

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
1.1	INTRODUCTION.....	1-1
1.1.1	<i>Transaction Bot Overview</i>	1-1
1.1.2	<i>Purpose</i>	1-1
1.1.3	<i>Audience</i>	1-2
1.1.4	<i>Accessibility</i>	1-2
1.1.5	<i>Access</i>	1-2
1.1.6	<i>Pre-requisites</i>	1-2
1.2	ARCHITECTURE.....	1-3
1.3	FEATURES OF BOT.....	1-3
1.3.1	<i>Support of Text and Voice Based inputs</i>	1-4
1.3.2	<i>Release Specific Indexing</i>	1-4
1.4	BOT CONFIGURATION.....	1-4
1.4.1	<i>Sample Workflow</i>	1-6
1.4.2	<i>BOT UI Elements</i>	1-9
1.4.3	<i>BOT Usability Workflow</i>	1-10
1.5	THIRD PARTY LICENSES.....	1-14
2.	DEVELOPER GUIDE FOR BOT CUSTOMIZATION.....	2-1
2.1	PRE-REQUISITES.....	2-1
2.2	OFSLL WRAPPER CUSTOMIZATION.....	2-1
2.3	ODA – DIALOG FLOW DEVELOPMENT.....	2-6
2.4	DEPLOYING WAR FILE ON WEBLOGIC SERVER.....	2-10
2.5	WEB APPLICATION UI FOR ACCESSING BOT.....	2-16
2.6	APP CONFIGURATION FOR ENABLING CHATBOT.....	2-16

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/ofsl transactionbot overview and developer guide.pdf

Following topics are discussed in this section:

- [Transaction Bot Overview](#)
- [Architecture](#)
- [Features of BOT](#)
- [Bot Configuration](#)
- [Third Party Licenses](#)
- [Developer Guide for BOT Customization](#)

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.

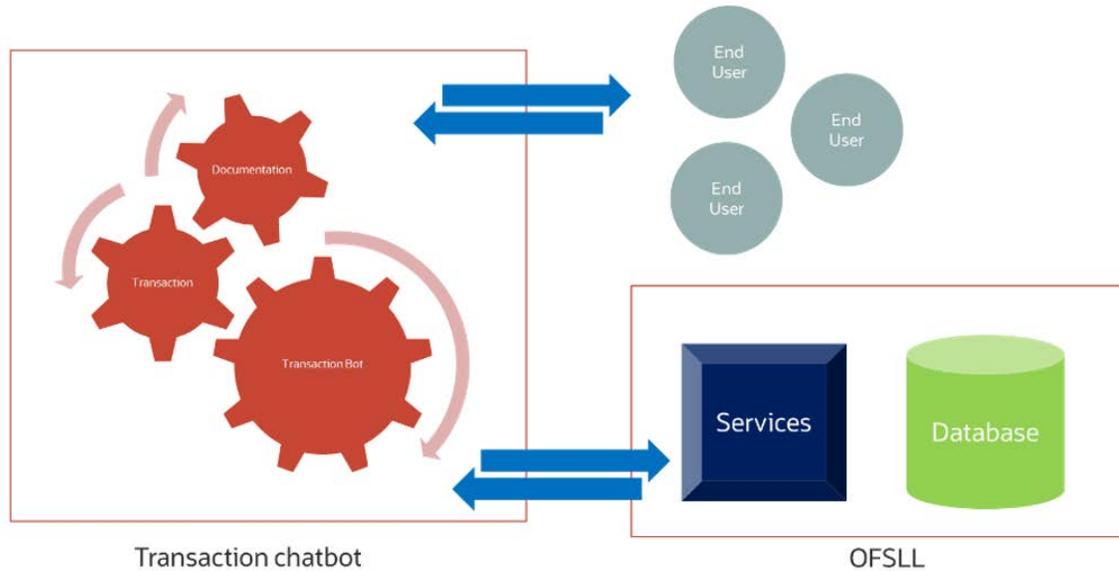
1.1.6 **Pre-requisites**

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 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/ofslbot/src.

This section contains the following:

- [Sample Workflow](#)
- [BOT UI Elements](#)
- [BOT Usability Workflow](#)

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.

SI. 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	

SI. No	Parameter Name	Fields	Description	Update required (Y/N)	Sample
5	googleAPIKey=	String	Captures the Google API key	N	
6	imageUrl=	Path	Captures the Image URL	N	
7	defaultHomeEntity=	String	Captures the home entity	N	
8	stockCode=	String	Captures the Stock Code	N	
9	moneyTransferPay=	String	Captures the Money Transfer Pay	N	
10	defaultBaseContext=	String	Captures the default base content	N	
11	sessionExpiresInMinutes = 15	Integer	Captures the Session timeout value	N	
12	ofssl.suffix = htm	String	Suffix of the files	N	Keep as .htm
13	ofssl.otmHttpUrl=https://docs.oracle.com/cd/	String	Captures the suffix for OTM Url	N	Keep as https://docs.oracle.com/cd/
14	ofssl.findex=/findex.htm	String	Captures the Findex path	N	Keep as /findex.htm
15	ofssl.index=index.htm	String	Captures the index.htm	N	Keep as index.htm
16	ofssl.video=/videos.htm	String	Captures the video file path	N	Keep as /video.htm
17	ofssl.ofsllReleaseNotes=/pdf/refdocs/ofssl_release_notes.pdf	String	Captures the OFSLL release notes suffix	N	Do not change

SI. No	Parameter Name	Fields	Description	Update required (Y/N)	Sample
18	ofssl.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	ofssl.splitSeparator==	String	Captures the Split separator	N	Do not change
20	ofssl.maxHitsResults=100	String	Captures the Max no of its results of the document query	Y (optional)	Change depending upon search results
21	ofssl.baseURL =	String	Captures the Service API URL	Y	Application URL
22	ofssl.username =	String	Captures the username of weblogic server	Y	Weblogic username
23	ofssl.pasd =	String	Captures the Password of weblogic server	Y	Weblogic password
24	ofssl.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	ofssl.releaseVersionUrl=	Path	Captures the Part Number	Y	Refer Release Specific Indexing table .
26	ofssl.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.



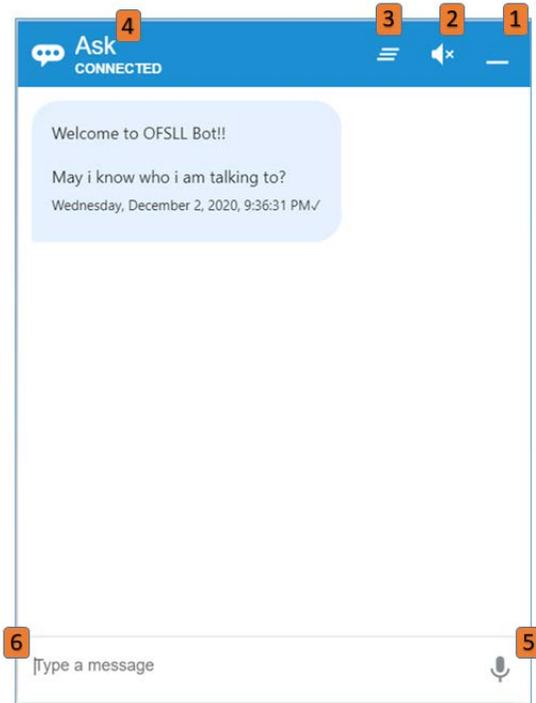
To Start with, enter your name and confirm if you want to continue using the bot. Based on your intent the bot starts building the answers.

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

SI.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.

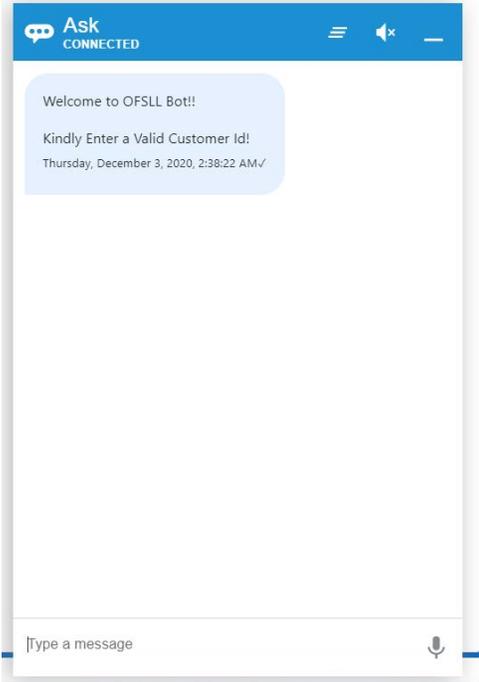
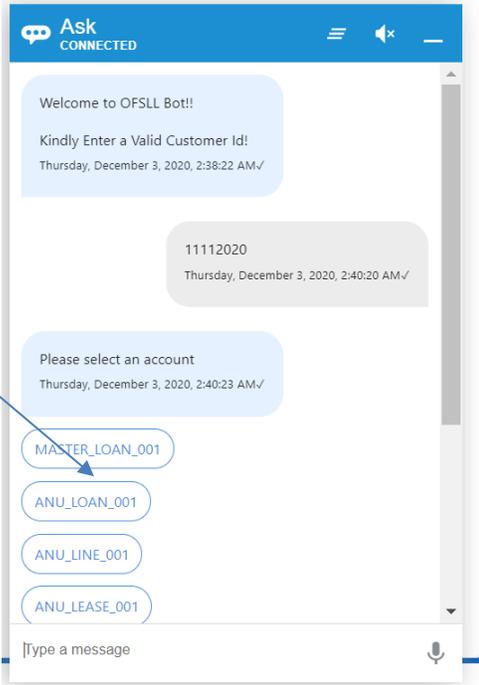
1.4.2 BOT UI Elements

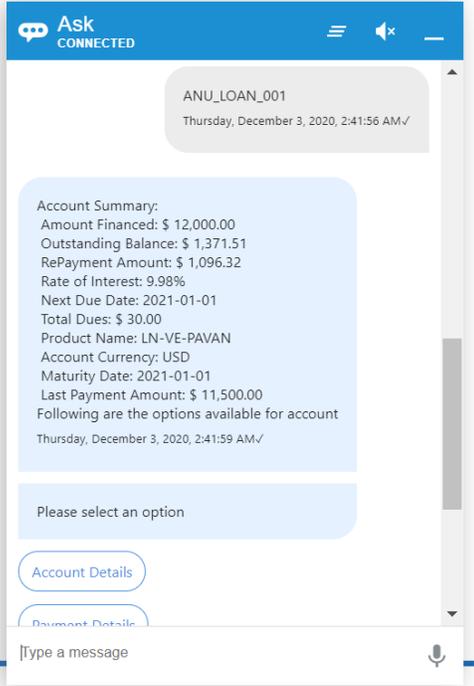
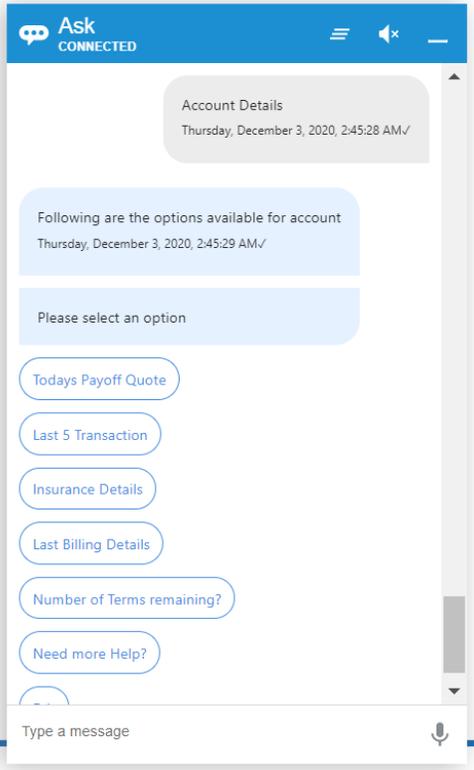


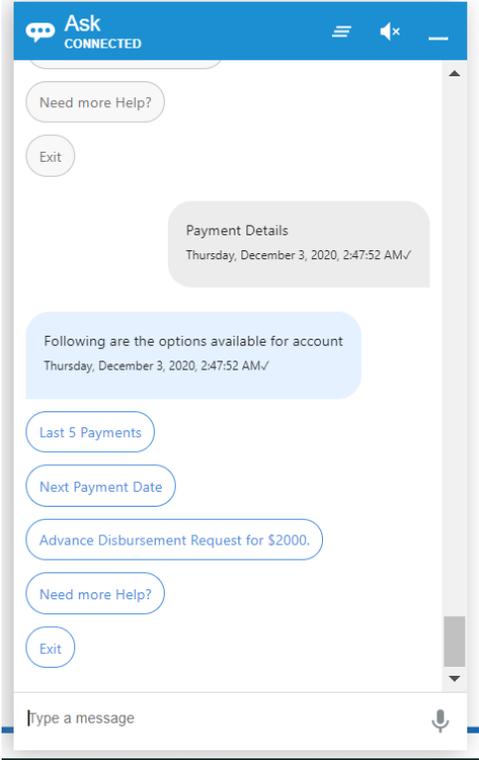
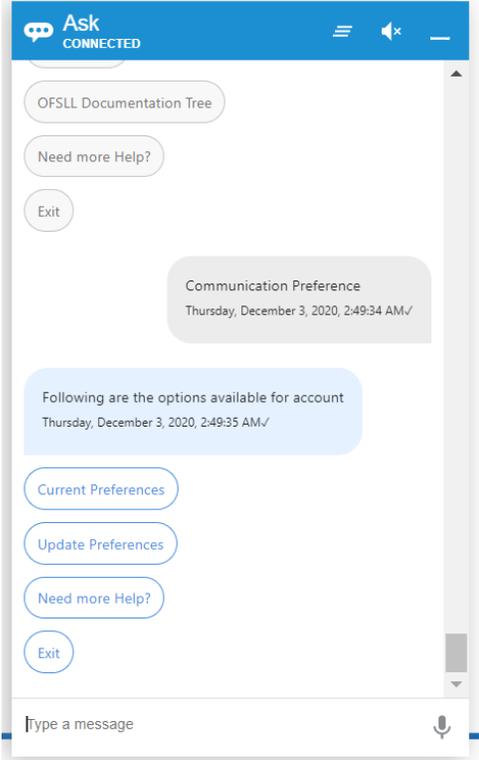
Sl.No	Option	View / Action
1	Minimize	Minimize BOT window
2	Speaker output	Enable BOT in speaker mode
3	Clear chat	Clear all messages in the BOT
4	Network Status	Connected: BOT connected to server Connecting: BOT awaiting network connection
5	Mic Input	Enable Mic for voice based input
6	Text Input	Enter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

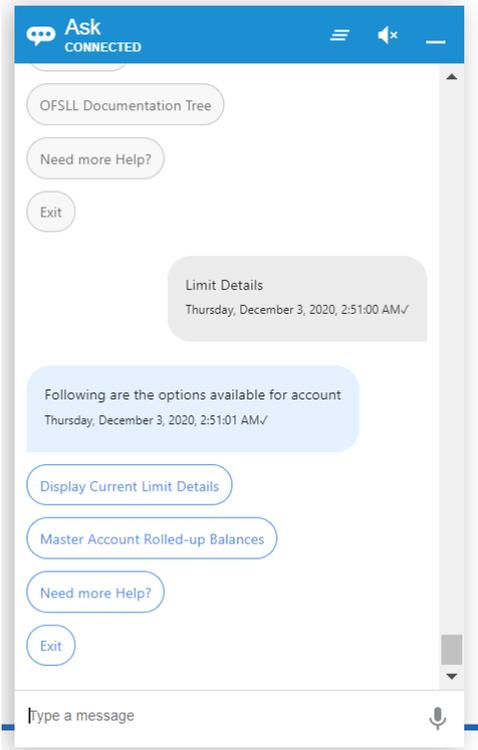
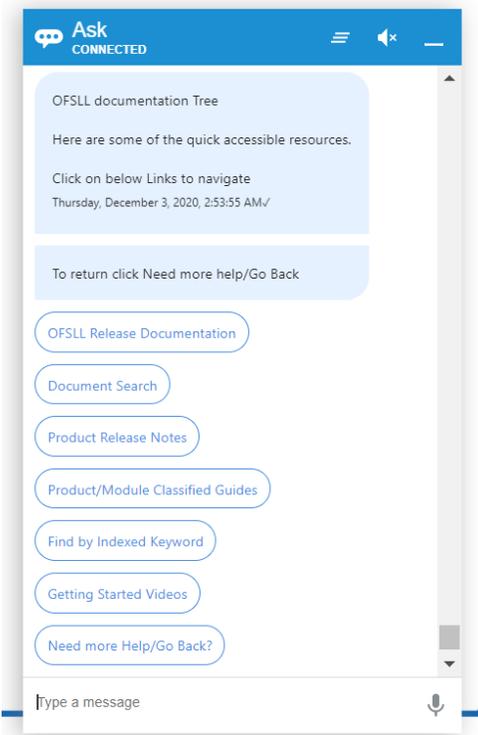
1.4.3 BOT Usability Workflow

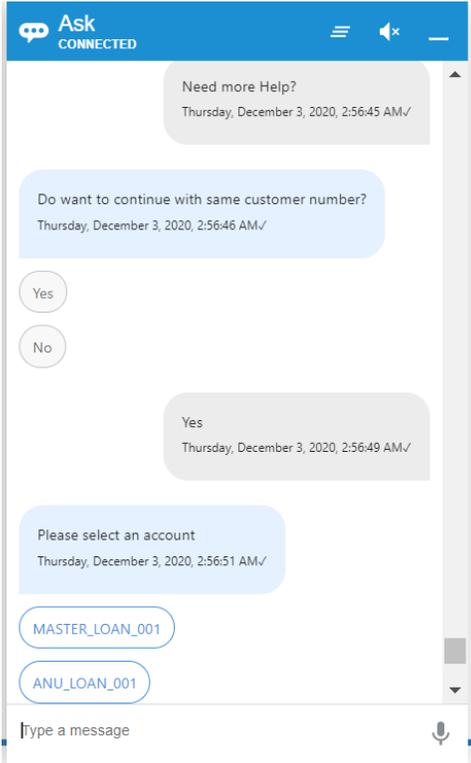
Following is a sample workflow indicating the following steps performed in chatbot:

Action	BOT Response
<p>Begin with entering a Customer ID / Account number.</p>	 <p>The screenshot shows a chatbot interface with a blue header bar containing the text 'Ask CONNECTED'. Below the header, a light blue message bubble contains the text: 'Welcome to OFSLL Bot!! Kindly Enter a Valid Customer Id! Thursday, December 3, 2020, 2:38:22 AM/'. At the bottom of the screen, there is a text input field with the placeholder text 'Type a message' and a microphone icon to its right.</p>
<p>Click on required account from the list of accounts belonging to the Customer ID</p>	 <p>The screenshot shows the chatbot interface after a user input. A grey message bubble contains the text: '11112020 Thursday, December 3, 2020, 2:40:20 AM/'. Below it, a light blue message bubble contains the text: 'Please select an account Thursday, December 3, 2020, 2:40:23 AM/'. Underneath this message, there is a list of four account options, each in a rounded blue button: 'MASTER_LOAN_001', 'ANU_LOAN_001', 'ANU_LINE_001', and 'ANU_LEASE_001'. A blue arrow points from the text in the 'Action' column to the 'MASTER_LOAN_001' button. At the bottom, the text input field and microphone icon are visible.</p>

Action	BOT Response
<p>View the Account Summary. Click Payment Details option form the list.</p>	 <p>ANU_LOAN_001 Thursday, December 3, 2020, 2:41:56 AM ✓</p> <p>Account Summary: Amount Financed: \$ 12,000.00 Outstanding Balance: \$ 1,371.51 RePayment Amount: \$ 1,096.32 Rate of Interest: 9.98% Next Due Date: 2021-01-01 Total Dues: \$ 30.00 Product Name: LN-VE-PAVAN Account Currency: USD Maturity Date: 2021-01-01 Last Payment Amount: \$ 11,500.00 Following are the options available for account Thursday, December 3, 2020, 2:41:59 AM ✓</p> <p>Please select an option</p> <p>Account Details Payment Details</p> <p>Type a message</p>
<p>View the account details with below menu options</p> <ul style="list-style-type: none"> Today's payoff Quote Last 5 Transactions Insurance Details Last Billing details Number of Terms remaining Need more help 	 <p>Account Details Thursday, December 3, 2020, 2:45:28 AM ✓</p> <p>Following are the options available for account Thursday, December 3, 2020, 2:45:29 AM ✓</p> <p>Please select an option</p> <p>Today's Payoff Quote Last 5 Transaction Insurance Details Last Billing Details Number of Terms remaining? Need more Help?</p> <p>Type a message</p>

Action	BOT Response
<p>View the payment details</p> <p>Last 5 payments</p> <p>Next Payment date</p> <p>Advance disbursement request for 2000 \$</p> <p>Need more help</p>	 <p>The screenshot shows a chatbot interface with a blue header labeled 'Ask CONNECTED'. Below the header, there are several input buttons: 'Need more Help?', 'Exit', and a grey bubble containing 'Payment Details Thursday, December 3, 2020, 2:47:52 AM/'. A blue bubble follows with the text 'Following are the options available for account Thursday, December 3, 2020, 2:47:52 AM/'. Below this, there are five more input buttons: 'Last 5 Payments', 'Next Payment Date', 'Advance Disbursement Request for \$2000.', 'Need more Help?', and 'Exit'. At the bottom, there is a text input field with the placeholder 'Type a message' and a microphone icon.</p>
<p>View the communication preference</p> <p>Current Preferences</p> <p>Update Preference</p> <p>Need more help</p>	 <p>The screenshot shows a chatbot interface with a blue header labeled 'Ask CONNECTED'. Below the header, there are several input buttons: 'OFSSL Documentation Tree', 'Need more Help?', and 'Exit'. A grey bubble contains 'Communication Preference Thursday, December 3, 2020, 2:49:34 AM/'. A blue bubble follows with the text 'Following are the options available for account Thursday, December 3, 2020, 2:49:35 AM/'. Below this, there are four more input buttons: 'Current Preferences', 'Update Preferences', 'Need more Help?', and 'Exit'. At the bottom, there is a text input field with the placeholder 'Type a message' and a microphone icon.</p>

Action	BOT Response
<p>View the Limit details</p> <p>Display current limit details</p> <p>Master Account rolled-up Balances</p> <p>Need more help</p>	 <p>The screenshot shows a chatbot interface with a blue header 'Ask CONNECTED'. The chat history includes a user message 'OFSLL Documentation Tree', a bot response 'Need more Help?', and a user message 'Exit'. A timestamped message 'Limit Details' is shown, followed by a list of options: 'Display Current Limit Details', 'Master Account Rolled-up Balances', 'Need more Help?', and 'Exit'. The input field at the bottom contains the text 'Type a message'.</p>
<p>Click on the “OFSLL documentation tree”</p> <p>The available options are</p> <p>OFSLL release documentation</p> <p>Document Search</p> <p>Product release notes</p> <p>Product classified guides</p> <p>Find by indexed keyword</p> <p>Getting started videos</p> <p>Need more help</p>	 <p>The screenshot shows a chatbot interface with a blue header 'Ask CONNECTED'. The chat history includes a user message 'OFSLL documentation Tree', a bot response 'Here are some of the quick accessible resources. Click on below Links to navigate', and a user message 'To return click Need more help/Go Back'. A list of options is shown: 'OFSLL Release Documentation', 'Document Search', 'Product Release Notes', 'Product/Module Classified Guides', 'Find by Indexed Keyword', 'Getting Started Videos', and 'Need more Help/Go Back?'. The input field at the bottom contains the text 'Type a message'.</p>

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

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:

- [OFSLL Wrapper customization](#)
- [ODA – Dialog Flow Development](#)
- [Deploying war file on WebLogic Server](#)
- [Web application UI for Accessing BOT](#)
- [Configure CSF Mapping in Weblogic](#)

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

```

ofssl.baseURL = <OFSLL REST service base URL
<http://<host>:<port>/OFSLLRestWS/service/api/resources>
ofssl.username = <OFSLL username>
ofssl.password = <OFSLL pass>
ofssl.suffix = htm
ofssl.otmHttpUrl=https://docs.oracle.com/cd/
ofssl.fIndex=/findex.htm
ofssl.index=index.htm
ofssl.video=/videos.htm
ofssl.ofsllReleaseNotes=/pdf/refdocs/ofsll_release_notes.pdf
ofssl.ofsllReleaseDoc=https://docs.oracle.com/en/industries/financial-
services/financial-lending-leasing/index.html
ofssl.splitSeperator==
ofssl.maxHitsResults=<max number of results returned>
ofssl.indexDir = <Release index directory path of server >
ofssl.releaseVersionUrl= <Release Part number>
ofssl.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(ODARRequestDTO 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("ofsl.indexDir")+prop.getProperty("ofsl.releaseNo");
            String releaseVersionUrl = prop.getProperty("ofsl.releaseVersionUrl");
            String urlPrefix = prop.getProperty("ofsl.otmHttpUrl");
            String splitSeperator = prop.getProperty("ofsl.splitSeperator");
            String releaseNo = prop.getProperty("ofsl.releaseNo");
            String urlPrefixPath = urlPrefix + releaseVersionUrl;
            String findIndexPath = prop.getProperty("ofsl.findIndex");
            String indexPath = prop.getProperty("ofsl.index");
            String videoPath = prop.getProperty("ofsl.video");
            String ofslReleaseNotesPath = prop.getProperty("ofsl.ofslReleaseNotes");
            String ofslReleaseDocPath = prop.getProperty("ofsl.ofslReleaseDoc");
            Integer maxHitsResults =
Integer.parseInt(prop.getProperty("ofsl.maxHitsResults"));
            File fileIndexDirPath = new File(indexDirPath);
            LuceneSearchHighlighter luceneSearchHighlighter = new
LuceneSearchHighlighter();
            List<String> fileList = new ArrayList <> ();

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

```

```

        if ((searchQuery.toLowerCase().trim().contains("#ofsl release document")))
    {
        releaseNo="All Release Version";
        fileList.add(searchQuery + splitSeperator + ofslReleaseDocPath +
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("#ofsl release notes"))) {
        fileList.add(searchQuery + splitSeperator + urlPrefixPath +
ofslReleaseNotesPath + 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(ODARRequestDTO 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: " +
dateFormatter(accountsDetails.getAccountDetailSummary().get(0).getStatementDetails().
get(0).getGenerationDate()) + "\n Closing Date: " +
dateFormatter(accountsDetails.getAccountDetailSummary().get(0).getStatementDetails().
get(0).getClosingDate()) + "\n Due Date: " +
dateFormatter(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>/wlsserver/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>/ofssl/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.

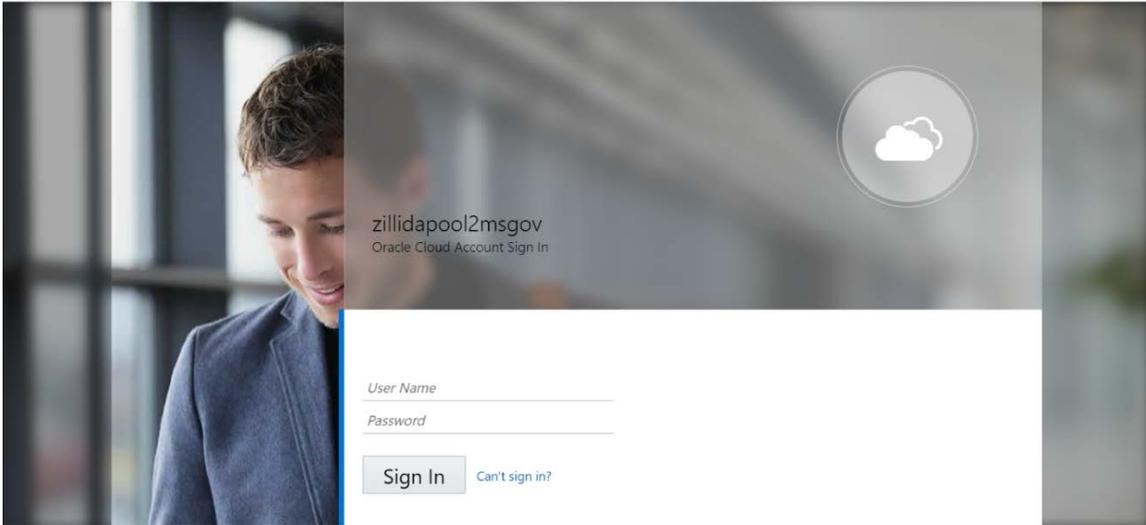
In-order to achieve a sequence of menu options, dialog flow development is required to be performed in ODA Oracle Digital Assistant. Following is a quick overview of steps involved:

- 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:

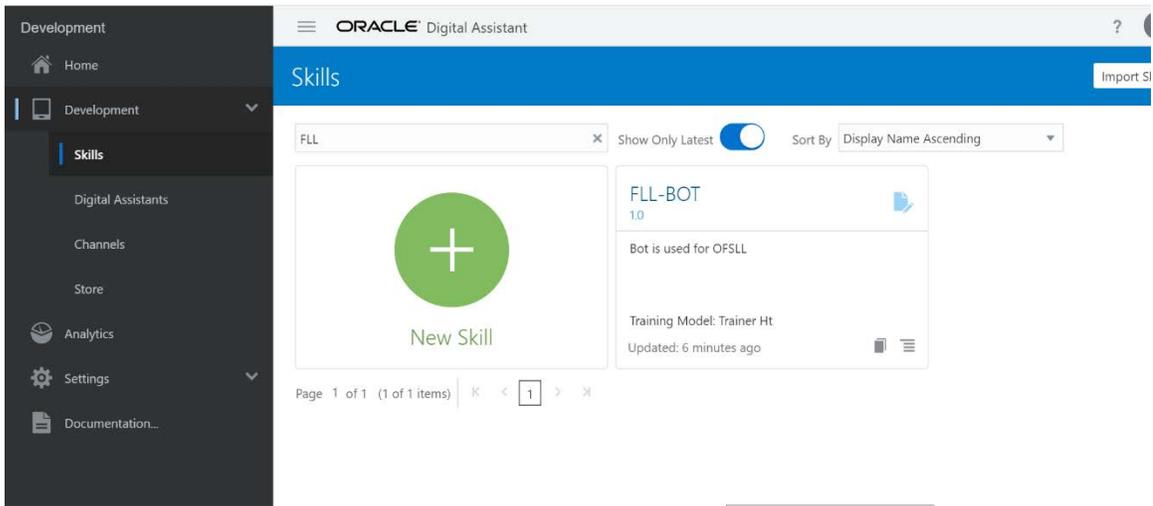
1. Login to ODA UI



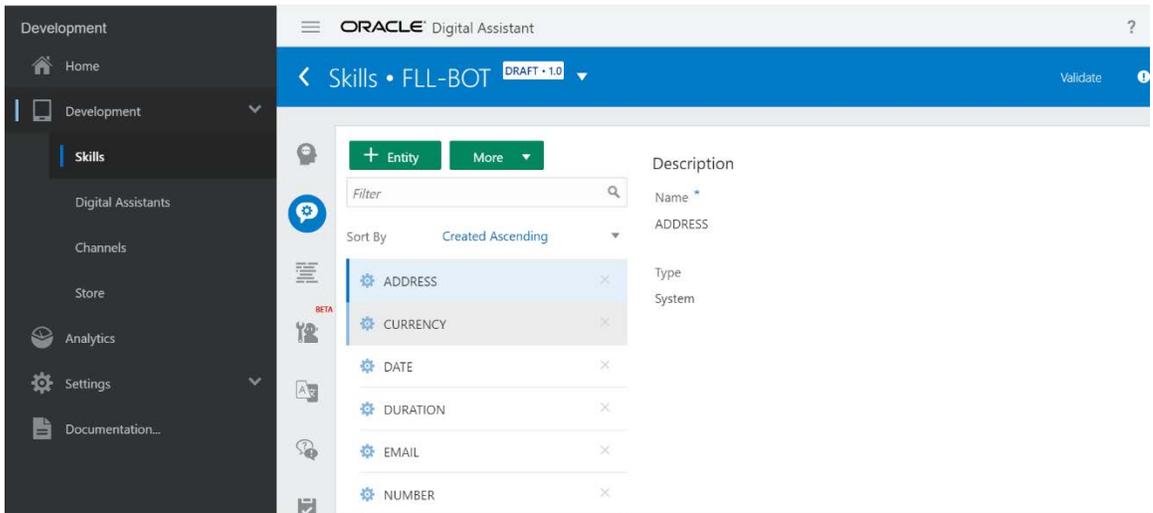
2. Go to Home



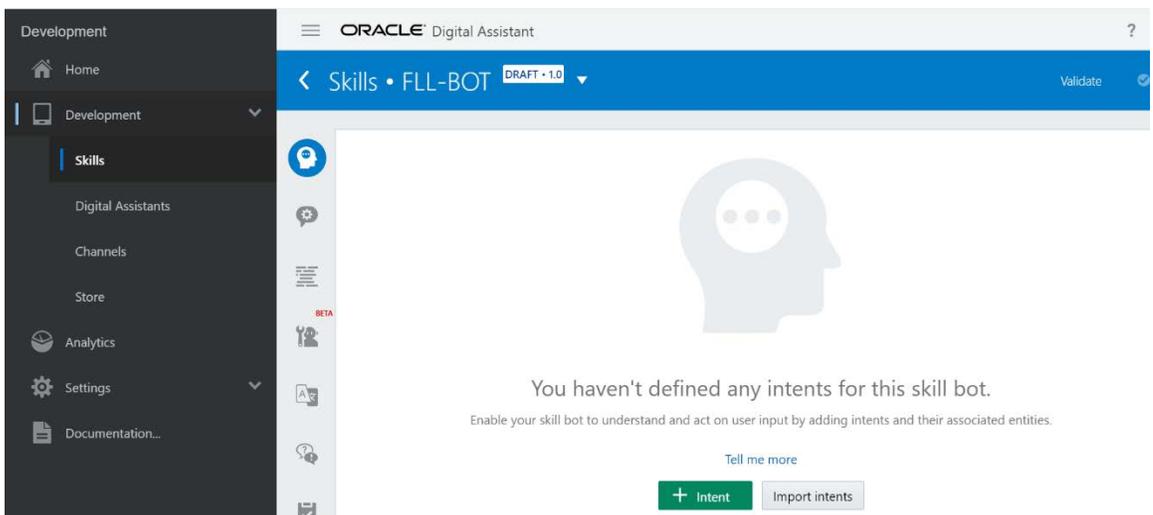
3. Create Skill/Digital Assistant.



4. Add Entities



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



6. Add Bot flow using Yaml

The screenshot shows the Oracle Digital Assistant interface. On the left is a navigation menu with options like Home, Development, Skills, Digital Assistants, Channels, Store, Analytics, Settings, and Documentation. The main area displays the configuration for a skill named 'FLL-BOT' in 'DRAFT 1.0' mode. A '+ Components' button is visible at the top. The configuration is shown in a code editor with the following YAML content:

```
1
2 metadata:
3   platformVersion: "1.0"
4   main: true
5   name: Testlease1
6   context:
7     variables:
8       temp: string
9
10 #variables for common purpose
11 iResult: nIresult
12 response: string
13 acc_nbr: string
14 transactiontype: string
15 acc_list: list
16 asset: string
17 oldAsset: string
18 serviceType: string
19 phoneNumber: string
20
21
22
23 #variables for fetch accounts
```

A 'Saving changes...' indicator is visible at the bottom right of the editor.

7. Add OFSLL REST Service

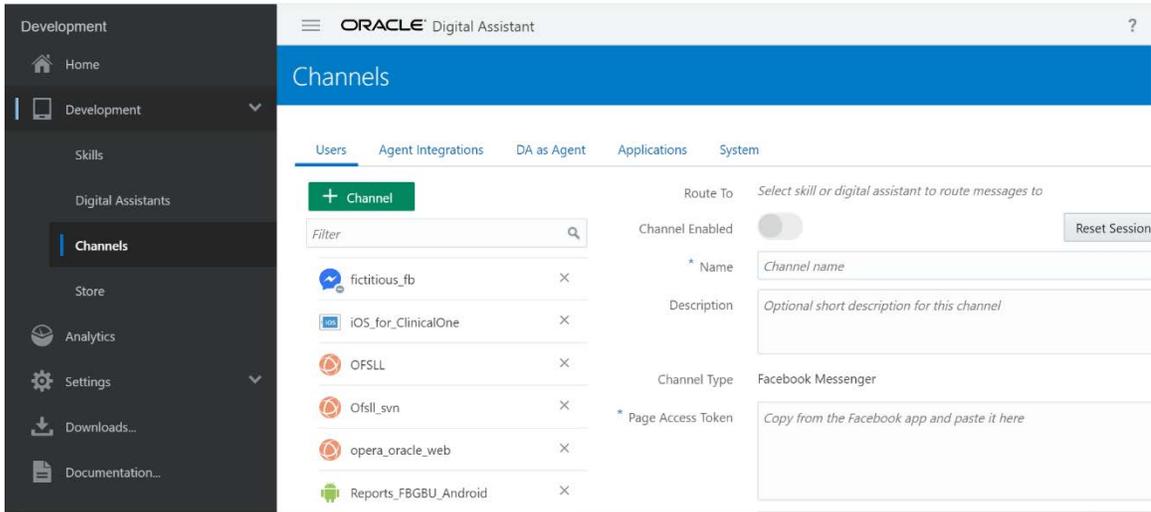
The screenshot shows a message in the Oracle Digital Assistant interface. The message reads: "You haven't configured any services for the custom components used by this skill bot." Below the message, it says: "Add the names, URLs, and authentication settings for the services that implement each of the custom components that you've used in a flow." There is a "Tell me more" link and a green "+ Service" button.

The screenshot shows the Oracle Digital Assistant interface for configuring a service. The service is named "Operations" and is currently "Service Enabled". The configuration details are as follows:

- Name: Operations
- Description: Operations
- Status: Ready
- Platform Version: 1.0
- Metadata URL: http://<server name>:<port #>/ofsl/v1/fulfillment
- User Name: [Empty field]
- Password: [Masked field]

There is an "Optional HTTP Headers" section with a note: "You can include HTTP headers here, if needed by your hosting service." and an "Add HTTP Header" button. The interface also shows a list of services on the left, including "Operations", "accountsummary", "advancedisbursement", "currentpreferences", "currentrateofinterest", "displaycurrentlimidetails", "fetchaccounts", "insuranceedetails", "lastbillingdetails", and "fulfillmentsummary".

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



- 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

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

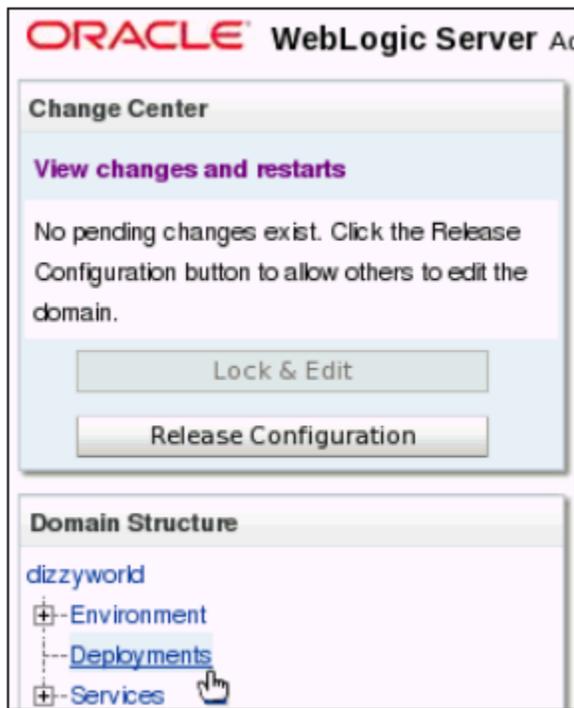
Note: Use the host name and port of the administration server of your domain.



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



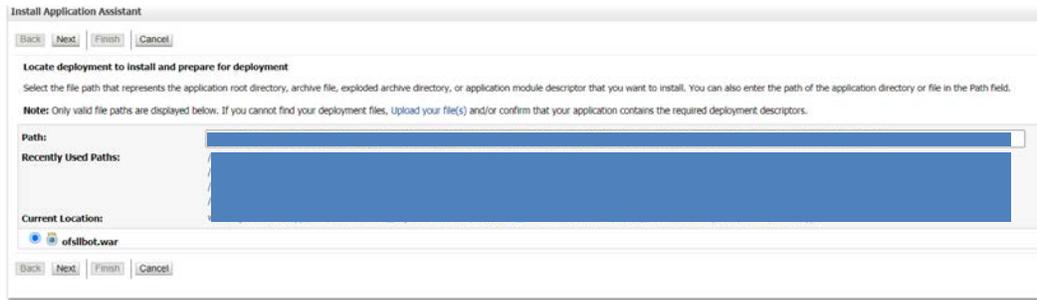
- Under Domain Structure, click 'Deployments'.



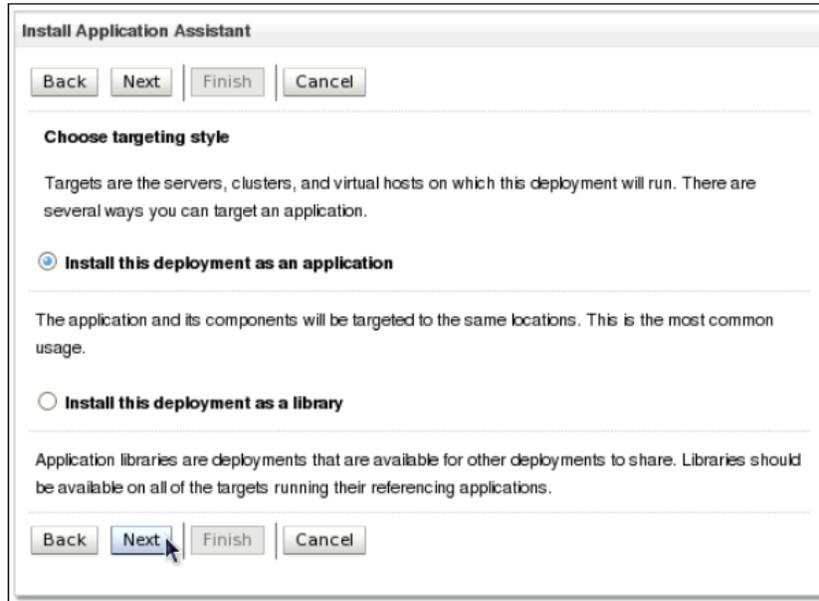
- On the right, under Deployments, click 'Install'.



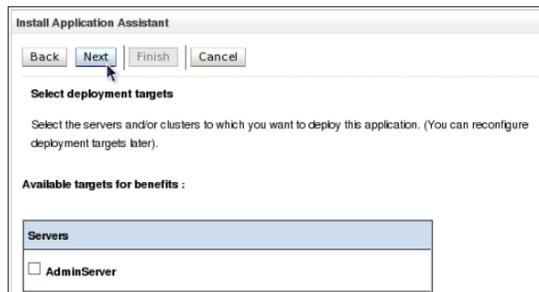
- Find the Current Location field. Use the links to browse to the location in which you placed the downloaded ofslbot.war. War file.
- On locating ofslbot.war, click the radio button next to it. Using the links and the radio button, the console auto populates the Path fields. Alternatively, you can type 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'.

The screenshot shows the 'Install Application Assistant' dialog box with the 'Optional Settings' section active. At the top, there are four buttons: 'Back', 'Next', 'Finish', and 'Cancel'. The 'Next' button is highlighted with a mouse cursor. Below the buttons, the text reads 'Optional Settings' and 'You can modify these settings or accept the defaults'. The 'General' section is expanded, showing the question 'What do you want to name this deployment?' with a text input field labeled 'Name:'. The 'Security' section is also expanded, showing the question 'What security model do you want to use with this application?' with four radio button options: 'DD Only: Use only roles and policies that are defined in the deployment descriptors.' (selected), 'Custom Roles: Use roles that are defined in the Administration Console; use policies that are defined in the deployment descriptor.', 'Custom Roles and Policies: Use only roles and policies that are defined in the Administration Console.', and 'Advanced: Use a custom model that you have configured on the realm's configuration page.'. The 'Source accessibility' section is expanded, showing the question 'How should the source files be made accessible?' with one radio button option: 'Use the defaults defined by the deployment's targets' (selected).

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

The screenshot shows the 'Install Application Assistant' dialog box with the 'Review your choices and click Finish' section active. At the top, there are four buttons: 'Back', 'Next', 'Finish', and 'Cancel'. The 'Finish' button is highlighted with a mouse cursor. Below the buttons, the text reads 'Review your choices and click Finish' and 'Click Finish to complete the deployment. This may take a few moments to complete.'. The 'Additional configuration' section is expanded, showing the question 'In order to work successfully, this application may require additional configuration. Do you want to review this application's configuration after completing this assistant?' with two radio button options: 'Yes, take me to the deployment's configuration screen.' and 'No, I will review the configuration later.' (selected).

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.

The screenshot shows the Oracle J2EE Administration Console interface. At the top, under the 'Messages' section, there are two green checkmark messages: 'The deployment has been successfully installed.' and 'You must also activate the pending changes to commit this, and other updates, to the active system.'

Below the messages is the 'Summary of Deployments' section, which has two tabs: 'Control' (selected) and 'Monitoring'. A text block explains that this page displays a list of Java EE applications and stand-alone application modules that have been installed to this domain. It notes that installed applications and modules can be started, stopped, updated (redeployed), or deleted from the domain by first selecting the application name and using the controls on this page. It also states that to install a new application or module for deployment to targets in this domain, the user should click the Install button.

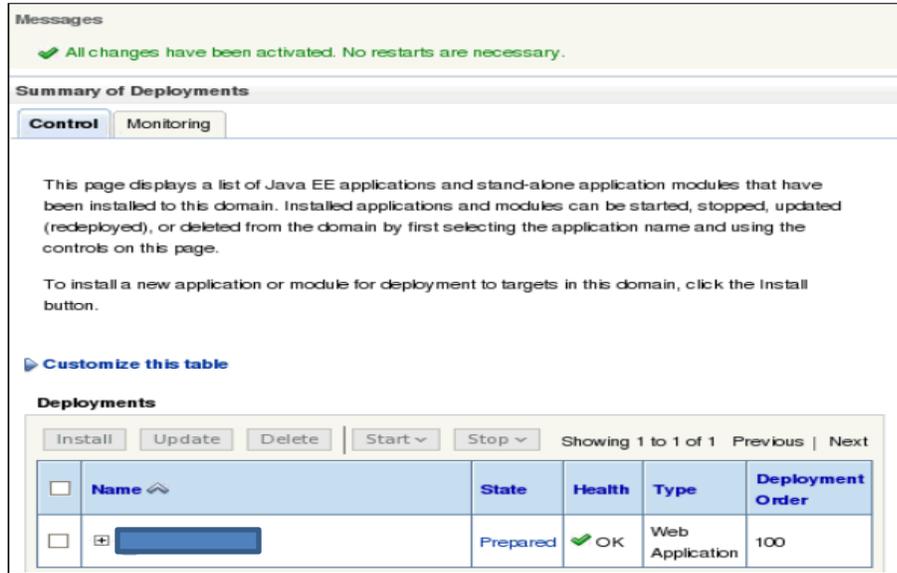
There is a link to 'Customize this table'.

The 'Deployments' section features a table with columns: Name, State, Health, Type, and Deployment Order. Above the table are buttons for 'Install', 'Update', 'Delete', 'Start', and 'Stop', along with pagination controls 'Showing 1 to 1 of 1 Previous | Next'. The table contains one entry with a checkbox, a plus sign, and a redacted name. The 'State' is 'distribute', 'Health' is 'Initializing', 'Type' is 'Web Application', and 'Deployment Order' is '100'. Below the table are the same control buttons and pagination as above.

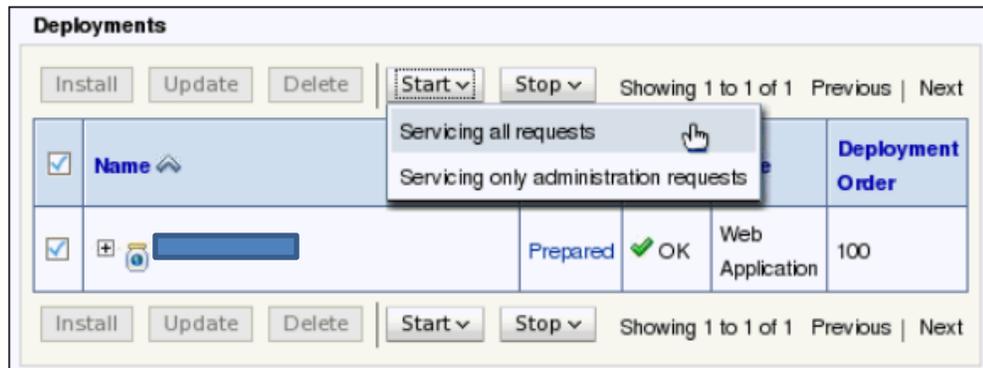
12. In the Change Center, click the Activate Changes button.

The screenshot shows the Oracle Change Center interface. It has a title bar 'Change Center' and a sub-header 'View changes and restarts'. Below this, a text block states: 'Pending changes exist. They must be activated to take effect.' There are two buttons: 'Activate Changes' (with a green checkmark icon) and 'Undo All Changes'. A mouse cursor is pointing at the 'Activate Changes' button.

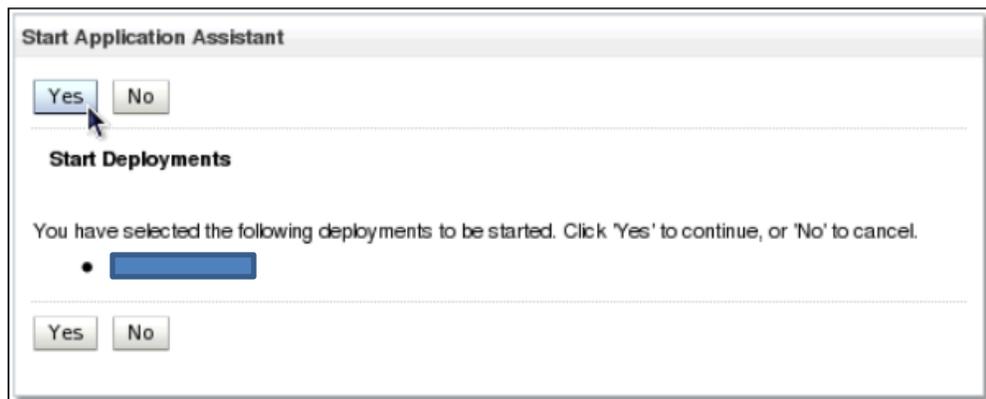
Notice the message indicating that the changes have been activated. In addition, notice the benefits application listed in the Deployments table is now in the "Prepared" state.



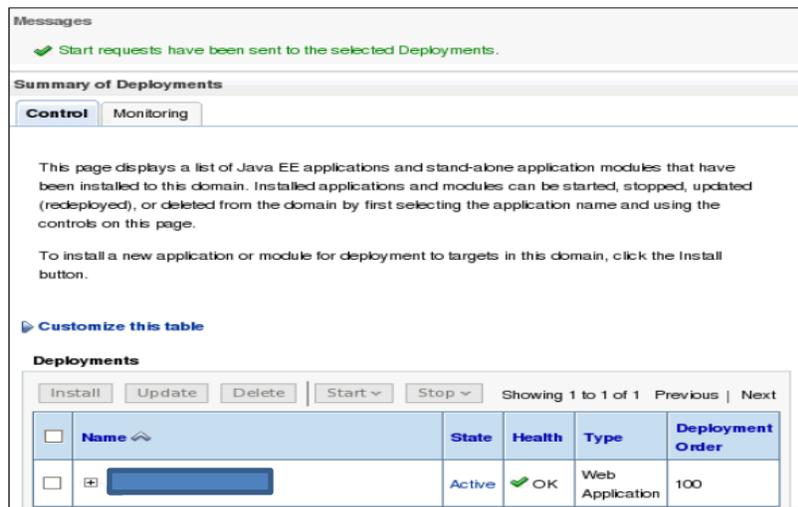
13. Select the checkbox against the left of the benefits application in the Deployments table. In the Start drop-down list, select 'Servicing all requests' option.



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:

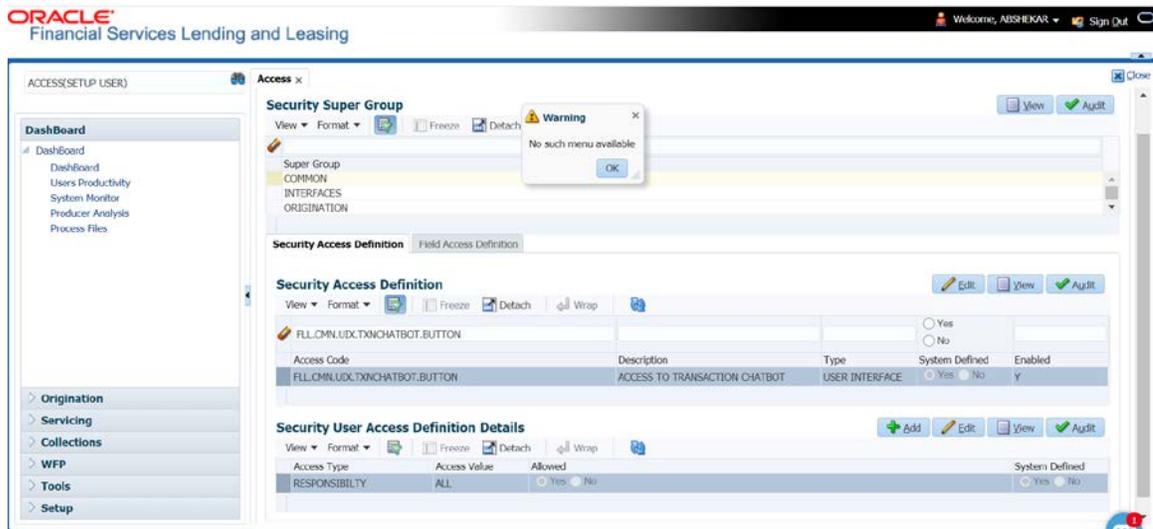
- In case you wish to launch BOT as separate application, Modify **index.html** in WebApp(or WebApp.war) and update the following 2 fields with required details:
 - URI: '<ODA host>',
 - channelId: 'published bot channel ID'
- In case you wish to integrate BOT in an existing front-end application, use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

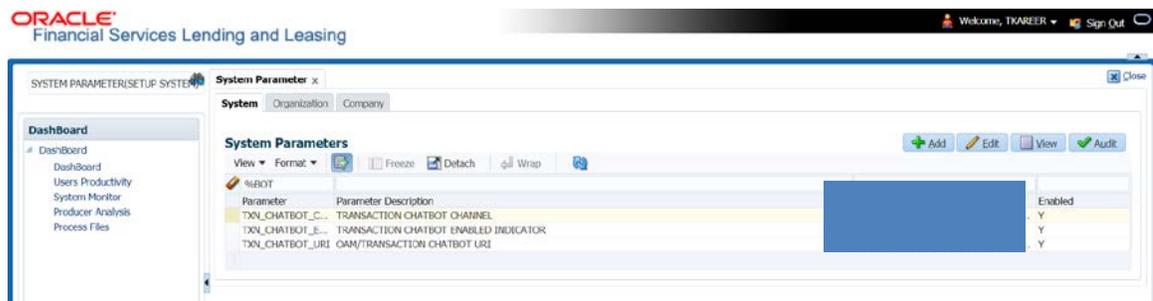
For additional information, contact Oracle Financial Services Lending and Leasing Product Engineering team.

2.6 App configuration for enabling chatbot

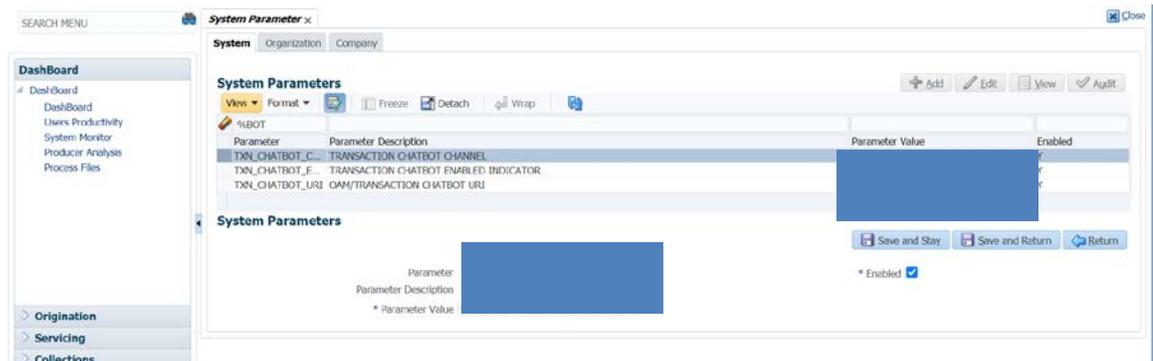
1. Enabling BOT and adding parameters:
 - Channel ID
 - URI
 - Enabled – Yes / No
2. Enable the FLS Access Key - FLL.CMN.UIX.TXNCHATBOT.BUTTON as indicated in the Access Setup screen.



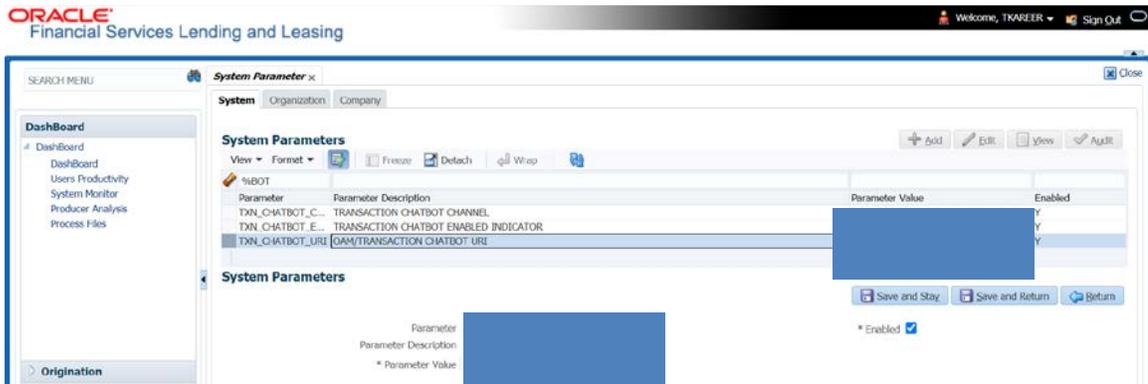
3. Search for "System parameter" in the box below



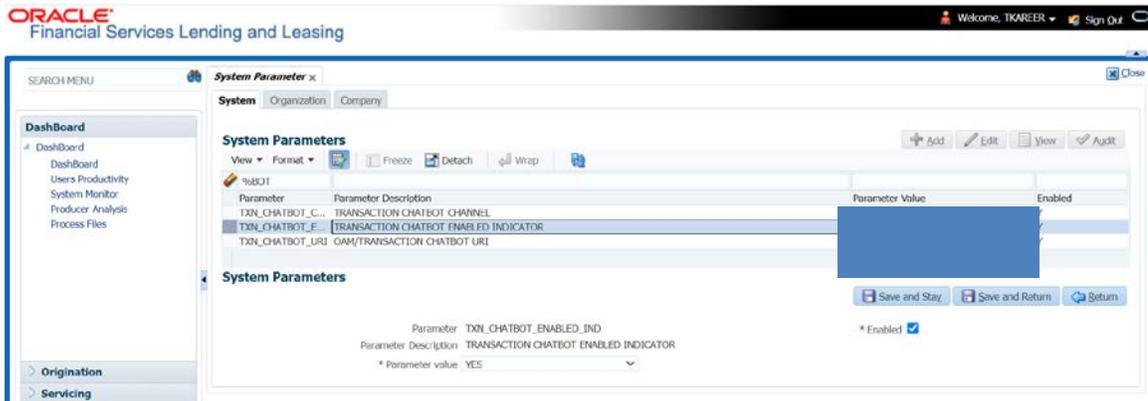
4. Enter the Channel id, click save and return.



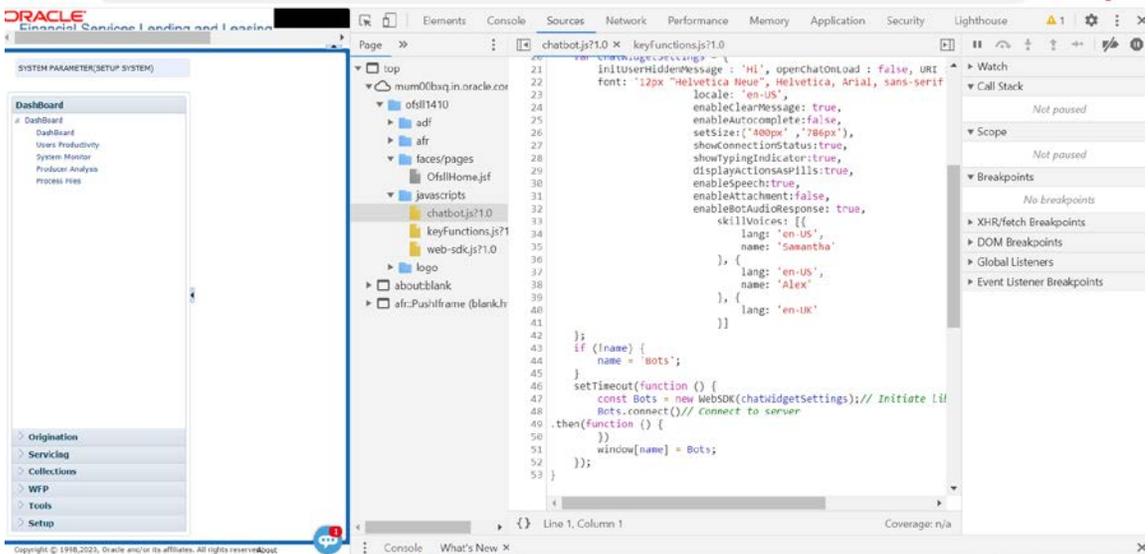
5. Enter the URI , click save and return



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';
    }
}

```

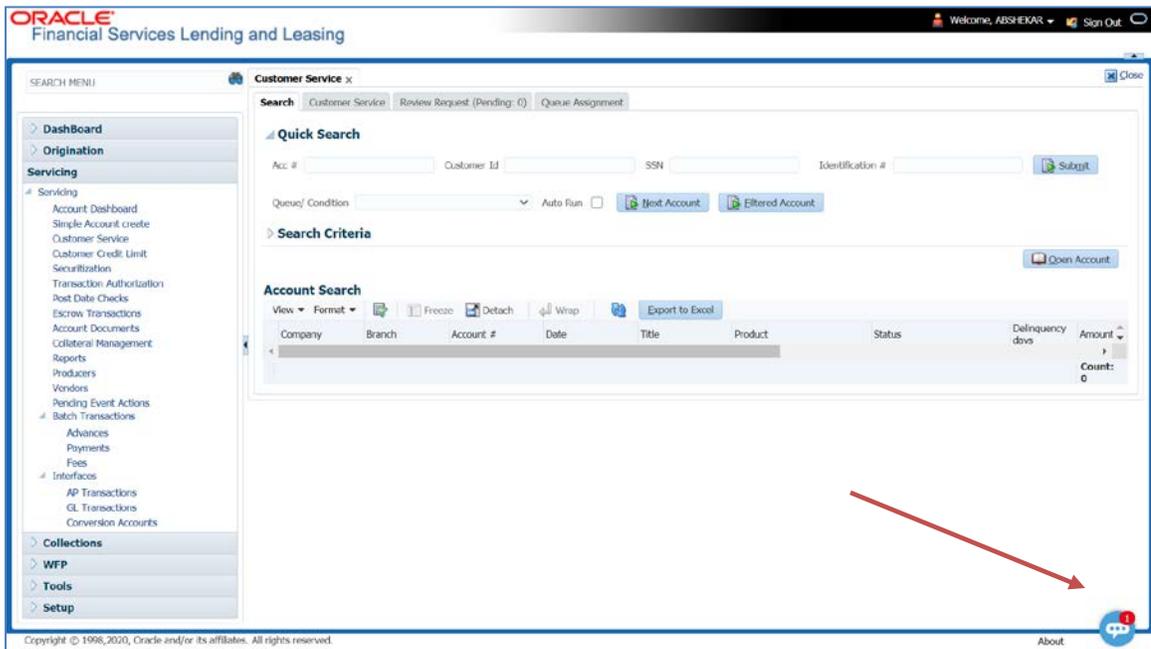
```

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

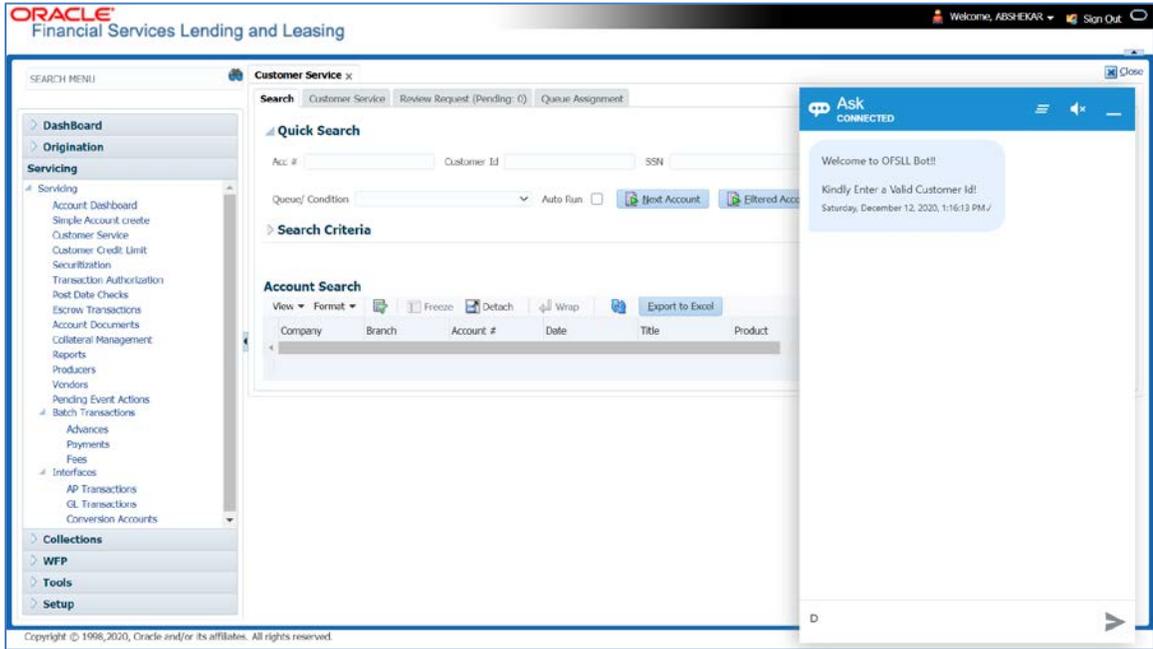
```

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

Copyright © 1998, 2020, Oracle and/or its affiliates. All rights reserved.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are “commercial computer software” pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate failsafe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish or display any part, in any form, or by any means. Reverse engineering, disassembly, or recompilation of this software, unless required by law for interoperability, is prohibited. The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

This software or hardware and documentation may provide access to or information on content, products and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services.