Transaction CHATBOT Integration with OFSLL
Overview and Developer Guide
Oracle Financial Services Lending and Leasing
Release 14.10.0.0.0
Part No. F35490-01
December 2020

ORACLE
Financial Services

Table of Contents

1. OFSLL Transaction Overview and Developer Guide

1.1 Introduction

OFSLL has an extended out of the box support for CHATBOT integration. This provides a new framework for direct user interaction with the system. However, since OFSLL is a back-office system there are additional external components required to be integrated to host and utilize the CHATBOT functionality.

Currently, OFSLL integration with CHATBOT is supported with some of the functionalities such that end users can search for documentation and / or query and fetch the account related information and/or perform other actions on an account with options presented in CHATBOT menu.

This document outlines the integrated framework and procedures required to implement certain features, but it is not a general-purpose configuration manual.

For latest version of this document, refer to
https://docs.oracle.com/cd/F35490_01/pdf/refdocs/ofsll_transactionbot_overview_and_developer_guide.pdf

Following topics are discussed in this section:

1.1.1 Transaction Bot Overview

OFSLL integrated Transaction bot (Transaction posting chatbot) is a functionality for product end-users to query account related details, outstanding dues and post simple account related updates as a transaction. In addition, there is also dynamic content search capability provided within the Transaction bot. For information on Documentation search using chatbot, refer to 'OFSLL Docubot Overview and Developer Guide'.

The Transaction ChatBot is hereafter is referred to as 'BOT' in the document.

1.1.2 Purpose

The purpose of this document is to demonstrate the capability of OFSLL BOT in handling transactional updates to accounts maintained in the system by integrating with Oracle Digital Assistant (ODA). This document is intended to detail the usability features and also to serve as a developer guide to understand the configuration procedures. However, the features and options presented are provided only as a sample and needs further customization based on requirements.

1.1.3 Audience

In general, this document is intended to all those parties and decision makers who are interested to know about OFSLL BOT integrated framework. The configuration sections are intended for system administrators, consulting and implementation teams who deploy customized solutions for customer.

1.1.4 Accessibility

The OFSLL BOT integrated framework is supported from OFSLL 14.10.0.0.0 release.

OFSLL being a back office system, only the data in the system is can be exposed using REST services and the interface for BOT facility is recommended to be configured on any 3rd party web application or customer self-service portal or lenders/financial services website for the benefit of end-users.

However, the account related services provided in this framework is just a sample and needs to be customized based on requirement. BOT is agnostic of which self-service site / portal is used to provide access and interface to the users for help documentation.

1.1.5 Access

Currently the framework supports basic authentication (not OAUTH). User Management and authentication needs to be handled as part of the implementation.

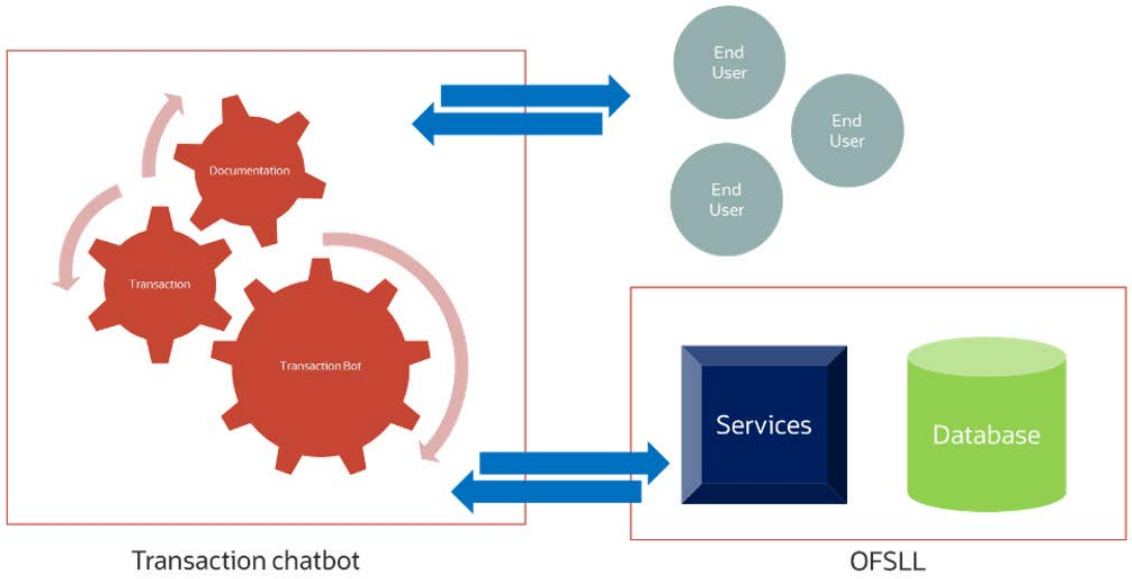
1.1.6 Pre-requisites

Following are the pre-requisites:

- The BOT is designed to work in ODA framework. The configuration is to be done as detailed in Developer Guide for BOT Customization section.
- Also the ODA Server Environment has to be licensed separately. For more information, refer to https://www.oracle.com/in/chatbots/digital-assistant-platform/
- Need to have release specific pre-indexed file for elastic search to work.
- Adequate space to store the indexed file directories in the respective folders.
- WebLogic server for deployment of war file.
- The parameters in 'Channel.Properties' file are to be configured before creating and deploying the .war file. For details, refer to 'BOT Configuration' section.

1.2 Architecture

The BOT connects to OFSLL and provides an interface to give results for the below mentioned table items. With the current structure BOT seamlessly integrates with Services and documentation of the current release of the product.



The documentation elastic search for OFSLL BOT requires pre-indexing of content. Hence, indexing is done for 14.10.0.0.0 release. The indexing process is done automatically using the third-party plugins such as Apache Lucene and Jsoup to identify unique keywords in HTML files. This generates indexed files which serves as common directory for searched keyword and the file instance where it exists.

For more information on third-party plugins used, refer to 'Third Party Licenses' section.

1.3 Features of BOT

Following are the unique features of OFSLL BOT:

- Account details view using Account # query
  - View Account Summary
  - Check the Next Payment Date
  - View and Update default Communication Preference
- Readily available navigation links to the following:
  - Link to all Release documentation
  - Dynamic Document Search option
  - Link to currently mapped Product Release notes
  - Listing of Product Module / Classified Guides
  - Link to list of indexed Keywords
  - Link to Getting Started Video gallery
- Intuitive Menu options:

- Option to clear chat data
- Speech Conversion – Voice based Input
- Network Status indication

1.3.1 Support of Text and Voice Based inputs

The BOT can support both Text and Voice based inputs to find information. This attempts to comply with multiple accessibility options.

The BOT is enabled with voice based inputs where in voice commands are accepted as input equivalent to typing or clicks. This option works on clicking the Mic button.

During text based input, the response is provided in the BOT interface. In a voice based input, the response is provided in both voice based response and BOT response simultaneously.

However, note that voice based input does not support to open a URL (link) reference.

1.3.2 Release Specific Indexing

Indexing is done for the following releases of OFSLL and indexed files are provided in respective folder. The mapping of Release number v/s Folder name and Part Number is indicated below:

Release No	Folder Name	Part Number
14.10.0.0.0	14.10	F53490_01

1.4 BOT Configuration

For the BOT to function, the following parameters are to be defined in the Properties file as indicated below. This file can be accessed from the location <OFSLL Installed Directory path>/ web_interface/ofsllbot/src.

This section contains the following:

The below tables lists all the parameters of the properties file. However, only those fields marked as 'Y' in Update required (Y/N) column are to be updated.

Sl. No	Parameter Name	Fields	Description	Update required (Y/N)	Sample
1	paymentPurposeRequired=Y	Boolean	Captures the Payment purpose Required	N	Y
2	accessToken=	String	Captures the access token	N	
3	proxyIP=	String	Captures the Proxy	N	
4	proxyPort=	Integer	Captures the Proxy Port	N	

Sl. No	Parameter Name	Fields	Description	Update required (Y/N)	Sample
5	googleAPI Key=	String	Captures the Google API key	N	
6	imageUrl=	Path	Captures the Image URL	N	
7	defaultHom eEntity=	String	Captures the home entity	N	
8	stockCode =	String	Captures the Stock Code	N	
9	moneyTran sferPay=	String	Captures the Money Transfer Pay	N	
10	defaultBas eContext=	String	Captures the default base content	N	
11	sessionExp iryInMinute s = 15	Integer	Captures the Session timeout value	N	
12	ofsll.suffix = htm	String	Suffix of the files	N	Keep as .htm
13	ofsll.otmHtt pUrl=https:/ /docs.oracl e.com/cd/	String	Captures the suffix for OTM Url	N	Keep as https://docs.oracl e.com/cd/
14	ofsll.fIndex =/findex.ht m	String	Captures the Findex path	N	Keep as /findex.htm
15	ofsll.index= index.htm	String	Captures the index.htm	N	Keep as index.htm
16	ofsll.video= /videos.htm	String	Captures the video file path	N	Keep as /video.htm
17	ofsll.ofsllRe leaseNotes =/pdf/refdo cs/ofsll_rel ease_notes .pdf	String	Captures the OFSLL release notes suffix	N	Do not change

Sl. No	Parameter Name	Fields	Description	Update required (Y/N)	Sample
18	ofsll.ofsllReleaseDoc=https://docs.oracle.com/en/industries/financial-services/financial-lending-leasing/index.html	String	Captures the OFSLL release doc URL	N	Do not change
19	ofsll.splitSeperator==	String	Captures the Split separator	N	Do not change
20	ofsll.maxHitsResults=100	String	Captures the Max no of its results of the document query	Y (optional)	Change depending upon search results
21	ofsll.baseURL =	String	Captures the Service API URL	Y	Application URL
22	ofsll.username =	String	Captures the username of weblogic server	Y	Weblogic username
23	ofsll.pasd =	String	Captures the Password of weblogic server	Y	Weblogic password
24	ofsll.indexDir =/folder path	Path	Captures the complete folder path where index files are placed (In this location, copy the index files from respective release folder)	Y	Change as per server indexed folder
25	ofsll.releaseVersionUrl=	Path	Captures the Part Number	Y	Refer Release Specific Indexing table.
26	ofsll.releaseNo=	Decimal	Captures the Release Number		Refer 'Folder Name' column Release Specific Indexing table.

1.4.1 Sample Workflow

While interacting with BOT, you need to input the basic details (like customer ID) to start and further drill down to explore multiple account options available.

Following image is an illustration of the workflow and also, one of the scenario is detailed as an example to indicate the BOT workflow in 'BOT Usability Workflow' section.

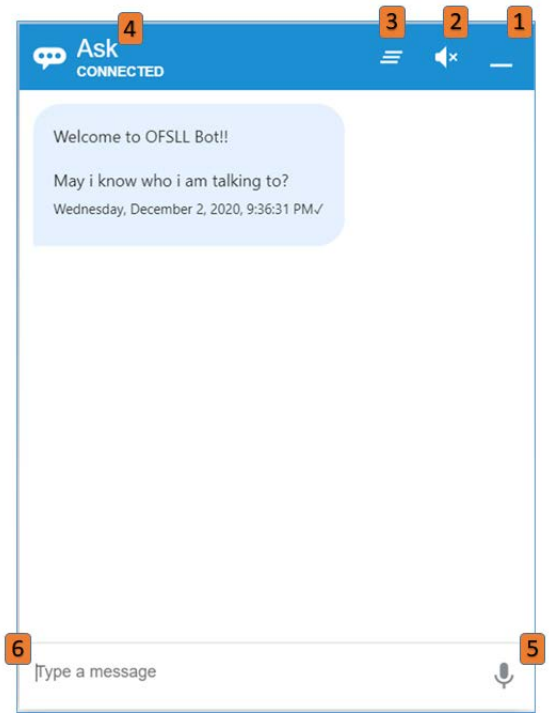
To Start with, enter your name and confirm if you want to continue using the bot.Based on your intent the bot starts building the answers.

Sl.No.	Menu
1	Begin with entering a Customer ID / Account number.
2	Click on required account from the list of accounts belonging to the Customer ID
3	View the Account Summary. Click Payment Details option form the list.
4	View the account details with below menu options Today's payoff Quote Last 5 Transactions Insurance Details Last Billing details Number of Terms remaining Need more help
5	View the payment details Last 5 payments Next Payment date Advance disbursement request for 2000 $ Need more help
6	View the communication preference Current Preferences Update Preference Need more help

Sl.No.	Menu
7	View the Limit details Display current limit details Master Account rolled-up Balances Need more help
8	Click on the “OFSLL documentation tree” The available options are OFSLL release documentation Document Search Product release notes Product classified guides Find by indexed keyword Getting started videos Need more help
9	Click “Need more help” The user has a the option to continue with the same customer id or enter new customer id

Also, one of the scenario is detailed as an example to indicate the Chatbot workflow in OFSLL. Refer to ‘BOT Usability Workflow’ section.

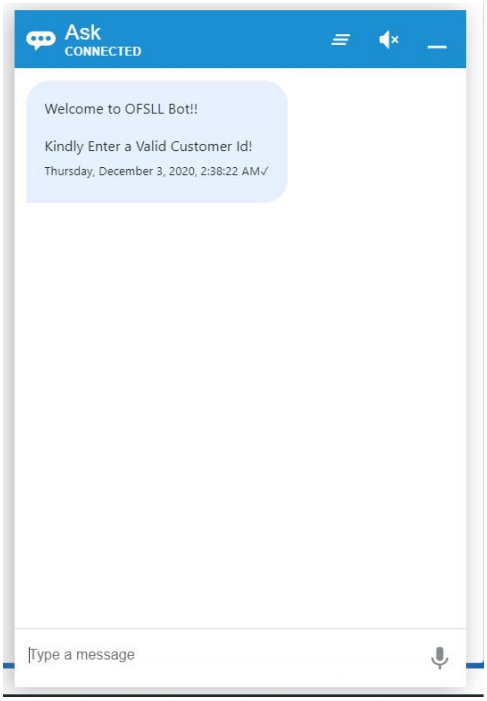
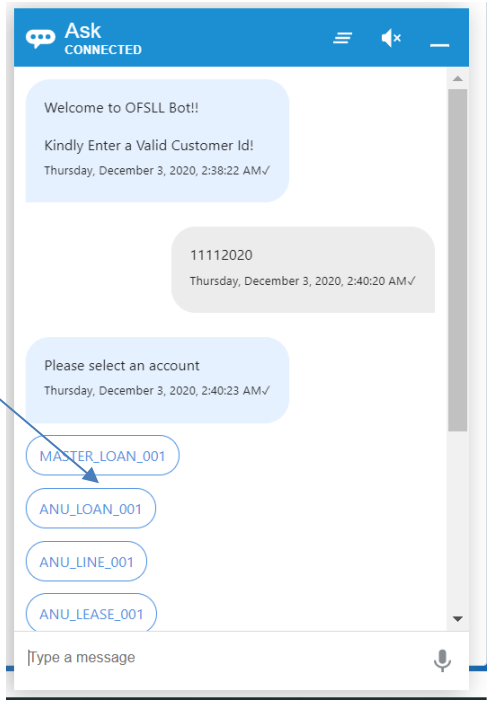
1.4.2 BOT UI Elements

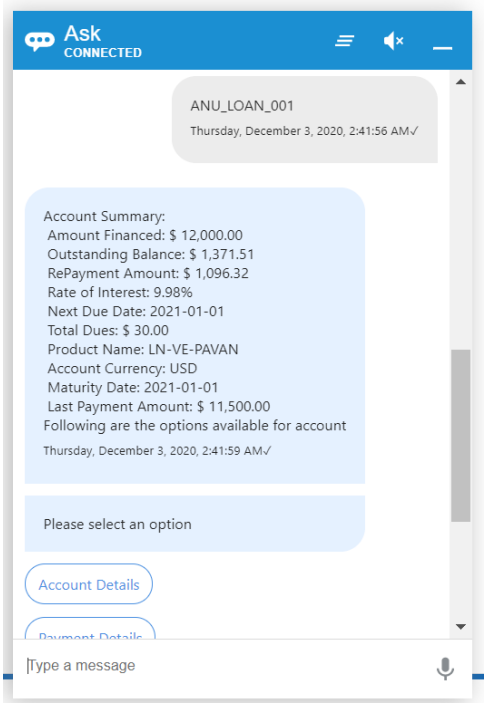
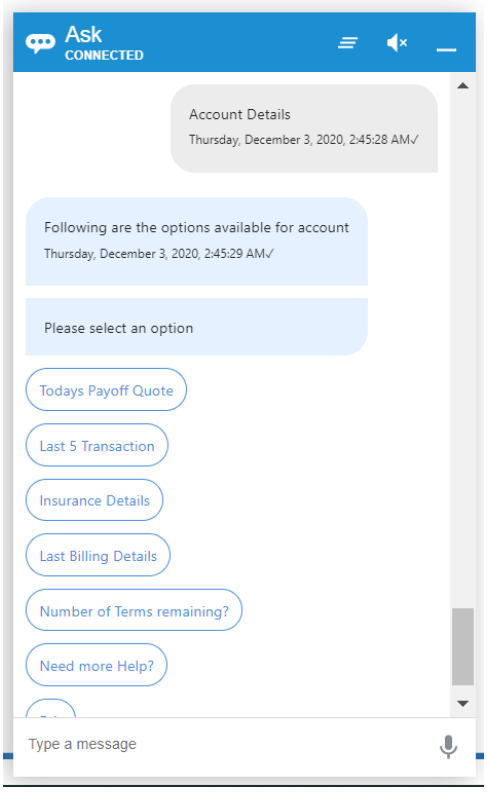


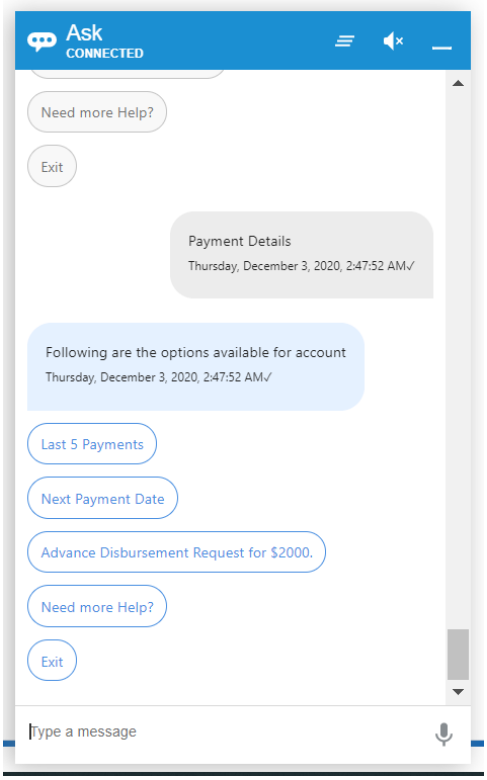
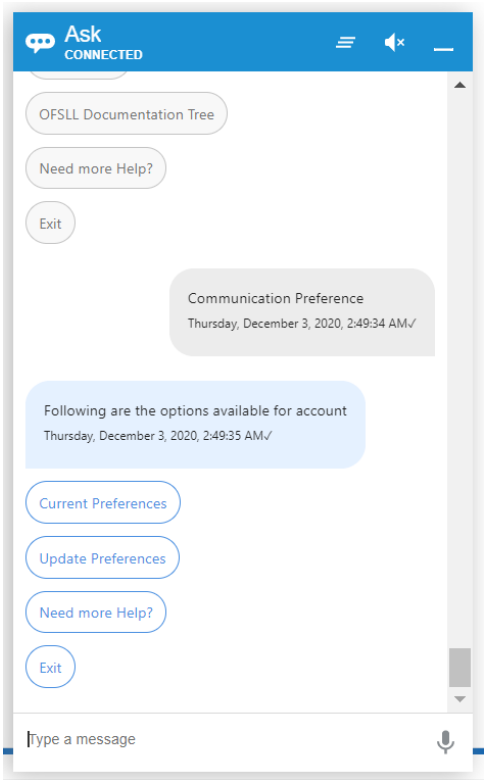
Sl.No	Option	View / Action
1	Minimize	Minimize BOT window
2	Speaker output	Enable BOT in speaker mode
3	Clear chat	Clear all messages in the BOT
4	Network Status	Connected: BOT connected to server Connecting: BOT awaiting network connection
5	Mic Input	Enable Mic for voice based input
6	Text Input	Enter search string using keyboard

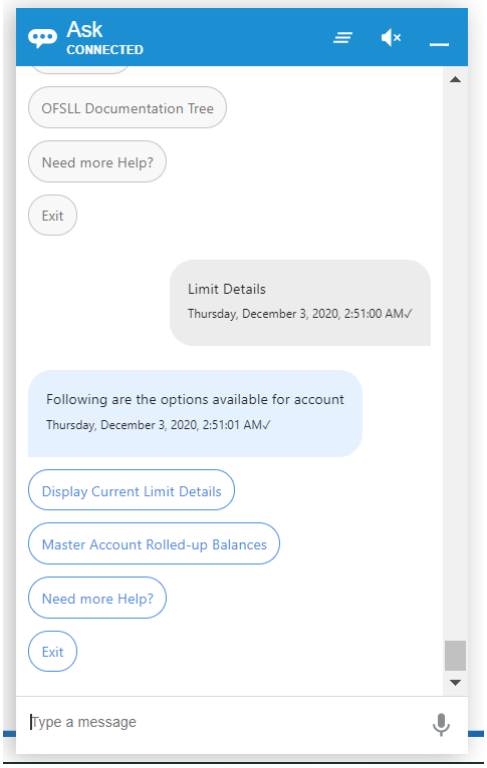
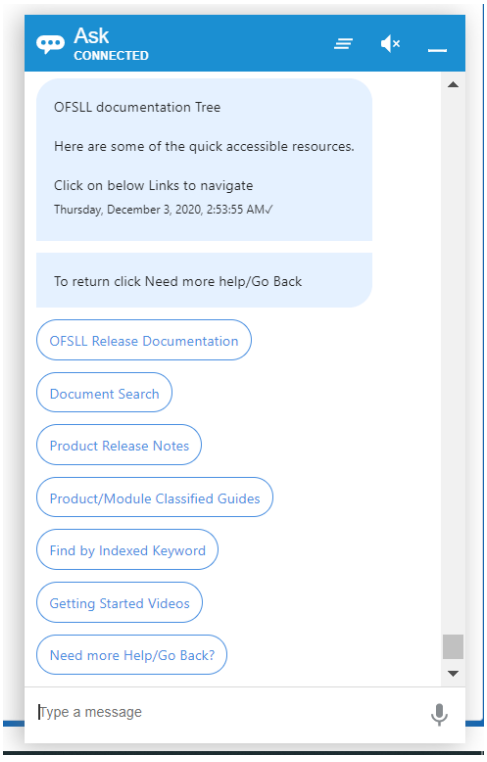
1.4.3 BOT Usability Workflow

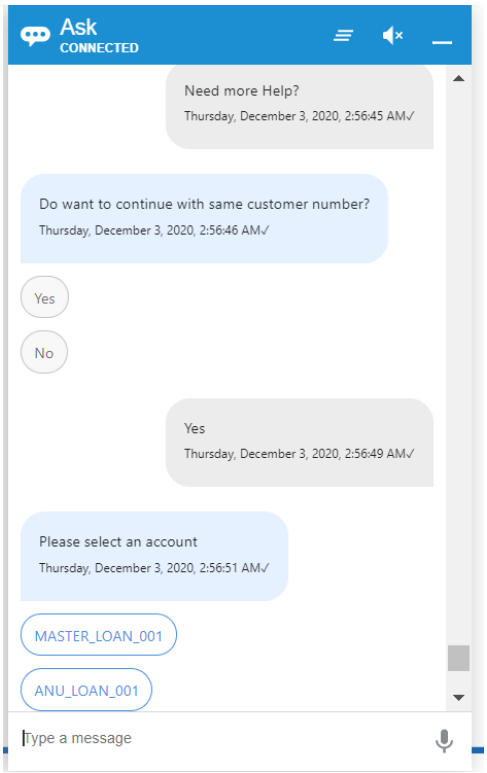
Following is a sample workflow indicating the following steps performed in chatbot:

Action	BOT Response
Begin with entering a Customer ID / Account number.	
Click on required account from the list of accounts belonging to the Customer ID	

Action	BOT Response
View the Account Summary. Click Payment Details option form the list.	
View the account details with below menu options **Today's payoff Quote** **Last 5 Transactions** **Insurance Details** **Last Billing details** **Number of Terms remaining** **Need more help**	

Action	BOT Response
View the payment details **Last 5 payments** **Next Payment date** **Advance disbursement request for 2000 $** **Need more help**	
View the communication preference **Current Preferences** **Update Preference** **Need more help**	

Action	BOT Response
View the Limit details **Display current limit details** **Master Account rolled-up Balances** **Need more help**	
Click on the “OFSLL documentation tree” **The available options are** **OFSLL release documentation** **Document Search** **Product release notes** **Product classified guides** **Find by indexed keyword** **Getting started videos** **Need more help**	

Action	BOT Response
Click “Need more help” **The user has a the option to continue with the same customer id or enter new customer id**	

1.5 Third Party Licenses

OFSLL BOT uses the following third party licenses:

- Apache Lucene, Version: 8.5.1

  The Apache Software Foundation, Technology: Lucene, Version: 8.5.1

  Files used (below are part of Apache Lucene 8.5.1)

  Lucene Core (8.5.1)

  Lucene query parser (8.5.1)

- JSOUP 1.13.1

  Jsoup is a Java library for working with real-world HTML.

  It provides a very convenient API for fetching URLs and extracting and manipulating data, using the best of HTML5 DOM methods and CSS selectors.

  jsoup implements the WHATWG HTML5 specification, and parses HTML to the same DOM as modern browsers do.

  scrape and parse HTML from a URL, file, or string

  find and extract data, using DOM traversal or CSS selectors

  manipulate the HTML elements, attributes, and text

  The purpose of using Jsoup in chatbot is to read the html elements <tags> <href> and use it as a added part of indexing

  Link : https://jsoup.org

For detailed information, refer to product licensing guide.

2. Developer Guide for BOT Customization

This section of the document intends to help you to set up and configure Oracle Digital Assistant (ODA) 'ASK' with the sample OFSLL wrapper. However, the instructions are provided in brief and for any additional information, contact Oracle Financial Services Lending and Leasing Product Engineering team.

Note: Currently this framework supports basic authentication provided by OFSLL REST service. OAUTH authentication is not supported. Additionally, OBDX (Oracle Banking Digital Experience) can be integrated for user authentication purpose. For more information, refer to documentation at https://docs.oracle.com/cd/E97825_01/webhelp/Content/obdx/core/authentn/authntctn.htm

Following topics are discussed in this section:

2.1 Pre-requisites

Following are the mandatory pre-requisites:

- OFSLL being a back-office system with limited capability, the following external components are to be integrated in a single framework:
  - ODA or Oracle Digital Assistant is a platform that allows to create and deploy digital assistants, which are AI-driven interfaces that help users accomplish a variety of tasks in natural language conversations.
  - OBDX or Oracle Banking Digital Experience as a Application Launching portal and for multi-factor authentication.
    --or--
  - Any 3rd party web application or customer self-service portal or lenders/financial services website to launch OFSLL BOT. In this case user authentication related integration needs to be handled as part of the implementation activity.
- Users need to have a capability to develop customized workflows using ODA development framework. A brief introduction is explained in 'ODA – Dialog Flow Development' section.
- User need to have a good understanding of OFSLL REST services and should be able to customize it accordingly.
- User needs to be well versed with OFSLL wrapper customization as explained in 'OFSLL Wrapper customization' section.

2.2 OFSLL Wrapper customization

Before starting OFSLL Wrapper customization, ensure that following files are present in jdeveloper/WebLogic installation with different versions which are required for compilation:

- eclipselink.jar
- jackson-annotations-2.9.5.jar
- jackson-core-2.9.5.jar

- jackson-databind-2.9.5.jar
- javax.ws.rs-api-2.0.1.jar
- jersey-client-2.28.jar
- jersey-common-2.29.jar
- servlet-api-2.5.jar
- jsoup-1.13.1.jar
- lucene-core-8.5.1.jar
- lucene-highlighter-8.5.1.jar
- lucene-memory-8.5.1.jar
- lucene-queries-8.5.1.jar
- lucene-queryparser-8.5.1.jar

Follow the below steps for OFSLL wrapper customization:

1. Import project into eclipse and modify channel.Properties to update below properties

```
ofsll.baseURL = <OFSLL REST service base URL
<http://<host>:<port>/OfsllRestWS/service/api/resources>>
ofsll.username = <OFSLL username>
ofsll.password = <OFSLL pass>
ofsll.suffix = htm
ofsll.otmHttpUrl=https://docs.oracle.com/cd/
ofsll.fIndex=/findex.htm
ofsll.index=index.htm
ofsll.video=/videos.htm
ofsll.ofsllReleaseNotes=/pdf/refdocs/ofsll_release_notes.pdf
ofsll.ofsllReleaseDoc=https://docs.oracle.com/en/industries/financial-
services/financial-lending-leasing/index.html
ofsll.splitSeperator==
ofsll.maxHitsResults=<max number of results returned>
ofsll.indexDir = <Release index directory path of server >
ofsll.releaseVersionUrl= <Release Part number>
ofsll.releaseNo=<Release No>
```

2. To add any new service modify com.ofss.ofsll.chatbot.restclient.ChatRestClient.java file.
   - Inside ChatRestClient Class add a new method with required actions
   - Add supporting JAXB files
   - Use the available supporting methods -- readInputStream, setChatBotResponse, createConnection, stringToJaxb etc.

Example for document search functionality is indicated below:

```
@Consumes(MediaType.APPLICATION_JSON)
        @Produces(MediaType.APPLICATION_JSON)
        @POST
        @Path("/lucenesearch")
         public Response lucenesearch(ODARequestDTO ibcsRequest) throws
IOException {
            final IChatbotAssembler chatbotAssembler =
ChatbotAssemblerFactory.getInstance().getChatbotAssembler("ODA");
            final HashMap < String,
            Object > map = (HashMap < String, Object > ) ibcsRequest.getProperties();
            String searchQuery = "";
            Properties prop = new Properties();
            try (InputStream propertiesFile =
this.getClass().getClassLoader().getResourceAsStream("channel.properties")) {
              prop.load(propertiesFile);
            }
            if (map != null && map.containsKey("query")) {
              searchQuery = (String) map.get("query");
            }
            ResponseDTO ibcsResponse = null;
            try {

              ChatbotResponseDTO chatbotResponse = new ChatbotResponseDTO();
              String indexDirPath =
prop.getProperty("ofsll.indexDir")+prop.getProperty("ofsll.releaseNo");
              String releaseVersionUrl = prop.getProperty("ofsll.releaseVersionUrl");
              String urlPrefix = prop.getProperty("ofsll.otmHttpUrl");
              String splitSeperator = prop.getProperty("ofsll.splitSeperator");
              String releaseNo = prop.getProperty("ofsll.releaseNo");
              String urlPrefixPath = urlPrefix + releaseVersionUrl;
              String findexPath = prop.getProperty("ofsll.fIndex");
              String indexPath = prop.getProperty("ofsll.index");
              String videoPath = prop.getProperty("ofsll.video");
              String ofsllReleaseNotesPath = prop.getProperty("ofsll.ofsllReleaseNotes");
              String ofsllReleaseDocPath = prop.getProperty("ofsll.ofsllReleaseDoc");
              Integer maxHitsResults =
Integer.parseInt(prop.getProperty("ofsll.maxHitsResults"));
              File fileIndexDirPath = new File(indexDirPath);
              LuceneSearchHighlighter luceneSearchHighlighter = new
LuceneSearchHighlighter();
              List<String> fileList = new ArrayList <> ();

              if ((searchQuery.toLowerCase().trim().contains("#ofsll release document")) ||
(searchQuery.toLowerCase().trim().contains("navigate to index page")) ||
(searchQuery.toLowerCase().trim().contains("#video gallery")) ||
(searchQuery.toLowerCase().trim().contains("#ofsll release notes")) ||
(searchQuery.toLowerCase().trim().contains("#index page"))) {
```

```
                if ((searchQuery.toLowerCase().trim().contains("#ofsll release document")))
{
                    releaseNo="All Release Version";
                  fileList.add(searchQuery + splitSeperator + ofsllReleaseDocPath +
splitSeperator+searchQuery+ splitSeperator+releaseNo);
                }
                if ((searchQuery.toLowerCase().trim().contains("navigate to index page"))) {
                  fileList.add(searchQuery + splitSeperator + urlPrefixPath + findexPath +
splitSeperator+searchQuery+ splitSeperator+releaseNo);
                }
                if ((searchQuery.toLowerCase().trim().contains("#index page"))) {
                  searchQuery = indexPath;
                  fileList = luceneSearchHighlighter.searchsinglepage(fileIndexDirPath,
searchQuery, maxHitsResults, splitSeperator);
                }
                if ((searchQuery.toLowerCase().trim().contains("#video gallery"))) {
                  fileList.add(searchQuery + splitSeperator + urlPrefixPath + videoPath +
splitSeperator+searchQuery+ splitSeperator+releaseNo);
                }
                if ((searchQuery.toLowerCase().trim().contains("#ofsll release notes"))) {
                  fileList.add(searchQuery + splitSeperator + urlPrefixPath +
ofsllReleaseNotesPath + splitSeperator+searchQuery+ splitSeperator+releaseNo);
                }
              } else {
                searchQuery = searchQuery.replaceAll("#", "");
                fileList = luceneSearchHighlighter.search(fileIndexDirPath, searchQuery,
maxHitsResults, splitSeperator);
              }
              String serviceOutputForChatBot = "";
              for (String obj: fileList) {
                if (serviceOutputForChatBot == "") {
                  serviceOutputForChatBot = obj.replace("\\", "/");
                } else {
                  serviceOutputForChatBot = serviceOutputForChatBot + "\n---\n" +
obj.replace("\\", "/");
                }
              }
              if (fileList.isEmpty()) {
                String errorOutputForChatBot = "Search is not found for : " + searchQuery;
                setChatBotResponse("failure", errorOutputForChatBot, chatbotResponse,
"response", "request");
              } else {
                List < String > srhchoices = new ArrayList < >();
                for (String obj: fileList) {
                  srhchoices.add(obj.replace("\\", "/"));
                }
                setChatBotResponse("success", srhchoices, chatbotResponse, "acc_srh",
"acc_srh");
```

```
            }
            ibcsResponse =
chatbotAssembler.fromChatbotResponseDTO((RequestDTO) ibcsRequest,
chatbotResponse);
          } catch(Exception e) {
            LOGGER.log(Level.SEVERE, e.getMessage());
          }
          return Response.status(Response.Status.OK).entity((Object)
this.buildResponse((Object) ibcsResponse)).build();
        }
```

Example for lastbillingdetails Service -- This uses Account Details Service

```
@Consumes(MediaType.APPLICATION_JSON)@Produces(MediaType.APPLICATION_
JSON)@POST@Path("/lastbillingdetails")
public Response lastbillingDetails(ODARequestDTO ibcsRequest) {
 final IChatbotAssembler chatbotAssembler =
ChatbotAssemblerFactory.getInstance().getChatbotAssembler("ODA");
 final HashMap < String,
 Object > map = (HashMap < String, Object > ) ibcsRequest.getProperties();
 String accountNumber = "";
 if (map != null && map.containsKey("acc_nbr")) {
  accountNumber = (String) map.get("acc_nbr");
 }
 ResponseDTO ibcsResponse = null;
 try {
  ChatbotResponseDTO chatbotResponse = new ChatbotResponseDTO();
  String requestURL = "/servicing/account/" + accountNumber +
"?displayassociateaccounts=N";
  HttpURLConnection conn = createConnection("GET", requestURL, "");

  if (conn.getResponseCode() != 200 && conn.getResponseCode() != 201 &&
conn.getResponseCode() != 202) {
    String errorOutput = readInputStream(conn, "error");
    AccountDetailResponseType accountsDetails =
stringToJaxb(AccountDetailResponseType.class, errorOutput);
    String errorOutputForChatBot = "\nBilling Details: \n " +
accountsDetails.getResult().getStatus().toString() + "\n" +
accountsDetails.getResult().getStatusDetails();
    setChatBotResponse("failure", errorOutputForChatBot, chatbotResponse,
"response", "request");
  } else {
    String serviceOutput = readInputStream(conn, "input");
    AccountDetailResponseType accountsDetails =
stringToJaxb(AccountDetailResponseType.class, serviceOutput);
```

```
String serviceOutputForChatBot = "\nBilling Details: " + "\n Generation Date: " +
dateFormater(accountsDetails.getAccountDetailSummary().get(0).getStatementDetails().
get(0).getGenerationDate()) + "\n Closing Date: " +
dateFormater(accountsDetails.getAccountDetailSummary().get(0).getStatementDetails().
get(0).getClosingDate()) + "\n Due Date: " +
dateFormater(accountsDetails.getAccountDetailSummary().get(0).getStatementDetails().
get(0).getDueDate()) + "\n Current Due Amount: " +
accountsDetails.getAccountDetailSummary().get(0).getStatementDetails().get(0).getCurr
entDueAmount();
    setChatBotResponse("success", serviceOutputForChatBot, chatbotResponse,
"response", "request");
   }
   ibcsResponse = chatbotAssembler.fromChatbotResponseDTO((RequestDTO)
ibcsRequest, chatbotResponse);
  } catch(Exception e) {
   LOGGER.log(Level.SEVERE, "Error: ", e);
  }
  return Response.status(Response.Status.OK).entity((Object)
this.buildResponse((Object) ibcsResponse)).build();
}
```

3. Export project as war file.
4. Deploy <WL_Home>/wlserver/common/deployable-libraries/jax-rs-2.0.war as Library on weblogic.
5. Deploy generated WAR in step 3 onto weblogic server.
6. Note down base service URL that is required while publishing in ODA.
   Example : http://<host>:<port>/ofsll/v1/fulfillment

2.3 ODA – Dialog Flow Development

Each menu option displayed in BOT are configured as an ‘Intent’ which is configured to perform a specific function or otherwise call a REST service in OFSLL.

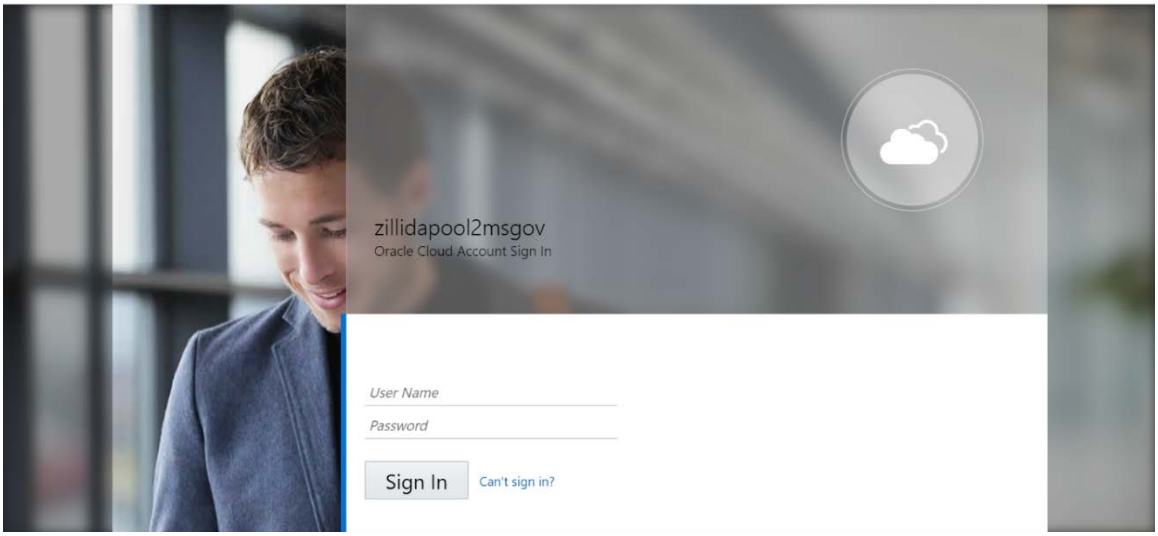
In-order to achieve a sequence of menu options, dialog flow development is required to be performed in ODA Oracle Digital Assistant. Following is a quick overview of steps involved:

- Login
- Creating Skill / Digital Assistant
- Defining Entity
- Adding Intents
- Updating Bot flow using Yaml
- Adding OFSLL REST service
- Configuring Channel for Publishing
- Publishing

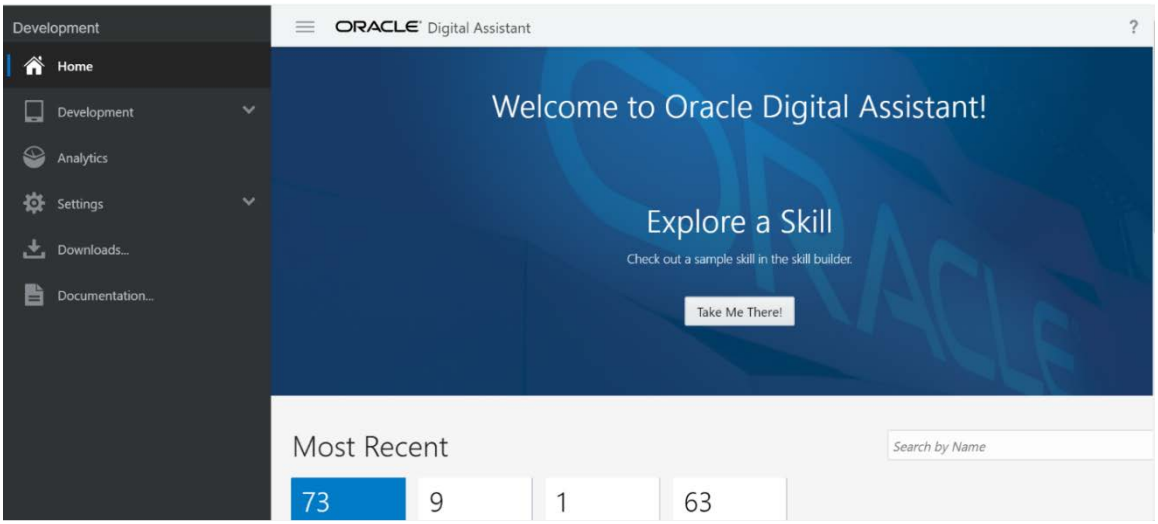
It is recommended to refer to ODA documentation for detailed information - https://docs.oracle.com/en/cloud/paas/digital-assistant/index.html

In the ODA - dialog flow development, you can either create new / import the given sample. The sequence of flow in creating a sample BOT in ODA is indicated below with illustration:

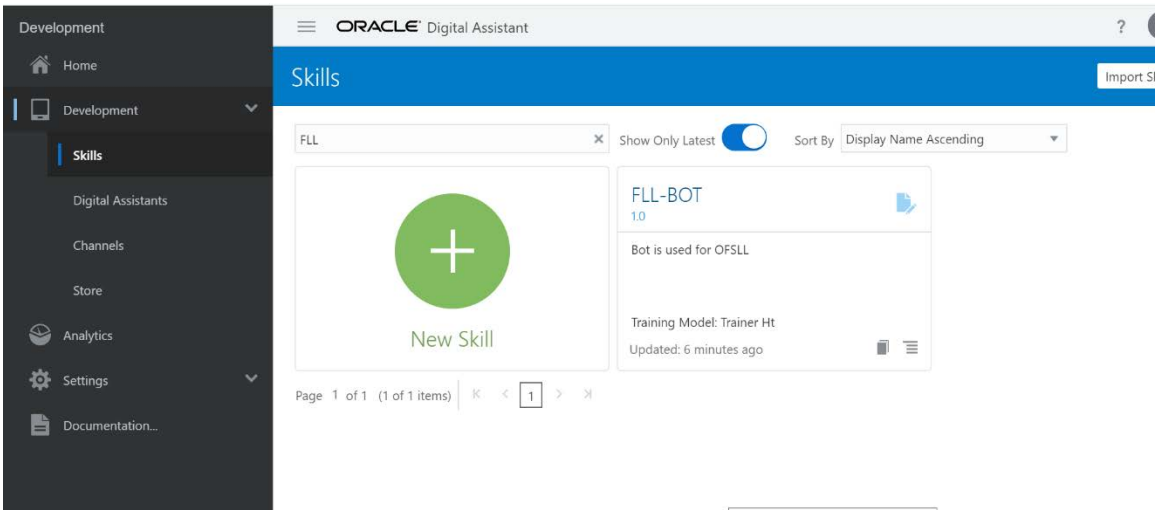
1. Login to ODA UI



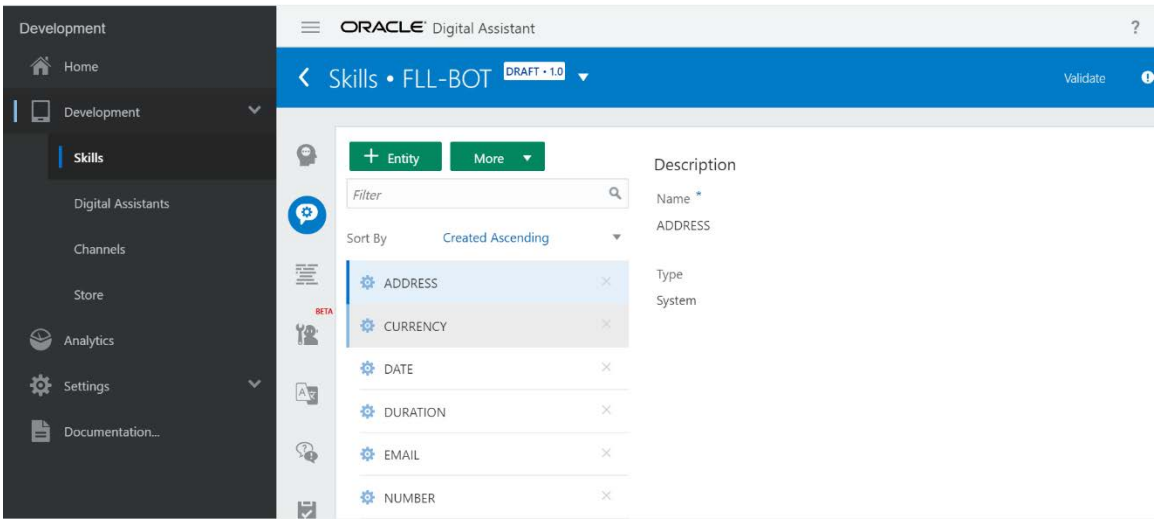
2. Go to Home



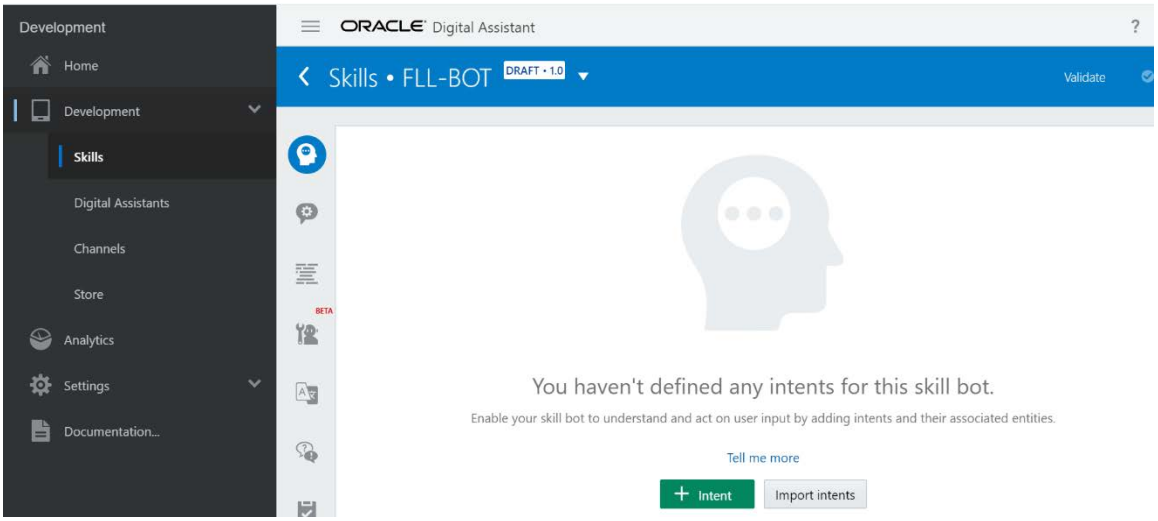
3. Create Skill/Digital Assistant.



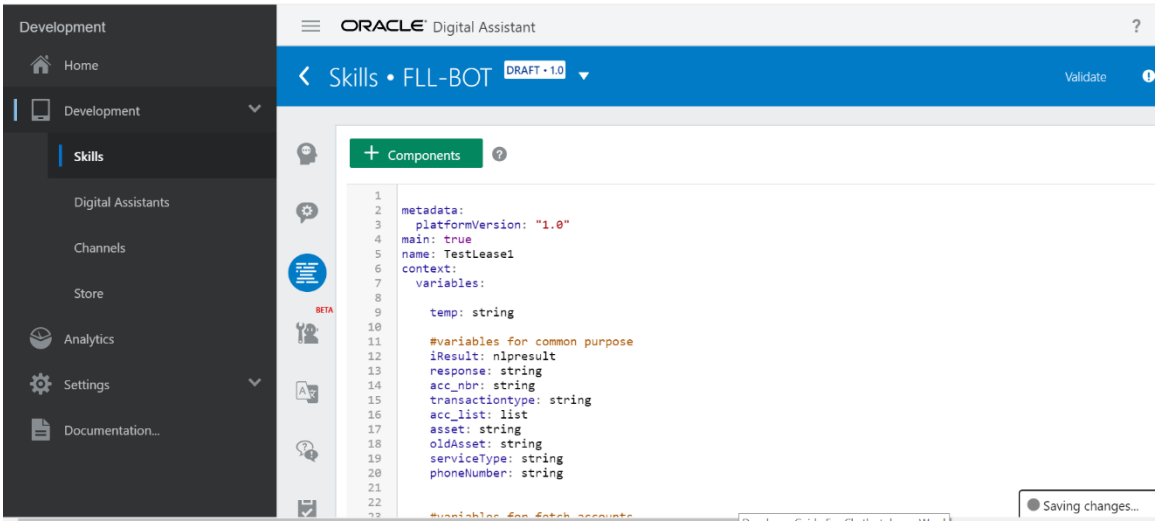
4. Add Entities



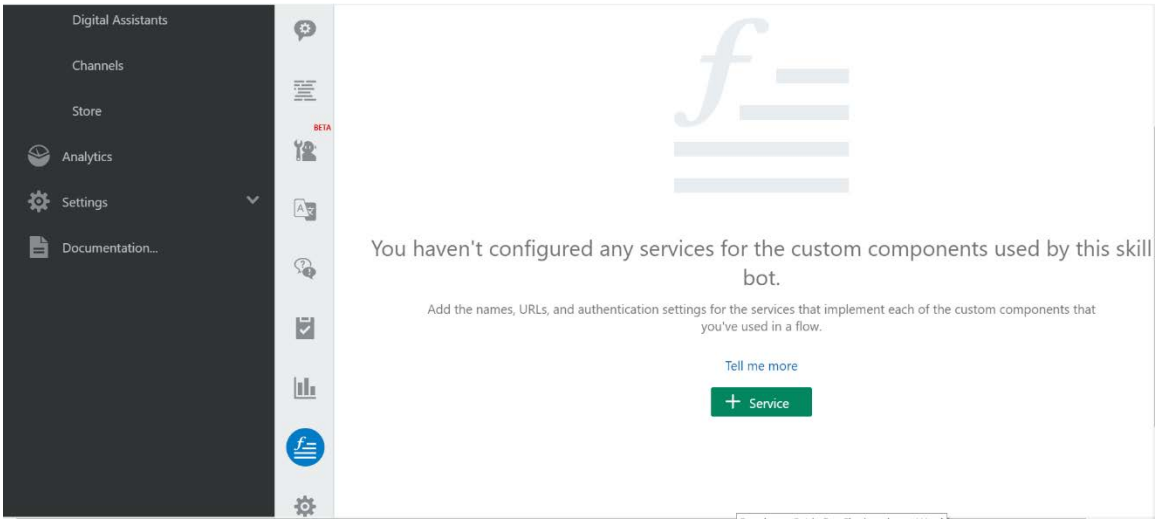
5. Add Intents. This involves defining Activity, Available option, Next level, Breakpoint, intermediate steps.

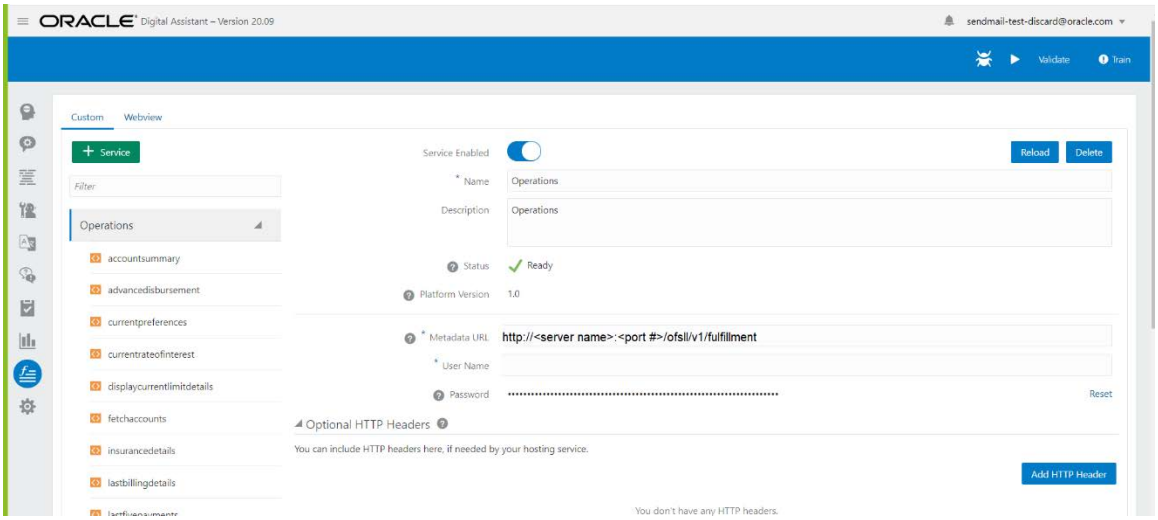


6. Add Bot flow using Yaml

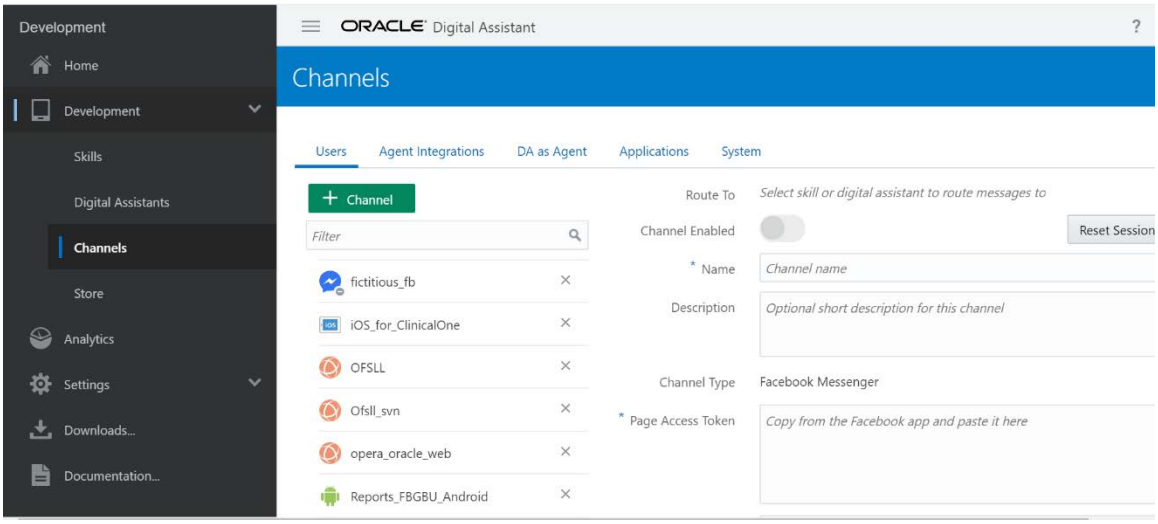


7. Add OFSLL REST Service





8. Add Channel. This indicates where it has to be published and in this sample application, only web channel is supported.
9. Enter the published URL as generated in step 2.6



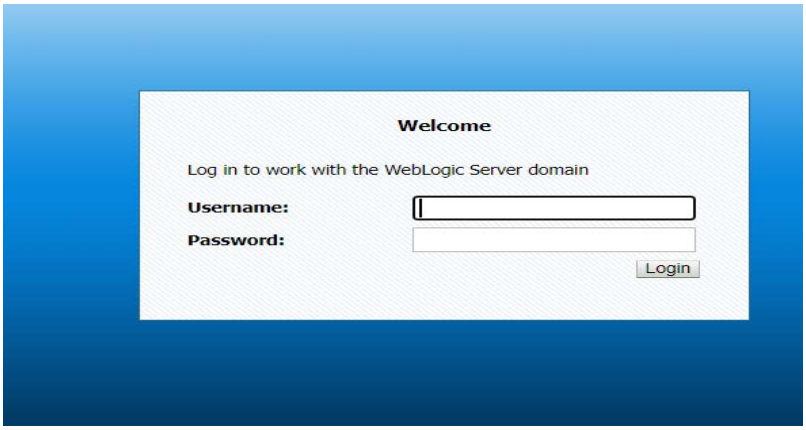
10. After completion of Skill, publish. On publishing, the draft is converted to final non-editable version and only final published version is accessible in bot.

Note: The 'ofsll-transaction-bot' is the sample ODA FLL application designed for the demo purpose. The same can be imported in any ODA environment tested, modified for new features.

2.4 Deploying war file on WebLogic Server

1. Login to Web Logic application server enterprise manager (e.g.:http://hostname:port/em). For example, http://host01.example.com:8001/console

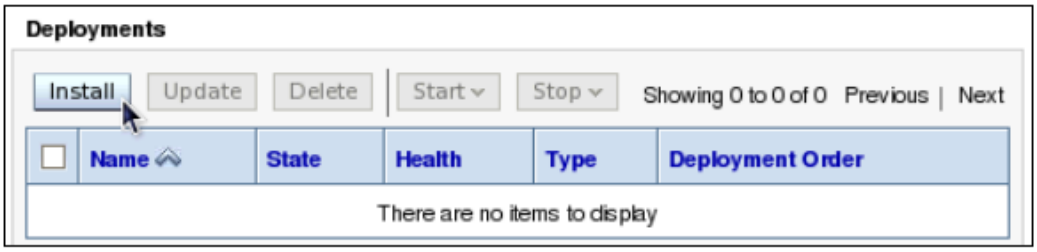
   Note: Use the host name and port of the administration server of your domain.



2. Enter valid login credentials.
3. Deploying an application is a change to the domain's configuration, so it must first be locked. In the Change Center. Click 'Lock & Edit'.

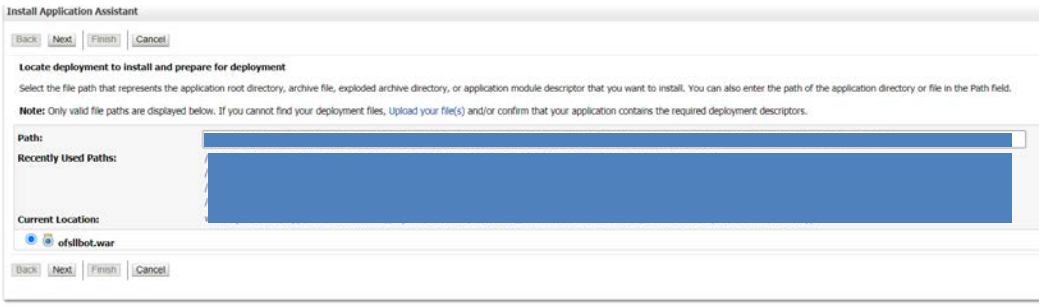
4. Under Domain Structure, click ‘Deployments’.

5. On the right, under Deployments, click ‘Install’.

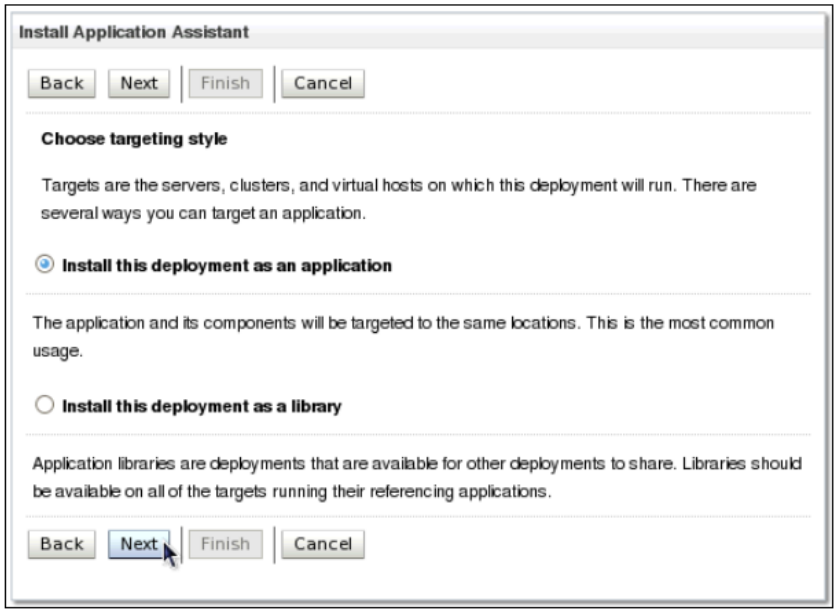


6. Find the Current Location field. Use the links to browse to the location in which you placed the downloaded ofsllbot.war. War file.

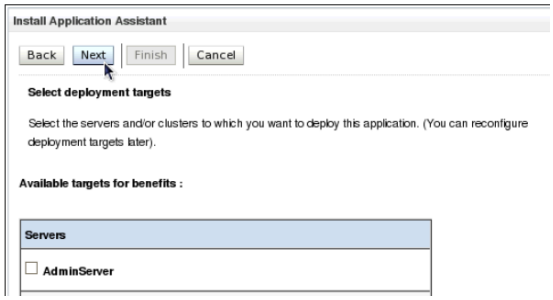
7. On locating ofsllbot.war, click the radio button next to it. Using the links and the radio button, the console auto populates the Path fields. Alternatively, you can type in the path and file name in the Path field yourself. Click ‘Next’.



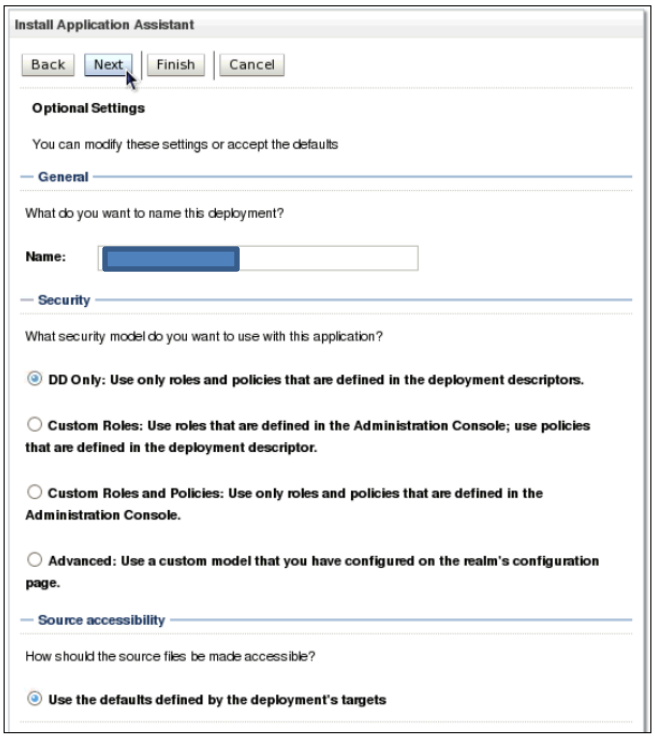
8. Ensure that 'Install this deployment as an application' option is selected. Click 'Next'.



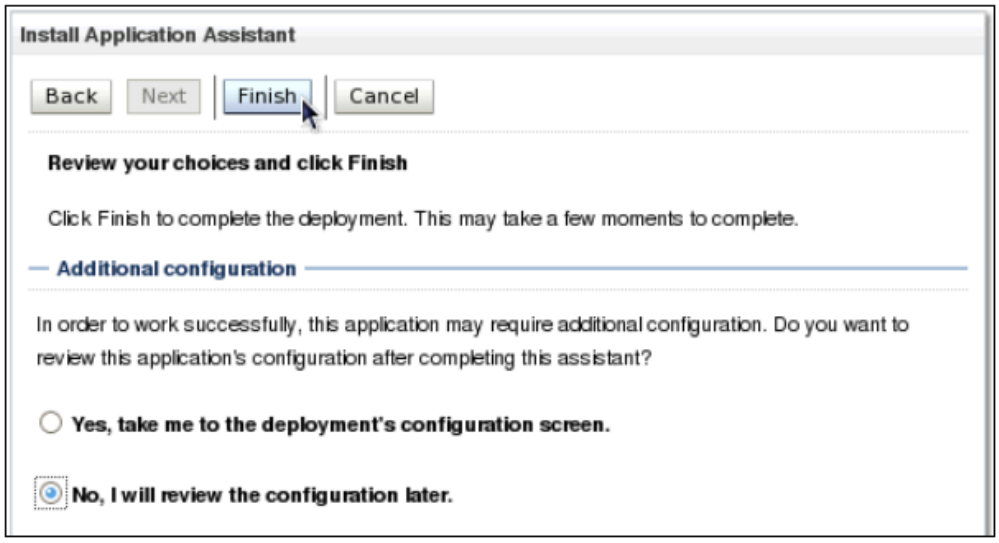
9. In the below window, click 'Next'.



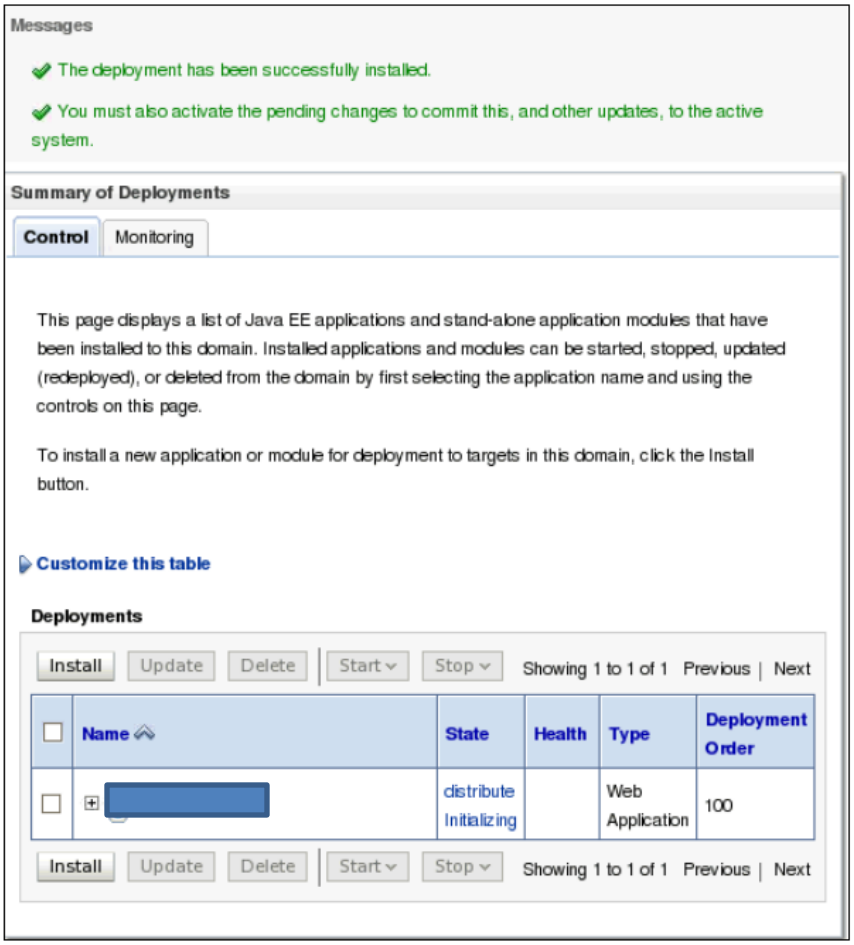
10. Retain the default values and click 'Next'.



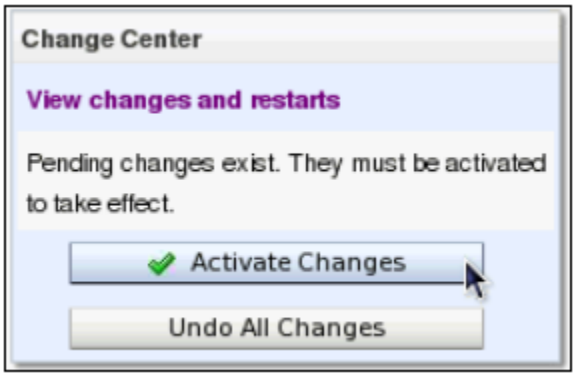
11. In the below window, select the option 'No, I will review the configuration later' and click 'Finish'.



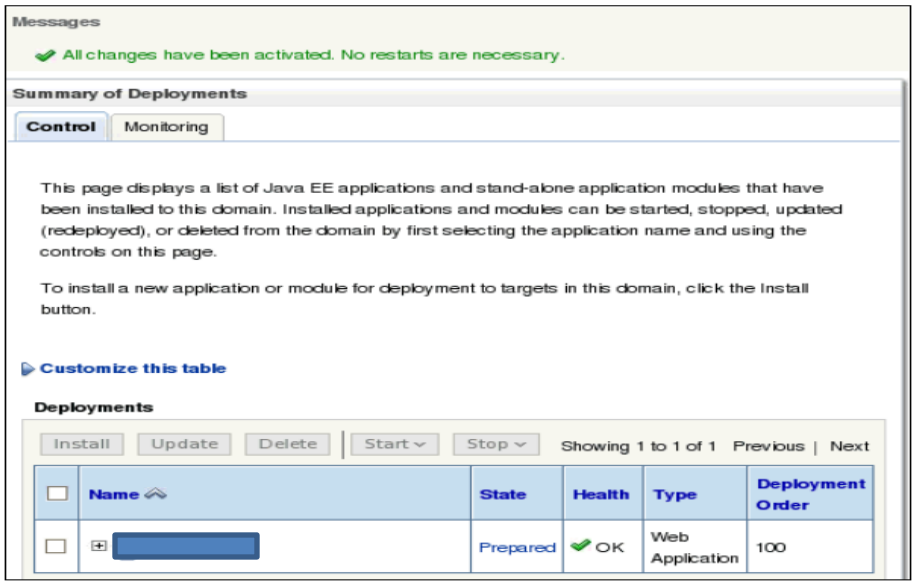
Once done view the messages indicating that the deployment was installed, but changes must be activated. In addition, notice the benefits application listed in the Deployments table.



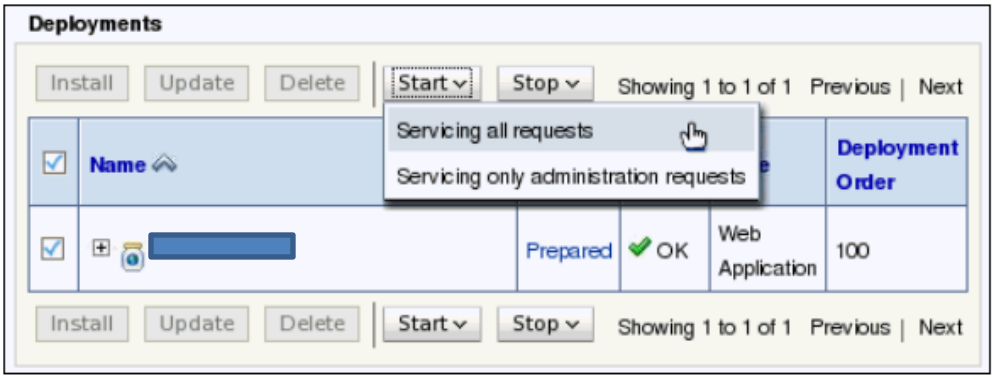
12. In the Change Center, click the Activate Changes button.



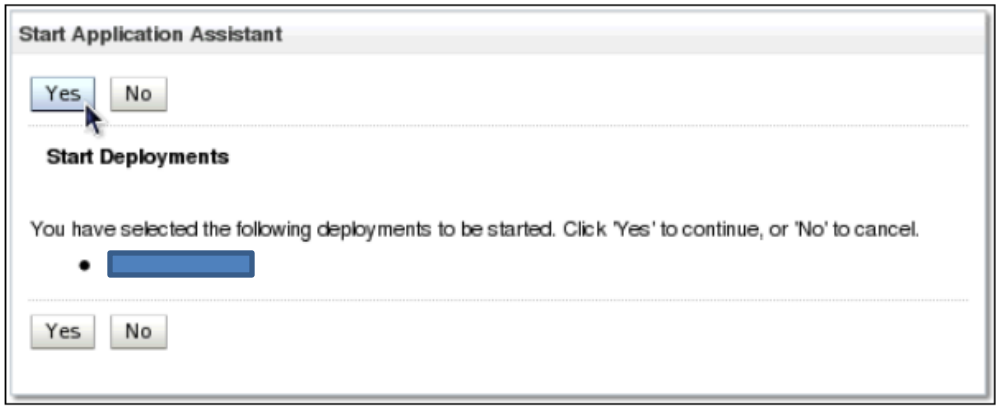
Notice the message indicating that the changes have been activated. In addition, notice the benefits application listed in the Deployments table is now in the "Prepared" state.



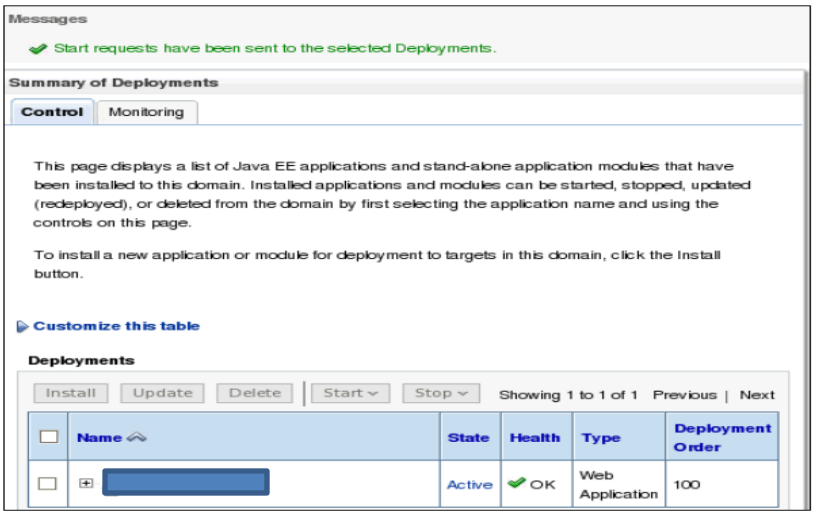
13. Select the checkbox against the left of the benefits application in the Deployments table. In the Start drop-down list, select ‘Servicing all requests’ option.



14. Click ‘Yes’ to continue.



15. A message is displayed indicating a start request was sent. Subsequently Notice that the state of application is ‘Active’ indicating that the application is accessible.



2.5 Web application UI for Accessing BOT

Web Application is User Interface where you can access the BOT functionality. The same can be integrated with OFSLL UI or any other front-end application such as customer support portal or financial institution website.

To configure WebApp, do one of the following:

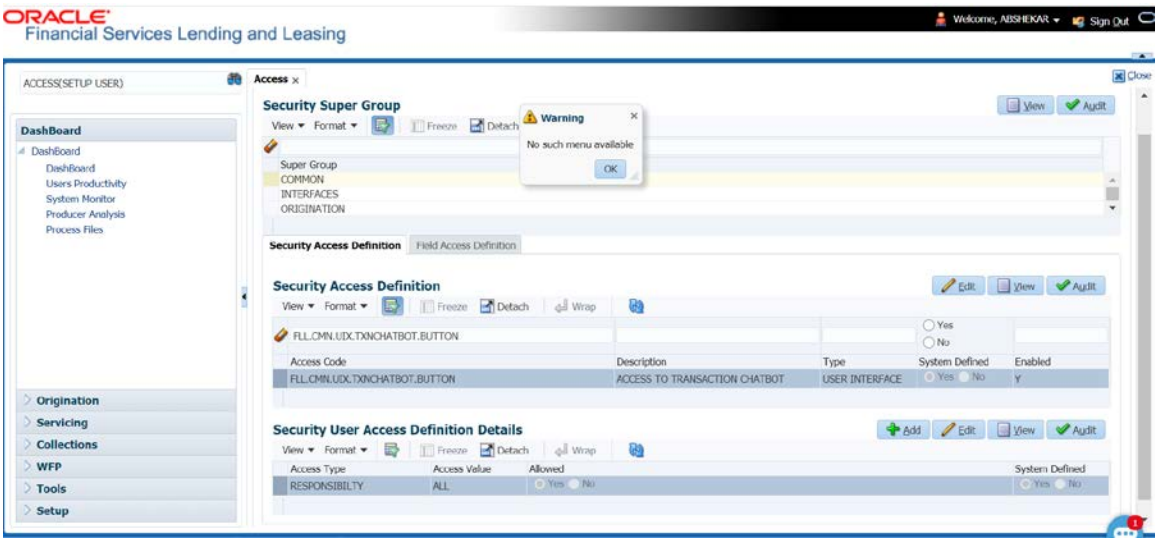
- In case you wish to launch BOT as separate application, Modify **index.html** in WebApp(or WebApp.war) and update the following 2 fields with required details:
  - URI: '<ODA host>',
  - channelId: 'published bot channel ID'
- In case you wish to integrate BOT in an existing front-end application, use the provided **index.html** with the modified value and **web-sdk.js**

The BOT needs to be published on the login page and the only way it come be done is by adding the above properties in the Weblogic

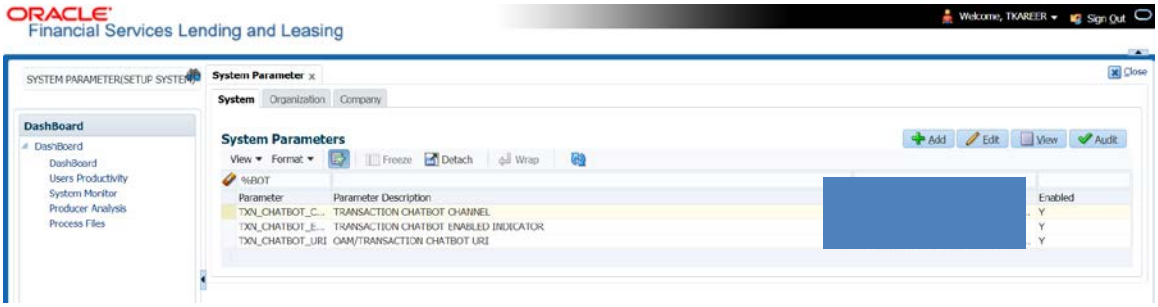
For additional information, contact Oracle Financial Services Lending and Leasing Product Engineering team.

2.6 App configuration for enabling chatbot

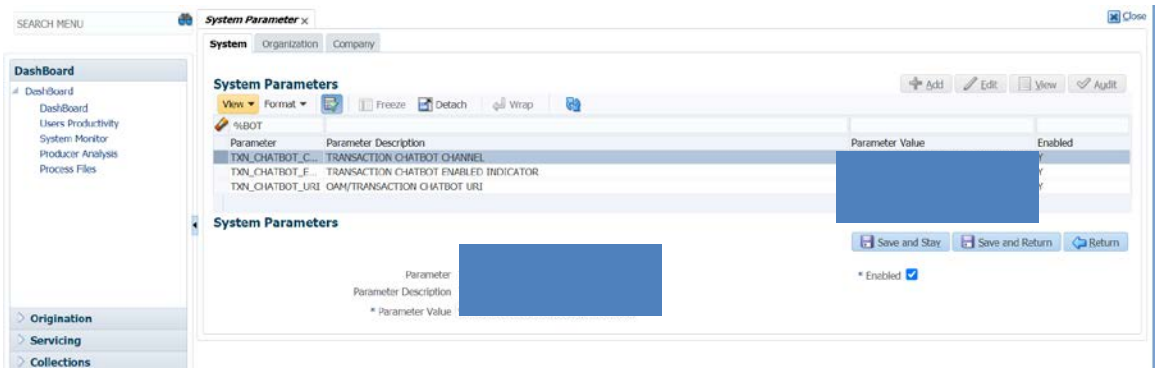
1. Enabling BOT and adding parameters:
   - Channel ID
   - URI
   - Enabled – Yes / No
2. Enable the FLS Access Key - FLL.CMN.UIX.TXNCHATBOT.BUTTON as indicated in the Access Setup screen.



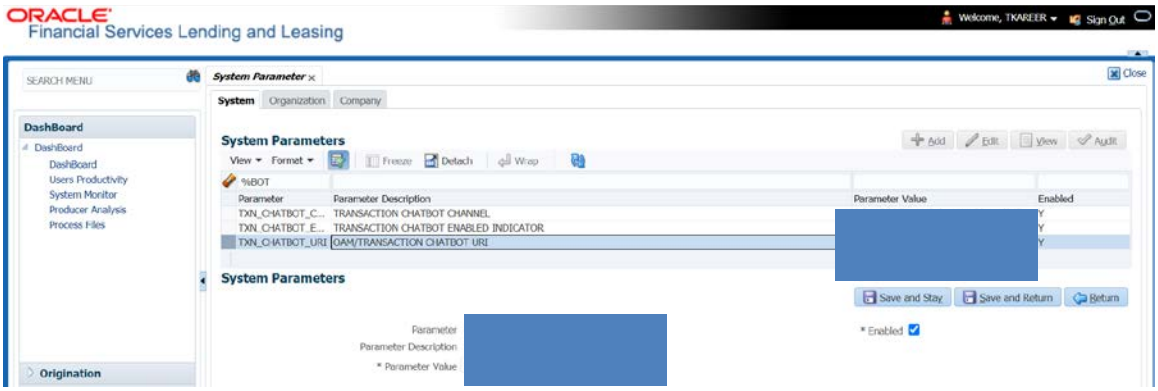
3. Search for “System parameter” in the box below



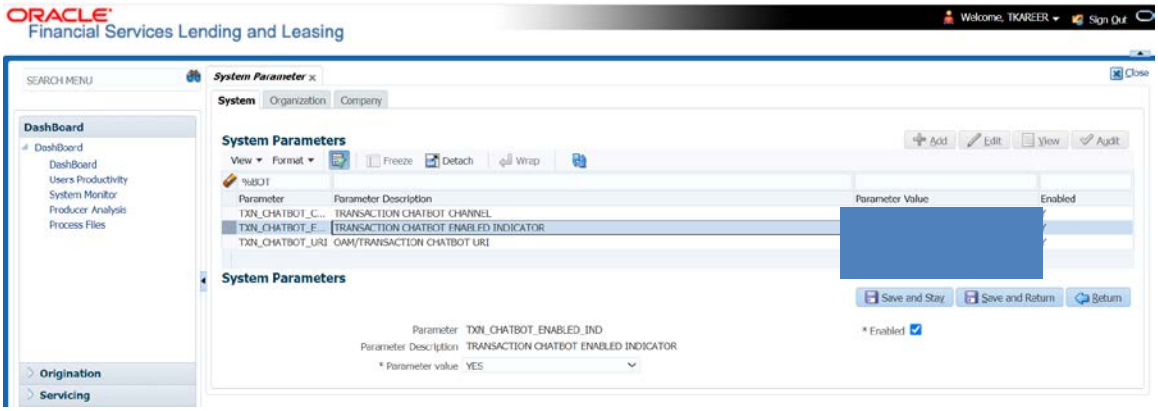
4. Enter the Channel id, click save and return.



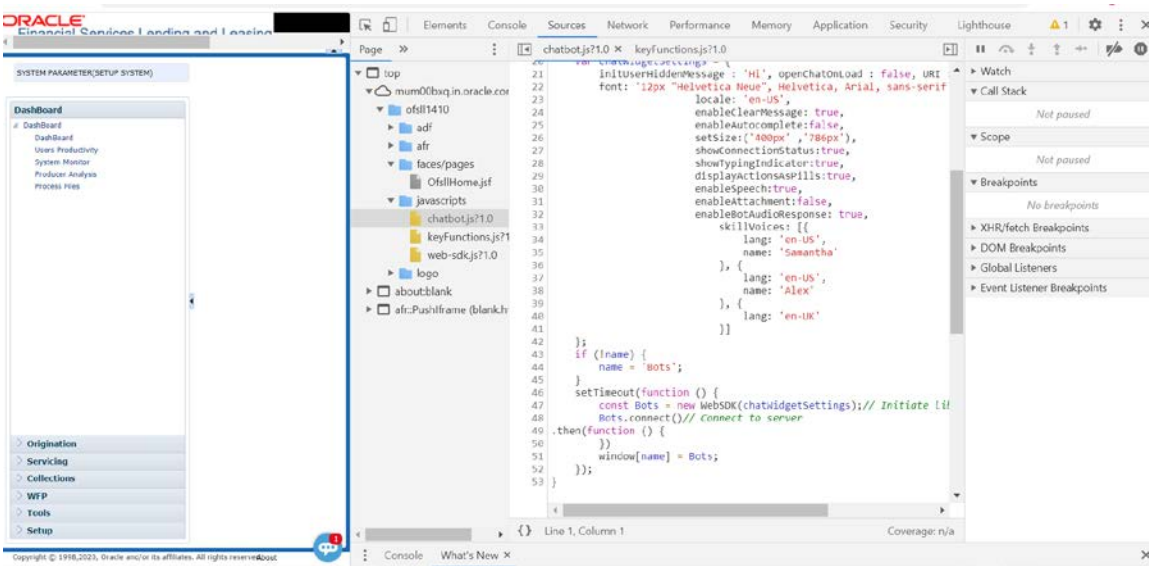
5. Enter the URI , click save and return



6. Enable Transaction bot



The below code needs to be implemented in the chatbot.js file as shown below:



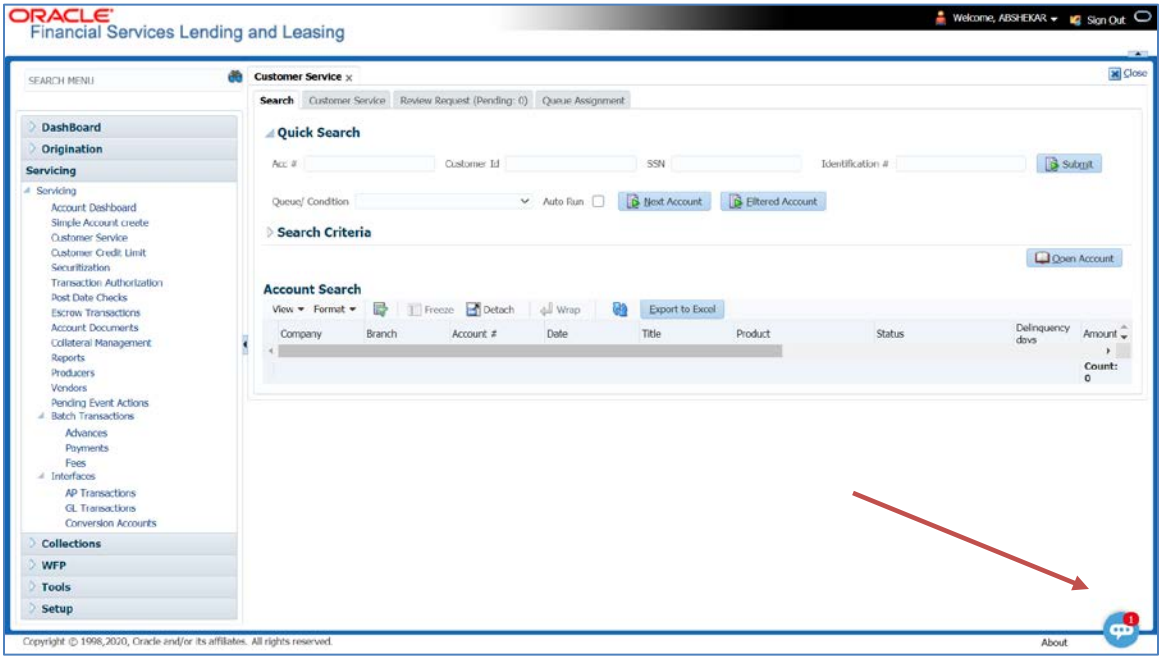
Ensure that no changes are done to the following js code:

```
function onLoginPageLoad(event) {
    var source = event.getSource();
    AdfCustomEvent.queue(source, "LoginChatbotEvent",
    {
        'someArg' : 'true'
    },
    true);
}

function onHomePageLoad(evt) {
    var eventSource = evt.getSource();
    AdfCustomEvent.queue(eventSource, "HomeChatbotEvent",
    {
        'someArg' : 'true'
    },
    true);
}

function initSdk(name, uri, channel) {
    var chatWidgetSettings = {
        initUserHiddenMessage : 'Hi', openChatOnLoad : false, URI : uri, channelId :
channel,
        font: '12px "Helvetica Neue", Helvetica, Arial, sans-serif',
                    locale: 'en-US',
                    enableClearMessage: true,
                    enableAutocomplete:false,
                    setSize:('400px' ,'786px'),
                    showConnectionStatus:true,
                    showTypingIndicator:true,
                    displayActionsAsPills:true,
                    enableSpeech:true,
                    enableAttachment:false,
                    enableBotAudioResponse: true,
                       skillVoices: [{
                          lang: 'en-US',
                          name: 'Samantha'
                       }, {
                          lang: 'en-US',
                          name: 'Alex'
                       }, {
                          lang: 'en-UK'
                       }]
    };
    if (!name) {
        name = 'Bots';
    }
```

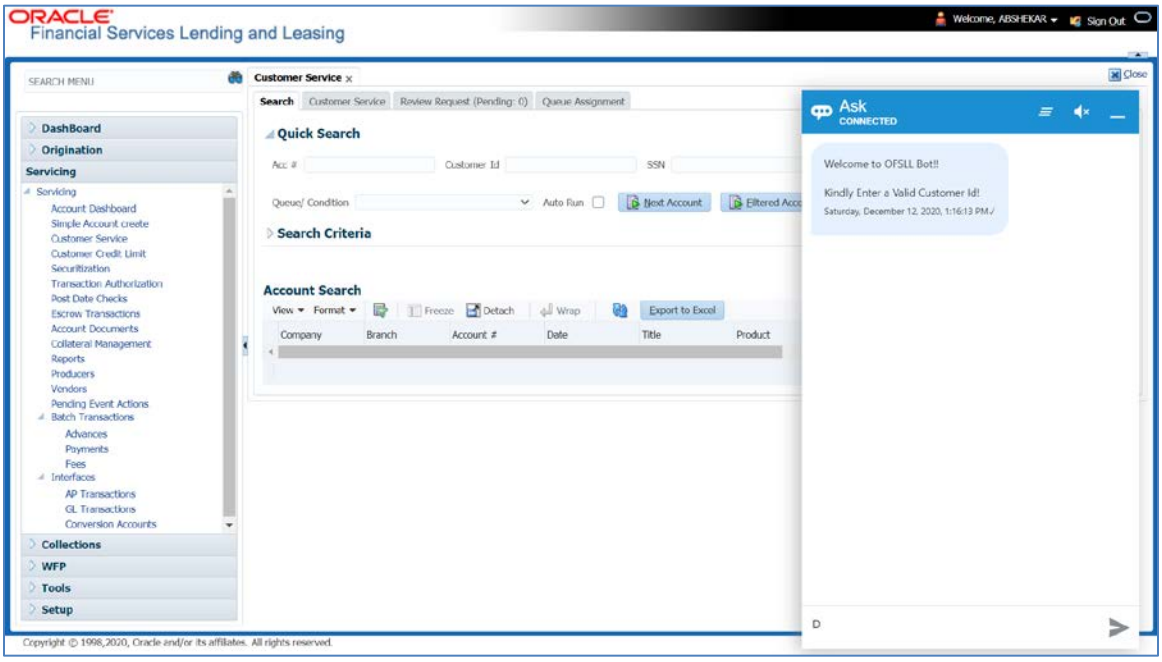
```
    setTimeout(function () {
        const Bots = new WebSDK(chatWidgetSettings);// Initiate library with configuration
        Bots.connect()// Connect to server
.then(function () {
        })
        window[name] = Bots;
    });
}
```

7. Web-sdk.js needs to be added from the << OFSLL Installed Directory >>/ /web_interface/ofsllbot/WebApp/scripts.

The BOT after launch and login form Web Application is as shown below:



On clicking bot icon, the interface is as displayed:



ORACLE
Financial Services

Transaction CHATBOT Integration with OFSLL
Oracle Financial Services Lending and Leasing Release 14.10.0.0.0
December 2020

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
https://www.oracle.com/industries/financial-services/index.html