ANNEXURE – 4

Oracle Banking Branch

Release 14.5.0.9.0

Part Number F85404-01

August 2023

ORACLE

Table of Contents

1. PREFACE 1-1

1.1 PURPOSE 1-1

1.2 AUDIENCE 1-1

1.3 DOCUMENT ACCESSIBILITY 1-1

1.4 ACRONYMS AND ABBREVIATIONS 1-1

1.5 RELATED DOCUMENTS 1-1

2. ANNEXURE - 4 2-2

2.1 INTRODUCTION 2-2

2.2 HOW TO DO MULTI NODE SETUP (HIGH AVAILABILITY ARCHITECTURE) 2-2

2.2.1 Configuration Server Related Changes 2-2

2.2.2 Plato UI Configuration Server Related Changes 2-3

2.2.3 setDomainEnv.sh related changes 2-3

2.2.4 Requirement of Load Balancers 2-4

2.3 SAMPLE HA PROXY CONFIGURATION 2-5

1. Preface

1.1 Purpose

This guide is a supporting document for the installation of Oracle Banking Microservices Architecture applications. The user can find the reference in the respective installation guides.

1.2 Audience

This guide is intended for WebLogic admin or ops-web team who are responsible for installing OFSS Banking Products.

1.3 Document 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 Acronyms and Abbreviations

Abbreviation	Description
SMS	Security Management System
CMC	Common Core
MOC	Manufacturing Operations Center

1.5 Related Documents

The Related document list are as follows:

- Oracle Banking Microservices Architecture Installation Guides
- Product Installation Guides

2. ANNEXURE - 4

2.1 Introduction

This guide is a supporting document for the installation of Oracle Banking Microservices Architecture applications. You can find the reference in the respective installation guides.

2.2 How to do Multi Node setup (High Availability Architecture)

2.2.1 Configuration Server Related Changes

The below changes are to be made in the PROPERTIES table pointed to by the Configuration Server:

- **For the Discovery Server:**

  PLATO Discovery Service should have an entry for its entire peer PLATO Discovery Services configured through **eureka.client.serviceUrl.defaultZone**. This will hold a comma-separated list of all the peer PLATO Discovery services.

  In addition, to enable the peer aware mode for the PLATO Discovery Service we should set the **eureka.client.register-with-eureka** to true.

ID	APPLICATION	PROFILE	LABEL	KEY	VALUE
1	plato-discovery-service	jdbc	jdbc	eureka.client.serviceUrl.defaultZone	http://<IP of the server where the first instance of PLATO Discovery Service is running>:<PORT where the first instance of PLATO Discovery Service is running>/plato-discovery-service/eureka,http://<IP of the server where the second instance of PLATO Discovery Service is running>:<PORT where the second instance of PLATO Discovery Service is running>/plato-discovery-service/eureka
2	plato-discovery-service	jdbc	jdbc	eureka.client.register-with-eureka	true
3	plato-discovery-service	jdbc	jdbc	server.port	<< PORT Number where the PLATO Discovery Service is running >>

- **For the Individual Services:**

  Each service should have an entry of all the PLATO Discovery Services configured through **eureka.client.serviceUrl.defaultZone**. This will hold a comma separated list of all the PLATO Discovery services.

ID	APPLICATION	PROFILE	LABEL	KEY	VALUE
1	<<service-name>>	jdbc	jdbc	eureka.client.serviceUrl.defaultZone	http://<IP of the server where the first instance of PLATO Discovery Service is running>:<PORT where the first instance of PLATO Discovery Service is running>/plato-discovery-service/eureka,http://<IP of the server where the second instance of PLATO Discovery Service is running>:<PORT where the second instance of PLATO Discovery Service is running>/plato-discovery-service/eureka

2.2.2 Plato UI Configuration Server Related Changes

For each of the product registered in **PRODUCT_SERVICES_ENV_LEDGER**, we need to change the URL to point to the Load Balancer of the PLATO API Gateway Service.

ID	PRODUCT_NAME	URL
1	<<PRODUCT NAME>>	<< HTTP URL OF THE LOAD BALANCER >>

2.2.3 setDomainEnv.sh related changes

- **For all the Micro Services:**

  Individual MICRO services should now access the PLATO Config Service via the Load Balancer URI (i.e). configured in the server runtime through the property **plato.services.config.uri**.

  The **plato.services.config.uri** must point to the URI of the load balancer. The format of the same would be as follows:

```
-Dplato.services.config.uri=http://<< IP OF THE LOAD BALANCER >>:<< PORT OF THE LOAD BALANCER >>
```

- **For the UI APPSHELL:**

  UI APPShell should now access the API Gateway Router Service via the Load balancer URI (i.e.) configured in the server runtime. For example, **Dapigateway.url**.

  The **apigateway.url** must point to the host and port of the load balancer. The format of the same would be as follows:

```
-Dapigateway.url=http://<< IP OF THE LOAD BALANCER >>:<< PORT OF THE LOAD BALANCER >>
```

If you need to install the services of Oracle Banking Microservices Architecture in more than two nodes, it is not possible to maintain the value of the eureka URL in the properties table due to the size restriction. In such cases, you can remove the following key from the properties table and add in the *setuseroverrides.sh* file.

```
-Deureka.client.serviceUrl.defaultZone
```

2.2.4 Requirement of Load Balancers

Load Balancers are required for PLATO API GATEWAY Service, PLATO Configuration Service, and PLATO UI APP SHELL.

2.2.4.1 PLATO API Gateway Router Service

PLATO API Gateway Router Service acts as a single point of entry for UI and External Systems to access the underlying services. This service will route requests to respective services via PLATO API GATEWAY Service. In a multi node deployment where multiple PLATO API Gateway Router Services are deployed, we would need a single URI for accessing the multi node deployments of the PLATO API Gateway Router Services. This Load Balancer would help us to achieve that functionality.PLATO Configuration Service

2.2.4.2 PLATO UI APP SHELL

The PLATO UI App Shell acts as the single user interface entry point for the users. In a multi-node deployment, where multiple instances of PLATO UI APP SHELL are deployed, users need a single URI for accessing the multi-node deployments of the PLATO UI APP SHELL. Load Balancer setup will help to achieve this.

In addition to the “App Shell,” the UI of the application is serviced by additional UI “component server” applications. These are for SMS, CMC, MOC, and the respective product domain too. All these UI component server applications need to be deployed in the same managed server, where PLATO UI APP SHELL war is deployed.

If the deployment is in a cluster with more than one managed server for UI applications, then all the UI applications need to be deployed in the clustered managed servers, and appropriate load balancer setup need to be done for all the UI applications.

2.3 Sample HA Proxy Configuration

A load balancer such as HAProxy, NGINX, Oracle HTTP Server, etc. may be used for high-availability." in case there isn't an existing general reference to load balancers.

Sample basic configuration in HAProxy for API Gateway (set in `etc/haproxy/haproxy.conf`)

frontend LBGateway

bind load.balancer.ip.address:port

default_backend LBGateway

backend LBGateway load.balancer.ip.address:port

mode http

balance roundrobin

option httpchk

option http-keep-alive

option forwardfor

option httpchk HEAD /app-shell

server <backend_server_name_1> api.gateway1.ip.address:api_gateway_port_1 check

server <backend_server_name_1> api.gateway2.ip.address:api_gateway_port_2 check