Plato Infrastructure Services Installation Guide
Oracle Banking Branch
Release 14.4.0.3.0
Part No. F39512-01
February 2021

ORACLE
FINANCIAL SERVICES

Table of Contents

ORACLE®

ORACLE®

1. Preface

1.1 Introduction

This guide helps you to install the Plato infrastructure 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.

It is recommended to use dedicated managed server for each of the Plato infrastructure 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 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 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

ORACLE

2. Database Setup

2.1 Introduction

In this section you are going to setup database related configuration for PLATO 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 Plato Infrastructure Domain Configuration

3.1.1 Prerequisites

- Machine should have Java JDK has installed.

- Oracle Fusion Middleware Infrastructure 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	jdbc/PLATOFEED	Plato-Feed-Server
PLATOALERTS	jdbc/PLATOALERTS	Plato-Alerts-Management-Server
PLATOBATCH	jdbc/PLATOBATCH	Plato-Batch-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. 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)

Plato 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)
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>

Plato 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  →  Plato  →  SMS  →  Common  →  Products
Software                              Core
```

7. Plato Infrastructure Software Deployment

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

7.1 Zookeeper Cluster Setup

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

7.1.1 Pre-requisite

JDK should be 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

7.1.2 Installation

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

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

- Inside the <zookeeper 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.

- Create a configuration file named zoo.cfg at   <zookeeper **home** directory> /zookeeper_3.6.2/config
  Add the following set of properties and values to that 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
node.
```

NOTE: Any odd number of zookeeper servers can be configured under the cluster.

ORACLE®

- Start the zookeeper on each node machine

  Navigate to <zookeeper home directory>/zookeeper_3.6.2 and execute the below command:

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

- 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 functioning 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
```

7.2 Kafka Cluster Setup

7.2.1 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

7.2.2 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, clear it.

- Edit the below lines in the
  <kafka home directory>/kafka_2.13-2.6.0/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
eeper_client_port>, …
```

- To start the Kafka, navigate to <kafka home directory>/kafka_2.13-2.6.0/ 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_2.13-2.6.0/ folder and run the below command:

```
/bin/kafka-topics.sh --create –zookeeper<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_2.13-2.6.0/* folder and run the below command:

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

- To describe the topic, navigate to *<kafka home directory>/kafka_2.13-2.6.0/* folder and run the below command:

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

- To start a producer, navigate to *<kafka home directory>/kafka_2.13-2.6.0/* 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

7.3 KAFKA Security Setup

7.3.1 Pre-requisite

Perform the following steps:

1. Install JDK in all node machines.

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

7.3.2 Installation

7.3.2.1 Generate Keystore

The command is mentioned below:

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

-keystore **keystore**

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

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. 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 Country Code for this Unit: The country in which your organization is physically located. For example US, UK, IN etc.

For 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>

7.3.2.2 <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 is displayed:

Certificate stored in file < KafkaCert.cer>

7.3.2.3 Import the Cert and generate TrustStore

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 mentioned in *Section 7.3.2.1*.
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.

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

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

7.3.2.4 Creation of users in Zookeeper

Follow the below steps to create user in Zookeeper:

1. Start the zookeeper. Refer to *Section 7.1.2* for the command.
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.

7.3.2.5 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 to *Section 7.2.2* for the command.

7.3.2.6 <u>Changes to Clients</u>

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®

7.3.2.7 <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

For example:

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

ORACLE

7.4 Tesseract Installation

7.4.1 Pre-requisite

7.4.1.1 Build Tools

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.

7.4.1.2 Dependent Libraries

These libraries should be present in the server. By default, these libraries are available in Oracle Linux. If these libraries are not present, 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, install the libraries from official Oracle repo or any other repo available for that distribution.

7.4.1.3 Installation Files

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

<Unzip the file>/THIRD_PARTY_SOFTWARES/Tesseract

The list of files present in the directory are as follows:

1. leptonica-1.76.0.tar.gz
2. tesseract-4.1.0.tar.gz
3. eng.traineddata
4. osd.traineddata

ORACLE

7.4.2 Installation

7.4.2.1 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.

7.4.2.1.1 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
sudo make install
```

If the installation is successful, then go to *Section 7.4.2.2*. Else, go to *Section 7.4.2.1.2*.

7.4.2.1.2   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.

ORACLE®

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

7.4.2.2 <u>Leptonica Configuration</u>

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

- Add the leptonica installation directory (For example: /usr/local/lib ,/usr/lib, /usr/lib64 etc) in library path.

- leptonica header path (For example: /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.)

- The **.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

7.4.2.3 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
sudo make install
```

- 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
```

7.4.2.4 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
```

- 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.

ORACLE

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

- 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
```

7.5 Conductor Installation

7.5.1 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.

7.5.2 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

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

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.

ORACLE

8. Security- SSL Encryption with SASL-SCRAM Authentication

8.1 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:

1. 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.

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, when configuring the SSL attributes of the Kafka Server.

3. First and Last Name (CN): Enter the domain name of the machine, for instance, 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 Country Code for this Unit: The country in which your organization is physically located, for example, US, UK, IN, etc.

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>

8.2 <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>

ORACLE

8.3 <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 2 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

So 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

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.

ORACLE

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

Start the zookeeper (command in *Section 7.1*) 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.

ORACLE

9. Plato Deployments

9.1 Pre-requisite

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 Plato application to run.

9.2 Plato Applications Deployment Order

```
Plato Config      Plato          Plato API-       Plato UI-        Plato Feed
Service      →    Discovery   →   Gateway     →    Config      →   Services
                  Service         Service          Service
```

```
Plato-Orch-Service                                 Plato Alerts
(To be deployed, only after  ←  Plato App-   ←    Management    ←   Plato Batch
 SMS-Servce is deployed)         shell             Services           Server
```
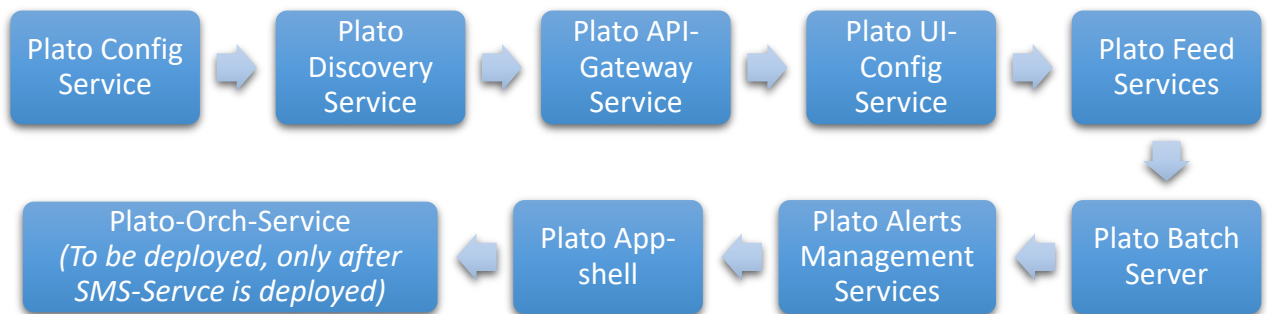
Installation Summary for Plato Services:

Application	Archive name	OSDC path	Targets
Plato-config-service	plato-config-service-5.8.0.war	{**unzip the file**}PLATO\plato-config-service\	Config Server
Plato-discovery-service	plato-discovery-service-5.4.0.war	{**unzip the file**}PLATO\plato-discovery-service\	Discovery Server
Plato-api-gateway	plato-api-gateway-5.7.0.war	{**unzip the file**}PLATO\plato-api-gateway\	Api Gateway
Plato-ui-config-service	plato-ui-config-service-5.5.0.war	{**unzip the file**}PLATO\plato-ui-config-service\	Plato UI Config
Plato-Orch-Service *(To be deployed after sms-service is deployed)*	plato-orch-service-5.8.0.war	{**unzip the file**}PLATO\plato-orch-service\	Plato-Orch-Service
Plato-Feed-Services	plato-feed-services-5.7.0.war	{**unzip the file**}PLATO\plato-feed-services\	Plato-Feed-Services

Application	Archive name	OSDC path	Targets
Plato-Batch-Server	Plato-Batch-Server-5.6.0.war	{**unzip the file**}PLATO\plato-batch-server\	Plato-Batch-Server
Plato-Alerts-Management-Services	Plato-Alerts-Management-Services-5.7.0.war	{**unzip the file**}PLATO\plato-alerts-management-services\	Plato-Alerts-Management-Server
Appshell	app-shell-5.8.0.war	{**unzip the file**}UI\app-shell.war	Appshell Server

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

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

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

9.4 <u>SSL Configuration</u>

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).

ORACLE

10. Restarts and Refresh

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

10.1 <u>Restarting Servers</u>

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

ORACLE

11.1 <u>Introduction</u>

This section describe about the logs area where after deployment of Plato Applications in WebLogic server.

11.2 <u>Logging Area</u>

Plato Application writes logs in the below area of the server:

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

Let's assume 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>

ORACLE

ORACLE®

Plato Infrastructure Services Installation Guide

February 2021

Version 14.4.0.3.0

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