

# **Oracle Banking Digital Experience**

**Chatbot Configuration Guide  
Release 19.1.0.0.0**

**Part No. F18558-01**

**May 2019**

**ORACLE®**

Oracle Financial Services Software Limited

Oracle Park

Off Western Express Highway

Goregaon (East)

Mumbai, Maharashtra 400 063

India

Worldwide Inquiries:

Phone: +91 22 6718 3000

Fax: +91 22 6718 3001

[www.oracle.com/financialservices/](http://www.oracle.com/financialservices/)

Copyright © 2019, 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 decompilation 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.

# Table of Contents

<b>1. Preface.....</b>	<b>4</b>
1.1 Intended Audience .....	4
1.2 Documentation Accessibility .....	4
1.3 Access to Oracle Support .....	4
1.4 Structure.....	4
1.5 Related Information Sources.....	4
<b>2. Purpose .....</b>	<b>5</b>
<b>3. Topology .....</b>	<b>6</b>
<b>4. Facebook Configurations .....</b>	<b>7</b>
4.1 ODA Configurations .....	11
4.2 OBDX Server Configurations .....	13
<b>5. Custom Webhook Configuration in ODA.....</b>	<b>17</b>
<b>6. Alexa Skill (Zig Bank) Configuration .....</b>	<b>19</b>
6.1 Define the Interaction Model .....	19
6.2 Create a Webhook channel.....	24
6.3 Configure the Endpoint .....	27

# 1. Preface

## 1.1 Intended Audience

This document is intended for the following audience:

- Customers
- Partners

## 1.2 Documentation Accessibility

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

## 1.3 Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit

<http://www.oracle.com/pls/topic/lookup?ctx=accandid=info> or visit

<http://www.oracle.com/pls/topic/lookup?ctx=accandid=trs> if you are hearing impaired.

## 1.4 Structure

This manual is organized into the following categories:

*Preface* gives information on the intended audience. It also describes the overall structure of the User Manual.

The subsequent chapters describes following details:

- Purpose
- Configuration / Installation.

## 1.5 Related Information Sources

For more information on Oracle Banking Digital Experience Release 19.1.0.0.0, refer to the following documents:

- Oracle Banking Digital Experience Licensing Guide

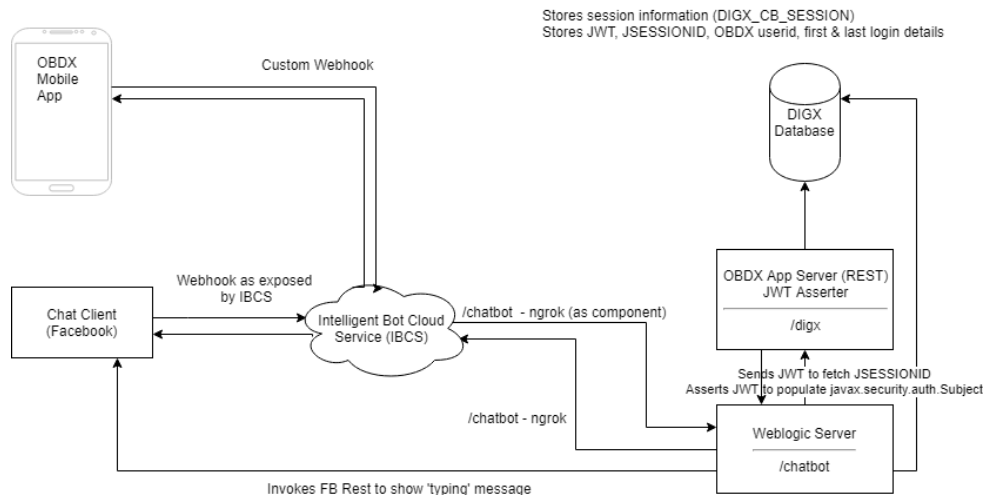
## 2. Purpose

OBDX 19.1 provides interface for Chatbot module, integrated with Oracle Digital Assistant (ODA) out of the box. It provides end users a chat interface to interact with the bank. Transactions like balance enquiry, fund transfers to payees, enquiring about banking products and details of ATM/Branches can be achieved through chat. This document provides steps to setup OBDX chatbot module with ODA. The prerequisites include –

- ODA setup
- Facebook credentials (optional)

[Home](#)

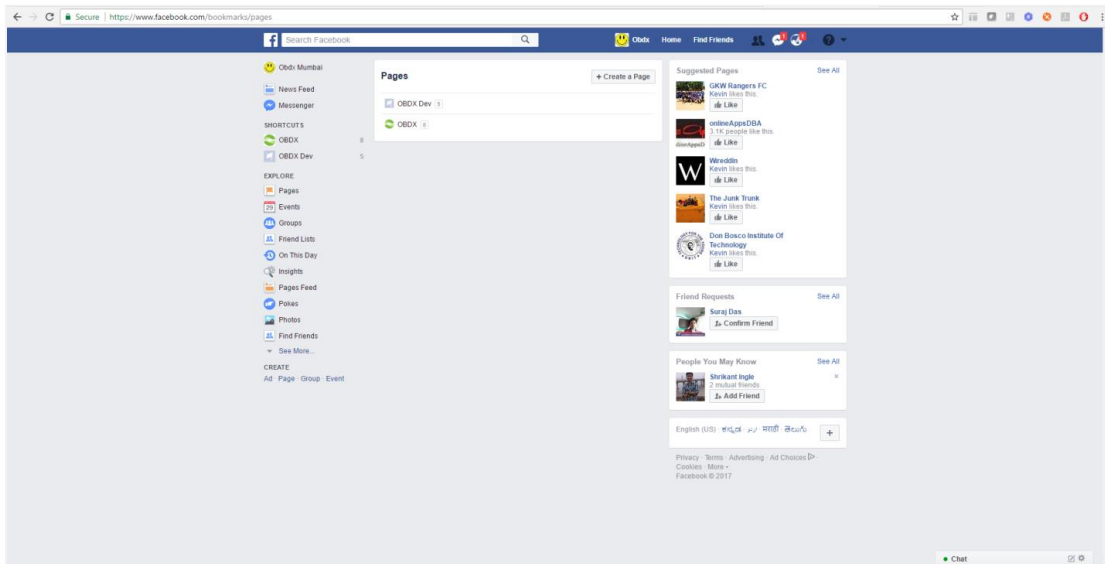
### 3. Topology

[Home](#)

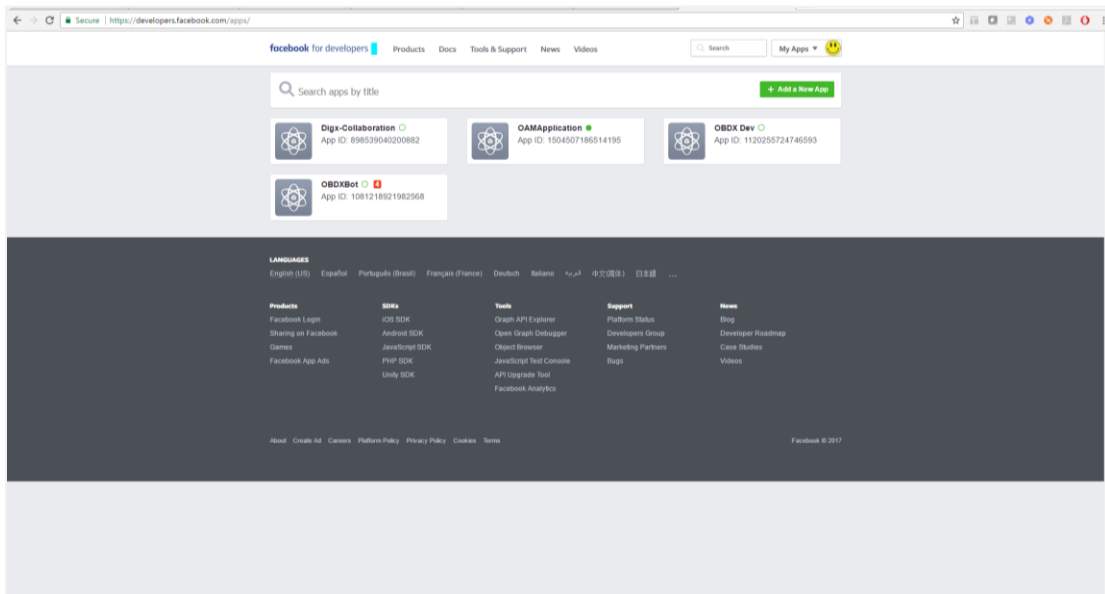
## 4. Facebook Configurations

Create a Facebook account for the Bank

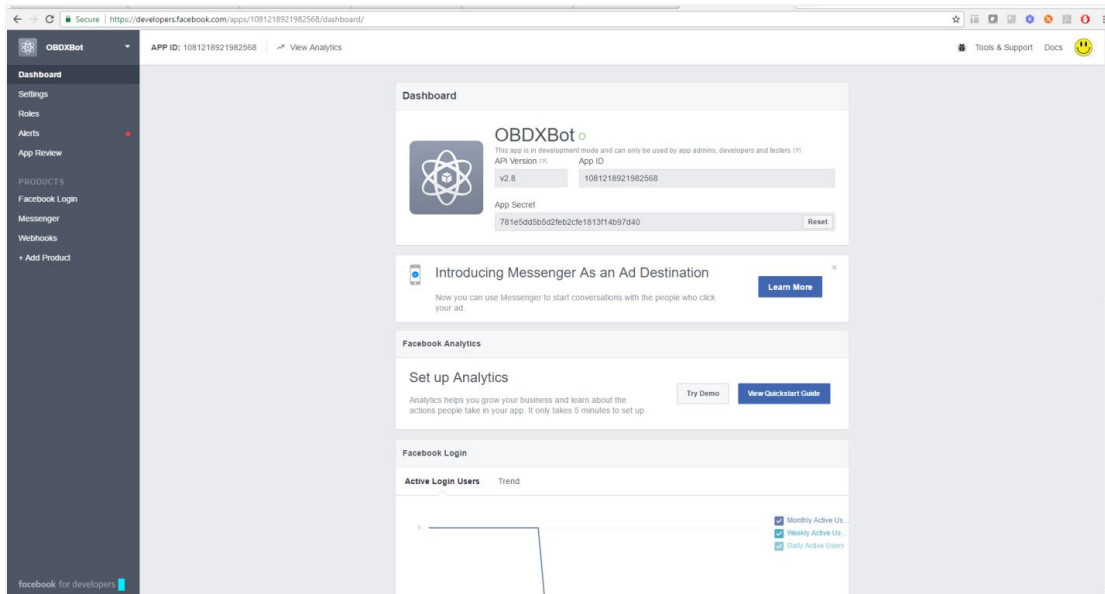
1. Login to Facebook with credentials.
2. Create a new page



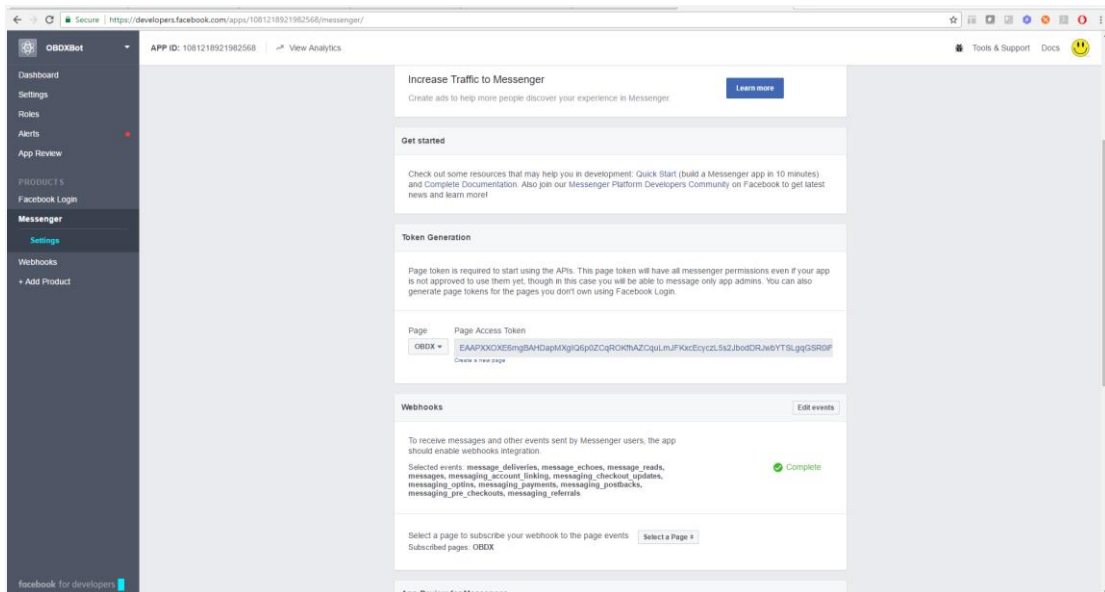
3. Navigate to developer link and create an application as shown below



4. Navigate to dashboard page and note the app secret

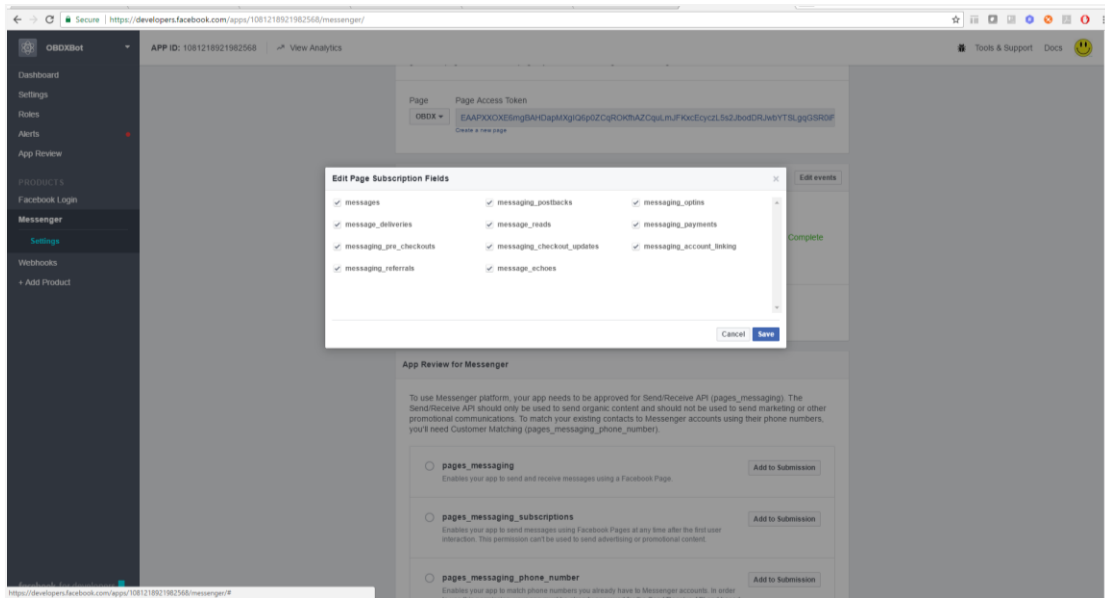


5. Navigate to Messenger > Settings page from left panel and in token generation section select the page created previously. Note the page access token.

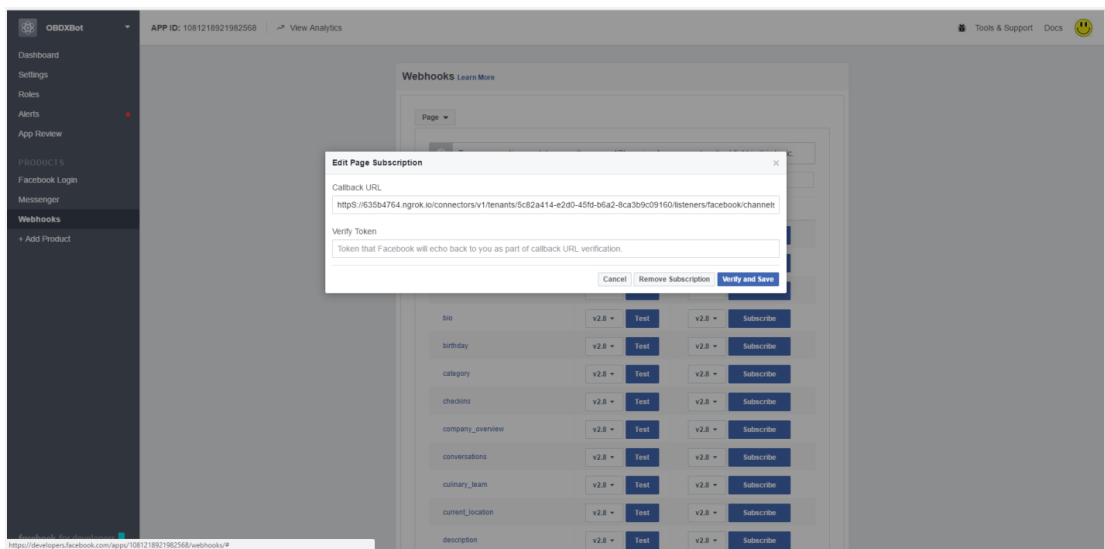


6. In the webhook section select the events

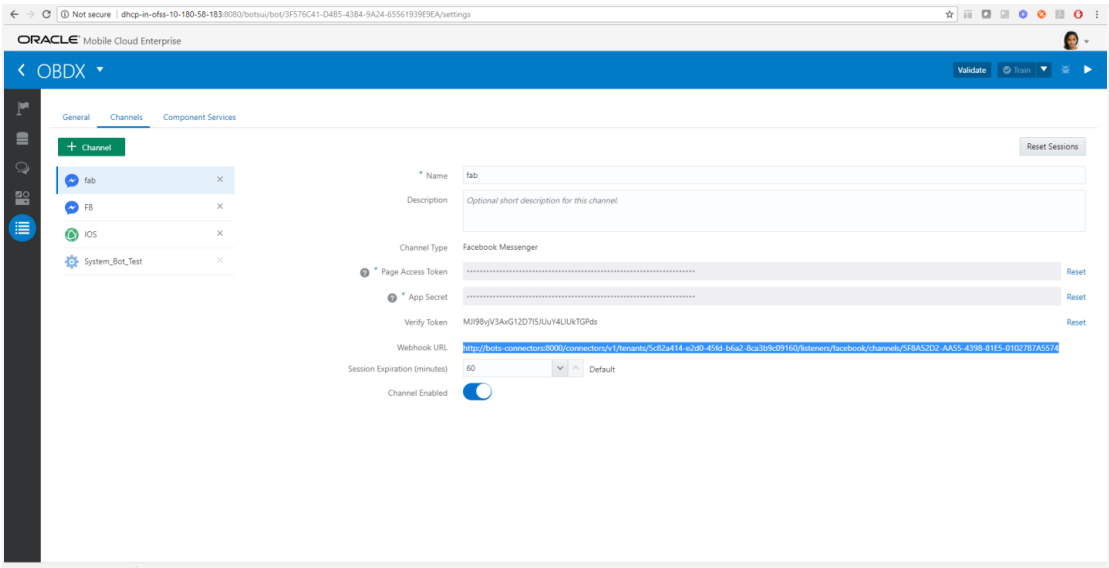




## 7. Navigate to Webhook > Edit Subscription

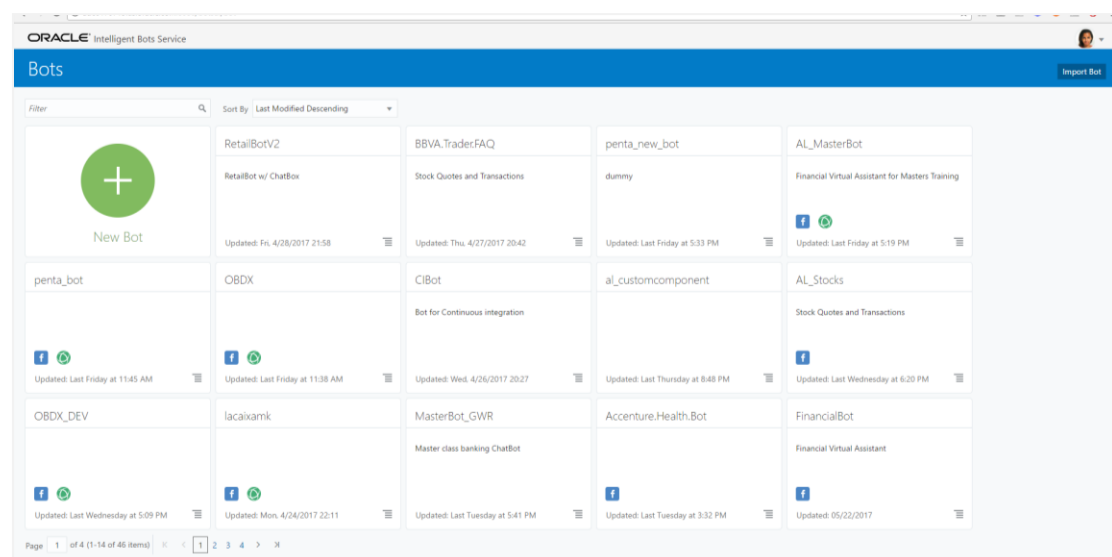
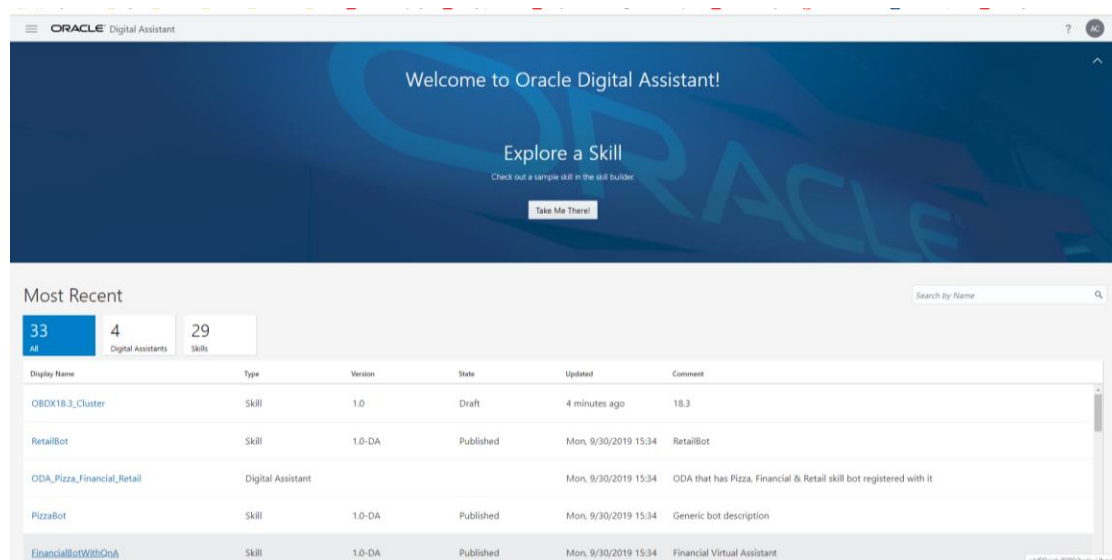


8. Add the ODA URL here.  
ODA URL should be available on the internet for Facebook to get access. This URL is obtained from below screen in ODA. The verify token is also obtained here.

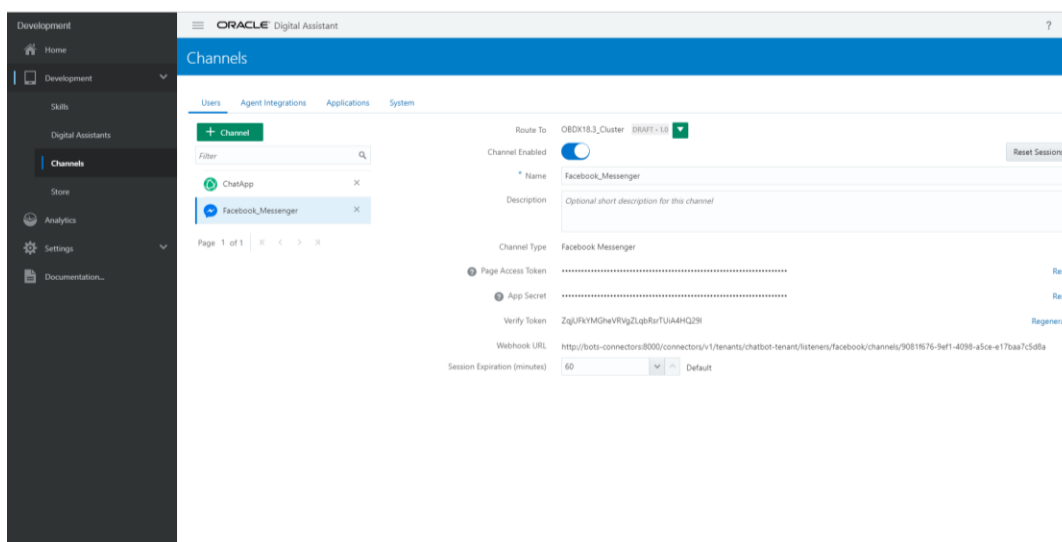


## 4.1 ODA Configurations

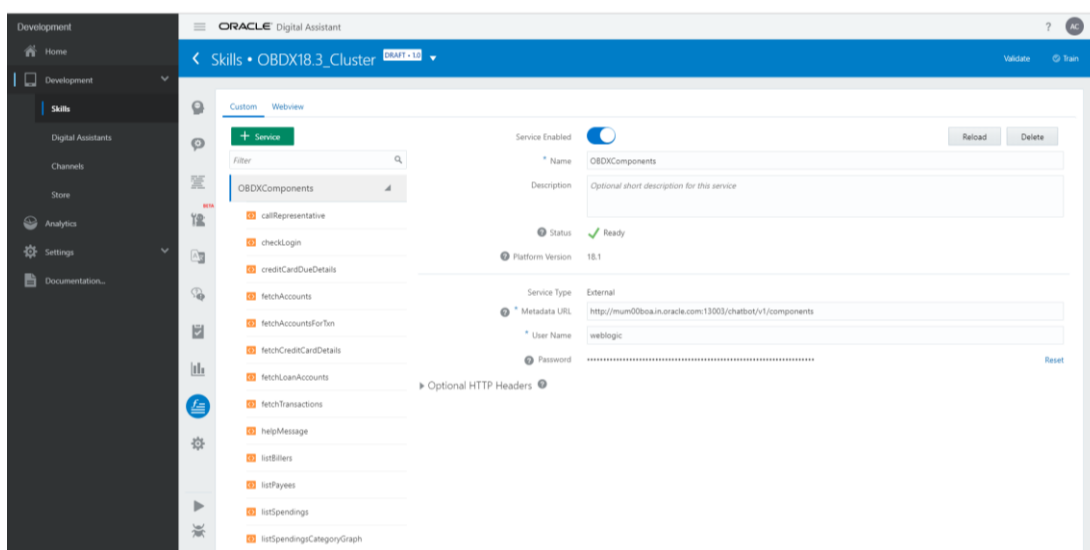
1. Login to ODA and import the OBDX bot shipped with OBDX installer. This is a zip (OBDX191.zip) file obtained in the installer in OBDX\_Installer/installables/chatbot/config directory. Import this by clicking the “Import Bot” on ODA dashboard.



2. Click on the OBDX Bot

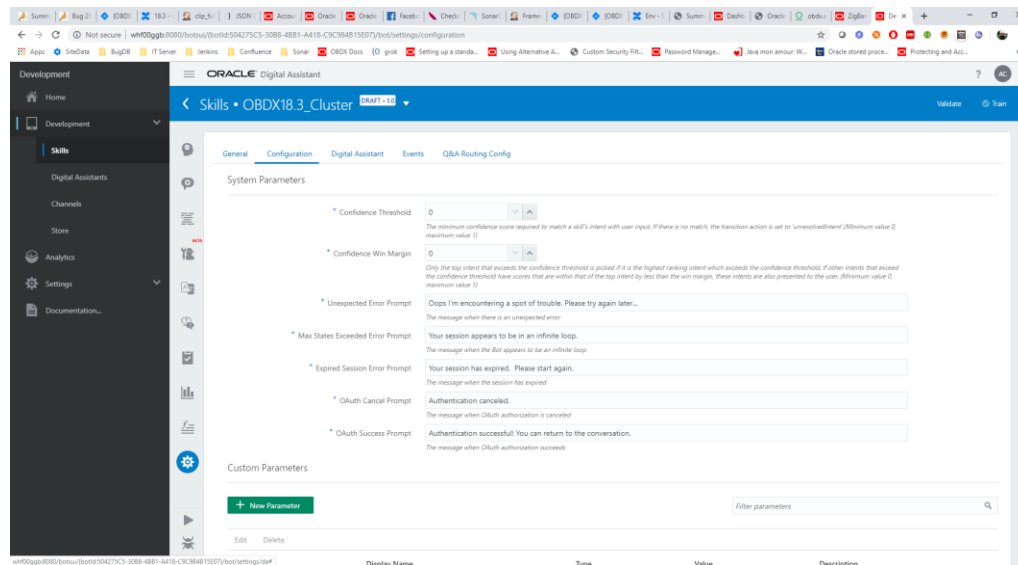


3. Update the Page access token and App Secret created previously in Facebook console.
4. Navigate to Webhook and enter the OBDX URL here. The OBDX setup and the ODA setup must be accessible over Internet.



5. Add username/password (in HTTP Basic authorization) of any user with Administrators role which can be used to login in OBDX Weblogic server.

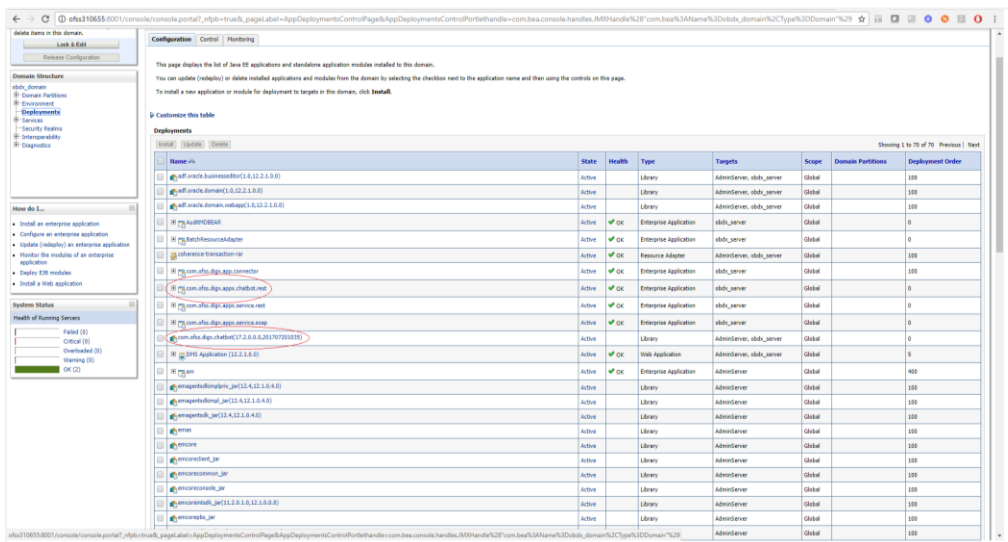
## 6. Configure confidence threshold (generally 0.5) as shown below



## 4.2 OBDX Server Configurations

Ensure that below applications are running on OBDX server

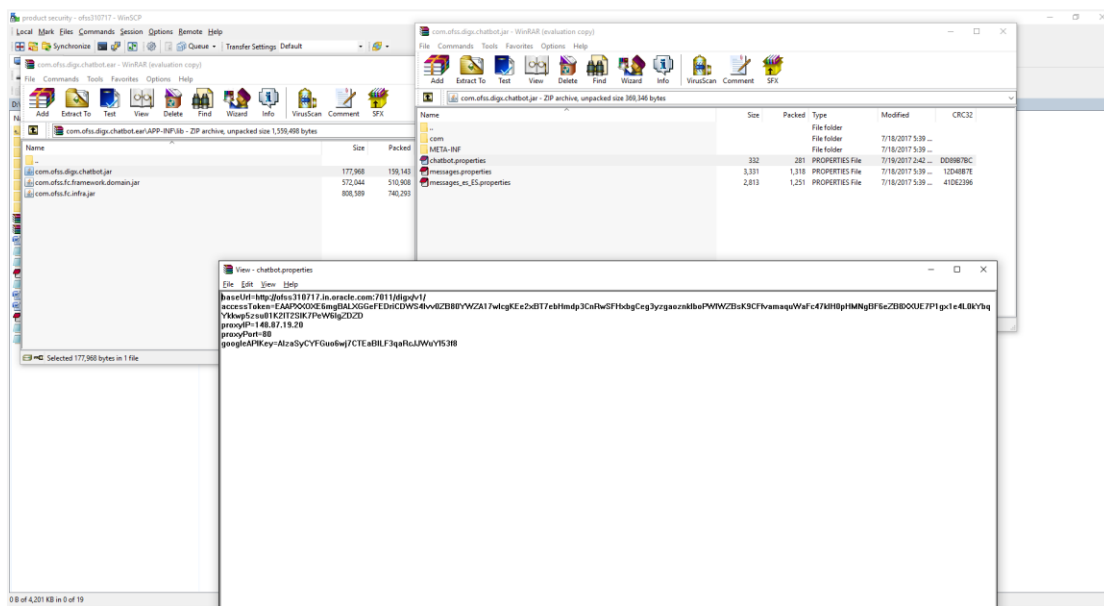
- com.ofss.digx.infra.channel
- com.ofss.digx.chatbot



In chatbot.properties in com.ofss.digx.chatbot.ear > com.ofss.digx.chatbot.jar, enter the base URL of the OBDX server where DIGX application is running.

**Note:** Enter the Weblogic port. If using OHS, that should not be patched with Webgate

If the server needs proxy to connect to internet, enter proxy details here else leave them blank. This call is required for the chatbot to display the typing.. icon in chat. The connection is directly from OBDX Chatbot application to Facebook. The access token of the Facebook page is also required here (which is generated on Facebook console in step 3e).



Redeploy the ear after above changes.

### Verification Steps

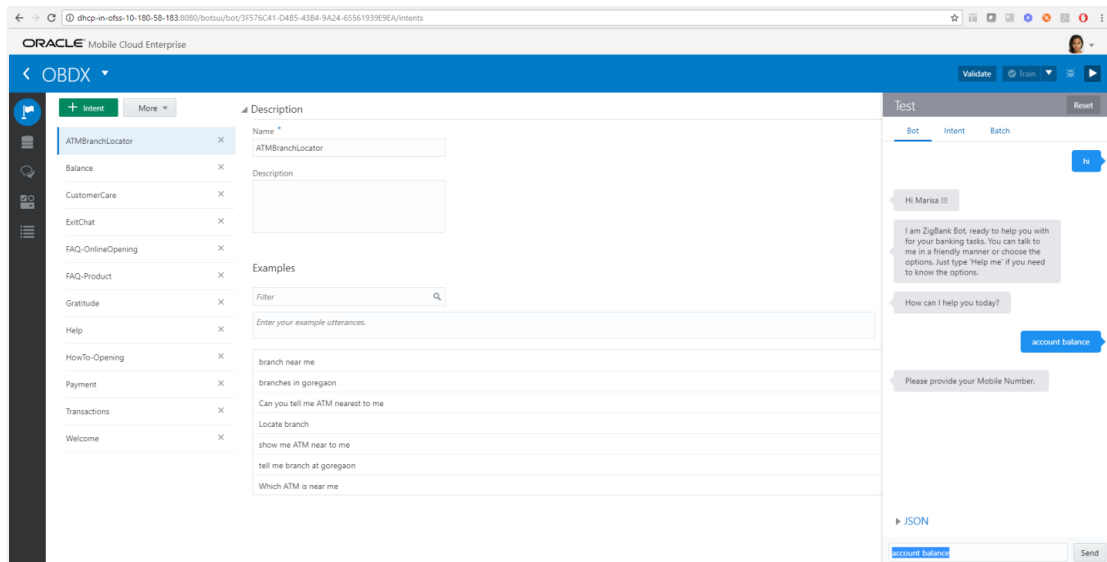
1. Check if OBDX server Chatbot application is running and accessible over the component URL given to ODA. Open a browser and enter the URL as entered in ODA component services.

```

{
  {
    "name": "login",
    "properties": {
      "mobileNumber": {
        "type": "string",
        "required": true
      }
    },
    "supportedActions": {
      "success": "success",
      "fail": "fail"
    }
  },
  {
    "name": "validateOTP",
    "properties": {
      "otp": {
        "type": "string",
        "required": true
      },
      "mobileNumber": {
        "type": "string",
        "required": true
      },
      "tokenRefId": {
        "type": "string",
        "required": true
      },
      "otpAttempts": {
        "type": "int",
        "required": true
      }
    },
    "supportedActions": {
      "success": "success",
      "fail": "fail"
    }
  },
  {
    "name": "quickBalance",
    "properties": {
      "accountNumber": {
        "type": "string",
        "required": true
      }
    },
    "supportedActions": {}
  },
  {
    "name": "checklogin",
    "properties": {}
  },
  {
    "name": "fetchAccounts",
    "properties": {
      "accountType": {
        "type": "string",
        "required": true
      }
    }
  }
}

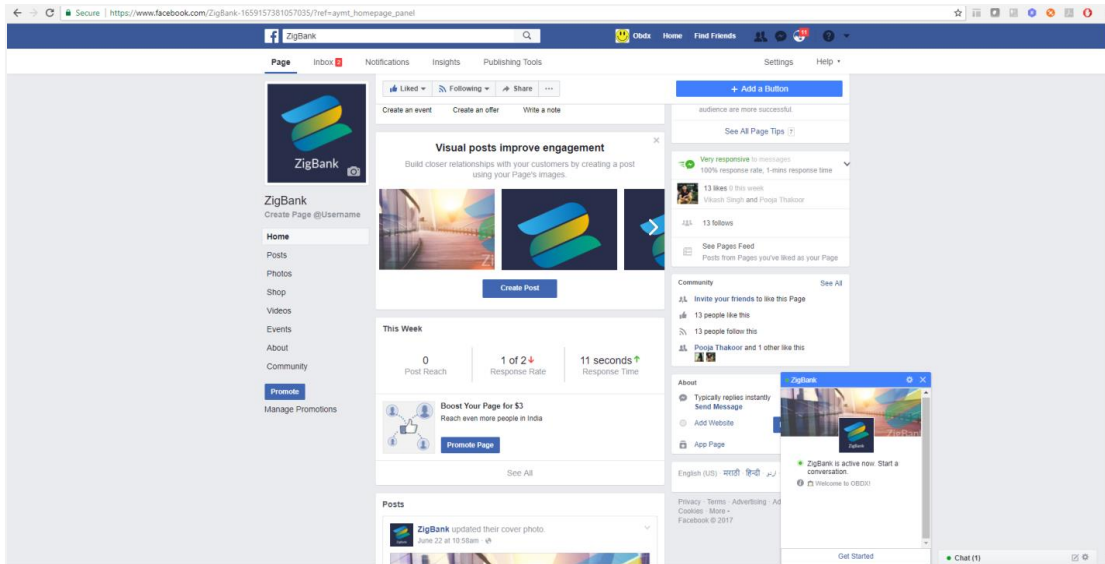
```

## 2. Login to ODA and click OBDXBot > Test

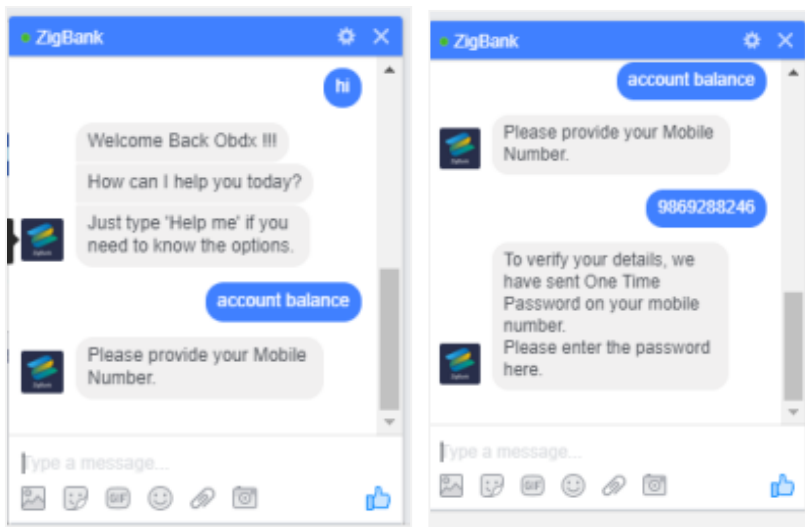


## Enquire about account balance

1. Login to Facebook > Navigate to the page and click > Send message



2. Click Get Started in the chat window > You should receive welcome message from ODA
3. Enquire about account balance > OTP should be received on the registered email address of the party in core banking

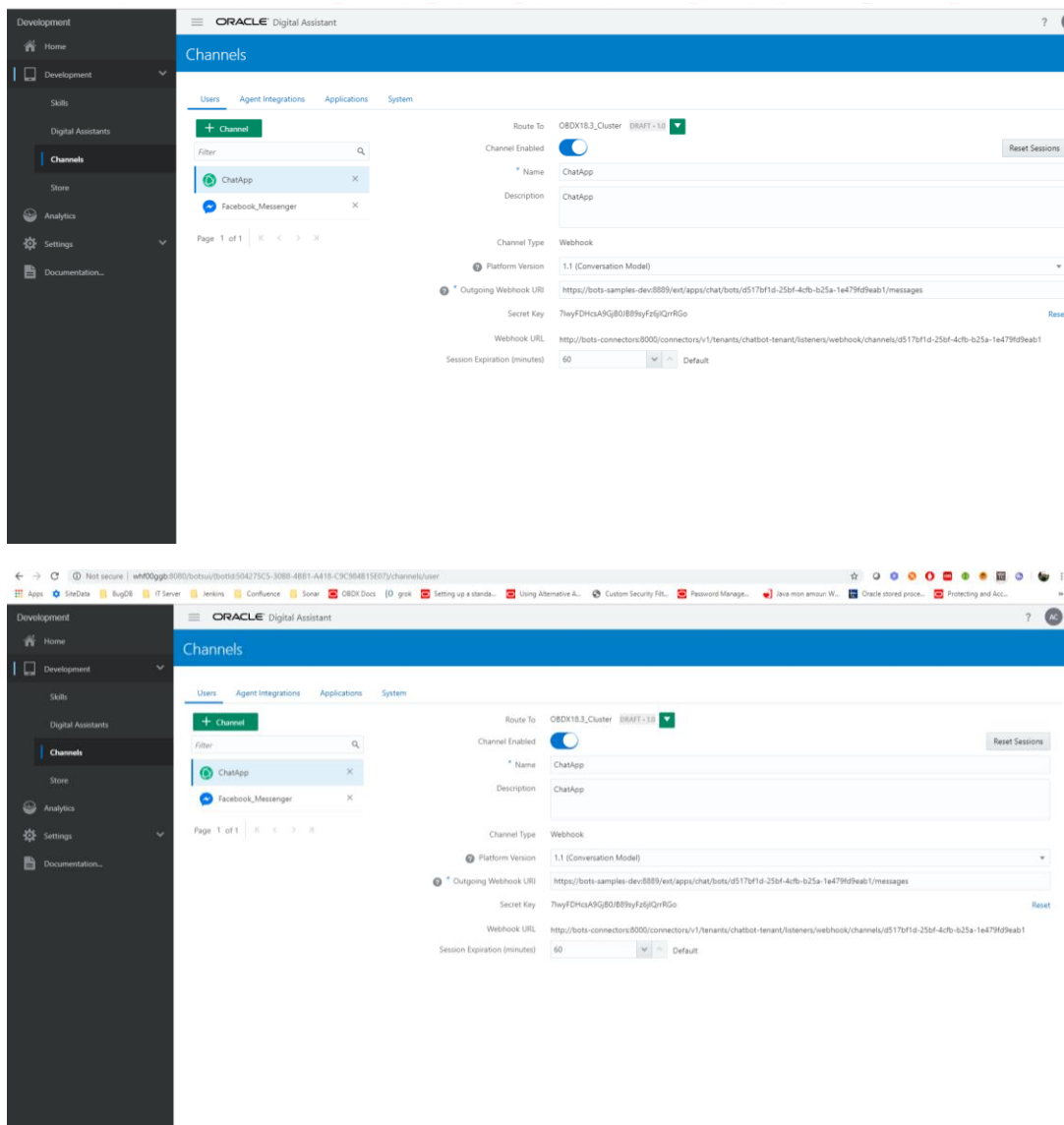


[Home](#)



## 5. Custom Webhook Configuration in ODA

1. In order to set up a channel apart from facebook, you need to add a custom webhook
2. Navigate to botsui bot's add webhook form.
3. Fill form with you details, in "Outgoing Webhook URI" field put: [https://bots-samples-dev.8889/ext/apps/chat/bots/channel\\_id/messages](https://bots-samples-dev.8889/ext/apps/chat/bots/channel_id/messages) and hit create button. Please note that bots-samples-dev is an internal identifier pointing to the samples server within bots. If you host your own chatserver with the webhook endpoint, please use it instead.
4. Replace "channel\_id" in "Outgoing Webhook URI" field with last parameter in url from "Webhook URL" field.



5. Open 'Admin UI' (e.g <http://ODA host:8888/ext/apps/chat/admin/>) and create new channel with "Secret Key" and "Webhook URL" from web hook form in bots UI.

Oracle Bot Cloud Service

Channels

+ New Channel

ChatApp

Name

ChatApp

Description

ChatApp

Secret Key

7heyFOrcuASG80UB99yFz6pQzRGo

Webhook URL

http://bots-connectors.8000/connectors/v1/tenants/chatbot-tenant/listeners/webhook/channels/0517bf18-238f-4c7b-827a-1e479595ea61

Save Remove

[Home](#)

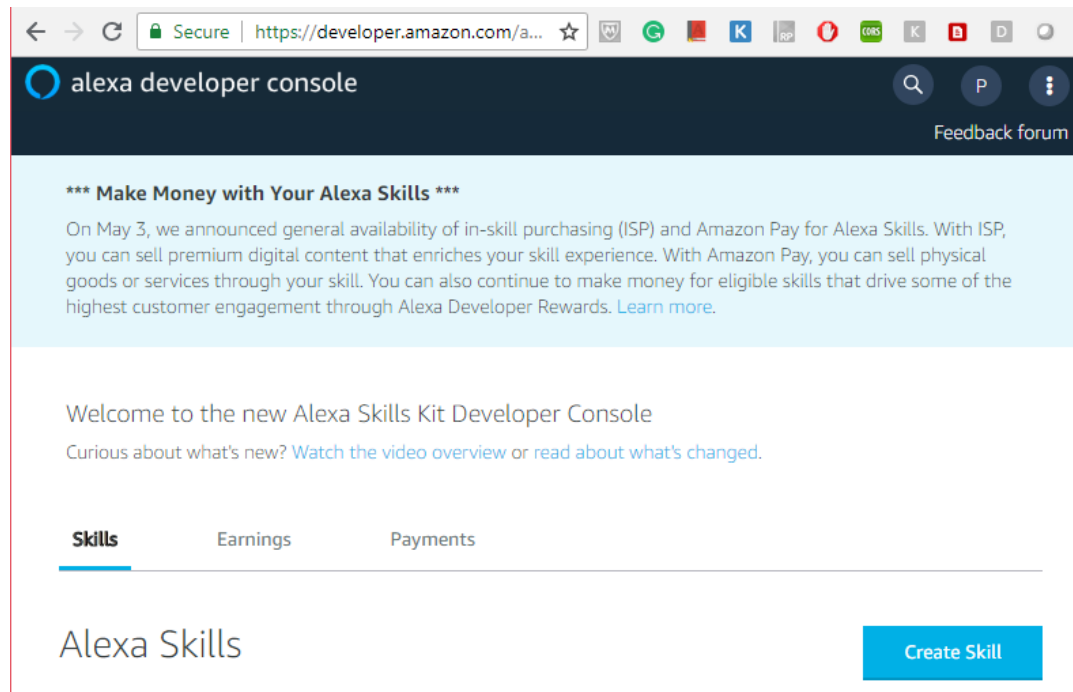
## 6. Alexa Skill (Zig Bank) Configuration

Creating an Alexa skill called *ZigBank* along with a Webhook channel lets you chat with a specific bot.

Add the skill information

Set up a developer account in the Amazon Developer Portal.

1. Open the [Amazon Developer Console](#).
2. Click on **'Create Skill'**



3. Enter ZigBot (or any name that you want to use to invoke this skill) as the Invocation Name.

### 6.1 Define the Interaction Model

1. Next, add the CommandBot intent, which sends a voice text to the configured bot. **Copy and Paste** following intent schema into the Developer Console's JSON Editor and then click on **'Save Model'**

```
{
  "interactionModel": {
    "languageModel": {
      "invocationName": "zigbank",
      "intents": [
        {
          "name": "CommandBot",
```

```

    "slots": [
      {
        "name": "command",
        "type": "CUSTOM_SLOT"
      },
      {
        "name": "amount",
        "type": "AMAZON.NUMBER"
      },
      {
        "name": "payee",
        "type": "AMAZON.Person"
      },
      {
        "name": "CURRENCY",
        "type": "CURRENCY_LIST"
      }
    ],
    "samples": [
      "{amount} {CURRENCY}",
      "{command}",
      "account ending with {command}",
      "anything {command}",
      "do something {command}"
    ]
  },
  {
    "name": "AMAZON.StopIntent",
    "samples": [
      "ok bye"
    ]
  },
  {
    "name": "AMAZON.NavigateHomeIntent",
    "samples": []
  }
],
"types": [
  {
    "name": "CUSTOM_SLOT",
    "values": [
      {
        "name": {
          "value": "0012",
          "synonyms": [
            "zero zero one two"
          ]
        }
      }
    ]
  }
]

```

```

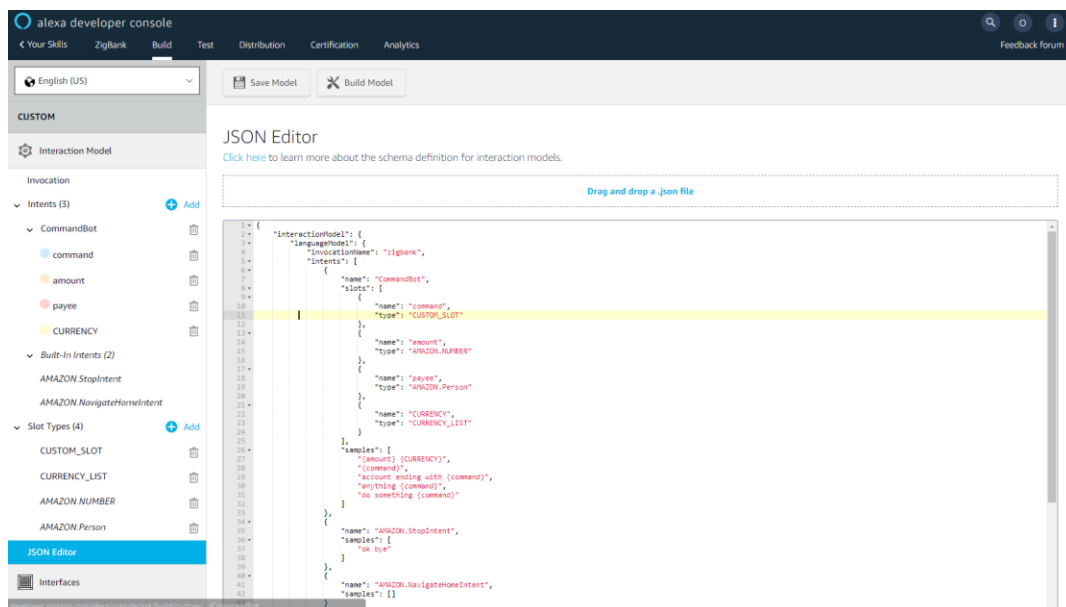
    },
    {
      "name": {
        "value": "0045",
        "synonyms": [
          "zero zero four five"
        ]
      }
    },
    {
      "name": {
        "value": "VODAFONE 4G",
        "synonyms": [
          "vodafone"
        ]
      }
    },
    {
      "name": {
        "value": "AIRTEL ",
        "synonyms": [
          "AIRTEL BROADBAND",
          "bharti airtel"
        ]
      }
    },
    {
      "name": {
        "value": "RELIANCE",
        "synonyms": [
          "REL"
        ]
      }
    }
  ],
  {
    "name": "CURRENCY_LIST",
    "values": [
      {
        "name": {
          "value": "GBP",
          "synonyms": [
            "gbp"
          ]
        }
      }
    ]
  },
  {

```

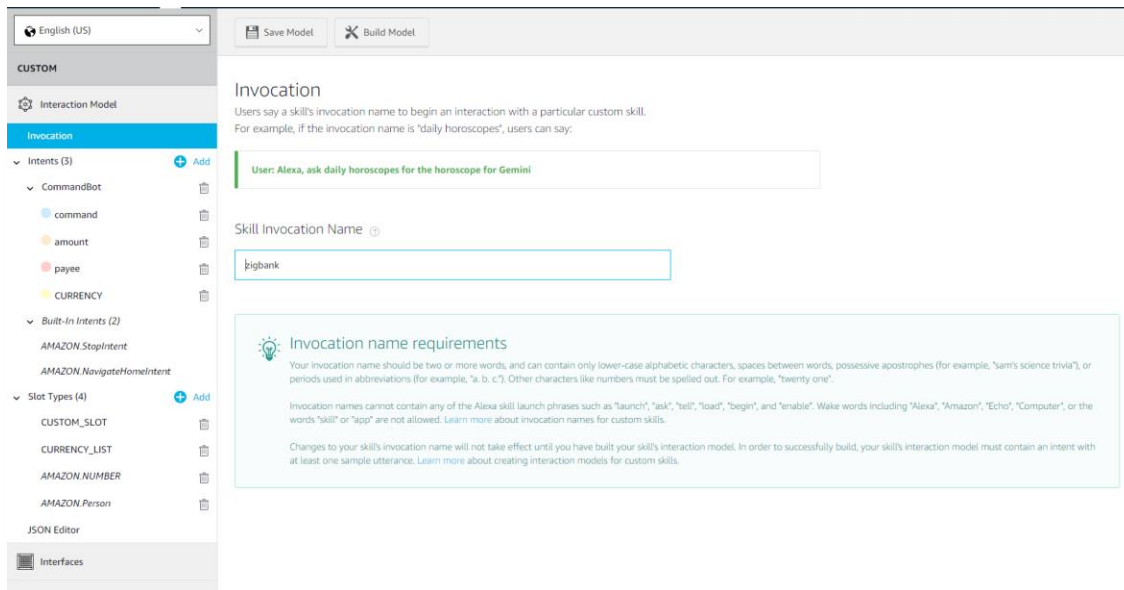
```

    "name": {
      "value": "EURO",
      "synonyms": [
        "euro"
      ]
    }
  ]
}

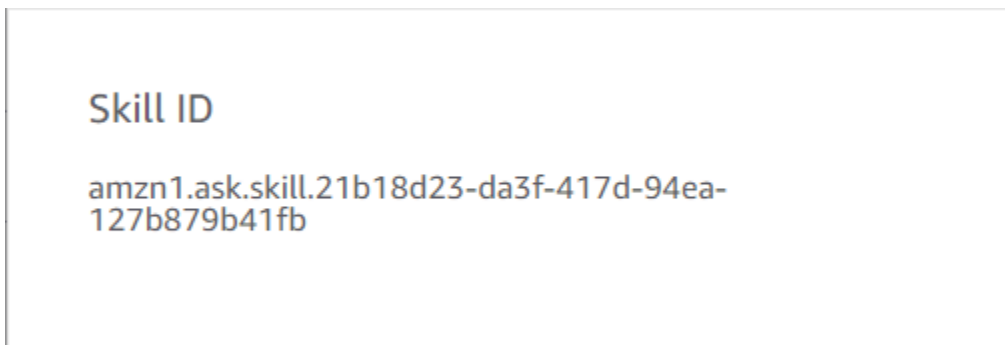
```



2. Click on '**Build Model**'

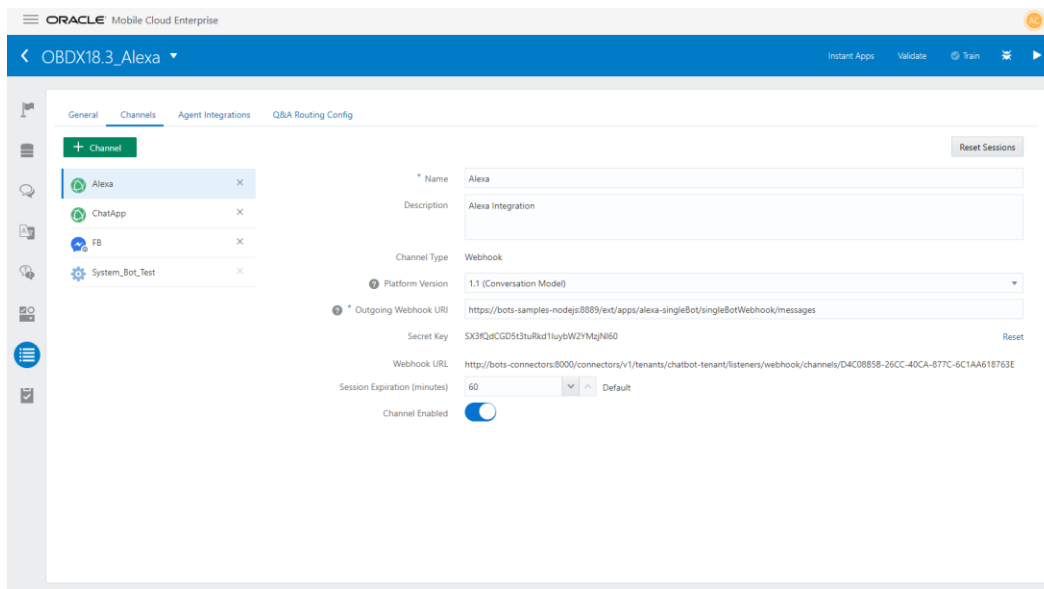
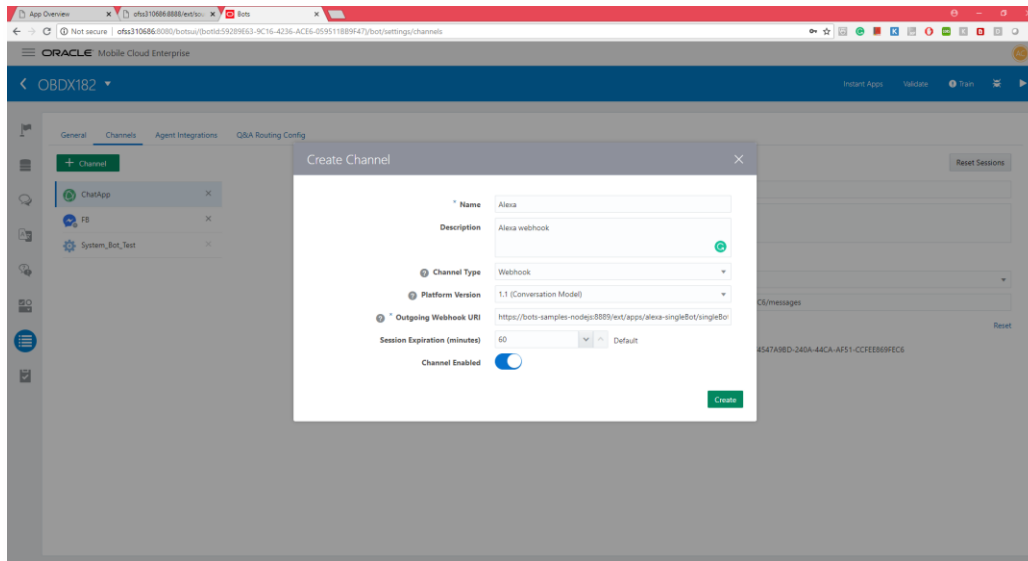


- Also note down the alexa skill id, we will need it in next step.



## 6.2 Create a Webhook channel

1. In the Bot Builder, create a webhook channel for your bot. In the Create Channel dialog, enter the outgoing Webhook URL as `https://bots-samples-nodejs:8889/ext/apps/alexa-singleBot/singleBotWebhook/messages`. This URL is where your bot will send its responses back to the Alexa ZigBot skill.



2. Keep the Secret Key and Webhook URL close by because you need to add them to the `app.js` file. Also, remember to set the amazon skill id (created in previous steps). For example:



```
var metadata = {
  allowConfigUpdate: true,
  waitForMoreResponsesMs: 200,
  amzn_appId: "amzn1.ask.skill.21b18d23-da3f-417d-94ea-127b879b41fb",
  channelSecretKey: 'SX3fQdCGD5t3tuRkd1luybW2YMzjNI60',
  channelUrl: 'http://bots-connectors:8000/connectors/v1/tenants/chatbot-tenant/listeners/webhook/channels/D4C08B5B-26CC-40CA-877C-6C1AA618763E'
};
```

app.js file is located at BOTS\_HOME/samples/nodejs/build/apps/alexa-singleBot/app.js

```
const alexa = require("alexa-app");
const _ = require("underscore");
const express = require('express');
const bodyParser = require('body-parser');
const PubSub = require('pubsub-js');
const Joi = require('joi');
const MessageModel = require('../bots-js-utils/lib/messageModel/MessageModel.js')(Joi);
const messageModelUtil = require('../bots-js-utils/lib/messageModel/messageModelUtil.js');
const botUtil = require('../bots-js-utils/lib/util/botUtil.js');
const webhookUtil = require('../bots-js-utils/lib/webhook/webhookUtil.js');

PubSub.immediateExceptions = true;

module.exports = function() {
  var self = this;

  //replace these settings to point to your webhook channel
  var metadata = {
    allowConfigUpdate: true, //set to false to turn off REST endpoint of allowing update of metadata
    waitForMoreResponsesMs: 200, //milliseconds to wait for additional webhook responses
    amzn_appId: "amzn1.ask.skill.21b18d23-da3f-417d-94ea-127b879b41fb",
    channelSecretKey: 'SX3fQdCGD5t3tuRkd1luybW2YMzjNI60',
    channelUrl: 'http://bots-connectors:8000/connectors/v1/tenants/chatbot-tenant/listeners/webhook/channels/D4C08B5B-26CC-40CA-877C-6C1AA618763E'
  };

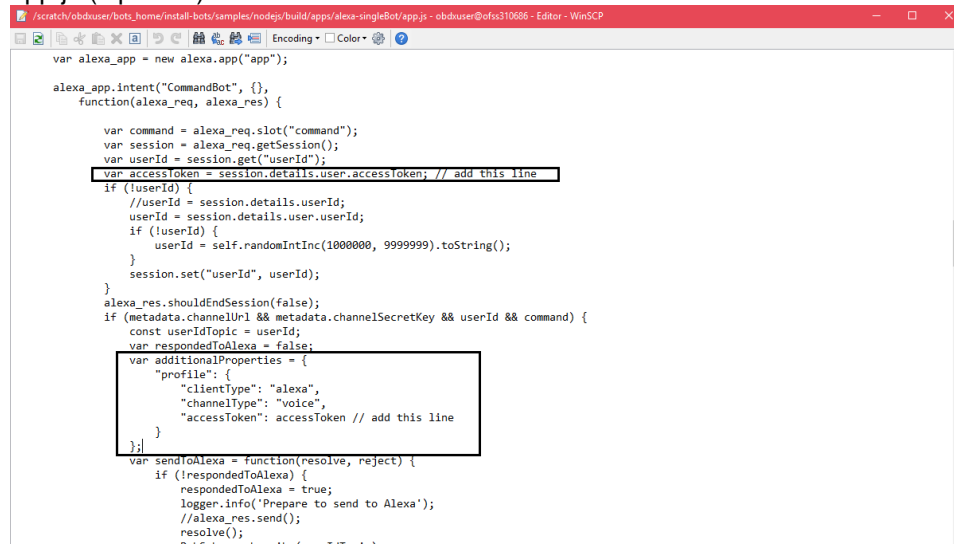
  this.randomIntInc = function(low, high) {
    return Math.floor(Math.random() * (high - low + 1) + low);
  };

  this.setConfig = function(config) {
    metadata = _.extend(metadata, _.pick(config, _.keys(metadata)));
  }
}
```

3. You can also edit the welcome message in the app.js.

```
alexapp.launch(function (alexa_req, alexa_res) {
  var session = alexa_req.getSession();
  session.set("startTime", Date.now());
  alexa_res.say("Welcome to Zig Bank. How may I help you?");
  alexa_res.shouldEndSession(false);
});
```

4. To pass accessToken to the OBDX Chatbot endpoint, add these additional properties in app.js (Optional)



```

var alexa_app = new alexa.app("app");

alexa_app.intent("CommandBot", {},
function(alexa_req, alexa_res) {

    var command = alexa_req.slot("command");
    var session = alexa_req.getSession();
    var userId = session.get("userId");
    var accessToken = session.details.user.accessToken; // add this line
    if (!userId) {
        //userId = session.details.userId;
        userId = session.details.user.userId;
        if (!userId) {
            userId = self.randomIntInc(1000000, 9999999).toString();
        }
        session.set("userId", userId);
    }
    alexa_res.shouldEndSession(false);
    if (metadata.channelUrl && metadata.channelSecretKey && userId && command) {
        const userIdTopic = userId;
        var respondedToAlexa = false;
        var additionalProperties = {
            "profile": {
                "clientType": "alexa",
                "channelType": "voice",
                "accessToken": accessToken // add this line
            }
        };
        var sendToAlexa = function(resolve, reject) {
            if (!respondedToAlexa) {
                respondedToAlexa = true;
                logger.info('Prepare to send to Alexa');
                //alexa_res.send();
                resolve();
            }
        };
    }
}

```

5. Restart the `bots-samples-nodejs` container.

## 6.3 Configure the Endpoint

1. Choose **HTTPS**
2. Enter the HTTPS ngrok URL for port 8888 that's appended with `ext/apps/alexa-singleBot/alexa/app`. For example: `https://<ngrok URL for port 8888>/ext/apps/alexa-singleBot/alexa/app`
3. Select SSL Certificate. Choose **'My development endpoint is a sub-domain of a domain that has a wildcard certificate from a certificate authority'**.
4. Save Endpoint

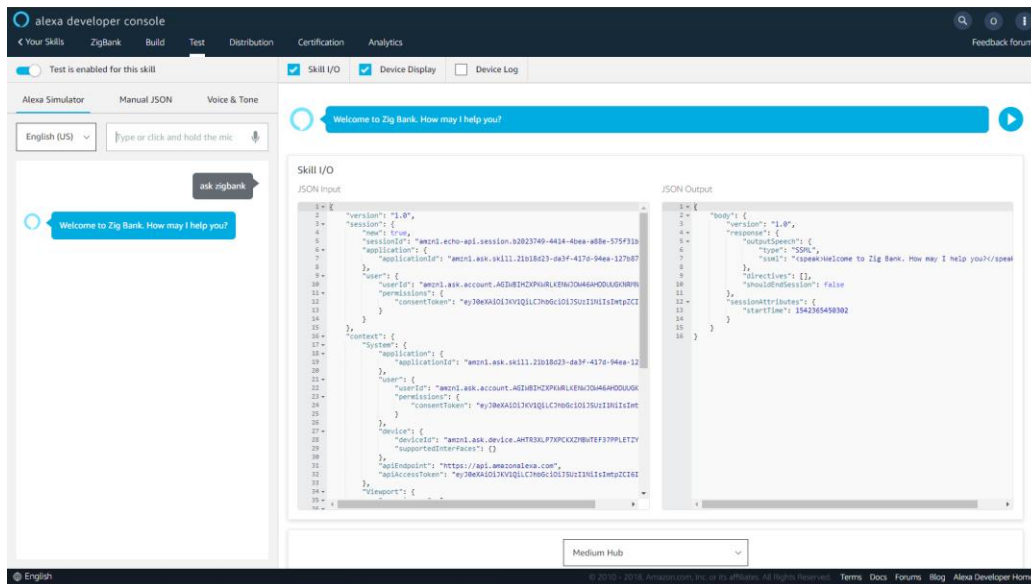
The screenshot shows the 'Endpoint' configuration page in the Amazon Developer Console. The left sidebar has a navigation menu with 'Endpoint' selected. The main area is titled 'Endpoint' and contains a 'Service Endpoint Type' section. Under 'Service Endpoint Type', 'HTTPS' is selected. The 'Default Region (Required)' is set to 'us-east-1'. The 'URL' field contains 'https://fcb67185.ngrok.io/ext/apps/alexa-singleBot/alexa/app'. The 'SSL certificate type' dropdown is set to 'My development endpoint is a sub-domain of a domain that has a wildcard certificate from...'. Below this, there are sections for 'North America (Optional)', 'Europe and India (Optional)', and 'Far East (Optional)', each with a 'URL' field and an 'SSL certificate type' dropdown.

### 6.3.1 Testing the singleBot Skill in the Amazon Developer Console

To test the skill, enter the following utterance in the Service Simulator

- ask zigbank

For each utterance, the Service Simulator window displays the response.



### 6.3.2 Testing ZigBot on an Echo Device

If your Echo device is logged to the same user account that accesses the Developer Console, then the ZigBot skill will be enabled in your Amazon Echo. Try out the same utterances, but start each one with "Alexa ask zibo..." If Alexa can't understand you, or your Echo's light ring is turned off, start over by saying, "Alexa ask zibo..." For example:

"Alexa ask zibo to show my balances"

==> "ssml": "<speak>For which account do you want your balance... </speak>"

"Alexa stop" or "stop" to end the interaction

If Alexa continually misunderstands you, take a look at the `bots-samples-nodejs` log to see which of your commands were picked up by Alexa and sent to the bot and which weren't. (docker logs <container id of bots-samples-nodejs>)

---

**Note:** Alexa might have trouble with some interactions, like following a web link or saying a number. Alexa spells out numbers (22 becomes twenty-two), which might be problematic if your bot is expecting a cardinal number. Also, Alexa won't wait for a result when web services are slow. When this happens, Alexa will state that your skill takes too long to respond.

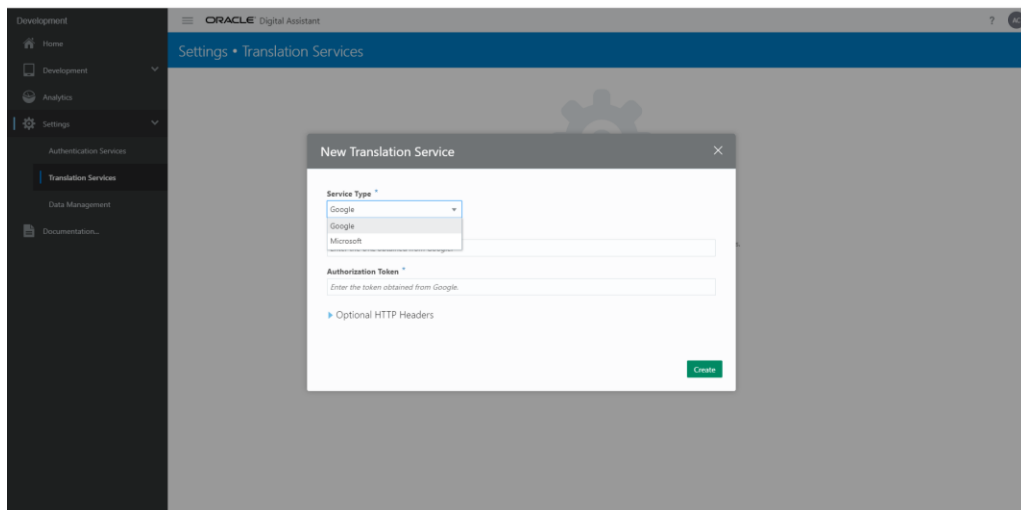
---

### 6.3.3 Configure Account Linking

Account linking lets you connect the identity of the user with a user's account in OBDX system.

1. Click on **Account Linking**
2. Select **Auth Code Grant**
3. Enter Authorization URL `https://<ngrok URL of OHS server port>/digx-auth/oauth2/authz`
4. Enter Access Token URL `https://<ngrok URL of OHS for port>/digx-auth/v1/token`
5. Enter client ID and client secret
6. Select Credentials in request body
7. Add Scope `OBDXVoiceAstServer.SC02` and `OBDXVoiceAstServer.SC05`(SC02 is default scope in case if no scopes are externally added.)
8. Click **Save**

### 6.3.4 Translation Services



In case input language to chatbot is other than english, configure translation services as shown above.