Oracle® Banking Microservices Architecture

Oracle Banking Microservices Platform Foundation Installation Guide

Release 14.6.0.0.0

F59266-01

May 2022

Oracle Banking Microservices Architecture Oracle Banking Microservices Platform Foundation Installation Guide, Release 14.6.0.0.0

F59266-01

Contents

Preface

Purpose

This guide helps 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 installation, WebLogic managed server creation and Oracle DB installation.

> **Note:**
>
> For the exact version to be installed, refer to **Tech Stack** section of **Release Notes**.

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

Audience

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

Acronyms and Abbreviations

The list of the acronyms and abbreviations that are used in this guide are as follows:

Table 1 Acronyms and Abbreviations

Abbreviation	Description
LDAP	Lightweight Directory Access Protocol
JWT	JSON Web Token
JWS	JSON Web Signature
JWE	JSON Web Encryption

List of Topics

This guide is organized as follows:

Table 2 List of Topics

Topics	Description
Database Setup	This topic provides the information about database setup.

Table 2 (Cont.) List of Topics

Topics	Description
Domain and Cluster Configuration	This topic provides the information about domain and cluster configuration.
Data Sources Creation	This topic provides the information about data sources creation.
Security Configuration and Tools Installation	This topic provides the information about security configuration and tools installation.
Deployments	This topic provides the information about deployments.
Multi Entity Configuration	This topic provides the information about Multi-Entity configuration.
Plato Orchestration Services	This topic provides the information about Plato Orchestration services.
Plato Feed Services	This topic provides the information about Plato Feed services.
Oracle Banking Microservices Architecture Software Deployment	This topic provides the information about the Oracle Banking Microservices Architecture Software Deployment.
Security- SSL Encryption with SASL-SCRAM Authentication	This topic provides the information about Security- SSL Encryption with SASL-SCRAM authentication.
Oracle Banking Microservices Architecture Deployments	This topic provides the information about Oracle Banking Microservices Architecture Deployment.
Restart and Refresh	This topic provides the information about restarts and refresh.
Logging Area	This topic provides the information about logging area.
Known Issues – Resolutions	This topic provides the information about Known Issues – Resolutions.

Organization

This guide allows the user to install the following services in same order as follows:

- WebLogic system environment settings
- Plato Config Service
- Plato Discovery Service
- Plato API Gateway Service
- 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

1 Database Setup

This topic describes about the database setup for Oracle Banking Microservices Architecture Installation.

Before proceeding with the below setup, make sure pre-installation setup is done.

Prerequisites

Before proceeding with below setup, ensure that the Schemas are being created.

It is recommended to have different schema for **Plato** and **Plato Security**. To configure Plato security, refer to Security Configuration chapter. Make sure that the schema user has the below rights:

Table 1-1 Schema User - Operations

DB OBJECT	CREATE	ALTER	DROP	NSERT	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 **Check Port Number** section in **ANNEXURE - 1**.

Make sure to configure Placeholder parameters in WebLogic server for plato-config-service, setDomain.env. For more details, refer to **Placeholder Update for Oracle Banking Microservices Architecture Services** section in **ANNEXURE - 1**.

2

Domain and Cluter Configuration

This topic describes about the domain and cluster configuration for Oracle Banking Microservices Architecture services.

Prerequisites

Before proceeding, make sure that the below installation is done.

- Machine should have Java installed.
- Oracle Fusion Middleware has to be installed on the machine.

> **Note:**
>
> For the exact version to be installed, refer to **Software Prerequisites** section in **License Guide**.

Domain Creation and Configuration

It is recommended to have different managed server in one domain for each application.

> **Note:**
>
> For creating Domain and Configuration, refer to **Create Domain and Cluster Configuration** section in **ANNEXURE - 1**.

3
Data Sources Creation

This topic provides the systematic instructions to create data sources for Oracle Banking Microservices Architecture.

Prerequisite

Before proceeding with Data source creation, ensure that the domain and cluster configuration is completed.

Data Sources List

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

Table 3-1 Data Sources List

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

Note:

For creating data source, refer to **Create Data Sources** section in **ANNEXURE - 1**.

4
Security Configuration and Tools Installation

This topic provides the information about security configuration and tools installation.

Prerequisites

Before proceeding, do the following steps:

- If the user wants to use LDAP for web application authentication with WebLogic as a provider for LDAP.

  > **Note:**
  >
  > Refer to the **WebLogic Embedded LDAP Setup** section in **ANNEXURE - 1** for the setup details.

- If the user wants to use OAuth without OAM (Spring OAuth), do the below 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 that the LDAP server details is given to the user as below:

    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.

Plato Security JWT

The Plato security module enables securing API microservices with JWT. The JWT are an open, industry standard RFC 7519 method for representing claims securely between two parties. The JWT is a compact, URL-safe means of representing claims transferred between two parties. The claims in the JWT are encoded as a JSON object, which is used as a payload of the JWS structure or as plain text of the JWE structure, enabling the claims to be digitally signed.

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. In case of **LDAP Directory Authentication**, change the below KEY with the provided LDAP details:

Table 4-1 LDAP Directory Authentication - Key Parameters

KEY	VALUE
LDAP_SERVER_CREDENTIAL_SALT	Enter LDAP server Credential salt e.g. 0.9482628451234567
CORS_ALLOWED_ORGINS	valid host names (comma delimited)
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. In case of **SSO Agent**, change the below KEY with the provided LDAP details:

Table 4-2 SSO Agent - Key Parameters

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

User Store

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

5 Deployments

This topic describes about the deployment of Oracle Banking Microservices Architecture.

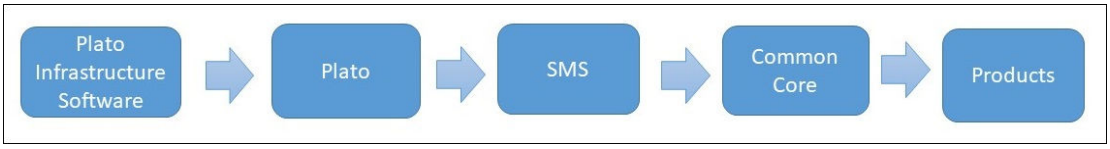
Prerequisites

Before proceeding with below setup, ensure that the previous steps are completed.

Deployment Order

The deployment order is shown below.

Figure 5-1 Deployment Orders



Flyway Configuration

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

- flyway.enabled = <Boolean>. If true, flyway will get executed. If false, flyway wouldn't be executed.
- spring.flyway.enabled=false (always).

6
Multi Entity Configuration

This topic describes the information about the multi entity configuration for Oracle Banking Microservices Architecture services.

Enable Multi Entity

By Default, Multi Entity is disabled.

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

Create Default Entity

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.

Create Multi-Entity Admin User

To create multi-entity user, perform below steps:

1. Execute the below scripts at the path in the OSDC zip.
   The user are 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_14.6.0.0.0.sql
   - For existing users:
     <<PRODUCT_NAME>>_INITIAL_SETUP\UPGRADE\PLATO\INS_PLATO_SEC_ME_ADMIN_USER_CREATION_14.6.0.0.0.sql.
2. Create the multi-entity admin user in the LDAP.

Create Entity

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

> **Note:**
>
> Refer to **Oracle Banking Multi Entity Deployment Guide** for the procedure to create an entity.

When the multi entity admin create an entity, the following processes executes in the background:

- The entity details are saved in the PLATO_TM_ENTITY table.
- The JNDIs are saved in the APPLICATION_LEDGER table.
- The flyway scripts for all the micro services gets executed in their respective schemas.
- Once the flyway execution is completed, a new role ENTITY_ADMIN is created in the entity. This step inserts the scripts into the following tables:
  - SMS_TM_ROLE
  - SMS_TW_ROLE
  - SMS_TM_ROLE_ACTIVITY
  - SMS_TW_ROLE_ACTIVITY
    This role is assigned to the entity admin user in the user creation step.
- The Head Office branch details are inserted into the CMC_TM_CORE_BRANCH and CMC_TW_CORE_BRANCH tables.
- The Bank details are inserted into the CMC_TM_CORE_BANK and CMC_TW_CORE_BANK tables.
- The System dates are inserted into the CMC_TM_SYSTEM_DATES and CMC_TW_SYSTEM_DATES tables.

Create User

Make sure that the entity creation step is complete before creating 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 **Oracle Banking Multi Entity Deployment Guide** for the procedure 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.

7

Plato Orchestration Services

This topic describes about the Plato Orchestration Services.

Migration Endpoint

> **Note:**
>
> This topic is applicable only to the existing customers.

The task blob usage is 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 must be executed to populate the data for the completed tasks in this table. In-Progress tasks, the data is automatically populated by the poller. This improves 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.

8
Plato Feed Services

This topic describes about the Plato Feed Services.

To avail the feature of record level approval functionality in Plato-Feed, the below property need to be maintained in the **RECORD_LEVEL_APPROVAL** column of **PLATO_TM_FEED** table.. If not maintained, the default behavior will be of file-level approval only.

- **property name** - RECORD_LEVEL_APPROVAL
- **property value** - Y or N
- **default value** - N

9

Oracle Banking Microservices Architecture Software Deployment

This topic describes about Oracle Banking Microservices Architecture software deployment.

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

9.1 Zookeeper Cluster Setup

This topic provides the systematic instructions to install Zookeeper cluster setup.

> **Note:**
>
> To restart the server, refer to the **Restart Server** section in **ANNEXURE - 1**.

Before proceeding, ensure that the below installation is done.

- JDK is installed in all node machines.
- Download zookeeper and extract the binary in all node machines. Zookeeper can be found at `<Unzip the file>/THIRD_PARTY_SOFTWARES/ZOOKEEPER/ARCHIVE`

1. Untar/unzip the zookeeper binary and move them to the folder that will be the zookeeper home directory.
2. Create two directories of Logs and Data inside the zookeeper home directory folder on all nodes with appropriate permission.

   > **Note:**
   >
   > Clear the logs folder if it is already present.

3. Create a myid file inside the `<zookeeper home directory>/`data folder.

   The myid file contains a single line text of machine ID. The server 1 myid has the text as 1 and nothing else. The ID must be unique within the ensemble and have a value between 1 and 255.

4. Create a configuration file named zoo.cfg at `<zookeeper home directory> / zookeeper_3.6.2/config`.
5. Add the below set of properties and values to the file.

```
dataDir= <zookeeper 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.

6. Start the zookeeper on each node machine.
7. Navigate to `<zookeeper home directory>/zookeeper_3.6.2` and execute the below command.

```
bin/zkServer.sh start
```

8. To see the leader and followers in the cluster, run the below command on each node.

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

9. To check the zoo cluster functioning (dynamic leader election), kill the zookeeper process on the leader node and check again with the below command on the remaining live zookeeper node.

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

9.2 Kafka Cluster Setup

This topic provides the systematic instructions for Kafka cluster setup.

Before proceeding, ensure that the below installation is done.

- JDK is 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`.

1. Untar/unzip the kafka binary and move them to the folder that will be the kafka home directory.
2. Create two directories of Logs and Data inside the kafka home directory folder on all nodes with appropriate permission.

> **Note:**
>
> Clear the logs folder if already present.

3. Edit the below lines in the `<kafka home directory>/kafka_2.13-2.8.1/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
eeper_client_port>, …
```

4. To start the Kafka, navigate to `<kafka home directory>/kafka_2.13-2.8.1/` 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.

5. To create topic, navigate to `<kafka home directory>/kafka_2.13-2.8.1/` folder and run the below command.

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

6. To list the available topic on kafka server, navigate to `<kafka home directory>/kafka_2.13-2.8.1/` folder and run the below command.

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

7. To describe the topic, navigate to `<kafka home directory>/kafka_2.13-2.8.1/` folder and run the below command.

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

8. To start a producer, navigate to `<kafka home directory>/kafka_2.13-2.8.1/` 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>
```

> **Note:**
>
> By default, port is taken as 9092 for the producer.

9. 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
```

Configure a Standalone Kafka Instance in Cluster Mode

If there is already a standalone Kafka instance with Oracle Banking Microservices Architecture services running on it, it is expected the topics are already created in the Kafka instances. In this case, use the below steps to enable replication of messages between Kafka brokers:

10. Download and edit the **increase.json** to have all the topics and their replication confirmation updated.

    Example: If you have a 3 node setup, the json will look like the attached sample. It is ideal to have the number of replicas equal to the number of brokers.

```
{"version":1,
  "partitions":[
     {"topic":"<Topic Name>","partition":0<if there is just one
partition, else there has to be a
     different record for each partition per topic>","replicas":
[comma separated list of broker ids]}
]}
```

11. Run the following command

```
.\bin\windows\kafka-reassign-partitions.bat --bootstrap-server
localhost:9092 --reassignment-json-file increase.json --execute
```

9.3 Kafka Security Setup

This topic describes about the Kafka Security setup.

Prerequisites

Before proceeding, ensure that the below installation is done.

- JDK is installed in all node machines.
- Kafka is downloaded and extracted the binary in all node machines. Kafka can be found at `<Unzip the file>/THIRD_PARTY_SOFTWARES/KAFKA/ARCHIVE`.

Generate Keystore

The items highlighted in bold 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
```

Table 9-1 Generate Keystore - Keyword Description

Keyword	Description
alias	Used to identify the public and private key pair created.
keyalg	It is a key algorithm used to generate the public and private key pair. The RSA key algorithm is recommended.
keysize	It is the size of the public and private key pairs generated. A key size of 1024 or more is recommended.
sigalg	It 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.
valdays	It is the number of days for which the certificate is to be considered valid. Please consult with your CA on this period.
keystore	It 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 prompts for the following attributes of the certificate and Keystore:

Table 9-2 Certificate and Keystore - Attributes

Attributes	Description
Keystore Password	Specify a password used to access the Keystore. This password needs to be specified later when configuring the identity store in Kafka server.

Table 9-2 (Cont.) Certificate and Keystore - Attributes

Attributes	Description
Key Password	Specify a password 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.
First and Last Name (CN)	Enter the domain name of the machine. For example, www.example.com.
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.
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.
Name of your City or Locality	The city in which your organization is physically located. For example, Bengaluru.
Name of your State or Province	The state/province in which your organization is physically located. For example, Karnataka.
Two-letter Country Code for this Unit	The country in which your organization is physically located. For example, US, UK, IN, etc.

Example 9-1 Execution

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>

Export Private Key as Certificate

Export private key as certificate 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>
```

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>`

Import the Cert and Generate TrustStore

To import the cert and generate TrustStore, the command is mentioned below:

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

Table 9-3 Certificate and TrustStore - Keyword Description

Keyword	Description
`alias`	It 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.
`cert_file`	It is the location of the file containing the PKCS#7 formatted reply from the CA, containing the signed certificate.
`truststore`	It is the location where the TrustStore should be generated.
`storepass`	It is the password for the TrustStore.

The user can generate two TrustStores from the same cert.

- One used for Kafka server

- One 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
```

Three Keystore files are required for this method as given in the table below:

Table 9-4 Keystore Files

File Name	Description
KafkaServerKeystore.jks	Keystore file for Kafka brokers
KafkaServerTrustStore.jks	TrustStore file for server
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. The user can generate and place these files on all the different servers of Kafka so that they can be accessed by server*.properties file. The `KafkaClientTrustStore.jks` should be placed on the server, which is accessible by the microservices also.

Create Users in Zookeeper

To create users in Zookeeper, follow below steps:

1. Start the zookeeper.

   > **Note:**
   >
   > Refer to Zookeeper Cluster Setup topic for more details.

2. Follow the below steps for user creation.

a. Execute the admin command for admin 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
```

> **Note:**
>
> The user created with admin as username and password is setup for the user for each scram mechanism. Here, the user **admin** is used for Kafka broker auth.

b. Execute the test command for test user creation.

```
./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
```

> **Note:**
>
> The user created with test as username and password is setup for the user for each scram mechanism. Here, the user **test** is used for client auth.

Configure Brokers

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

1. Add the following properties to Kafka in the server.properties file.
   SSL-SCRAM Settings for SSL Configuration (Recommended)

```
listeners=SSL://localhost:9092
advertised.listeners=SSL://localhost:9092
ssl.endpoint.identification.algorithm=
ssl.truststore.location=/scratch/Data/Certificates/KafkaServerTrustStore.jks
ssl.truststore.password=oracle123
ssl.keystore.location/scratch/Data/Certificates/KafkaServerKeystore.jks
ssl.keystore.password=oracle123
ssl.key.password=oracle123
security.inter.broker.protocol=SSL
```

   Entries in the properties table for each kafka consumer/producer service

   - 'spring.cloud.stream.kafka.binder.configuration.ssl.truststore.location'
   - 'spring.cloud.stream.kafka.binder.configuration.ssl.truststore.password'
   - 'spring.cloud.stream.kafka.binder.configuration.security.protocol' value = ‘SSL’

- 'ssl.endpoint.identification.algorithm' = ''

SSL-SCRAM Settings for SASL-SSL Configuration (Not recommended)

```
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.ScramL
oginModule required username="admin" password="admin-secret";
```

2. Specify the absolute path of the Kafka Server Truststore and Keystore and its respective passwords.
3. Modify the host name and IP in the listeners and advertised.listeners properties field accordingly.
4. Start the Kafka servers.

> **Note:**
>
> Refer to the command in Kafka Cluster Setup topic.

Clients Changes (Kafka Consumer and Producer Services)

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.

Table 9-5 List of PROPERTIES

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.username	<Zookeeper user created for clients>
spring.cloud.stream.kafka.binder.jaas.options.password	<Zookeeper user encrypted password for clients>
spring.cloud.stream.kafka.binder.configuration.ssl.truststore.location	<location of client trust store certificate>
spring.cloud.stream.kafka.binder.configuration.ssl.truststore.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 the user clicks 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 topic **Method 3 – Using env files and setUserOverrides.sh file** in ***ANNEXURE-1***.

Important Commands

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

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

For example,

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

9.4 Tesseract Installation

This topic describes systematic instructions of Tesseract installation.

Prerequisite

Build Tools
Make sure that the following build tools are available:

- GNU Autotools—autoconf, automake, libtool
- CMake (Optional, we use CMake if autoconf fails to build leptonica).
  Both should be available inside Oracle yum.

Dependent Libraries
The libraries must be on the server. By default, they are available on Oracle Linux. If libraries are not present, please install through yum with the following command:

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

Following are the library names:

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

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

Installation Files
Download the installation files required to install and set up Tessract. Files are available at `<Unzip the file>/THIRD_PARTY_SOFTWARES/Tesseract`.

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

- leptonica-1.80.0.tar.gz
- tesseract-4.1.0.tar.gz
- eng.traineddata
- osd.traineddata

Leptonica Installation

Tesseract uses Leptonica internally for image processing. Leptonica can be built and installed by autoconf or CMake. The installation can be done using Autoconf and CMake.

> **Note:**
>
> If the user already have full 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)

> **Note:**
>
> In this topic, we execute all commands with sudo. The user can skip based on your user permission details.

Install through Autoconf

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

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

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

Here, the core number is used as 4 to build the software.

If the processor does not have multiple cores, the user can use normal make command `sudo make`.

> **Note:**
>
> If the installation is successful, then go to **Leptonica Configuration** . Else, go to **Install through CMake**.

Install through CMake

- If the installation through Autoconf fails to generate the configure file or has any other error, follow the commands below 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
```

Leptonica Configuration

- Configure the Leptonica path so that the Tesseract Leptonica installation can be found.
- Add the leptonica installation directory in library path. **Example**: `/usr/local/lib ,/usr/lib, /usr/lib64` etc.
- Configure the Leptonica header path. **Example**: `/usr/local/include/leptonica`.
- Setup the Pkgconfig path and 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 still unable to find lept.pc file.

It gives configuration errors. For example, 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.).

> **Note:**
>
> .so files are usually present in the server at /usr/local/lib.

- Type **whereis libleptonica** or **locate libleptonica** to find the path and copy the .so files in /usr/lib64 path.

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

Tesseract Installation

- Copy the Tesseract tarball tesseract-4.1.0.tar.gz to the server (installation directory). For example, `/scratch`.
- Copy the Tesseract trained files eng.traineddata, osd.traineddata to the server.
- Execute below commands sequentially to build and install Tesseract.

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

> **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 on where you want to install.

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

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

Tesseract Configuration

- Execute the below commands to set the Tesseract library path.

```
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
```

- Some programs search for the `tessdata` directory in a different path (`usr/share/tesseract/4/tessdata`). Copy the existing tessdata directory to the path (either in `/usr/share or /usr/local/share` based on your installation).

```
cd /usr/share
sudo mkdir tesseract(execute if tesseract directory is not p resent)
cd tesseract
sudo mkdir 4
cd /usr/share
sudo cpR tessdata /usr/share/tesseract/4
```

- Run the below command to set tessdata prefix.

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

The Tesseract is now installed.

- Verify the version with below command.

```
tesseract --version
```

It shows the tesseract version (4.1.0), leptonica version (1.76.0) along with other default libraries (libjpeg, libjpeg-turbo, libpng, libtiff, zlib).

9.5 Conductor Installation

This topic provides the systematic instructions to install conductor.

Before proceeding, ensure that the below steps are done.

- Make sure that the `datasource jdbc/PLATO-O` is created. The maximum capacity attribute of the datasource connection pool must be greater than 100.
- Make sure that the Domain and cluster configuration steps are completed.

> **Note:**
>
> The conductor-server.war file needs to be deployed on a separate manged server due to its load and size.

1. Set the required properties in the config.properties file found in {**unzip the file**} `THIRD_PARTY_SOFTWARES\CONDUCTOR_SERVER\CONFIG`.

   Refer to the following 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
```

3. Deploy the conductor-server.war file in the weblogic.

To deploy application, refer to **Deploy Application** section in **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”.

Table 9-6 config.properties

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 `rpm -ivh`	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 DEFAULTENTITY is entity Id jdbc/PLATO-O is 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
multi.entity.enabled	By default, it is false. To enable multi-entity, set it to true.
decider.sweep.disable	true
db	oracle
isSSLenabled	By default, it is false. To enable SSL, set it to true
security.protocol	Kafka SSL property. Valid only if SSL is enabled. Value has to be set as SASL_SSL

Table 9-6 (Cont.) config.properties

Property Name	Property Description
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.

9.6 Report Service Installation

This topic provides the systematic instructions to install report service.

Prerequisites
Before proceeding, ensure that the below steps are done.

- Make sure that the data source is created.

Table 9-7 Data Source List

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 below wars are installed along with the ones mentioned above.
  - CMC Core Service
  - CMC Base Service
  - CMC Currency Service
  - CMC Component Service
  - Plato Report Service
  - SMS Component Server
  - App Shell

Install setUserOverrides.sh file
Perform the following steps:

1. Create a file called **setUserOverrides.sh** inside the Web Logic bin location.
2. The following formats of the **setUserOverrides.sh** file and the list of parameters that need to be passed 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
```

//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
```

Plato Reporting Deployment Order

Figure 9-1 Plato Reporting Deployment Order

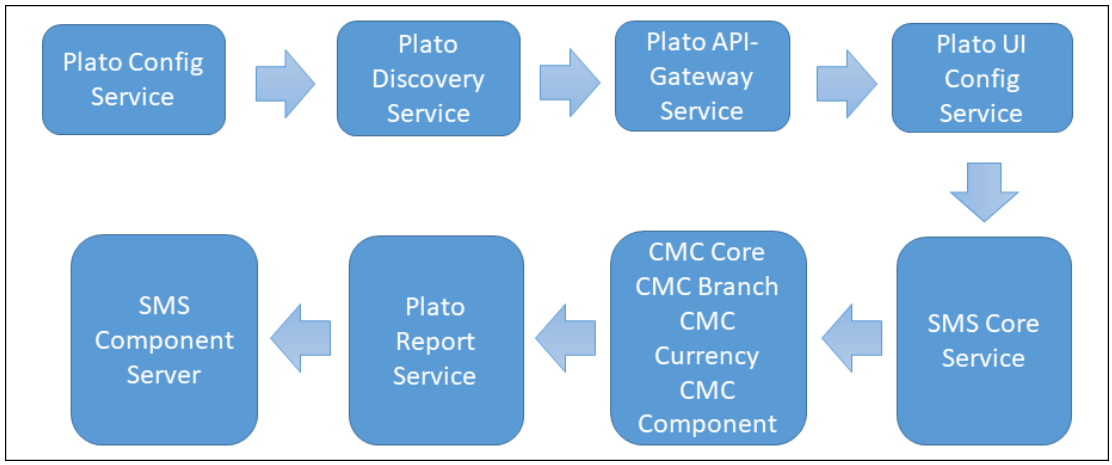


Table 9-8 Installation Summary for Plato Reporting Service:

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

Table 9-8 (Cont.) Installation Summary for Plato Reporting Service:

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

10

Security with SSL Encryption with SASL-SCRAM Authentication

This topic describes about Security - SSL Encryption with SASL-SCRAM authentication.

Generate Keystore

The items highlighted in bold 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
```

Table 10-1 Generate Keystore - Keyword Details

Keyword	Description
alias	Used to identify the public and private key pair created.
keyalg	It is a key algorithm used to generate the public and private key pair. The RSA key algorithm is recommended.
keysize	It is the size of the public and private key pairs generated. A key size of 1024 or more is recommended.
sigalg	It 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.
valdays	It is the number of days for which the certificate is to be considered valid. Please consult with your CA on this period.
keystore	It 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 prompts for the following attributes of the certificate and Keystore:

Table 10-2 Generate Keystore - Attributes

Attributes	Description
Keystore Password	Specify a password used to access the Keystore. This password needs to be specified later when configuring the identity store in Kafka server.
Key Password	Specify a password 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.

Table 10-2 (Cont.) Generate Keystore - Attributes

Attributes	Description
First and Last Name (CN)	Enter the domain name of the machine. For example, www.example.com.
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.
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.
Name of your City or Locality	The city in which your organization is physically located. For example, Bengaluru.
Name of your State or Province	The state/province in which your organization is physically located. For example, Karnataka.
Two-letter Country Code for this Unit	The country in which your organization is physically located. For example, US, UK, IN, etc.

Example 10-1 Sample Execution

Listed below is the result of a sample execution.

```
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>

Export Private Key as Certificate

Export private key as certificate 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>
```

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`>

Import the Cert and Generate TrustStore

To import the cert and generate TrustStore, the command is mentioned below:

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

Table 10-3 Generate TrustStore - Keyword Details

Keyword	Description
`alias`	It 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.
`cert_file`	It is the location of the file containing the PKCS#7 formatted reply from the CA, containing the signed certificate.
`truststore`	It is the location where the TrustStore should be generated.
`storepass`	It is the password for the TrustStore.

The user can generate two TrustStores from the same cert.

- One used for Kafka server
- One 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
```

Three Keystore files are required for this method as given in the table below:

Table 10-4 Keystore Files

File Name	Description
`KafkaServerKeystore.jks`	Keystore file for Kafka brokers
`KafkaServerTrustStore.jks`	TrustStore file for server
`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. The user can generate and place these files on all the different servers of Kafka so that they can be accessed by server*.properties file. The `KafkaClientTrustStore.jks` should be placed on the server, which is accessible by the microservices also.

Create Users in Zookeeper

To create users in Zookeeper, follow below steps:

1. Start the zookeeper.

   > **Note:**
   >
   > Refer to Zookeeper Cluster Setup topic.

2. Follow the below steps for user creation.
   a. Execute the admin command for admin 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
```

> **Note:**
>
> The user created with admin as username and password is setup for the user for each scram mechanism. Here, the user **admin** is used for Kafka broker auth.

b. Execute the test command for test user creation.

```
./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
```

> **Note:**
>
> The user created with test as username and password is setup for the user for each scram mechanism. Here, the user **test** is used for client auth.

11

Oracle Banking Microservices Architecture Deployments

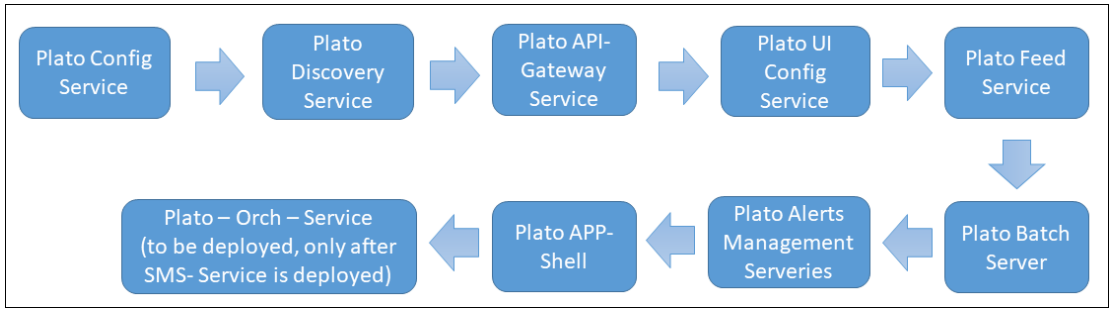
This topic describes about the Oracle Banking Microservices Architecture deployments.

Prerequisites

Before proceeding, make sure that the previous steps are completed.

Oracle Banking Microservices Architecture Applications Deployment Order

Figure 11-1 Deployment Order



The below table provides the deployments required on each server to run the Oracle Banking Microservices Architecture Application.

Table 11-1 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
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

Table 11-1 (Cont.) Installation Summary for Oracle Banking Microservices Architecture Services

Application	Archive name	OSDC path	Targets
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

Note:

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

Eventhub based applications should not be deployed in the admin server.

Steps to Deploy as Application

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

SSL Configuration

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

Table 11-2 SSL Configuration - Parameters

Key	Default Value	Purpose
${apigateway.protocol}	https	Only for API gateway protocol. it must be **https only**.
${eureka.protocol}	https	For inter-service communication protocol. Values can be **http** or **https**.

Table 11-2 (Cont.) SSL Configuration - Parameters

Key	Default Value	Purpose
${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:

- Appshell needs to be secured with SSL.
- Api-Gateway needs to be secured with SSL.
- Appshell to Api-gateway communication should happen over SSL.
  The api-gateway URL mentioned as -D parameter for appshell should be SSL enabled (https-based).

12

Restart and Refresh

This topic describes about restart and refresh the servers.

Once everything is deployed, restart all the managed servers. For each application, call path `/refresh` to refresh the configuration properties.

Restart the Servers

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

13
Logging Area

This topic describes the logging area where the Oracle Banking Microservices Architecture Applications deployed in the WebLogic server.

Plato 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 is created **platoinfra_domain** in the server `/scratch/oracle/middleware/user_projects/domains/platoinfra_domain`.

Logging area for Plato

`=<URL>`

14

Known Issues - Resolutions

This topic describes about the 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.

Index

Z