Oracle Banking Microservices Platform Foundation Installation Guide

Oracle Banking

Release 14.5.0.10.0

Part Number F90623-01

November 2023

ORACLE®

Table of Contents

ORACLE®

ORACLE

1. Preface

1.1 Introduction

This guide helps you to install the Oracle Banking Microservices Architecture services on designated environment. It is assumed that all the prior setup is already done related with WebLogic 12c installation, WebLogic managed server creation and Oracle DB installation.

It is recommended to use dedicated managed server for each of the Oracle Banking Microservices Architecture services.

1.2 Audience

This document is intended for WebLogic admin or ops-web team who are responsible for installing the OFSS banking products.

1.3 Documentation Accessibility

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

1.4 Related Documents

The Related document list are as follows:

- Oracle Banking Microservices Architecture Installation Guides

- Product Installation Guides

1.5 Organization

This installation user guide would allow you to install following services in same order:

- WebLogic system environment settings

- Plato Config Service

- Plato Discovery Service

- Plato Apigateway Router

- Plato UI Config Service

- Plato O (Conductor)

- Plato Orch Service

- Plato Feed Services

- Plato Batch Server

- Plato Alerts Management Services

- Security configuration and tool installation

- Plato Rules

- Plato Reports Services

ORACLE

2. Database Setup

2.1 Introduction

In this section you are going to setup database related configuration for Oracle Banking Microservices Architecture Installation. Before you proceed ensure pre-installation setup is done.

2.2 Pre-requisite

Before you proceed with the document, ensure Schemas are being created. It is recommended to have different schema for **Plato** and **Plato Security**. To configure Plato security refer Security Configuration chapter. Make sure that the schema user has the below rights:

DB OBJECT	OPERATION					
	CREATE	**ALTER**	**DROP**	**INSERT**	**UPDATE**	**DELETE**
TABLE	Y	Y	N	Y	Y	Y
VIEW	NA	NA	NA	NA	NA	NA
SEQUENCE	Y	Y	Y	NA	NA	NA
PACKAGE	NA	NA	NA	NA	NA	NA
PACKAGE BODY	NA	NA	NA	NA	NA	NA
INDEX	Y	Y	Y	NA	NA	NA
SYNONYM	NA	NA	NA	NA	NA	NA
FUNCTION	NA	NA	NA	NA	NA	NA
TRIGGER	NA	NA	NA	NA	NA	NA
TYPE	NA	NA	NA	NA	NA	NA

To know the server port number, refer to **How to check port number** section in ANNEXURE-1.

Ensure to configure Placeholder parameters in Weblogic server for Plato Config service, setDomain.env. To know more, refer to **Place Holder update for Plato-Config-Services** section in ANNEXURE-1.

ORACLE®

3.   Domain and Cluster Configuration

3.1   Oracle Banking Microservices Architecture Domain Configuration

3.1.1   Prerequisites

- Machine should have Java has installed.

- Oracle Fusion Middleware has to be installed on the machine.

    NOTE: For the exact version to be installed, refer to **Software Pre-requisites** section in **License Guide**.

3.1.2   Domain Creation and Configuration

It is recommended to have different managed server in one domain for each application. For Creating Domain and Configuration, refer to **How to create and Cluster Configuration** in ANNEXURE-1.

ORACLE®

4. Data Sources Creation

4.1 Prerequisite

Before you proceed with Data source creation, make sure Domain and cluster configuration steps completed.

4.2 Data sources List

The table below lists the data sources to be created on each managed server prior to deployment of applications onto managed servers.

Data source Name	Data source JNDI	Targets
PLATO	jdbc/PLATO	Config Server, API Gateway Server, Plato Feed Server, Plato-Alerts-Management-Server,Plato-Batch-Server, Appshell Server
PLATOSEC	jdbc/PLATO_SECURITY	Config Server, API Gateway Server
PLATO_UI	jdbc/PLATO_UI_CONFIG	Plato UI Config Server, Appshell Server
CONDUCTOR	jdbc/PLATO-O	Plato-O, Plato Orch Server
PLATOFEED	According to the JNDI created for each entity, for DEFAULTENTITY, the JNDI should be jdbc/PLATOFEED	Plato-Feed-Server
PLATOALERTS	According to the JNDI created for each entity, for DEFAULTENTITY, the JNDI should be jdbc/PLATOALERTS	Plato-Alerts-Management-Server
PLATOBATCH	According to the JNDI created for each entity, for DEFAULTENTITY, the JNDI should be jdbc/PLATOBATCH	Plato-Batch-server
PLATORULE	According to the JNDI created for each entity, for DEFAULTENTITY, the JNDI should be jdbc/PLATORULE	Plato-Rules-Server
REPORTSERVICE	According to the JNDI created for each entity, for DEFAULTENTITY, the JNDI should be jdbc/ jdbc/REPORTSERVICE	Plato-Report-Server

For creating data source, refer to **How to create Data sources section** in ANNEXURE-1.

ORACLE

5. Security Configuration and Tools Installation

5.1 Pre-requisite

Before you proceed, do the following steps:

- In case you are planning to use LDAP for web application authentication with Weblogic as provider for LDAP. Please first go through the steps of Embedded Weblogic setup steps in ANNEXURE 1.

- In case you are planning to use OAuth without OAM(i.e. Spring OAuth), do the following **change in Weblogic configuration:**
  In the config.xml file of the concerned domain in Weblogic add the following script at the end of **security-configuration** tag (Just before the line **</security-configuration>)**

```
<enforce-valid-basic-auth-credentials>false</enforce-valid-
basic-auth-credentials>
```

To use the Standard LDAP directory authentication for Online Web Application authentication, make sure LDAP server details is provided to you:

Like LDAP_URL, USER_STORE, LDAP_SERVER_CREDENTIAL_SALT, LDAP_SERVER_USER, LDAP_SERVER_BASE, LDAP_SERVER_CREDENTIAL, LDAP_USER_SEARCH_BASE, LDAP_USER_PREFIX, CORS_ALLOWED_ORGINS, LDAP_SERVER_CREDENTIAL_SALT etc.

5.2 Plato Security JWT

Plato security module enables securing API micro services with JWT (JSON Web Tokens).

JSON Web Tokens are an open, industry standard RFC 7519 method for representing claims securely between two parties. JSON Web Token (JWT) is a compact, URL-safe means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is used as the payload of a JSON Web Signature (JWS) structure or as the plaintext of a JSON Web Encryption (JWE) structure, enabling the claims to be digitally signed.

5.3 Plato Security Configuration (Online Web Application Authentication)

Oracle Banking Microservices Architecture recommend to create new schema for security to keep the security related database objects at one place. If the environment is configured for multi-tenant, we require a security schema per tenant.

All the Plato security configurations are maintained at SECURITY_CONFIG table

Steps to configure in the table:

1. Change in case of **LDAP directory authentication** the below KEY with provided LDAP details:

KEY	VALUE
LDAP_SERVER_CREDENTIAL_SALT	Enter LDAP server Credential salt e.g. 0.9482628451234567
CORS_ALLOWED_ORGINS	valid host names (comma delimited)

KEY	VALUE
LDAP_URL	Enter LDAP Server URL Example: ldap://wxy00abc:9001
LDAP_SERVER_USER	Enter LDAP Server USERID Example: uid=admin
LDAP_SERVER_BASE	Enter LDAP server BASE Example: dc=oracle,dc=com
LDAP_SERVER_CREDENTIAL	Enter LDAP server encrypted password using provided jwt algorithm Example: m0o/F3UvlwvBSv5C/TSckA== (use plato encryption utlity to generate encrypted password)
LDAP_USER_SEARCH_BASE	Enter LDAP User search Base Example: ou=people
LDAP_USER_PREFIX	Enter LDAP User Prefix Example: uid

2. Change in case of **SSO Agent** the below KEY with provided LDAP details**:**

KEY	VALUE
IS_SSO_CONFIGURED	True
CORS_ALLOWED_ORGINS	valid host names (comma delimited)

5.4 <u>User Store</u>

Oracle Banking Microservices Architecture supports following user stores for authentication Users Maintained at table. Plato security can authenticate the users maintained at table (APP_USER) in the security schema. However, we do not recommend to use this option.

ORACLE®

6. Deployments

6.1 Pre-requisite

Before you proceed with below, make sure previous steps are completed.

6.2 Deployment Order

Plato Infrastructure Software → Plato → SMS → Common Core → Products
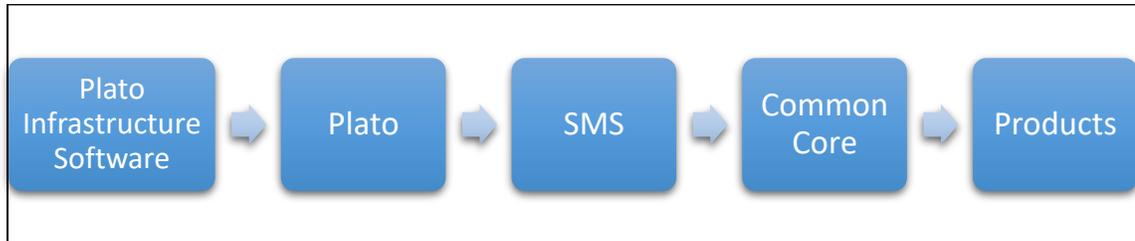
6.3 Flyway configuration

The following parameters have to be added as jvm arguments for controlling flyway:

1. flyway.enabled = <Boolean>. If true, flyway will get executed. If false, flyway wouldn't be executed

2. spring.flyway.enabled=false (always).

ORACLE®

7.   Multi Entity Configuration

7.1   Enable Multi Entity

By Default, Multi Entity is disabled.
To enable Multi Entity, add jvm argument as -Dmulti.entity.enabled=true.

7.2   Default Entity Creation

Default entity creation is described as follows:

- A new column ENTITY_ID will be introduced in the APPLICATION_LEDGER table in the PLATO schema with default value as "*DEFAULTENTITY*". This will get executed as a part of flyway for plato-config-service-{version}.war.

- A new table "SERVICE_REGISTRY" will be introduced in the PLATO schema. This table will contain the AppId and microservice name of all the microservices. This will get executed as a part of flyway for plato-config-service-{version}.war.

- A new table "PLATO_TM_ENTITY" will be introduced in the PLATO SECURITY schema with a single entry for "DEFAULTENTITY". This will get executed as a part of the flyway scripts for plato-api-gateway-{version}.war

- A new table "PLATO_TM_USER_ENTITY_MAPPING" will be introduced in the PLATO SECURITY schema which will also get executed as part of the flyway scripts for plato-api-gateway-{version}.war

- **Only for Existing Customers** - For users already maintained in SMS, users must be replicated to "PLATO_TM_USER_ENTITY_MAPPING" for the "*DEFAULTENTITY*".

- The sample query is as follows:
  INSERT INTO *PLATO_TM_USER_ENTITY_MAPPING (ID, USER_ID , ENTITY_ID ,HOME_ENTITY,MULTI_ENTITY_ADMIN,USER_NAME,ENTITY_ADMIN,EMAIL,START_DATE,END_DATE*)
  SELECT *ID , USER_LOGIN_ID ,'DEFAULTENTITY' ,'Y','N',USER_NAME,'N',USER_EMAIL,START_DATE,END_DATE* FROM *PLATOSMS.SMS_TM_USER*
  PLATOSMS – SMS schema for the DEFAULTENTITY

- If the customer wishes to change the default entity ID, it can be done by changing the ENTITY_ID column value in the PLATO_TM_ENTITY , APPLICATION_LEDGER & PLATO_TM_USER_ENTITY_MAPPING table. It is considered that the entity schemas are same and only entity ID is changed.

ORACLE

7.3 Multi Entity Admin User Creation

Perform the following steps

1. Execute the scripts at the below path in the OSDC zip. You will be prompted to enter the multi entity admin user Id and language code.

   - For new users :
     <<PRODUCT_NAME>>_INITIAL_SETUP\INSTALL\PLATO\INS_PLATO_SEC_ME_ADMIN_USER_CREATION_14.5.0.0.0.sql

   - For existing users :
     <<PRODUCT_NAME>>_INITIAL_SETUP\UPGRADE\PLATO\INS_PLATO_SEC_ME_ADMIN_USER_CREATION_14.5.0.0.0.sql.

2. Create the multi entity admin user in the LDAP.

7.4 Entity Creation

Using the Multi entity admin created in the previous step, log in to the app-shell and create the entities.

NOTE: Refer to Common Core User Guide for the steps to create an entity.

When the multi entity admin create an entity the following processes will execute in the background:

- The entity details will be saved in the PLATO_TM_ENTITY table.

- The JNDIs will be saved in the APPLICATION_LEDGER table.

- The flyway scripts for all the micro services will get executed in their respective schemas.

- Once the flyway execution is completed a new role "ENTITY_ADMIN" will be created in the entity. This step will insert scripts into the following tables:

  - SMS_TM_ROLE

  - SMS_TW_ROLE

  - SMS_TM_ROLE_ACTIVITY

  - SMS_TW_ROLE_ACTIVITY

  This role will be assigned to the entity admin user in the user creation step.

- The Head Office branch details will be inserted into the CMC_TM_CORE_BRANCH and CMC_TW_CORE_BRANCH tables.

- The Bank details will be inserted into the CMC_TM_CORE_BANK and CMC_TW_CORE_BANK tables.

- The System dates will be inserted into the CMC_TM_SYSTEM_DATES and CMC_TW_SYSTEM_DATES tables.

7.5 <u>User Creation</u>

Make sure that the entity creation step is complete before proceeding to create users.

- Create the users in the LDAP.

- Multi entity admin must login to the app-shell and create entity admins and users.

  NOTE: Refer to Common Core User Guide for the steps to create users.

- The entity admins and user details will be stored in the PLATO_TM_USER_MAPPING table in the security schema.

- For the entity admins scripts will be executed in the SMS schema in the following tables to assign the ENTITY_ADMIN role to the entity admin users.

  - sms_tm_user

  - sms_tw_user

  - sms_tm_user_role_branch

  - sms_tw_user_role_branch

  - sms_tm_user_application

  - sms_tw_user_application

- The entity admins need to log in to the app-shell, and provide the missing user details, assign roles and branches to users.

ORACLE

8.   Plato Orchestration Services

8.1 Migration Endpoint

NOTE: This section is applicable only to the existing customers.

The task blob usage has been removed for GET endpoints in *plato-orch-service* for task list screens. The table *HTASK_ADDN_DTLS* contains the task related details. A migration endpoint needs to be executed to populate the data for the completed tasks in this table. In Progress tasks data will be automatically populated by the poller. This will improve the performance in Free Tasks/My Tasks/Completed Tasks/Supervisor Tasks inquiry.

To populate the table *HTASK_ADDN_DTLS* with previously COMPLETED tasks (for tasks not present in *task_in_progress* table), a migration API needs to be executed.

GET Request:

Endpoint: http://<host>:<port>/plato-orch-service/api/v1/extn/migrate

Headers: Sample inputs shown below.

appId: platoorch

branchCode: 000

Content-Type: application/json

entityId: DEFAULTENTITY

To verify if the **HTASK_ADDN_DTLS** table entries are consistent with others, execute the following script and check if the count comes as zero.

SELECT COUNT(*) FROM TASK t
WHERE JSON_VALUE(json_data, '$.status') = 'COMPLETED'
AND JSON_VALUE (json_data, '$.taskType') = 'WAIT'
AND TASK_ID NOT IN (SELECT TASK_ID FROM HTASK_ADDN_DTLS);

NOTE: For future tasks and previous non-completed tasks present in *task_in_progress* table, poller keeps checking the *task_in_progress* table, and populates the *HTASK_ADDN_DTLS* table.

9. Plato Infrastructure Software Deployment

Once everything is deployed, the managed servers. For each application, call path "/refresh" for refreshing the configuration properties.

9.1 Zookeeper Cluster Setup

To restart the server, refer to.**How to restart** section in ANNEXURE-1.

9.1.1 Pre-requisite

JDK should be installed in all node machines.

Download kafka_2.13-3.4.0 and extract the binary in all node machines. Kafka can be found at <Unzip the file>/THIRD_PARTY_SOFTWARES/KAFKA/ARCHIVE.

NOTE: Please note that the zookeeper that we will be using is bundled within Kafka.

9.1.2 Installation

- Untar/unzip the kafka binary and move them into a folder which will be the kafka home directory.

- Create two directories named logs and data inside the kafka home directory folder in all the nodes with appropriate permission. If logs folder is already present, please clear it.

- Inside the <kafka home directory>/data folder create a myid file. The myid file consists of a single line containing only the text of that machine's id. So myid of server 1 would contain the text "1" and nothing else. The id must be unique within the ensemble and should have a value between 1 and 255.

- Edit the configuration file named zookeeper.properties at <kafka home directory> /kafka_2.13-3.4.0/config

  Add the following set of properties and values to that file:

```
dataDir= <kafka home directory>/data
tickTime=2000
clientPort= Zookeeper client Port value (2181)
initLimit=10
syncLimit=5

server.1=<hostname> :< peer port> :< leader port>
#1 is the id that we put in myid file.

server.2= <hostname> :< peer port> :< leader port>
#2 is the id that we will put in myid file of second
node.

server.3=<hostname> :< peer port> :< leader port>
#3 is the id that we will put in myid file of third
```

NOTE: Any odd number of zookeeper servers can be configured under the cluster. The properties with the key as server.x are only required if you want to have a multiNode kafka setup.

ORACLE

- Start the zookeeper on each node machine

  Navigate to <kafka home directory>/ kafka_2.13-3.4.0 and execute the below command

  ```
  /bin/zookeeper-server-start.sh /config/zookeeper.properties
  ```

- To see who is the leader and followers in the cluster, run the below command on each node

  ```
  echo stat | nc localhost 2181
  ```

- To check the zoo cluster functionting i.e dynamic leader election, kill the zookeeper process on the leader node and check again with the following commands on the remaining live zookeeper node.

  ```
  echo stat | nc localhost 2181
  ```

9.2 Kafka Cluster Setup

9.2.1 Pre-requisite

JDK should be installed in all node machines.

Download Kafka and extract the binary in all node machines. Kafka can be found at <Unzip the file>/THIRD_PARTY_SOFTWARES/KAFKA/ARCHIVE.

9.2.2 Installation

- Untar/unzip the kafka binary and move them into a folder which will be the kafka home directory.

- Create two directories named logs and data inside the kafka home directory folder in all the nodes with appropriate permission. If logs folder is already present, please clear it.

  NOTE: Please skip the above 2 steps if already done during zookeeper setup.

- Edit the below lines in the
  <kafka home directory>/kafka_2.13-3.4.0/config/server.properties.

  ```
  broker.id= (Unique Integer which identifies the kafka broker in the
  cluster.)
  listeners=PLAINTEXT://<hostname>:<Kafka broker listen port(9092)>
  log.dirs=<kafka home directory>/logs
  log.retention.hours= <The number of hours to keep a log file before
  deleting it (in hours),tertiary to log.retention.ms property>
  log.retention.bytes= <The maximum size of the log before deleting it>
  log.segment.bytes= <The maximum size of a single log file>
  log.retention.check.interval.ms= <The frequency in milliseconds that
  the log cleaner checks whether any log is eligible for deletion>
  zookeeper.connect=<zookeeper_hostname_1>:<zookeeper_client_port>,<zook
  eeper_hostname_2>:<zookeeper_client_port>,<zookeeper_hostname_3>:<zook
  ```

ORACLE

- To start the Kafka, navigate to <kafka home directory>/kafka_2.13-3.4.0/ folder and run the below command on each node.

```
export JMX_PORT=[PORT VALUE]

nohup bin/kafka-server-start.sh config/server.properties &
```

The Default value of JMX Port is 9999.
Tail the log for server status.

- To create topic, navigate to <kafka home directory>/kafka_2.13-3.4.0/ folder and run the below command:

```
/bin/kafka-topics.sh --create --bootstrap-server
<hostname>:<client port> --replication-factor 3 --partitions
3 --topic <topic name>
```

- To list the available topic on kafka server, navigate to *<kafka home directory>/kafka_2.13-3.4.0/* folder and run the below command:

```
./bin/kafka-topics.sh --list --bootstrap-server
<hostname>:<client port>
```

- To describe the topic, navigate to *<kafka home directory>/kafka_2.13-3.4.0/* folder and run the below command:

```
./bin/kafka-topics.sh --describe --topic <topic name> --
bootstrap-server <hostname>:<client port>
```

- To start a producer, navigate to *<kafka home directory>/kafka_2.13-3.4.0/* folder and run the below command:

```
export JMX_PORT=[PORT VALUE]//Different Value from the server
JMX port

./bin/kafka-console-producer.sh --broker-list
<hostname>:<port>, <hostname>:<port>, --topic <topic name>
```

By default, port is taken as 9092 for the producer.

- To start a consumer console for viewing the received messages sent by the producer, use the following command:

```
export JMX_PORT=[PORT VALUE]//Different Value from the server
JMX port

./bin/kafka-console-consumer.sh --bootstrap-server
<hostname>:<port>,<hostname>:<port>, --topic <topic_name> --
from-beginning
```

ORACLE

9.3 <u>KAFKA Security Setup</u>

9.3.1 <u>Pre-requisite</u>

JDK should be installed in all node machines.

Download Kafka and extract the binary in all node machines. Kafka can be found at <Unzip the file>/THIRD_PARTY_SOFTWARES/KAFKA/ARCHIVE

9.3.2 <u>Installation</u>

9.3.2.1 <u>Generate Keystore</u>

The items highlighted in bold are placeholders, and should be replaced with suitable values while running the following command.

*keytool -genkeypair -alias **alias** -keyalg **keyalg** -keysize **keysize** -sigalg **sigalg** -validity **valDays***

*-keystore **keystore***

In the above command,

1. alias is used to identify the public and private key pair created.

2. keyalg is the key algorithm used to generate the public and private key pair. The RSA key algorithm is recommended.

3. keysize is the size of the public and private key pairs generated. A key size of 1024 or more is recommended.

4. sigalg is the algorithm used to generate the signature. This algorithm should be compatible with the key algorithm and should be one of the values specified in the Java Cryptography API Specification and Reference.

5. valdays is the number of days for which the certificate is to be considered valid. Please consult with your CA on this period.

6. keystore is used to specify the location of the JKS file. If no JKS file is present in the path provided, one will be created.

The command will prompt for the following attributes of the certificate and keystore:

1. Keystore Password: Specify a password that will be used to access the keystore. This password needs to be specified later, while configuring the identity store in Kafka Server.

2. Key Password: Specify a password that will be used to access the private key stored in the keystore. This password needs to be specified later, while configuring the SSL attributes of the Kafka Server.

3. First and Last Name (CN): Enter the domain name of the machine. For example: www.example.com

4. Name of your Organizational Unit: The name of the department or unit making the request. Use this field to further identify the SSL Certificate you are creating. For example, by department or by physical server.

5. Name of your Organization: The name of the organization making the certificate request, for example, Oracle Financial Services. It is recommended to use the company or organization's formal name, and this name entered here must match the name found in official records.

6. Name of your City or Locality: The city in which your organization is physically located. For example: Bengaluru.

7. Name of your State or Province: The state/province in which your organization is physically located. For example: Karnataka.

8. Two-letter CountryCode for this Unit: The country in which your organization is physically located. For example US, UK, IN etc.

Example:

A sample execution of the command is mentioned below:

keytool -genkeypair -alias certificates -keyalg RSA -keysize 1024 -sigalg SHA512withRSA -validity 365 -keystore /scratch/Data/Certificates/KafkaServerKeystore.jks

Enter keystore password:<Enter a password to protect the keystore>

Re-enter new password:<Confirm the password keyed above>

What is your first and last name?

[Unknown]: <domain name>.oracle.com

What is the name of your organizational unit?

[Unknown]: <application name>

What is the name of your organization? [Unknown]: Oracle Financial Services

What is the name of your City or Locality?

[Unknown]: Bengaluru

What is the name of your State or Province?

[Unknown]: Karnataka

What is the two-letter country code for this unit?

[Unknown]: IN

Is CN= name.oracle.com, OU=Test, O=Oracle Financial Services, L= Bengaluru, ST= Karnataka, C=IN correct? [no]: yes

Enter key password for < password >

RETURN if same as keystore password): <Enter a password to protect the key>

Re-enter new password: <Confirm the password keyed above>

9.3.2.2 Export Private Key as Certificate

The command is given below:

keytool -export -alias <alias_name> -file <export_certificate_file_name_with_location.cer> -

keystore <keystore_name.jks> -keypass <Private key Password> -storepass <Store Password>

Example:

keytool -export -alias certs -file /scratch/Data/Certificates/KafkaCert.cer -keystore /scratch/Data/Certificates/KafkaServerKeystore.jks -keypass oracle123 -storepass oracle123

If successful, the following message is displayed:

Certificate stored in file < KafkaCert.cer>

ORACLE

9.3.2.3 <u>Import the Cert and generate TrustStore</u>

The command is given below:

*keytool -import -alias **alias** -file **cert_file** -keystore **truststore** –storepass **storepass***

In the above command:

1. *alias* is used to identify the public and private key pair. Specify the alias of the key pair used to create the CSR in the earlier step mentioned in section 7.3.1.2.
2. *cert_file* is the location of the file containing the PKCS#7 formatted reply from the CA, containing the signed certificate.
3. *truststore* is the location where the truststore should be generated.
4. *storepass* is the password for the truststore.

Generate two truststores from the same cert. One is used for Kafka server and another is used for clients.

Example:

keytool -import -alias certs -file /scratch/Data/Certificates/KafkaCert.cer –keystore /scratch/Data/Certificates/KafkaServerTrustStore.jks -storepass oracle123

keytool -import -alias certs -file /scratch/Data/Certificates/KafkaCert.cer -keystore /scratch/Data/Certificates/KafkaClientTrustStore.jks -storepass oracle123

Hence, the following three keystore files would be needed for this method:

1. *KafkaServerKeystore.jks* : keystore file for Kafka brokers
2. *KafkaServerTrustStore.jks* : Truststore file for server
3. *KafkaClientTrustStore.jks* : Truststore file for client

To validate the server, each client should import the KafkaClientTrustStore.jks file.

NOTE: The truststore files should be generated using the same CA. Generate and place these files on all the different servers of Kafka so that it can be accessed by server*.properties file. The KafkaClientTrustStore.jks should be placed on the server, which is accessible by the microservices also.

ORACLE®

9.3.2.4 Creation of users in Zookeeper

Follow the below steps to create user in Zookeeper:

1. Start the zookeeper. Refer command in Section *10.1.2*.

2. Execute the below commands for the user creation.

   ./kafka-configs.sh --zookeeper localhost:2181 --alter --add-config "SCRAM-SHA-256=[password=admin-secret],SCRAM-SHA-512=[password=admin-secret]" --entity-type users --entity-name admin

   ./kafka-configs.sh --zookeeper localhost:2181 --alter --add-config "SCRAM-SHA-256=[iterations=8192,password=test-secret],SCRAM-SHA-512=[password=test-secret]" --entity-type users --entity-name test

Two users are created above with user names as test and admin, and two different passwords are setup for each user one for each scram mechanism. Here, the user 'admin' is used for Kafka broker auth and 'test' is used for client auth.

9.3.2.5 Configuring Brokers

Some modifications need to be made in the server.properties file of kafka server. The following properties need to be added in server.properties file of kafka.

SSL-SCRAM Settings

ssl.endpoint.identification.algorithm=

ssl.truststore.location=/scratch/Data/Certificates/KafkaServerTrustStore.jks

ssl.truststore.password=orcl@123

ssl.keystore.location/scratch/Data/Certificates/KafkaServerKeystore.jks

ssl.keystore.password=orcl@123

ssl.key.password=orcl@123

sasl.enabled.mechanisms= SCRAM-SHA-256

sasl.mechanism.inter.broker.protocol= SCRAM-SHA-256

security.inter.broker.protocol=SASL_SSL

listeners=SASL_SSL://whf00phz:9093

advertised.listeners=SASL_SSL://10.40.162.113:9093

listener.name.sasl_ssl.scram-sha-256.sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required username="admin" password="admin-secret";

NOTE: Provide the absolute path of the Kafka Server Truststore and keystore, and its respective passwords. Modify the hostname and IP in the listeners and advertised.listeners properties field accordingly.

Start the Kafka servers. Refer command in Section *10.2.2*.

ORACLE

9.3.2.6 Changes to Clients

These attributes should be available in application.yml of any custom service that connects to SSL/Authentication enabled Kafka broker. Values for these needs to be released to the PROPERTIES table.

Key	Value
spring.cloud.stream.kafka. binder.brokers	<hostname:port>
spring.cloud.stream.kafka. binder.zknodes	<hostname:port>
spring.cloud.stream.kafka. binder.jaas.options.userna me	<Zookeeper user created for clients>
spring.cloud.stream.kafka. binder.jaas.options.passw ord	<Zookeeper user encrypted password for clients>
spring.cloud.stream.kafka. binder.configuration.ssl.tru ststore.location	<location of client trust store certificate>
spring.cloud.stream.kafka. binder.configuration.ssl.tru ststore.password	<Pass code of client truststore certificate>

To encrypt the password, use the following API of plato-config-service:

API: http://hostname:port/config-service/encrypt

Request Type: Text

Request Body: Password

For example, when you hit the above API for the following passwords we get the response of encrypted value:

test-secret : 36c11a239ffafbe229d888e7d21f0508a38a2501fd5592b1fe54e30889dd57ed

While inserting to properties table, append the encrypted values with the keyword {cipher} to get it decrypted by the config-service during fetch as given in below example:

For more information on adding properties to plato-config-deploy.env, refer to the section "Method 3 – Using env files and setUserOverrides.sh file" in ANNEXURE-1.

ORACLE®

9.3.2.7 <u>Important Commands</u>

To view the messages getting sent in Kafka, save the below lines in a file, and name it as ssl.properties.

ssl.truststore.location=/scratch/Data/Certificates/KafkaClientTrustStore.jks

ssl.truststore.password=orcl@123

security.protocol=SASL_SSL

ssl.endpoint.identification.algorithm=

sasl.mechanism=SCRAM-SHA-256

*sasl.jaas.config=org.apache.kafka.common.security.scram.ScramLoginModule required *

 *username="obvam_new" *

 password="obvam-secret";

NOTE: Update the truststore location and the password.

To view the messages being published use the below command:

./kafka-console-consumer.sh --bootstrap-server kafka-server --topic topicName --consumer.config absolute-path-of-consumer-config --from-beginning

Example:

./kafka-console-consumer.sh --bootstrap-server localhost:9092 --topic test_topic -- consumer.config =/scratch/kafka/config/ssl.properties --from-beginning

9.4 <u>Tesseract Installation</u>

9.4.1 <u>Pre-requisite</u>

9.4.1.1 <u>Build Tools</u>

Ensure that the following build tools are available:

- GNU Autotools—autoconf, automake, libtool
- CMake (Optional, we will use this only if autoconf fails while building leptonica)

Both should be available inside Oracle yum.

9.4.1.2 <u>Dependent Libraries</u>

These libraries should be present in the server. By default, these libraries are available in Oracle Linux. If these libraries are not present, please install it through yum with the following command:

```
sudo yum install <LIBRARY_NAME>
```

Following are the library names:

- libjpeg
- libtiff
- zlib
- libjpeg-turbo
- libwebp
- libpng-devel
- libtiff-devel
- libwebp-devel

NOTE: If you are using any distribution other than Oracle Linux, please install the libraries from official Oracle repo or any other repo available for that distribution.

9.4.1.3 <u>Installation Files</u>

Download installation files required to install and set up tesseract. Files are available at the following location:

<Unzip the file>/THIRD_PARTY_SOFTWARES/Tesseract

Please find below the list of files present in the directory:

1. leptonica-1.80.0.tar.gz
2. tesseract-4.1.0.tar.gz
3. eng.traineddata
4. osd.traineddata

ORACLE®

9.4.2 Installation

9.4.2.1 Leptonica Installation:

Tesseract uses leptonica internally for image processing. Leptonica can be build and installed either by autoconf or by CMake.

In this document, we will cover installation using both autoconf and CMake.

NOTE: If you already have all access to all installation directory then sudo is not required.

>sudo LINUX_COMMAND (In case the user does not have file access permissions)

>LINUX_COMMAND (In case the user has all access. Example: DBA user, Root user)

In this document, we will execute all the commands with sudo. You can omit it based upon your user's permission details.

9.4.2.1.1   Installation through Autoconf

- Copy the downloaded leptonica tarball (leptonica-1.76.0.tar.gz) in server (in the installation directory. Ex: /scratch)
- Execute below commands sequentially to install leptonica through autoconf

NOTE: In line 4, we used **sudo make -j4**. Here 4 is the number of CPU core. Generally, you can use **sudo make -jn** where n is the number of core. It will make the build process much faster.

In the document, we will use 4 as core number to build the software.

If the processor does not have multiple cores you can use normal make command **sudo make**.

```
sudo tar xvf leptonica-1.76.0.tar.gz
cd leptonica-1.76.0
sudo ./configure
sudo make -j4
```

If the installation is successful, then go to 10.4.2.2. Else, go to 10.4.2.1.2.

ORACLE

9.4.2.1.2 Installation through CMake

In case autoconf fails to generate the configure file or there is any other error, then proceed with the below steps, to build through CMake.

```
sudo tar xvf leptonica-1.76.0.tar.gz
cd leptonica-1.76.0
sudo mkdir build
cd build
sudo cmake ..
sudo make -j4
```

9.4.2.2 <u>Leptonica Configuration</u>

- Leptonica path should be configured such that tesseract can find the leptonica installation.

- Please add the leptonica installation directory (Ex: /usr/local/lib ,/usr/lib, /usr/lib64 etc) in library path.

- leptonica header path (Ex: /usr/local/include/leptonica) should be configured.

- Pkgconfig path also need to be set up.

  Execute the below mentioned commands to set the path:

```
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/local/lib/pkgconfig/
export PKG_CONFIG_PATH=$PKG_CONFIG_PATH:/usr/lib64/pkgconfig/
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
export LIBLEPT_HEADERSDIR=/usr/local/include/leptonica
```

NOTE: Sometimes, tesseract will still be unable to find lept.pc file.

It will give configuration errors (ex: Leptonica 1.74 or higher is required). In that case locate the lept.pc file (usually present at /usr/local/lib/pkgconfig/) with the command **locate lept.pc** and copy the same in /usr/lib64 directory.

```
sudo cp /usr/local/lib/pkgconfig/lept.pc /usr/lib64/pkgconfig/
```

- Similarly, some services might not be able to get libleptonica shared object files (.so files, ex: liblept.so, libleptonica.so etc.)

- .so files are usually present in the server at /usr/local/lib. You can type **whereis libleptonica** or **locate libleptonica** to find the path. Then copy the .so files in /usr/lib64 path.

```
cd /usr/local/lib
sudo cp -a *liblept* /usr/lib64
```

ORACLE

9.4.2.3 Tesseract Installation

- Copy the tesseract tarball tesseract-4.1.0.tar.gz in server (in the installation directory. Ex: /scratch)

- Copy the tesseract trained files eng.traineddata, osd.traineddata in the server

- Execute below commands sequentially to build and install tesseract

NOTE: /usr/bin is the directory where tesseract binary will be present if you pass prefix=/usr in configure. You can provide the path based upon where you want to install.

```
sudo tar xvf tesseract-4.1.0.tar.gz
cd tesseract-4.1.0
sudo ./autogen.sh
sudo ./configure --prefix=/usr
sudo make -j4
```

- Copy the traineddata files in tessdata directory. If you use prefix=/usr, tessdata directory will be present at /usr/share. If you use prefix=/usr/local, tessdata directory will be present at /usr/local/share.

```
sudo cp osd.traineddata /usr/share/tessdata
sudo cp eng.traineddata /usr/share/tessdata
```

9.4.2.4 Tesseract Configuration

- Set the tesseract library path by executing the below commands:

```
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/lib
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/usr/local/lib
```

- Sometimes services are unable to find libtesseract shared object files (.so files) present in system (Usually at /usr/lib). In that case copy the libtesseract files in /usr/lib64

```
cd /usr/lib
sudo cp -a *libtesseract* /usr/lib64
```

ORACLE®

- Some of the programs search for the tessdata directory in a different path usr/share/tesseract/4/tessdata). Copy the existing tessdata directory (which will be present in either /usr/share or /usr/local/share based on your installation) in that path.

```
cd /usr/share
sudo mkdir tesseract (execute if tesseract
directory is not present)
cd tesseract
sudo mkdir 4
cd /usr/share
```

- Set tessdata prefix by running following command.

```
export TESSDATA_PREFIX=/usr/share/tesseract/4/tessdata
```

- Tesseract is now installed.
- You can verify the version with following command. It will give the tesseract version (4.1.0), leptonica version (1.76.0) along with other default libraries (libjpeg, libjpeg-turbo, libpng, libtiff, zlib).

```
tesseract --version
```

9.5 Conductor Installation

9.5.1 Pre-requisite

Perform the following steps:

1. Ensure that the datasource jdbc/PLATO-O is created. The maximum capacity attribute of the datasource connection pool should be greater than 100.

2. Make sure that the Domain and cluster configuration steps completed.

   NOTE: The conductor-server.war file needs to be deployed in a separate manged server because of its load and size.

9.5.2 Installation

Perform the following steps:

1. Required properties should be set in the config.properties file found in {**unzip the file**} THIRD_PARTY_SOFTWARES\CONDUCTOR_SERVER\CONFIG. Refer to the below table to find the description of properties in the config.properties. This file should be placed at <<CONFIG.PROPERTIES LOCATION >>.

2. An additional environment variable is required for setting up the conductor. Include the below mentioned –Dparam along with the existing environment variables.
   -Dconductor.properties = << CONFIG.PROPERTIES LOCATION >>/config.properties

ORACLE®

3. Deploy the conductor-server.war file in the weblogic. To deploy application, refer to **How to deploy section** ANNEXURE-1.

4. To control the Log Level, the following property has to be added as a –Dparam along with the existing environment variables:
-Dplato.conductor.logging.level = << LOG-LEVEL >>

By Default, the log level is "DEBUG". To obtain only ERROR logs value has to be given as "ERROR" and to completely switch of conductor logs, the value has to be set as "OFF"

Property Name	Property Description
flyway.enabled	Set this to true to enable flyway and false to disable flyway.
flyway.setbaselineOnMigrate	Set this to true to enable flyway baselineOnMigrate and false to disable.
eureka.registration.enabled	Should be set to true to enable discovery registration.
eureka.hostName	plato-o
eureka.instanceId	plato-o:<port-number>
eureka.serviceUrl.default	Discovery service URL (http://<hostname>:<port>/plato-discovery-service/eureka)
eureka.registerWithEureka	true
eureka.name	plato-o
eureka.vipAddress	plato-o
eureka.port	Port Number on which the conductor server war file is deployed.
conductor.entity.list	*DEFAULTENTITY~jdbc/PLATO-O* Here – *DEFAULTENTITY* – is entity Id *jdbc/PLATO-O* – JNDI name of Conductor Datasource The entity added, need to make changes in this property. Multiple entities can be added using "," as a delimiter. For example, *ENTITY1~jdbc/PLATO-O1, ENTITY2~jdbc/PLATO-O2*
workflow.elasticsearch.instanceType	*EXTERNAL*

ORACLE

Property Name	Property Description
multi.entity.enabled	*By default, it is false.* *To enable multi-entity, set it to true.*
db	*oracle*
ssl.truststore.location **ssl.truststore.password** **sasl.mechanism** **sasl.jaas.config**	*All are kafka SSL related properties.* *These are valid only if SSL is enabled* *ALL ARE ENVIRONMENT SPECIFIC.*

ORACLE

9.6 <u>Report Service Installation</u>

9.6.1 <u>Pre-requisite</u>

Make sure that the data source is created.

Data source Name	Data source JNDI	Targets
PLATOCMC	jdbc/CMNCORE	Plato Common Core Server
PLATOSMS	jdbc/sms	Plato-SMS-Server
REPORTSERIVCE	jdbc/REPORTSERVICE	Plato-Report-Service-Server

Make sure that the wars mentioned below are installed along with the ones mentioned above.

- CMC Core Service
- CMC Base Service
- CMC Currency Service
- CMC Component Service
- Mock Extractor
- Obreports Component Server
- Plato Report Service
- SMS Component Server
- App Shell

9.6.2 <u>Installation</u>

9.6.2.1 <u>setUserOverrides.sh file</u>

Perform the following steps:

1. Create a file called **setUserOverrides.sh** inside the Weblogic bin location.

2. The following formats of the **setUserOverrides.sh** file and the list of parameters that need to be passed in order to run the plato services properly.

   NOTE: Below are the list of **-D params** (ENV Variables), which needs to be set for all the individual services. Set a single **–Dparam** as follows:

   JAVA_OPTIONS="${JAVA_OPTIONS} -DParam =<ParamValue>"

   export JAVA_OPTIONS

ORACLE

//Plato Report Service

-Dflyway.domain.placeholders.report-service.hostname=<http://<REPORT_SERVOICE_HOSTNAME>:<REPORT_SERVICE_PORT>/report-service/api/>

-Dflyway.domain.placeholders.report-service.server.port=<REPORT_SERVICE_PORT>

-Dflyway.domain.placeholders.report-service.domain.jndi=<JNDI_SCHEMA>

-Dflyway.domain.placeholders.report-service.template-metadata-directory=/scratch/OBMA/report-service/template_metadata

-Dflyway.domain.placeholders.report-service.output-directory=/scratch/OBMA/report-service/output/

-Dflyway.domain.placeholders.report-service.fop-config-file=/scratch/OBMA/report-service/fop.xconf

9.6.2.2 Plato Reporting Deployment Order
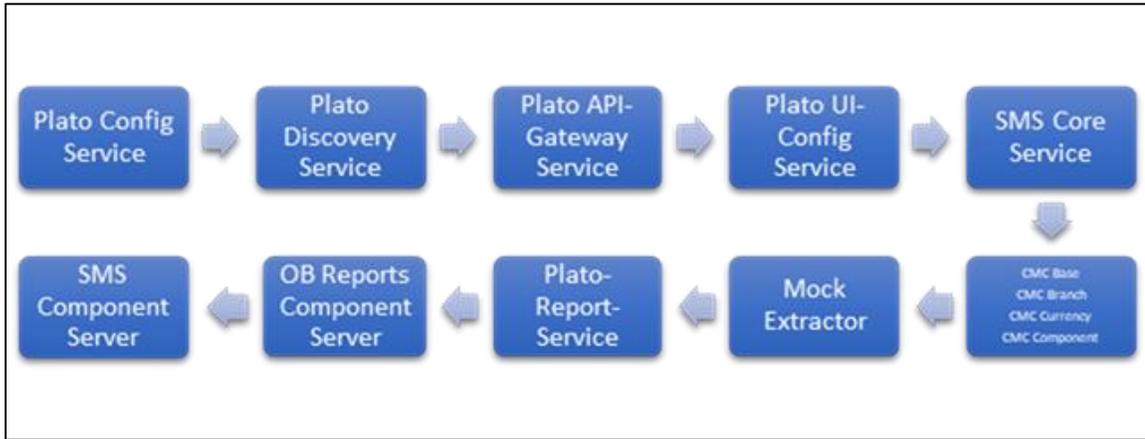
Installation Summary for Plato Reporting Service:

Application	Archive name	OSDC path	Targets
sms-core-Service	sms-core-Service-{version}.war	{**unzip the file**}PLATO\sms-core-service\	Sms-core-Service
cmc-base-services	cmc-base-services-{version}.war	{**unzip the file**}PLATO\cmc-base-service\	cmc-base-Service
cmc-branch-services	cmc-branch-services-{version}.war	{**unzip the file**}PLATO\cmc-branch-service\	cmc-branch-Service
cmc-currency-services	cmc-currency-services-{version}.war	{**unzip the file**}PLATO\cmc-currency-service\	cmc-currency-Service
cmc-component-server	cmc-component-services-{version}.war	{**unzip the file**}PLATO\cmc-component-service\	cmc-component-Service
mock-extractor	mock-extractor-{version}.war	{**unzip the file**}PLATO\mock-extractor\	Mock-extractor

9-18

ORACLE®

Application	Archive name	OSDC path	Targets
plato-report-Services	plato-report-Services-{version}.war	{**unzip the file**}PLATO\plato-report-services\	Plato-report-Server
obreports-component-Server	obreports-component-Server-{version}.war	{**unzip the file**}PLATO\obreports-component-server\	Obreports-component-Server
sms-component-server	sms-component-services-{version}.war	{**unzip the file**}PLATO\sms-component-service\	sms-component-Service

NOTE: Refer to OSDC file for the exact version number for each service.

ORACLE

10. Security- SSL Encryption with SASL-SCRAM Authentication

10.1 Generate Keystore:

The items highlighted in blue are placeholders, and should be replaced with suitable values when running the command.

keytool -genkeypair -alias alias -keyalg keyalg -keysize keysize -sigalg sigalg -validity valDays -

keystore keystore

In the above command:

1. *alias* is used to identify the public and private key pair created.

2. *keyalg* is the key algorithm used to generate the public and private key pair. The RSA key algorithm is recommended.

3. *keysize* is the size of the public and private key pairs generated. A key size of 1024 or more is recommended.

4. *sigalg* is the algorithm used to generate the signature. This algorithm should be compatible with the key algorithm and should be one of the values specified in the Java Cryptography API Specification and Reference.

5. *valdays* is the number of days for which the certificate is to be considered valid. Please consult with your CA on this period.

6. *keystore* is used to specify the location of the JKS file. If no JKS file is present in the path provided, one will be created.

The command will prompt for the following attributes of the certificate and keystore:

7. Keystore Password: Specify a password that will be used to access the keystore. This password needs to be specified later, when configuring the identity store in Kafka Server.

8. Key Password: Specify a password that will be used to access the private key stored in the keystore. This password needs to be specified later, when configuring the SSL attributes of the Kafka Server.

9. First and Last Name (CN): Enter the domain name of the machine, for instance, www.example.com.

10. Name of your Organizational Unit: The name of the department or unit making the request. Use this field to further identify the SSL Certificate you are creating, for example, by department or by physical server.

11. Name of your Organization: The name of the organization making the certificate request, for example, Oracle Financial Services. It is recommended to use the company or organization's formal name, and this name entered here must match the name found in official records.

12. Name of your City or Locality: The city in which your organization is physically located, for example Bengaluru.

13. Name of your State or Province: The state/province in which your organization is physically located, for example Karnataka.

14. Two-letter CountryCode for this Unit: The country in which your organization is physically located, for example US, UK, IN etc.

ORACLE

For example:

Listed below is the result of a sample execution of the command:

keytool -genkeypair -alias certificates -keyalg RSA -keysize 1024 -sigalg SHA512withRSA -

validity 365 -keystore /scratch/Data/Certificates/KafkaServerKeystore.jks

Enter keystore password:<Enter a password to protect the keystore>

Re-enter new password:<Confirm the password keyed above>

What is your first and last name?

[Unknown]: <domain name>.oracle.com

What is the name of your organizational unit?

[Unknown]: <application name>

What is the name of your organization? [Unknown]: Oracle Financial Services

What is the name of your City or Locality?

[Unknown]: Bengaluru

What is the name of your State or Province?

[Unknown]: Karnataka

What is the two-letter country code for this unit?

[Unknown]: IN

Is CN= name.oracle.com, OU=Test, O=Oracle Financial Services, L= Bengaluru, ST= Karnataka, C=IN correct? [no]: yes

Enter key password for < password >

RETURN if same as keystore password): <Enter a password to protect the key>

Re-enter new password: <Confirm the password keyed above>

ORACLE

10.2 <u>Export Private Key as Certificate</u>

The command is mentioned below:

keytool -export -alias <alias_name> -file <export_certificate_file_name_with_location.cer> -

keystore <keystore_name.jks> -keypass <Private key Password> -storepass <Store Password>

For example:,

keytool -export -alias certs -file /scratch/Data/Certificates/KafkaCert.cer -keystore

/scratch/Data/Certificates/KafkaServerKeystore.jks -keypass oracle123 -storepass oracle123

If successful, the following message will be displayed:

Certificate stored in file < KafkaCert.cer>

10.3 <u>Import the Cert and generate TrustStore</u>

The command is mentioned below:

keytool -import -alias alias -file cert_file -keystore truststore –storepass storepass

In the above command:

1. *alias* is used to identify the public and private key pair. Specify the alias of the key pair used to create the CSR in the earlier step.
2. *cert_file* is the location of the file containing the PKCS#7 formatted reply from the CA, containing the signed certificate.
3. *truststore* is the location where the truststore should be generated.
4. *storepass* is the password for the truststore.

Generate two truststores from the same cert. One used for kafka server and one for clients.

For example:

keytool -import -alias certs -file /scratch/Data/Certificates/KafkaCert.cer –keystore

/scratch/Data/Certificates/KafkaServerTrustStore.jks -storepass oracle123

keytool -import -alias certs -file /scratch/Data/Certificates/KafkaCert.cer -keystore

/scratch/Data/Certificates/KafkaClientTrustStore.jks -storepass oracle123

Hence, three keystore files would be needed for this method:

1. *KafkaServerKeystore.jks* : keystore file for Kafka brokers
2. *KafkaServerTrustStore.jks* : Truststore file for server
3. *KafkaClientTrustStore.jks* : Truststore file for client

ORACLE

The *KafkaClientTrustStore.jks* file need to be imported by every client to validate the server.

NOTE: The truststore files should be generated using the same CA. Generate and place these files on all the different servers of kafka so that it can be accessed by server*.properties file. The *KafkaClientTrustStore.jks* should be placed on the server, which is accessible by the microservices also.

10.4 <u>Creation of users in Zookeeper</u>

Start the zookeeper (command in Section *9.1*) and execute the below commands for the user creation.

./kafka-configs.sh --zookeeper localhost:2181 --alter --add-config "SCRAM-SHA-256=[password=admin-secret],SCRAM-SHA-512=[password=admin-secret]" --entity-type users --entity-name admin

./kafka-configs.sh --zookeeper localhost:2181 --alter --add-config "SCRAM-SHA-256=[iterations=8192,password=test-secret],SCRAM-SHA-512=[password=test-secret]" --entity-type users --entity-name test

Two users are created above with test and admin as usernames and two different passwords are setup for each user one for each scram mechanism. Here, the user 'admin' is used for Kafka broker auth and 'test' is used for client auth.

11. Oracle Banking Microservices Architecture Deployments

11.1 Pre-requisite

Please note that the app-shell and api-gateway are updated with new encoding/decoding logic in the platojwtauth call. By default, the new encoding/decoding logic will be used, but if you want to use the previous encoding/decoding format, you can do so by adding the below property in the managed servers where app-shell and api-gateway are deployed.

Property	Value
-DEncryptionFlag	**true** - new encoding/decoding logic will be applied (more secure) OR **false** - old encoding/decoding logic will be applied.

Before you proceed with below, make sure that the previous steps are completed. Below table give details of the deployments required on each Server for the Oracle Banking Microservices Architecture application to run.

11.2 Oracle Banking Microservices Architecture Applications Deployment Order
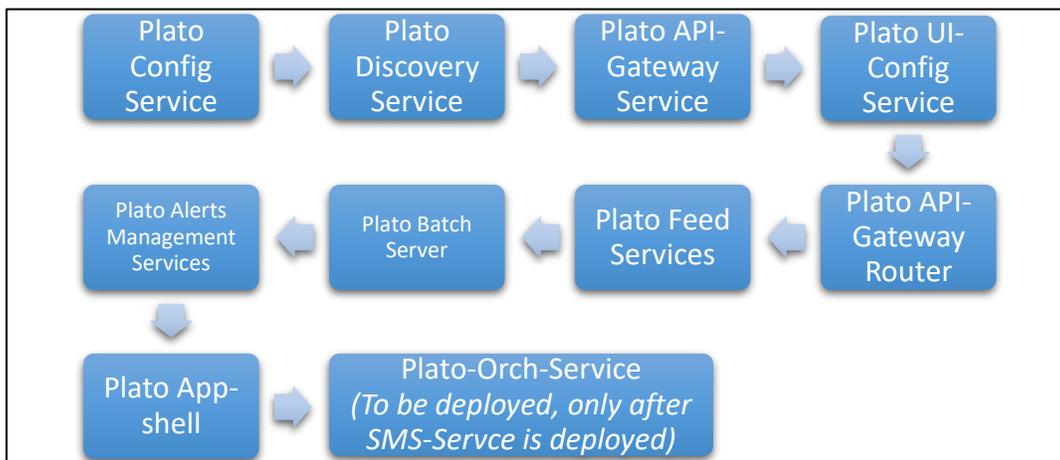
Installation Summary for Oracle Banking Microservices Architecture Services:

Application	Archive name	OSDC path	Targets
Plato-config-service	plato-config-service-{version}.war	{**unzip the file**}PLATO\plato-config-service\	Config Server
Plato-discovery-service	plato-discovery-service-{version}.war	{**unzip the file**}PLATO\plato-discovery-service\	Discovery Server

Application	Archive name	OSDC path	Targets
Plato-api-gateway	plato-api-gateway-{version}.war	{**unzip the file**}PLATO\plato-api-gateway\	Api Gateway
Plato-ui-config-service	plato-ui-config-service-{version}.war	{**unzip the file**}PLATO\plato-ui-config-service\	Plato UI Config
Plato-apigateway-router	plato-apigateway-router-{version}.jar	{**unzip the file**}PLATO\plato-apigateway-router\	Plato-Apigateway-Router
Plato-Orch-Service *(To be deployed after sms-service is deployed)*	plato-orch-service-{version}.war	{**unzip the file**}PLATO\plato-orch-service\	Plato-Orch-Service
Plato-Feed-Services	plato-feed-services-{version}.war	{**unzip the file**}PLATO\plato-feed-services\	Plato-Feed-Services
Plato-Batch-Server	plato-batch-server-{version}.war	{**unzip the file**}PLATO\plato-batch-server\	Plato-Batch-Server
Plato-Alerts-Management-Services	plato-alerts-management-services-{version}.war	{**unzip the file**}PLATO\plato-alerts-management-services\	Plato-Alerts-Management-Server
Plato-Rule-Services	plato-rule-service-{version}.war	{**unzip the file**}PLATO\plato-rule-service\	Plato-Rule-Server
Plato-Report-Services	plato-report-services-{version}.war	{**unzip the file**}PLATO\plato-report-services\	Plato-Report-Server
Plato-Swagger-Api	plato-swagger-api-{version}.war	{**unzip the file**}PLATO\plato-swagger-api\	Plato-Swagger-Api Server
Appshell	app-shell-{version}.war	{**unzip the file**}UI\app-shell-{version}.war	Appshell Server

Refer to OSDC file for the exact version number for each service.

NOTE: Eventhub based applications should not be deployed in admin server

11.3 <u>Steps to Deploy as Application</u>

To deploy application, refer to **How to deploy section** in ANNEXURE-1.

11.4 <u>SSL Configuration</u>

The below parameters should be available into JVM for SSL configuration.

Key	Default Value	Purpose
${apigateway.protocol}	https	Only for API gateway protocol. it must be **https only**.

ORACLE

${eureka.protocol}	https	For inter-service communication protocol. Values can be **http** or **https**.
${prefer.ip.address.enabled}	false	For prefer IP address flag. it must be **false** only.
${nonsecure.port.enabled}	false	For disabling inter-service communication on non secure port. Values can be **false** or **true**
${secure.port.enabled}	true	For allowing inter-service calls on secure port. Values scan be **false** or **true**

We recommend only https-based connections. Below are the recommendations:

1.   Appshell needs to be secured with SSL.

2.   Api-Gateway needs to be secured with SSL.

3.   Appshell to Api-gateway communication should happen over SSL.The api-gateway url mentioned as -D parameter for appshell should be ssl enabled (i.e. https-based) and must point to plato-apigateway-router.

12. Restarts and Refresh

Once everything is deployed, the managed servers. And for each application call path "/refresh" for refreshing the configuration properties.

12.1 <u>Restarting Servers</u>

To restart the server, refer to **How to restart** section in ANNEXURE-1.

13. Logging Area

13.1 <u>Introduction</u>

This section describes the logs area where after deployment of Oracle Banking Microservices Architecture Applications in the WebLogic server.

13.2 <u>Logging Area</u>

Oracle Banking Microservices Architecture Application writes logs in the below area of the server:

<WEBLOGIC_DOMAIN_CONFIG_AREA>/ logs/plato-api-gateway.log

For example, consider that a domain has been created **platoinfra_domain** in the following area of the server "/scratch/oracle/middleware/user_projects/domains/platoinfra_domain".

Logging area for Plato
=<URL>

14. Known Issues – Resolutions

For deploying any application, if there is an issue with ID column conflict for table *product_services_ledger* (PLATO_UI_CONFIG schema), change the current value of DB sequence (PRODUCT_SVCS_LEDGER_ID_SEQ) to maximum value present in ID column for table *product_services_ledger*.

ORACLE®

Oracle Banking Microservices Platform Foundation Installation Guide

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