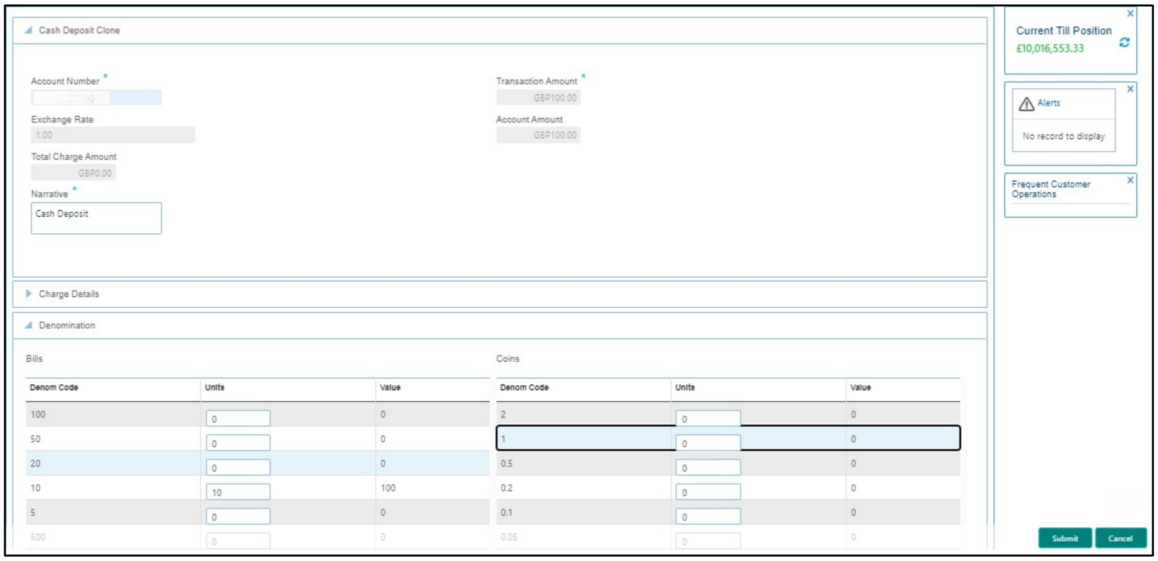
The below mentioned tables need to be checked in branch common schema:

- 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 5: Cash Deposit Clone



Account Number * [illegible]
Exchange Rate 1.00
Total Charge Amount GBP0.00
Narrative * Cash Deposit
Transaction Amount * GBP100.00
Account Amount GBP100.00

Current Till Position £10,016,553.33

No record to display

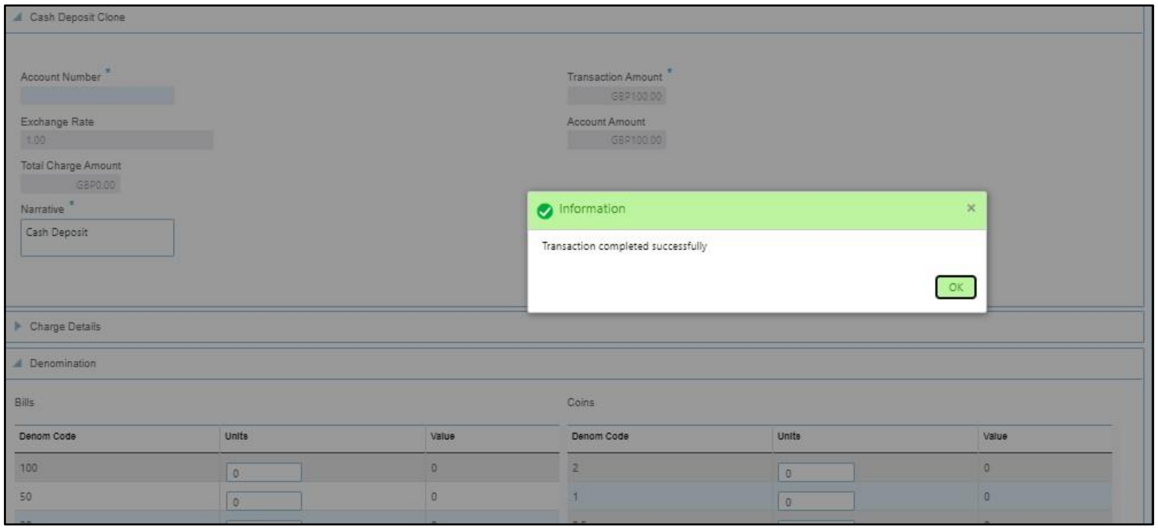
Bills

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

Figure 6: Information Message



Account Number *

Transaction Amount * GBP100.00

Exchange Rate 1.00

Account Amount GBP100.00

Total Charge Amount GBP0.00

Narrative * Cash Deposit

Information

Transaction completed successfully

Bills

Denom Code	Units	Value
100	0	0
50	0	0

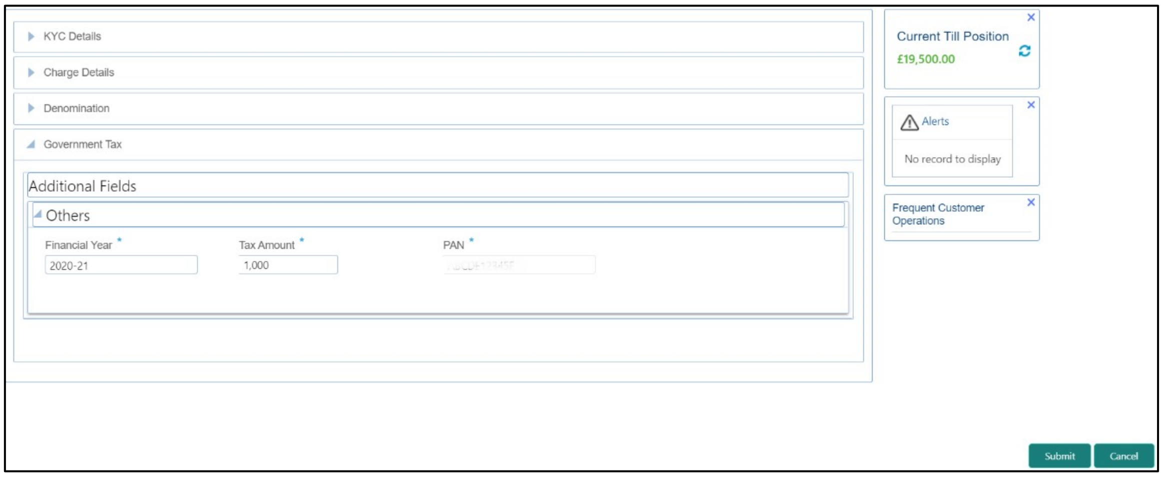
Coins

Denom Code	Units	Value
2	0	0
1	0	0

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 7: Additional Fields Segment



Financial Year * 2020-21

Tax Amount * 1,000

PAN *

Current Till Position £19,500.00

No record to display

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.

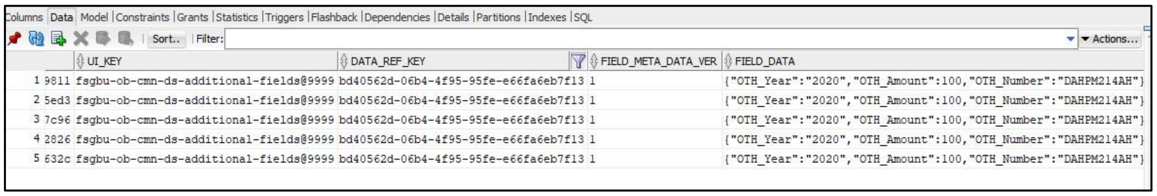
```
<wls:library-ref>
        <wls:library-name>obremo-srv-cmn-common-txn</wls:library-name>
</wls:library-ref>
```
Response:

```
{
        "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,
"recievedAccountCcy": null,
"recievedAccountAmt": 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: Common Core Additional Attributes

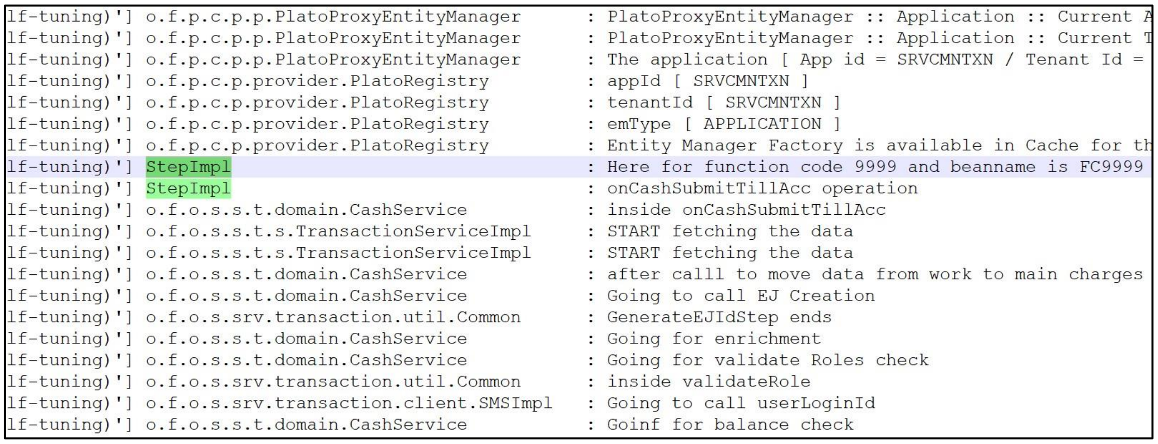


		UI_KEY	DATA_REF_KEY	FIELD_META_DATA_VER	FIELD_DATA
1	9811	fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}
2	5ed3	fsgbu-ob-cmn-ds-additional-fields@9999	bd40562d-06b4-4f95-95fe-e66fa6eb7f13	1	{"OTH_Year":"2020","OTH_Amount":100,"OTH_Number":"DAHPM214AH"}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

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

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

Figure 9: Common Code

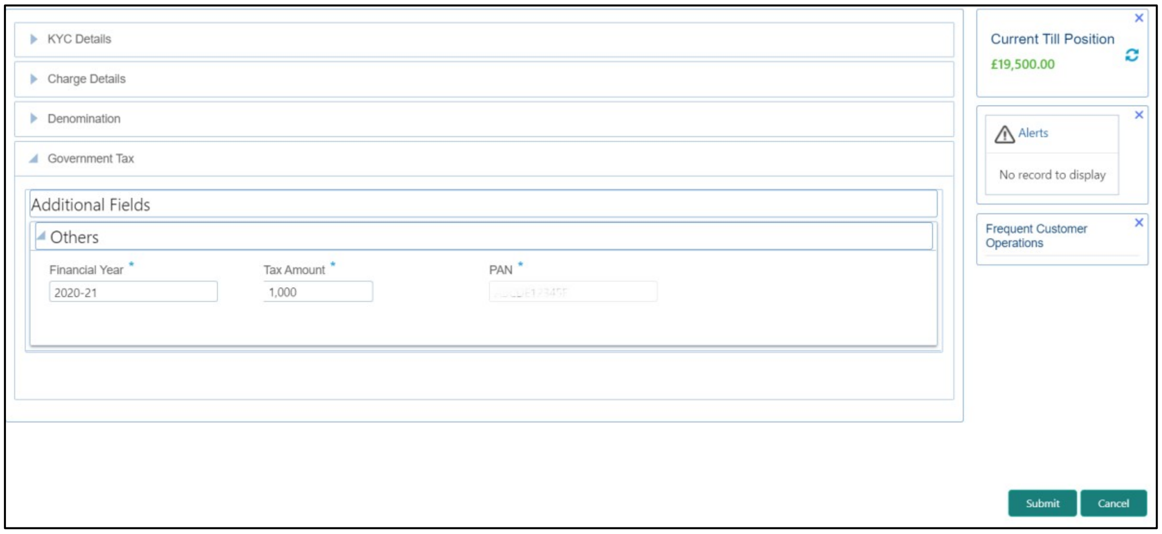


```
lf-tuning)'] o.f.p.c.p.p.PlatoProxyEntityManager      : PlatoProxyEntityManager :: Application :: Current A
lf-tuning)'] o.f.p.c.p.p.PlatoProxyEntityManager      : PlatoProxyEntityManager :: Application :: Current T
lf-tuning)'] o.f.p.c.p.p.PlatoProxyEntityManager      : The application [ App id = 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
```

Exact Clone with Additional Fields Using Extensible Code

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

Figure 10: Additional Fields Segment



Financial Year * 2020-21
Tax Amount * 1,000
PAN * [redacted]
Current Till Position £19,500.00

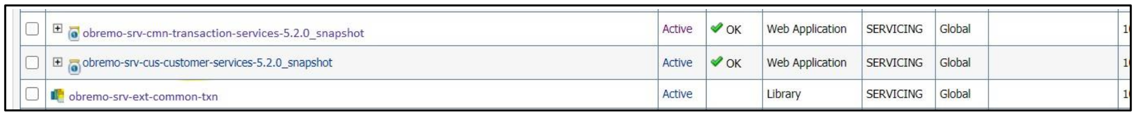
A library reference is added weblogic.xml (**obremo-srv-ext-common-txn**) for extensibility. A new jar **obremo-srvcmn-common-txn**, which holds the most of the code of transaction service and can be a dependency in the external jar.

<wls:library-ref>

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

Jar Deployment in Weblogic:

Figure 11: 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:

{

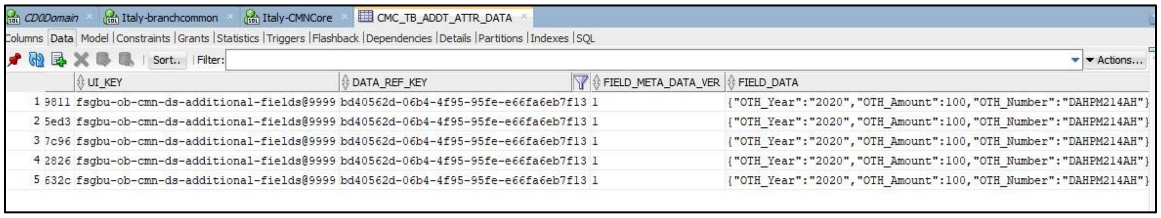
"data": {

```
 "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,
"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": ""
}
```

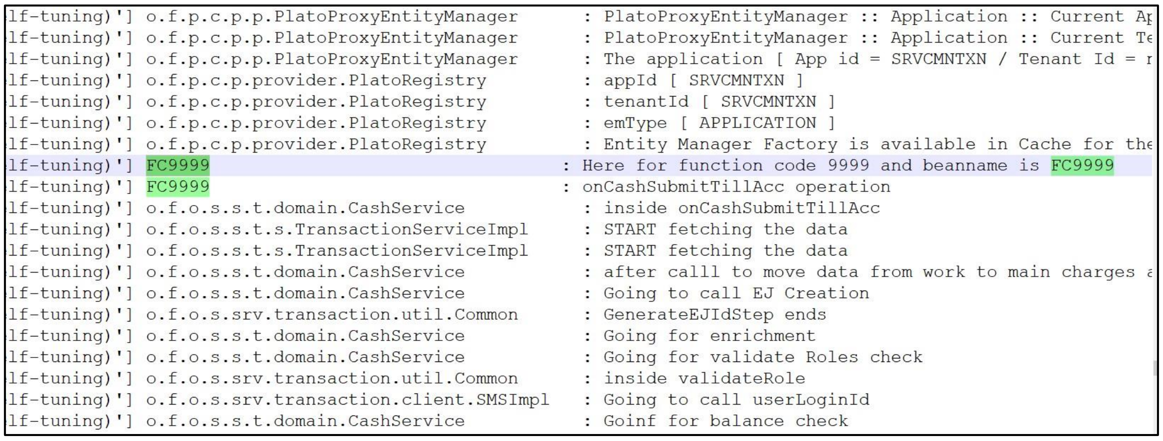
Figure 12: Common Core Additional Attributes



		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 13: 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
```

Extensibility Use Cases for OBX

New Transaction screen – 1499 (Clone of 1401)

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

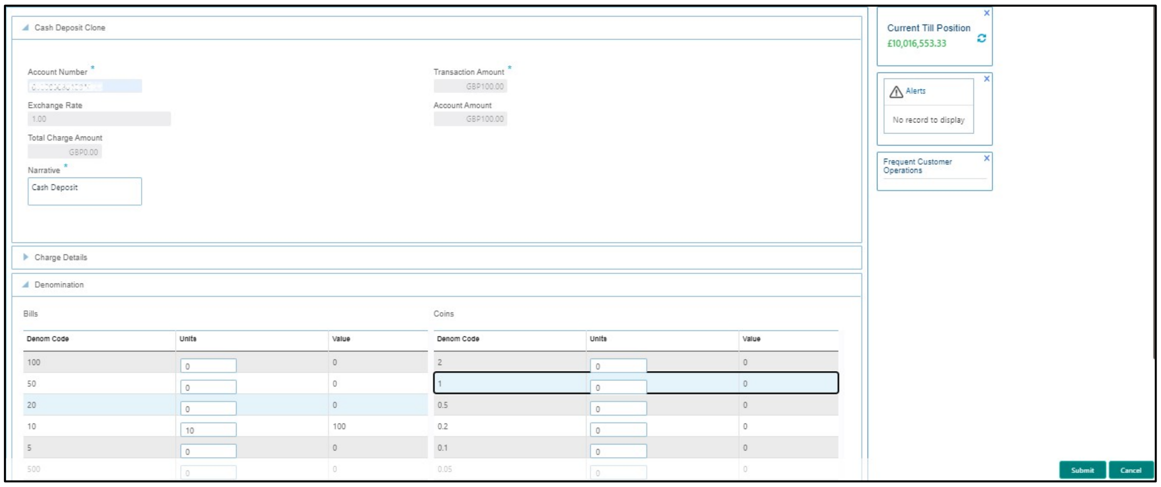
The below mentioned tables need to be checked in in Common Core schema:

- 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 14: Cash Deposit Clone



Cash Deposit Clone

Account Number * [redacted]

Transaction Amount * GBP100.00

Exchange Rate 1.00

Account Amount GBP100.00

Total Charge Amount GBP0.00

Narrative * Cash Deposit

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

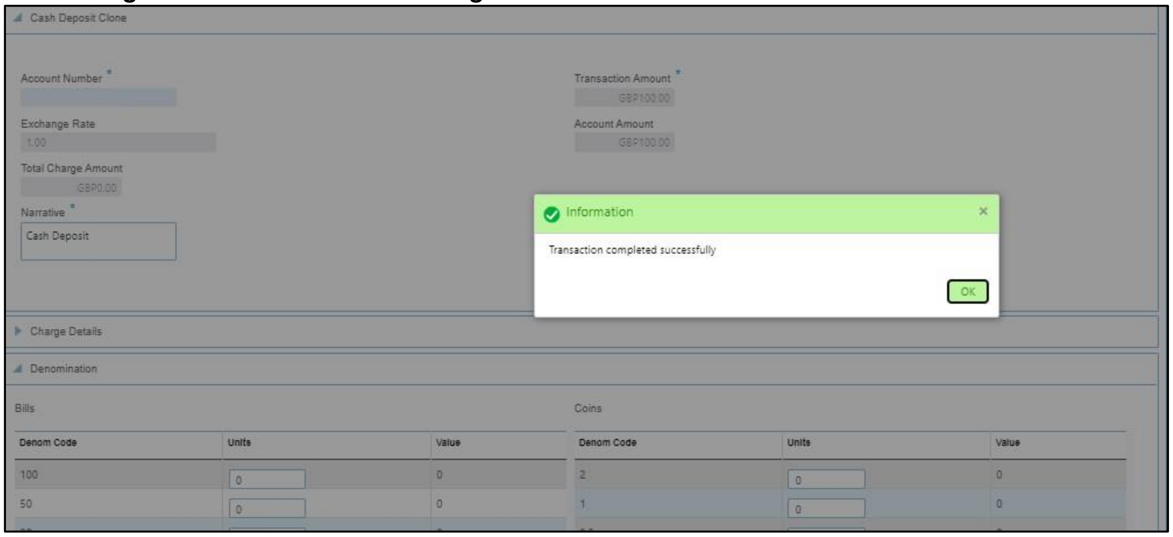
Current Till Position £10,016,553.33

Alerts

No record to display

Frequent Customer Operations

Figure 15: Information Message



Cash Deposit Clone

Account Number *

Transaction Amount * GBP100.00

Exchange Rate 1.00

Account Amount GBP100.00

Total Charge Amount GBP0.00

Narrative * Cash Deposit

Information

Transaction completed successfully

Charge Details

Denomination

Bills

Denom Code	Units	Value
100	0	0
50	0	0

Coins

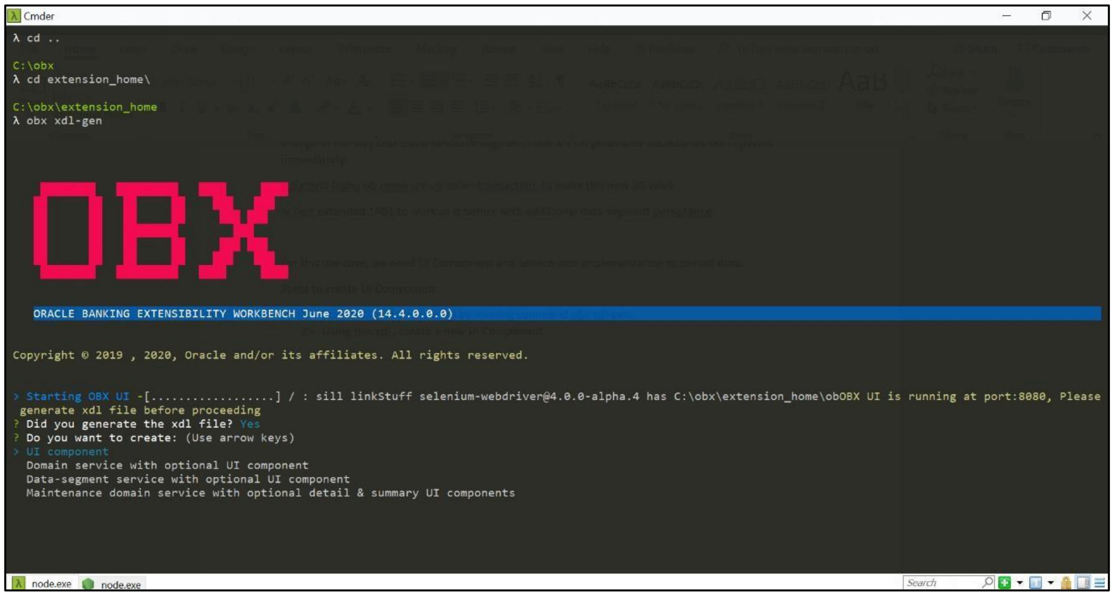
Denom Code	Units	Value
2	0	0
1	0	0

New Data Segment in Existing 1401 Screen

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 16: XDL Generation



```
λ cd ..

C:\obx
λ cd extension_home\

C:\obx\extension_home
λ obx xdl-gen

   ORACLE BANKING EXTENSIBILITY WORKBENCH June 2020 (14.4.0.0.0)

Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.

> Starting OBX UI -[..................] / : sill linkStuff selenium-webdriver@4.0.0-alpha.4 has C:\obx\extension_home\obOBX 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
```

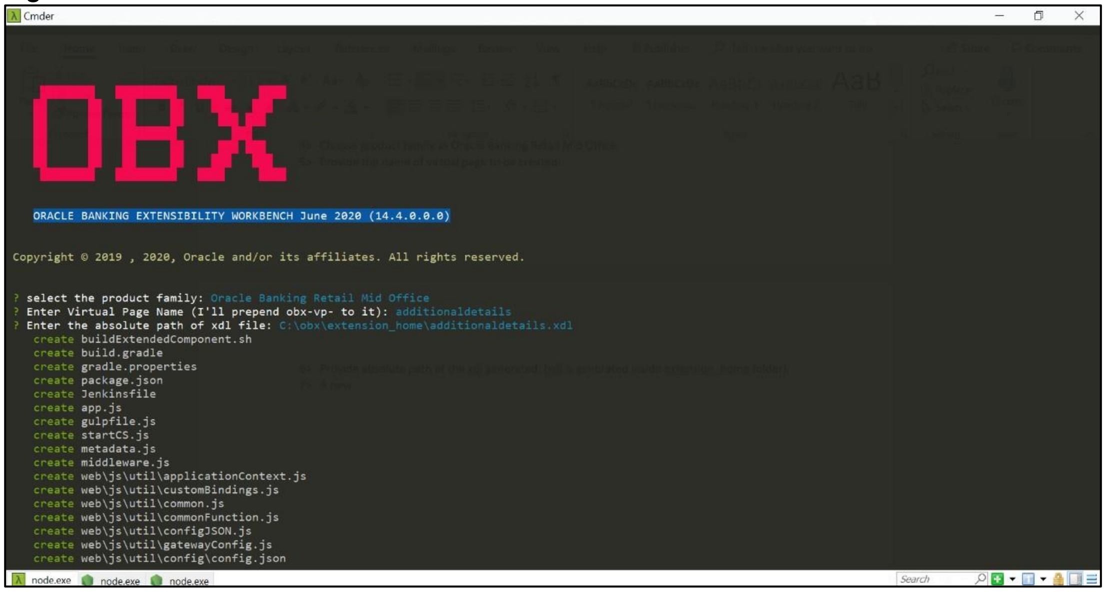
3. Select the option **UI Component**.

4. Choose 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).

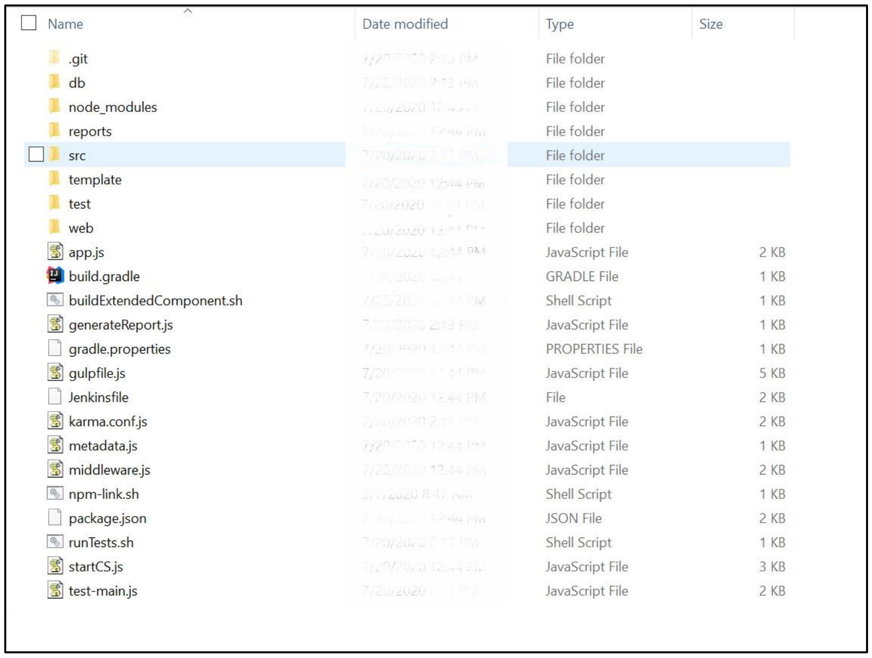
NOTE: A new UI Component will be created in extension_home folder with prefix obx-vp/obx-ds. In the **Cmder** tab, OBX will prompt to modify Metadata.js file of the newly created component. In addition, the component-server will start running at port 8002.

Figure 17: XDL Path



```
   ORACLE BANKING EXTENSIBILITY WORKBENCH June 2020 (14.4.0.0.0)

Copyright © 2019 , 2020, 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): additionaldetails
? Enter the absolute path of xdl file: C:\obx\extension_home\additionaldetails.xdl
   create buildExtendedComponent.sh
   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
   create web\js\util\applicationContext.js
   create web\js\util\customBindings.js
   create web\js\util\common.js
   create web\js\util\commonFunction.js
   create web\js\util\configJSON.js
   create web\js\util\gatewayConfig.js
   create web\js\util\config\config.json
```

Figure 18: Extension Home Folder



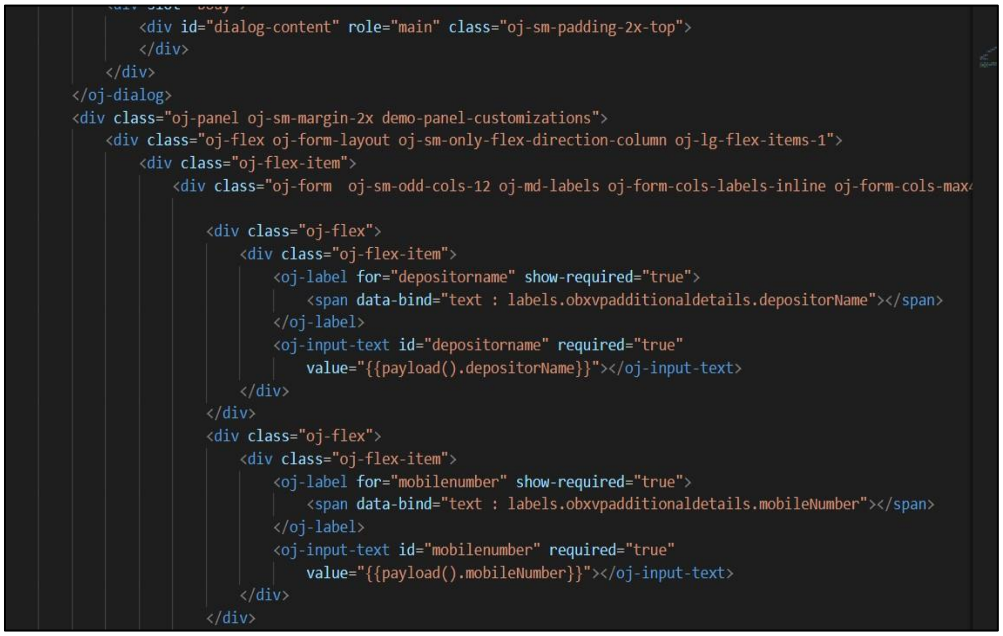
Name	Date modified	Type	Size
.git	[illegible]	File folder	
db	[illegible]	File folder	
node_modules	[illegible]	File folder	
reports	[illegible]	File folder	
src	[illegible]	File folder	
template	[illegible]	File folder	
test	[illegible]	File folder	
web	[illegible]	File folder	
app.js	[illegible]	JavaScript File	2 KB
build.gradle	[illegible]	GRADLE File	1 KB
buildExtendedComponent.sh	[illegible]	Shell Script	1 KB
generateReport.js	[illegible]	JavaScript File	1 KB
gradle.properties	[illegible]	PROPERTIES File	1 KB
gulpfile.js	[illegible]	JavaScript File	5 KB
Jenkinsfile	[illegible]	File	2 KB
karma.conf.js	[illegible]	JavaScript File	2 KB
metadata.js	[illegible]	JavaScript File	1 KB
middleware.js	[illegible]	JavaScript File	2 KB
npm-link.sh	[illegible]	Shell Script	1 KB
package.json	[illegible]	JSON File	2 KB
runTests.sh	[illegible]	Shell Script	1 KB
startCS.js	[illegible]	JavaScript File	3 KB
test-main.js	[illegible]	JavaScript File	2 KB

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.

HTML Changes

The HTML fields look like *Figure 19: HTML Changes* for all the screens. 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.

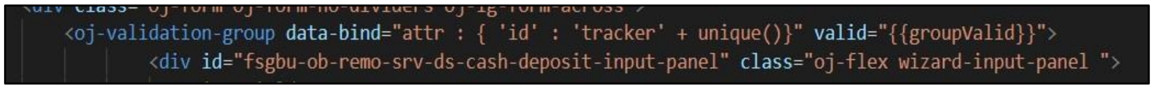
Figure 19: HTML Changes



```
            <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**

20: Validation



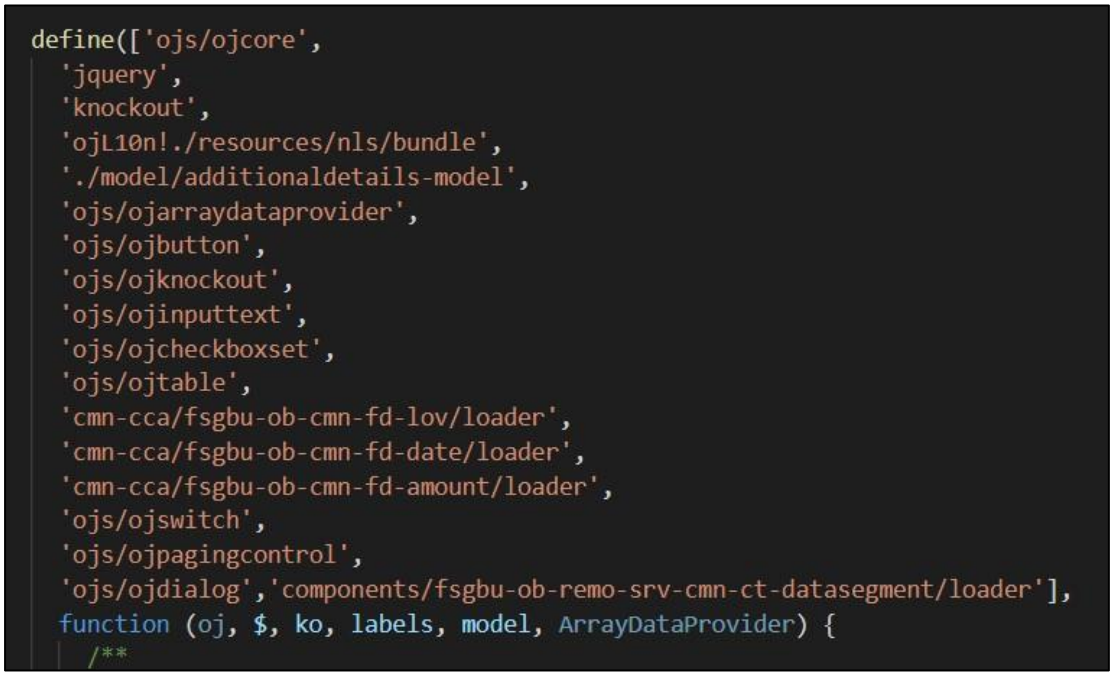
```
    <oj-validation-group data-bind="attr : { 'id' : 'tracker' + unique()}" valid="{{groupValid}}">
        <div id="fsgbu-ob-remo-srv-ds-cash-deposit-input-panel" class="oj-flex wizard-input-panel ">
```

JS Changes

Perform the following steps to implement JS changes:

1. Add all the dependencies in define block.

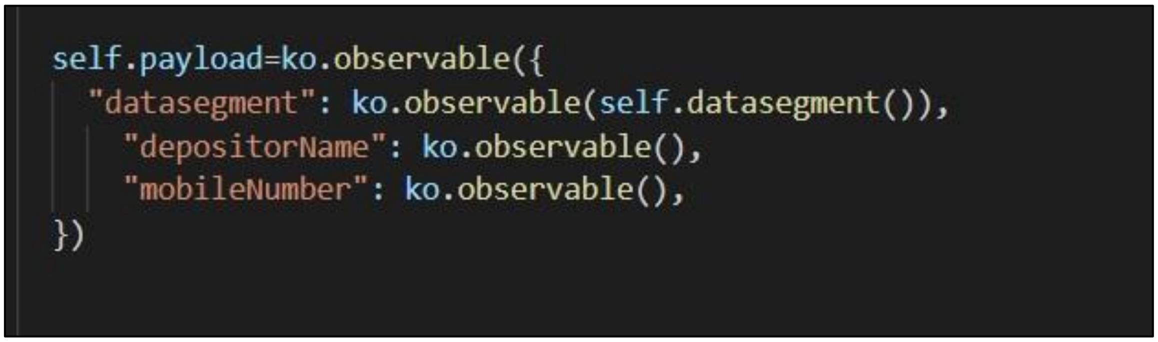
Figure 21: JS Changes



```
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) {
    /**
```

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.

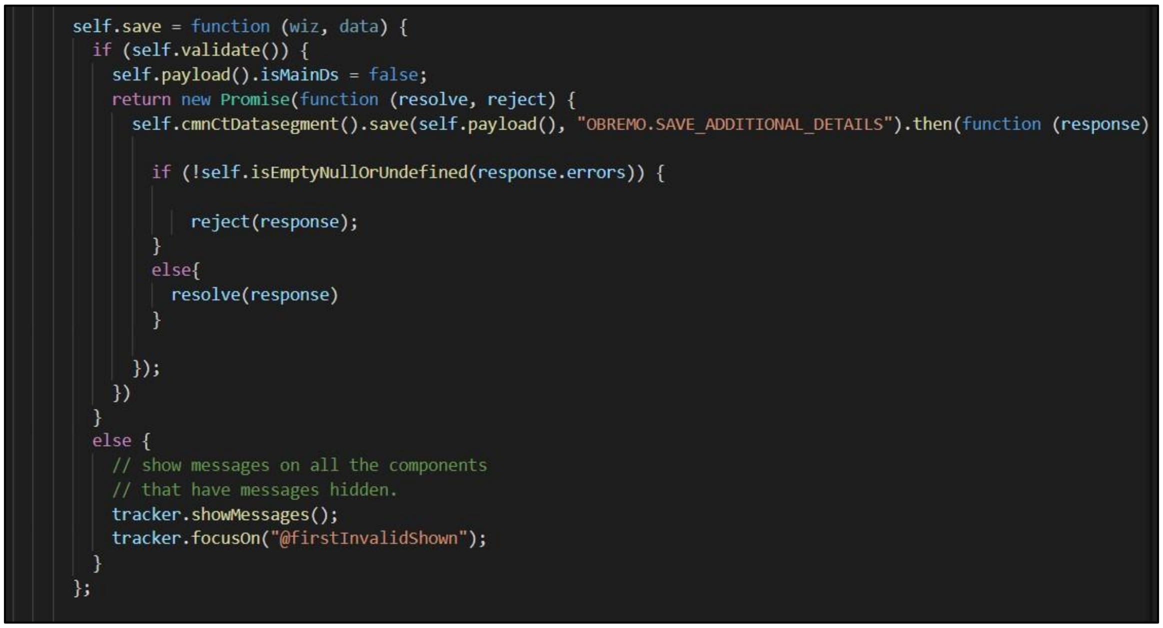
Figure 22: JS Self Payload



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

Save method implementation will look like *Figure 23: Save Method*. 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.

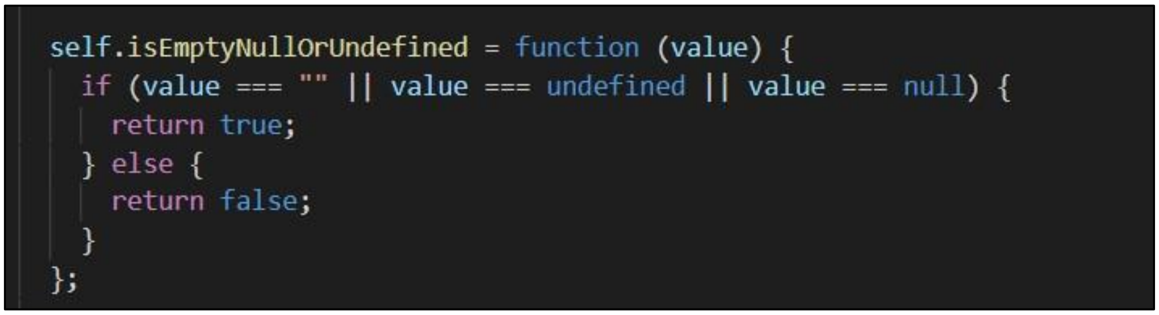
Figure 23: Save Method



```
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");
  }
};
```

The function null check is as shown below:

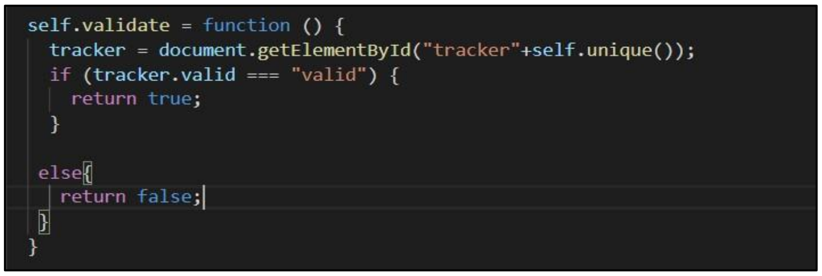
Figure 24: Function Null Check



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

The validate function is shown in the *Figure 25: Validate Function*, which will check all mandatory fields during save.

Figure 25: Validate Function

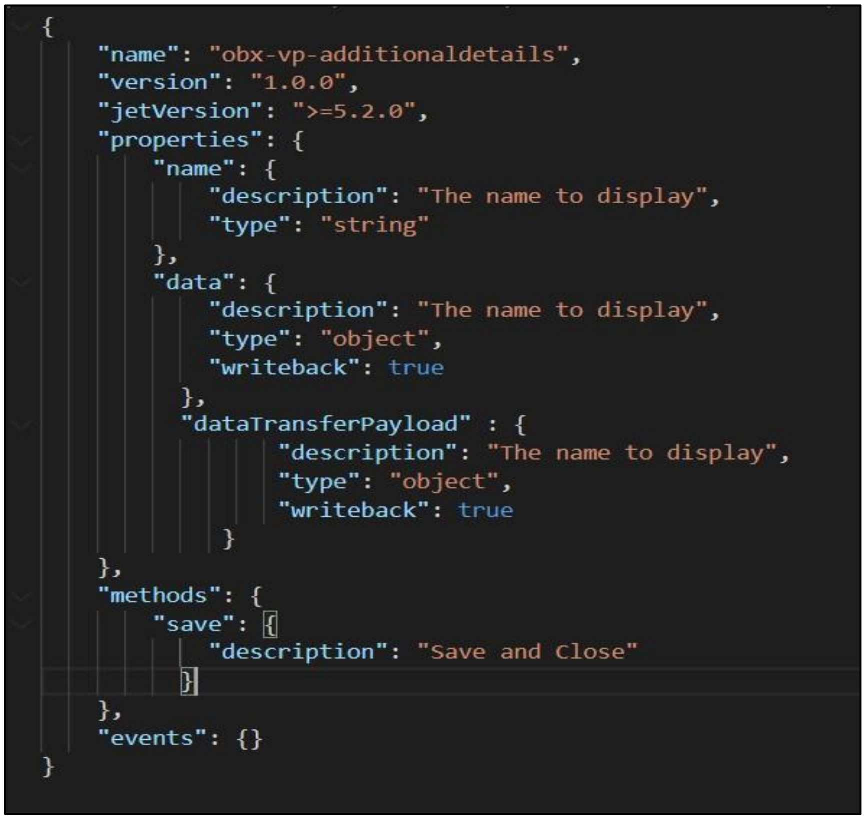


```
self.validate = function () {
  tracker = document.getElementById("tracker"+self.unique());
  if (tracker.valid === "valid") {
    return true;
  }

 else{
   return false;
 }
}
```

JSON Changes

The **data** and **datatransferPayload** properties needs 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 26: 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": {}
}
```

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.

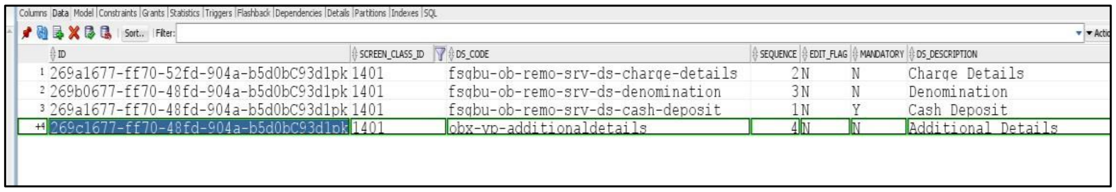
   NOTE: The OBX generates SQL script with default HEADER_APPID as PXDSSRV001 for all components. This script can be changed and used.

2. Create extended war for the component and deploy.

Database Changes

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 *Figure 27: Data Segment Entry*.

Figure 27: Data Segment Entry



	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

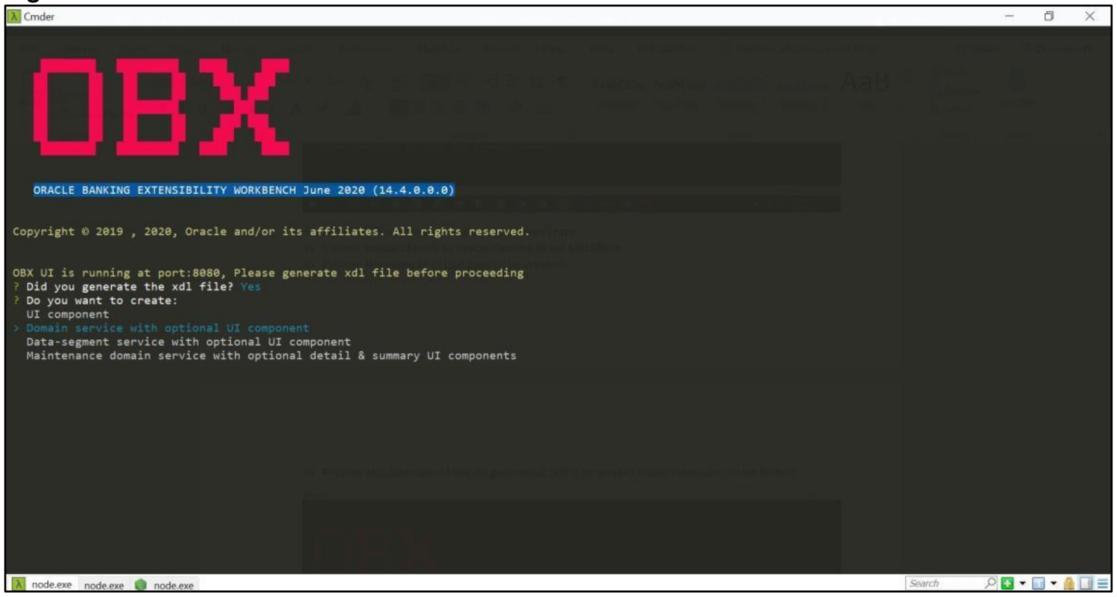
3. If the service is created separately than UI Component, change the endpoint URL in SQL script for table **PRODUCT_SERVICES_LEDGER** accordingly.

Service Component

1. Start OBX and use the XDL file that is already generated.

2. Select the domain service with optional UI component.

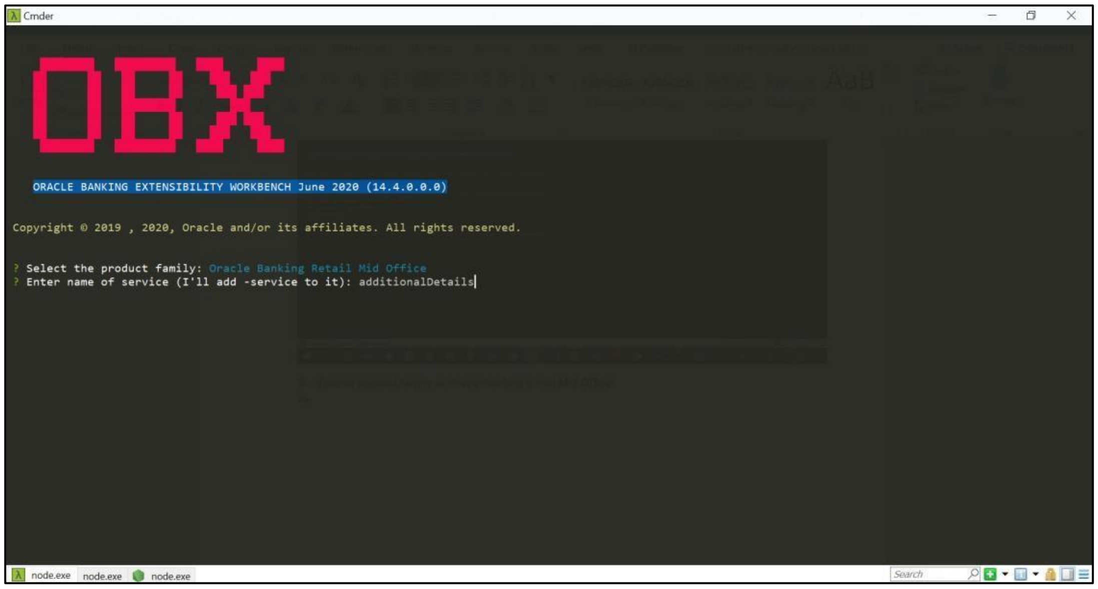
Figure 28: Domain Service



```
ORACLE BANKING EXTENSIBILITY WORKBENCH June 2020 (14.4.0.0.0)

Copyright © 2019 , 2020, Oracle and/or its affiliates. All rights reserved.

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:
  UI component
> Domain service with optional UI component
  Data-segment service with optional UI component
  Maintenance domain service with optional detail & summary UI components
```

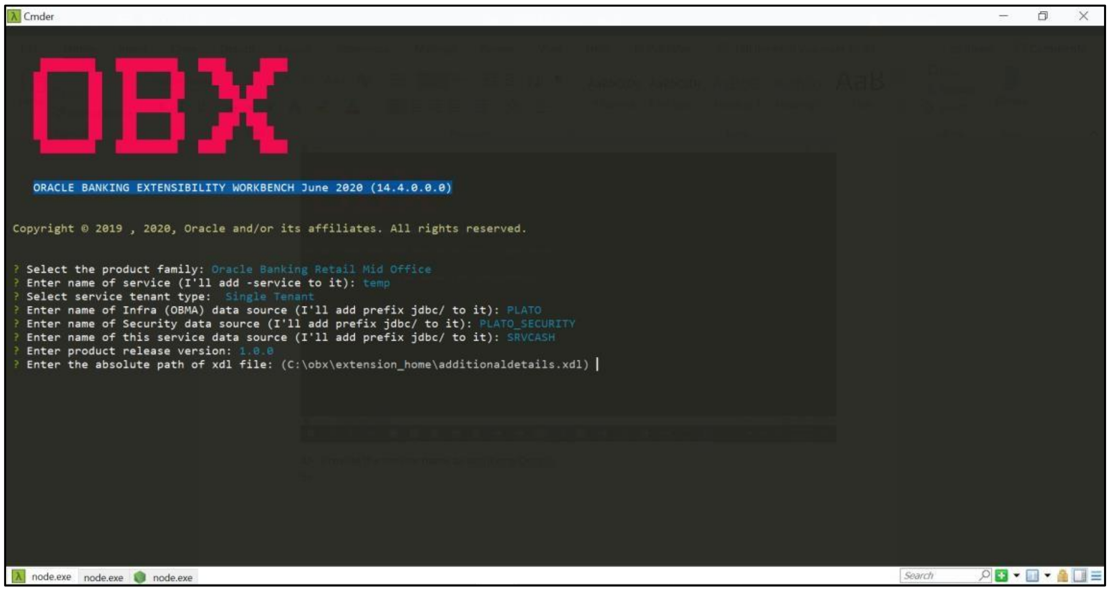
3. Select product family as **Oracle Banking Retail Mid Office**.

Figure 29: Product Family


```
ORACLE BANKING EXTENSIBILITY WORKBENCH June 2020 (14.4.0.0.0)

Copyright © 2019 , 2020, 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): additionalDetails
```

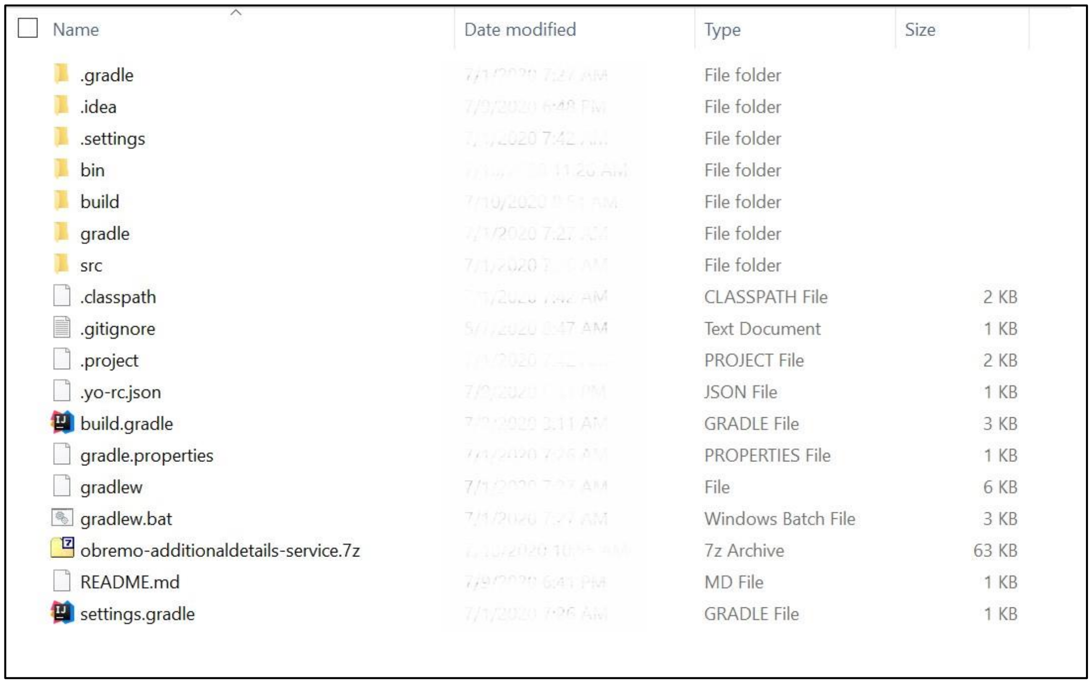
4. Specify the service name as **additionalDetails** and all the remaining details as mentioned in the *Figure 30: Service Name*.

Figure 30: Service Name


```
ORACLE BANKING EXTENSIBILITY WORKBENCH June 2020 (14.4.0.0.0)

Copyright © 2019 , 2020, 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): temp
? 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): SRVCASH
? Enter product release version: 1.0.0
? Enter the absolute path of xdl file: (C:\obx\extension_home\additionaldetails.xdl)
```

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

Figure 31: Extension Home Folder



Name	Date modified	Type	Size
.gradle	[illegible]	File folder	
.idea	[illegible]	File folder	
.settings	[illegible]	File folder	
bin	[illegible]	File folder	
build	[illegible]	File folder	
gradle	[illegible]	File folder	
src	[illegible]	File folder	
.classpath	[illegible]	CLASSPATH File	2 KB
.gitignore	[illegible]	Text Document	1 KB
.project	[illegible]	PROJECT File	2 KB
.yo-rc.json	[illegible]	JSON File	1 KB
build.gradle	[illegible]	GRADLE File	3 KB
gradle.properties	[illegible]	PROPERTIES File	1 KB
gradlew	[illegible]	File	6 KB
gradlew.bat	[illegible]	Windows Batch File	3 KB
obremo-additionaldetails-service.7z	[illegible]	7z Archive	63 KB
README.md	[illegible]	MD File	1 KB
settings.gradle	[illegible]	GRADLE File	1 KB

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. If you need to create a new schema, mention that in table **PRODUCT_SERVICES_CTX_LEDGER** while running UI Component Script.

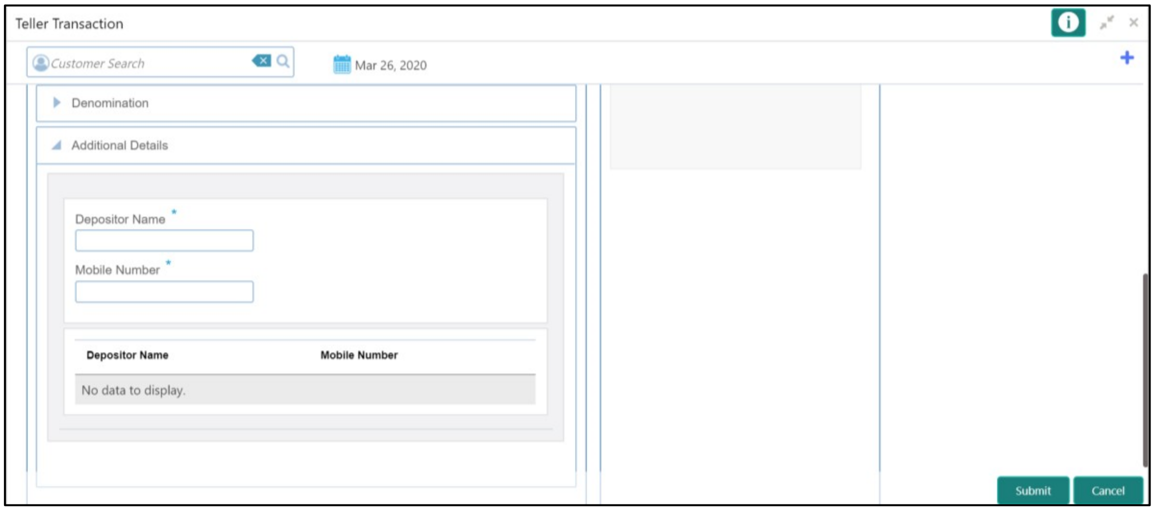
7. Restart plato servers once this change is completed.

8. 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.

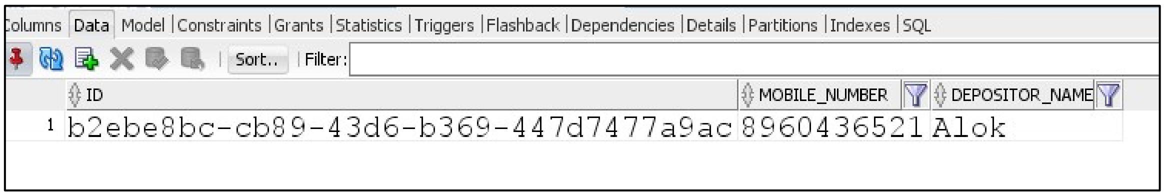
9. Fill the necessary details and click **Submit**, the data for new DS will be saved in new table.

Figure 32: Additional Details Segment



Depositor Name	Mobile Number
No data to display.	

Figure 33: Updated Data in New Table


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

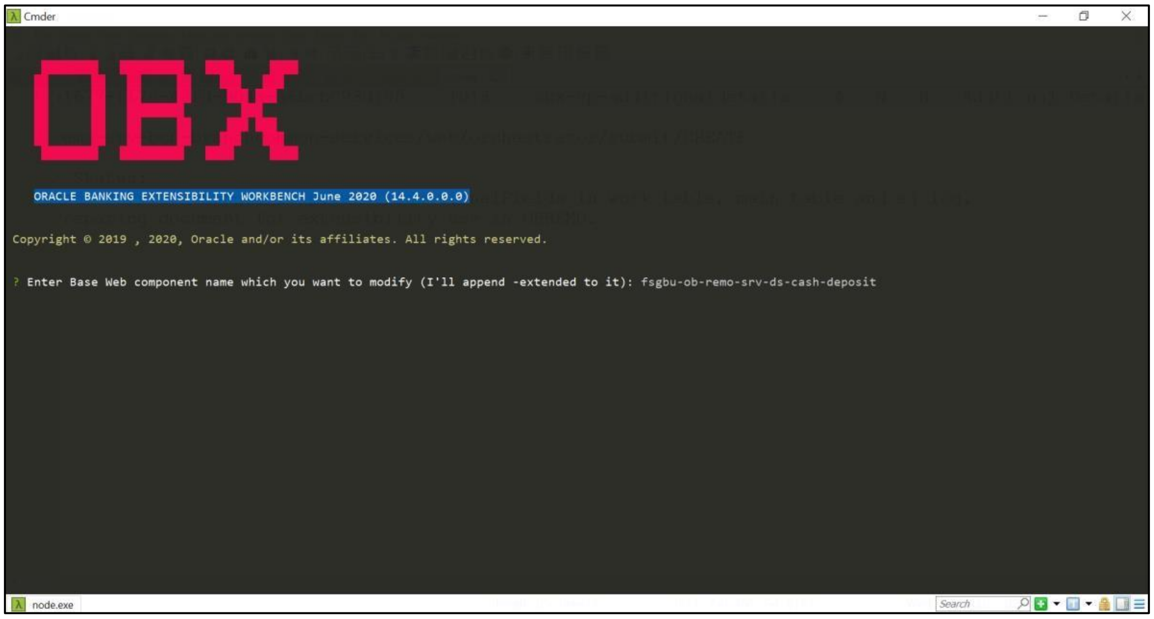
	ID	MOBILE_NUMBER	DEPOSITOR_NAME
1	b2ebe8bc-cb89-43d6-b369-447d7477a9ac	8960436521	Alok

New Field in Existing Base Data Segment

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:

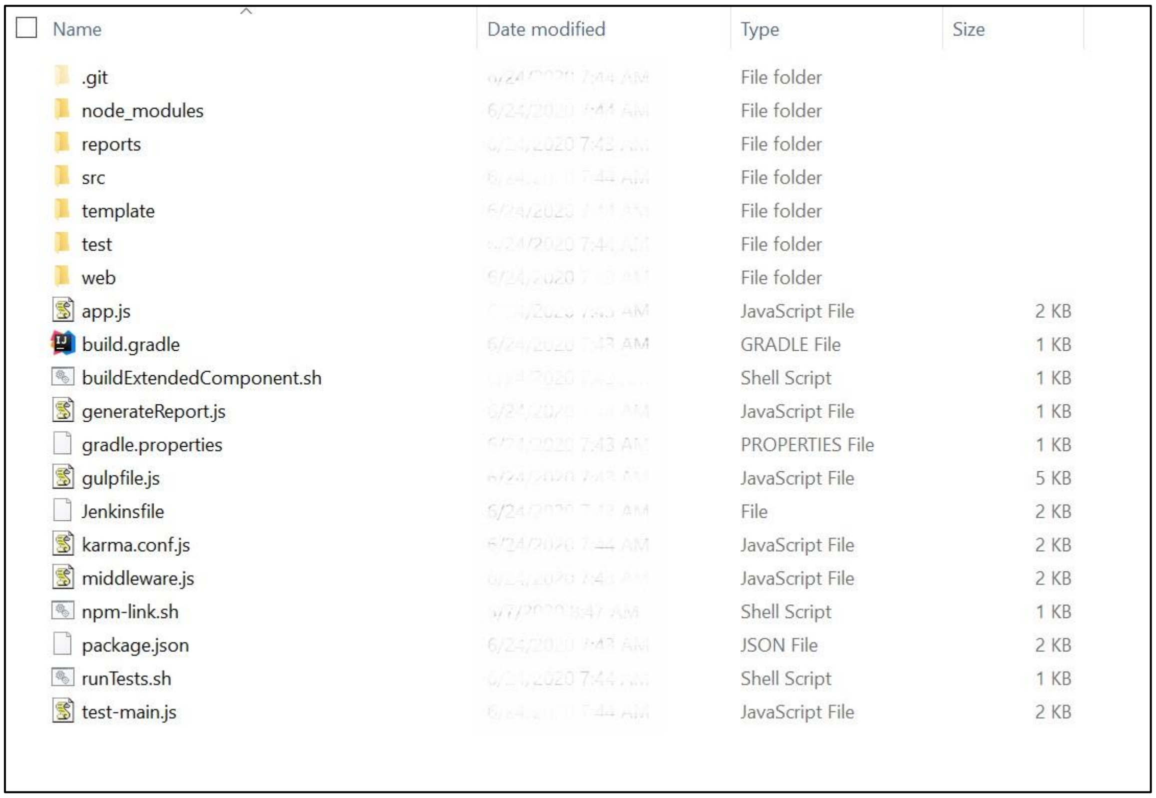
1. Modify the base component **cca** and create an extended component. To do this, start OBX and run the command **obx ui --mb**. It will prompt for name of base web component.
2. Specify the name of base web component. A folder will be created with base component name appending -extended at the end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

Figure 34: Base Web Component



```
ORACLE BANKING EXTENSIBILITY WORKBENCH June 2020 (14.4.0.0.0)

Copyright © 2019 , 2020, 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-ds-cash-deposit
```

Figure 35: Extended Folder



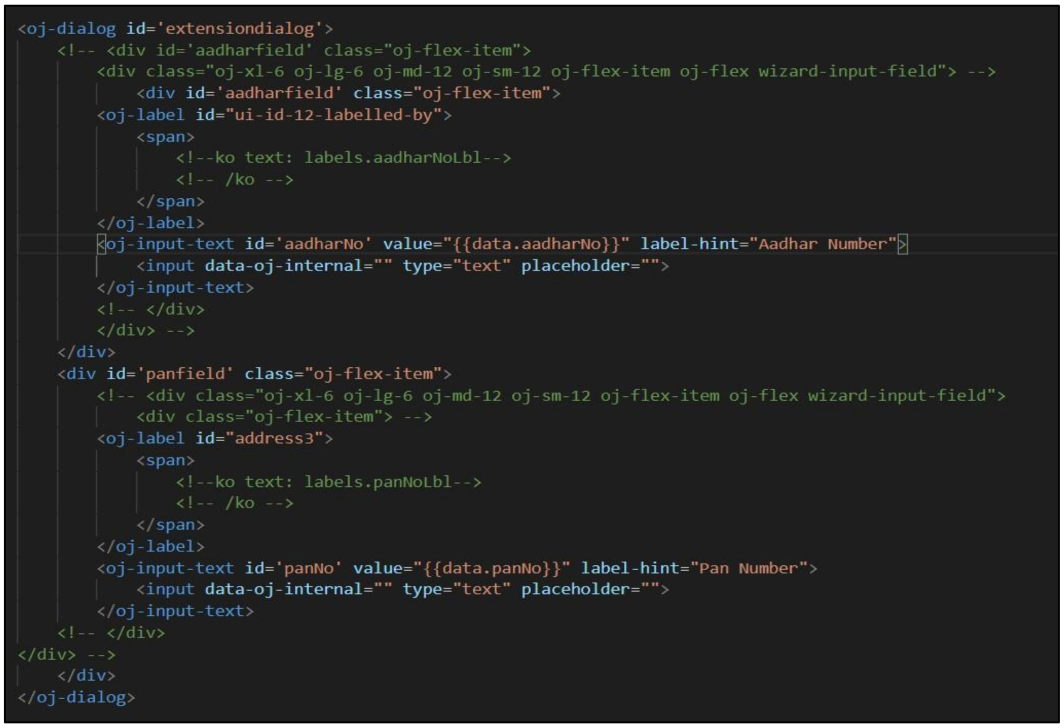
Name	Date modified	Type	Size
.git	[redacted]	File folder	
node_modules	[redacted]	File folder	
reports	[redacted]	File folder	
src	[redacted]	File folder	
template	[redacted]	File folder	
test	[redacted]	File folder	
web	[redacted]	File folder	
app.js	[redacted]	JavaScript File	2 KB
build.gradle	[redacted]	GRADLE File	1 KB
buildExtendedComponent.sh	[redacted]	Shell Script	1 KB
generateReport.js	[redacted]	JavaScript File	1 KB
gradle.properties	[redacted]	PROPERTIES File	1 KB
gulpfile.js	[redacted]	JavaScript File	5 KB
Jenkinsfile	[redacted]	File	2 KB
karma.conf.js	[redacted]	JavaScript File	2 KB
middleware.js	[redacted]	JavaScript File	2 KB
npm-link.sh	[redacted]	Shell Script	1 KB
package.json	[redacted]	JSON File	2 KB
runTests.sh	[redacted]	Shell Script	1 KB
test-main.js	[redacted]	JavaScript File	2 KB

NOTE: Changes needed in the extended component from UI side.

HTML Changes (Extended Components)

The extended component contains the boiler plate codes, in which you need to make the changes as shown in the *Figure 36: HTML Changes (Extended Component)*. After you make the necessary changes, the additional fields will be added after the existing fields in the base component.

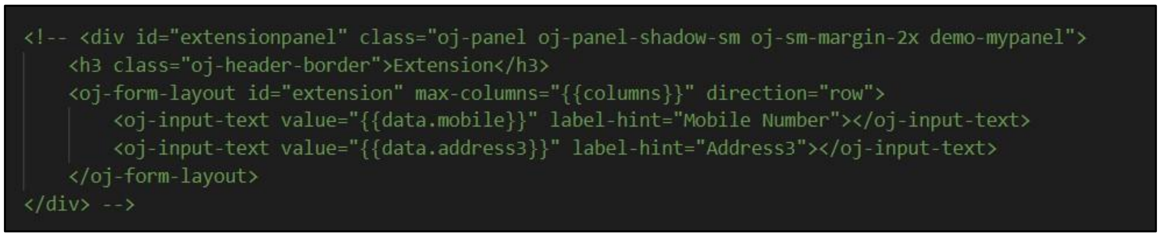
Figure 36: HTML Changes (Extended Component)



```
<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 37: Extension Panel

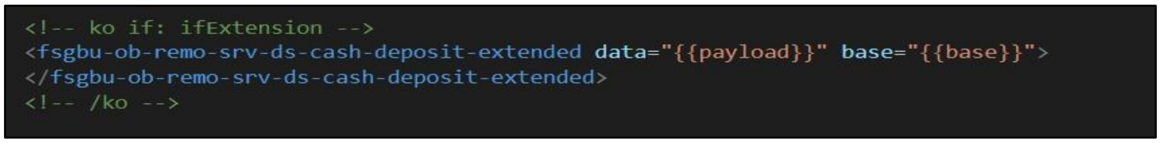


```
<!-- <div id="extensionpanel" class="oj-panel oj-panel-shadow-sm oj-sm-margin-2x demo-mypanel">
    <h3 class="oj-header-border">Extension</h3>
    <oj-form-layout id="extension" max-columns="{{columns}}" direction="row">
        <oj-input-text value="{{data.mobile}}" label-hint="Mobile Number">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

HTML Changes (Base Component)

Perform the HTML changes in the base component as shown in *Figure 38: HTML Changes (Base Component)*.

Figure 38: HTML Changes (Base Component)

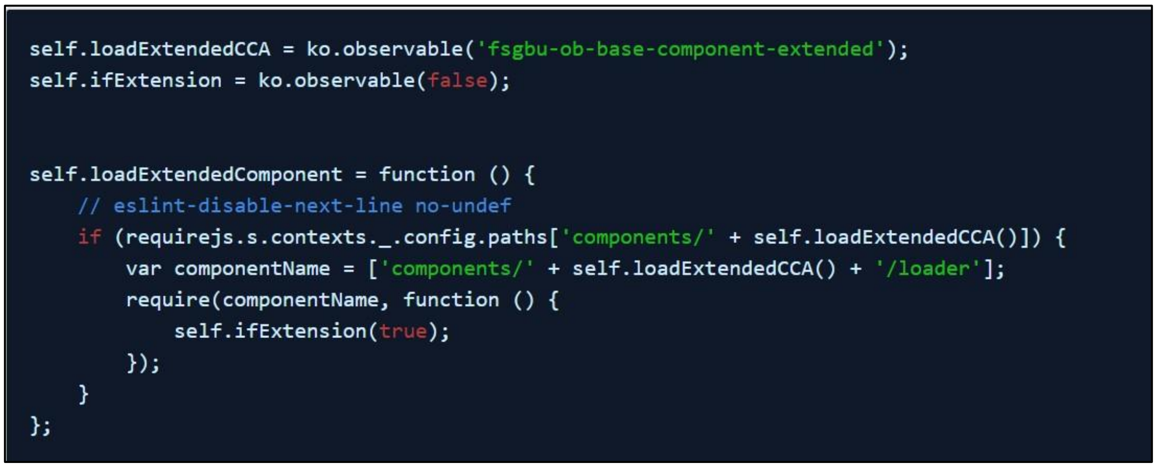


```
<!-- ko if: ifExtension -->
<fsgbu-ob-remo-srv-ds-cash-deposit-extended data="{{payload}}" base="{{base}}">
</fsgbu-ob-remo-srv-ds-cash-deposit-extended>
<!-- /ko -->
```

JS Changes (Base Component)

Perform the JS changes in the base component as shown in *Figure 39: JS Changes (Base Component)*.

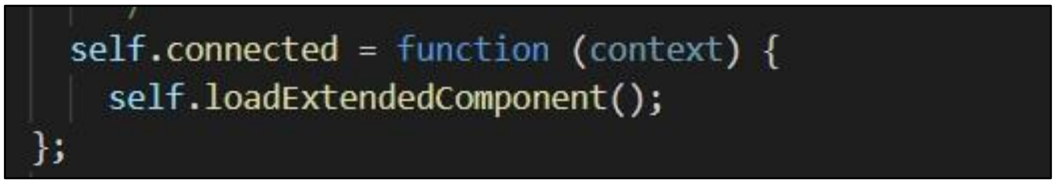
Figure 39: JS Changes (Base Component)



```
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 40: Self Connected Method

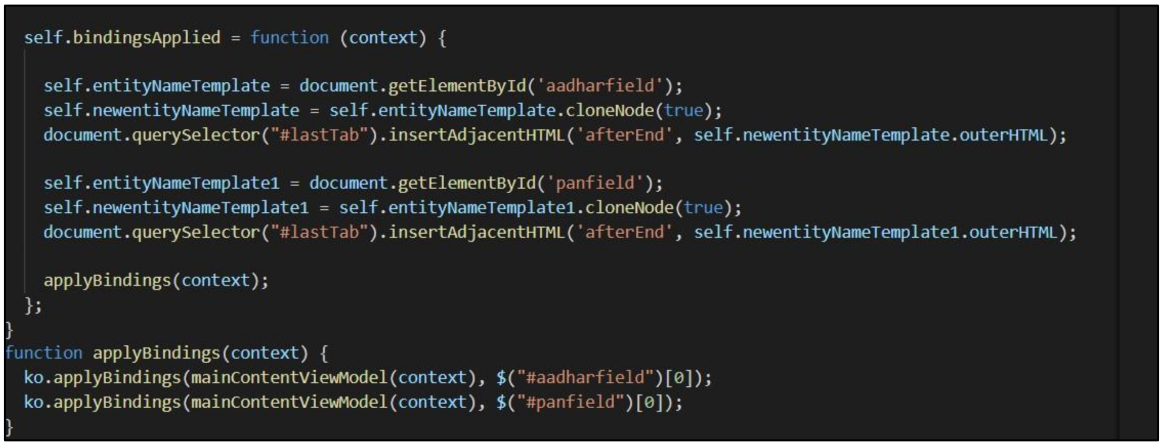


```
  self.connected = function (context) {
    self.loadExtendedComponent();
};
```

JS Changes (Extended Component)

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 41: JS Changes (Extended Component)

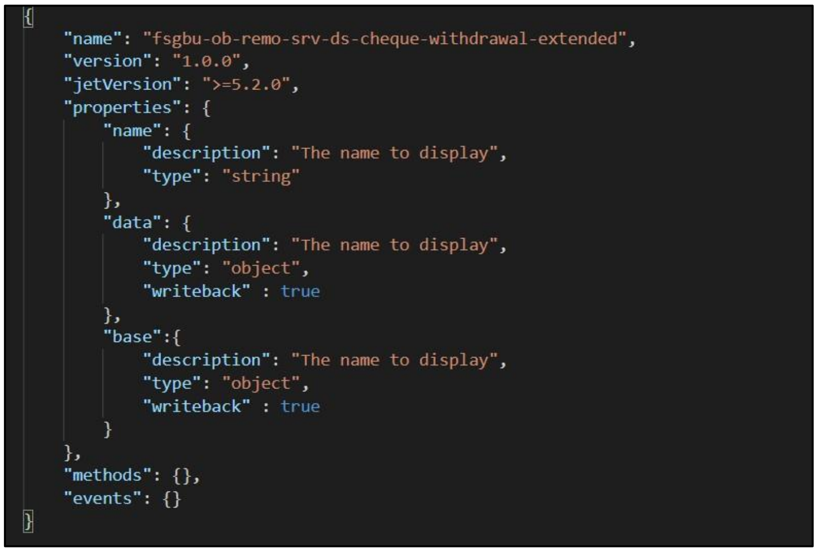


```
  self.bindingsApplied = function (context) {

    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]);
}
```

JSON Changes (Extended Component)

Perform the HTML changes as shown in *Figure 42: JSON Changes (Extended Component)* to add data and base property for extended component.

Figure 42: JSON Changes (Extended Component)

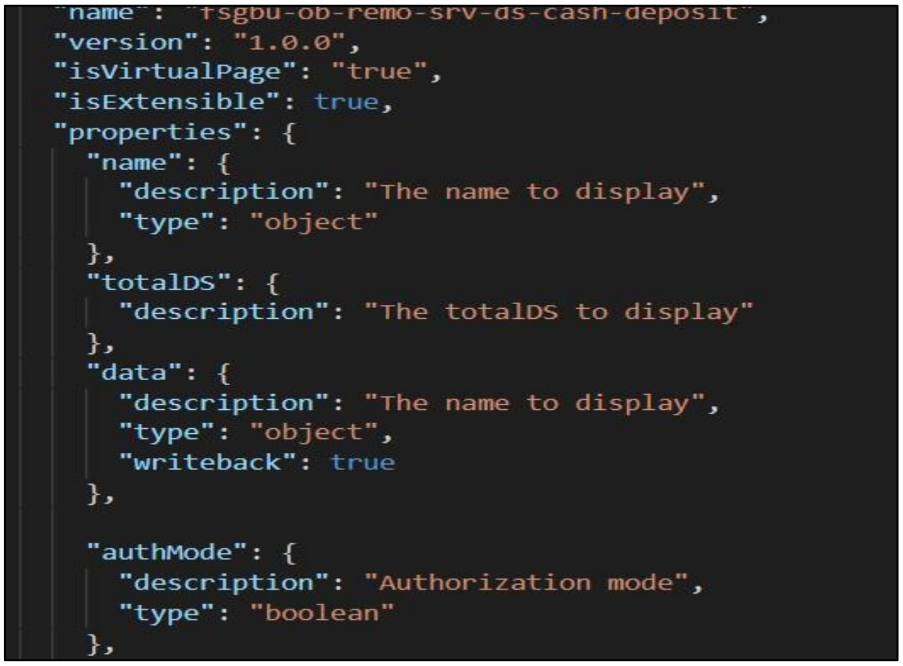


```
{
    "name": "fsgbu-ob-remo-srv-ds-cheque-withdrawal-extended",
    "version": "1.0.0",
    "jetVersion": ">=5.2.0",
    "properties": {
        "name": {
            "description": "The name to display",
            "type": "string"
        },
        "data": {
            "description": "The name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

JSON Changes (Base Component)

In base component JSON file, the properties **isExtensible** 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 43: 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"
    },
```

DB Changes

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.

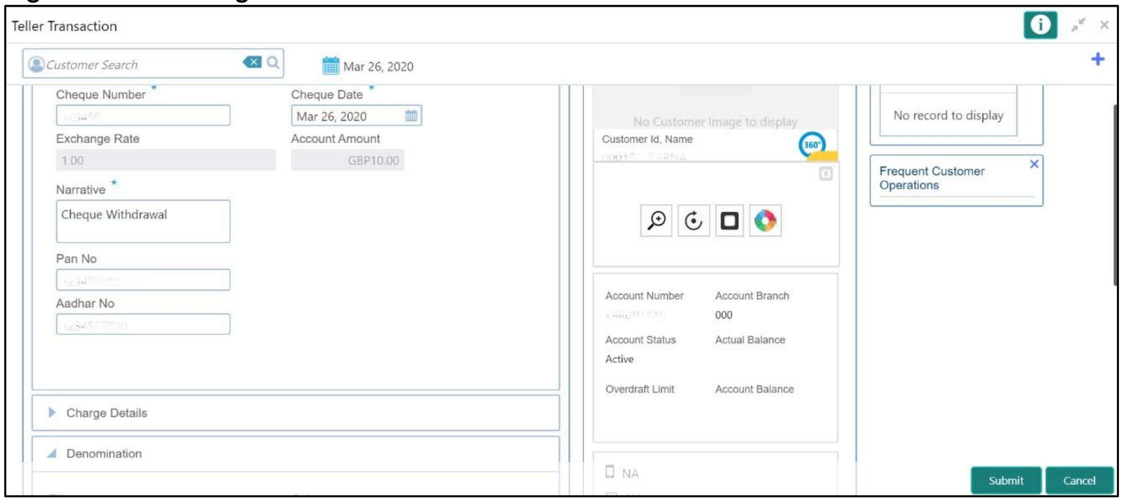
3. For persistence of data in main table, add **additionalFields** with data type String in model class.

4. Deploy the changed service, extended war component, and changed appshell.

   NOTE: After deployment, the two additional fields named **Pan No** and **Aadhaar No** 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 44: Data Segment with Additional Fields



Teller Transaction

Customer Search

Mar 26, 2020

Cheque Number *

Cheque Date *: Mar 26, 2020

Exchange Rate: 1.00

Account Amount: GBP10.00

Narrative *: Cheque Withdrawal

Pan No

Aadhar No

Charge Details

Denomination

No Customer Image to display

Customer Id, Name

Account Number

Account Branch: 000

Account Status: Active

Actual Balance

Overdraft Limit

Account Balance

NA

No record to display

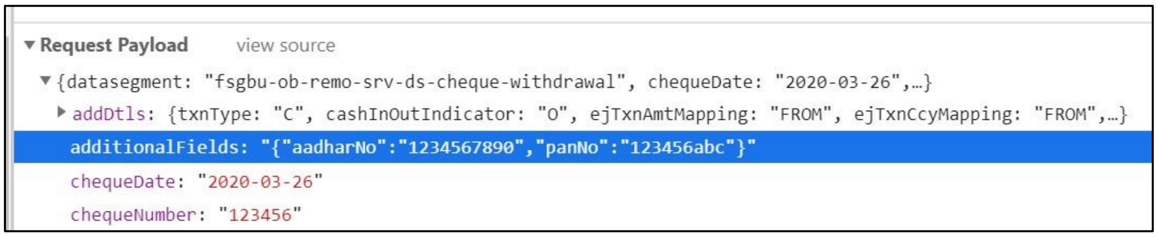
Frequent Customer Operations

Submit

Cancel

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

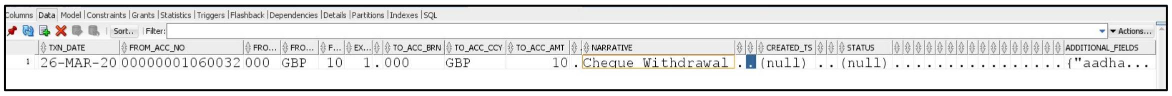
Figure 45: Request Payload



```
▼ 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"
```

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

Figure 46: SRV_TB_CH_CASH_TXN Table



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

Sort... Filter: Actions...

	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...

Index

Reference and Feedback

Reference

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

- Oracle Banking Extensibility Workbench Installation Guide

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

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 or if you still need assistance, please contact documentation team.