Oracle® Fusion Middleware

Administering Oracle User Messaging Service

14*c* (14.1.2.0.0)
F89236-02
February 2025

Oracle Fusion Middleware Administering Oracle User Messaging Service, 14*c* (14.1.2.0.0)

F89236-02

Primary Author: Oracle Corporation

Contents

4 Configuring Oracle User Messaging Service

Preface

This guide describes how to administer Oracle User Messaging Services (UMS). This includes how to configure and deploy user messaging drivers and other components, how to enable security in UMS, and how to monitor UMS using Oracle Enterprise Manager Fusion Middleware Control.

Audience

This document is intended for UMS administrators, who are responsible for configuring and monitoring UMS.

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.

Access to Oracle Support

Oracle customers that have purchased support have access to electronic support through My Oracle Support. For information, visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info or visit http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs if you are hearing impaired.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documents

For more information, see the following documents:

- *Release Notes for Oracle Fusion Middleware Infrastructure*
- *Developing Applications with Oracle User Messaging Service*
- *User Messaging Service Java API Reference*
- *WLST Command Reference for Infrastructure Components*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in Oracle User Messaging Service

This chapter describes the features and improvements in Oracle User Messaging Service (UMS). The following topics introduce the new and changed features of UMS and other significant changes in this guide, and provides pointers to additional information.

New and Changed Features for Release 14c (14.1.2.0.0)

- **OAuth Support for Email Driver**: UMS provides support for OAuth 2.0 based authentication to Gmail and Microsoft exchange which enables users to authorize one app or service to sign into another without divulging private information such as passwords. For more information, see Configuring Email Driver with OAuth in Administering Oracle User Messaging Service.
- **Enhancements to UMS Drivers**: UMS supports the new Google Cloud Messaging (GCM) driver used for mobile push notification service that can send mobile push notifications to Android applications. Enhancements to APNS and XMPP drivers include updates to configuration, common properties, and custom properties and support for Twitter Driver is dropped. For more information, see Administering Oracle User Messaging Service.
- **Support for Saving Driver Properties**: UMS provides support to save driver specific properties in the UMS database. When the UMS driver appears, the properties stored in the DRIVERPROPERTIES table for specific fields override the values in the file and the driver instance uses those property values from the database to process requests. For more information, see Saving Driver Properties in Administering Oracle User Messaging Service.
- **SMPP Custom Property**: New SMPP custom property field Optional Params is introduced which enables passing of additional parameters along with SMS. For more information, see SMPP Custom Properties in Administering Oracle User Messaging Service.

1

Introduction to Oracle User Messaging Service

This chapter introduces you to Oracle User Messaging Service (UMS), and includes the following topics:

- Overview
  UMS enables two-way communication between users and deployed applications.
- Introduction to Oracle User Messaging Service Configuration
  To enable UMS to send and receive messages, use Oracle Enterprise Manager Fusion Middleware Control to set the UMS environment by configuring the appropriate drivers in the domain or cluster. UMS includes drivers that support messaging through various channels, for instance, email, IM, SMS.

Overview

UMS enables two-way communication between users and deployed applications.

Key features include:

- Support for a variety of messaging channels: Messages can be sent and received through various channels, for instance, email, instant messaging (IM) (XMPP), short message service (SMS) (SMPP).
- Two-way messaging: In addition to sending messages from applications to users (referred to as *outbound* messaging), users can initiate messaging interactions (inbound messaging). For example, a user can send an email or text message to a specified address; the message is routed to the appropriate application that can then respond to the user or invoke another process according to its business logic.
- User messaging preferences: End users can use a web interface to define preferences for how and when they receive messaging notifications. Applications immediately become more flexible; rather than deciding whether to send to a user's email address or IM client, the application can simply send the message to the user, and let UMS route the message according to the user's preferences.

> **Note:**
>
> The User Messaging Preferences UI is available at `http://host:port/sdpmessaging/userprefs-ui` , or `https://host:sslport/sdpmessaging/userprefs-ui`.

- Robust message delivery: UMS keeps track of delivery status information provided by messaging gateways, and makes this information available to applications so that they can respond to a failed delivery. Or, applications can specify one or more *failover* addresses for a message in case delivery to the initial address fails. Using the failover capability of UMS frees application developers from having to implement complicated retry logic. This retry logic is also supported by the automatic resend feature that is introduced in 12*c*.
- Pervasive integration within Oracle Fusion Middleware: UMS is integrated with other Fusion Middleware components providing a single consolidated bi-directional UMS.

  - Integration with Oracle BPEL Process Manager: Oracle JDeveloper includes prebuilt BPEL activities that enable messaging operations. Developers can add messaging capability to a SOA composite application by dragging and dropping the desired activity into any workflow.
  - Integration with human workflow: UMS enables the human workflow service engine to send actionable messages to and receive replies from users over email.
  - Integration with Oracle BAM: Oracle BAM uses UMS to send email alerts in response to monitoring events.
  - Integration with Oracle WebCenter Portal: UMS APIs are available to developers building applications for Oracle WebCenter Portal: Spaces. For more information on Oracle WebCenter Portal, please refer to *Oracle Fusion Middleware Building Portals with Oracle WebCenter Portal*.

- Components
  There are three types of components that comprise UMS. These components are standard Jakarta EE applications, making it easy to deploy and manage them using the standard tools provided with Oracle WebLogic Server.
- Architecture
  This section describes the system architecture of the User Messaging Service.

Components

There are three types of components that comprise UMS. These components are standard Jakarta EE applications, making it easy to deploy and manage them using the standard tools provided with Oracle WebLogic Server.

- UMS Server: The UMS Server orchestrates message flows between applications and users. The server routes outbound messages from a client application to the appropriate driver, and routes inbound messages to the correct client application. The server also maintains a repository of previously sent messages in a persistent store, and correlates delivery status information with previously sent messages.
- UMS Drivers: UMS Drivers connect UMS to the messaging gateways, adapting content to the various protocols supported by UMS. Drivers can be deployed or undeployed independently of one another depending on what messaging channels are available in a given installation.
- UMS Client applications: UMS client applications implement the business logic of sending and receiving messages. A UMS client application might be a SOA application that sends messages as one step of a BPEL workflow, or a WebCenter Portal Spaces application that can send messages from a web interface.

In addition to the components that comprise UMS itself, the other key entities in a messaging environment are the external gateways required for each messaging channel. These gateways are not a part of UMS or Oracle WebLogic Server. Since UMS Drivers support widely-adopted messaging protocols, UMS can be integrated with existing infrastructures such as a corporate email servers or XMPP (Jabber) servers. Alternatively, UMS can connect to outside providers of SMS services that support SMPP.

Architecture

This section describes the system architecture of the User Messaging Service.

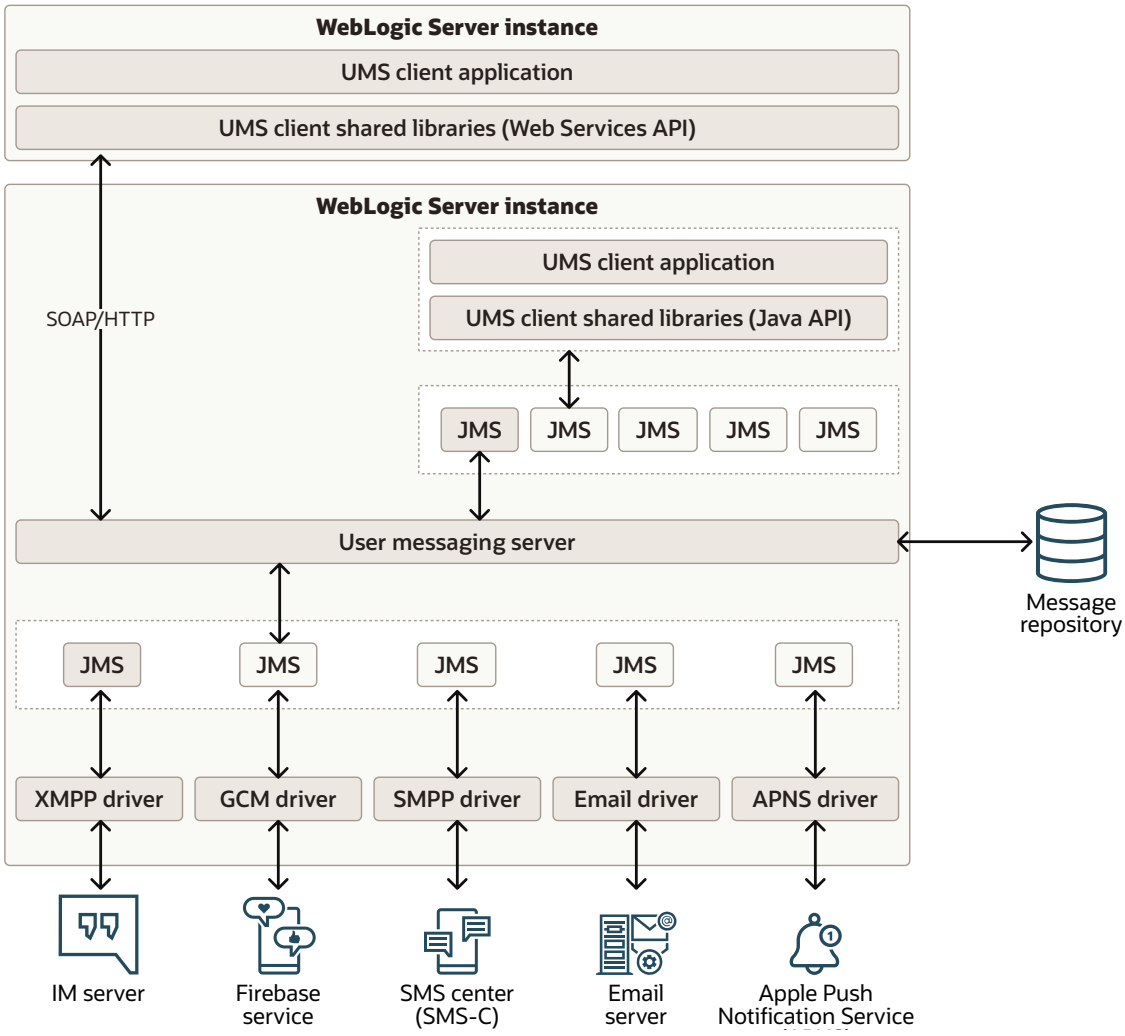
In 14*c*, UMS is available as a part of JRF. This enables easier upper stack integration. For more information about configuring your domain using JRF templates, refer to chapter

Configuring your Oracle Fusion Middleware Infrastructure Domain in *Oracle Fusion Middleware Installing and Configuring the Oracle Fusion Middleware Infrastructure*.

For maximum flexibility, the components of UMS are separate Jakarta EE applications. This allows them to be deployed and managed independently of one another. For example, a particular driver can be stopped and reconfigured without affecting message delivery on all other channels.

Exchanges between UMS client applications and the UMS Server occur as SOAP/HTTP web service requests for UMS Web Services API clients, or through remote Jakarta Enterprise Beans (EJBs) and JMS calls for UMS Java API clients. Exchanges between the UMS Server and UMS drivers occur through JMS queues.

Figure 1-1 UMS Architecture



Introduction to Oracle User Messaging Service Configuration

To enable UMS to send and receive messages, use Oracle Enterprise Manager Fusion Middleware Control to set the UMS environment by configuring the appropriate drivers in the domain or cluster. UMS includes drivers that support messaging through various channels, for instance, email, IM, SMS.

For more information about configuring UMS, see Configuring Oracle User Messaging Service

For workflow participants to receive the notifications, they must register the devices that they use to access messages through User Communication Preferences. For more information, see chapter User Communication Preferences in *Oracle Fusion Middleware Developing Applications with Oracle User Messaging Service*.

> **Note:**
>
> Some details in the API may vary between the underlying protocols. Study the driver description chapter in the Administration Guide (http://docs.oracle.com/middleware/1213/ums/administer/ns_descriptions.htm#UMSAG97610) and in particular the UMS API Programmer Notes sections.

2

Getting Started with Oracle User Messaging Service

This chapter helps you get started with Oracle User Messaging Service (UMS). It discusses how to install and upgrade UMS. It also discusses the procedures needed for achieving high availability and moving from a test to a production environment.

- Installing User Messaging Service
  This section describes the procedures to install the User Messaging Service.
- Upgrading User Messaging Service
  This section describes how to upgrade the User Messaging Service to UMS 14c.
- Scalability and High Availability
  You can achieve a highly scalable environment for UMS. To achieve high scalability, UMS scales horizontally by adding new identically configured nodes. This means, the same type of drivers and UMS applications are deployed on each node.
- Moving from a Test to a Production Environment
  This framework eases the moving of Oracle Fusion Middleware components from a test environment to a production environment.

Installing User Messaging Service

This section describes the procedures to install the User Messaging Service.

For detailed information about installing and configuring UMS, see *Oracle Fusion Middleware Installing and Configuring the Oracle Fusion Middleware Infrastructure*. The installation of UMS includes the following main procedures:

- Running the Repository Creation Utility (RCU) to create the database schemas

  When running the RCU, select **User Messaging Service** in the list of components. For information about creating the database schemas, see Configuring your Oracle Fusion Middleware Infrastructure Domain in *Oracle Fusion Middleware Installing and Configuring the Oracle Fusion Middleware Infrastructure*.
- Extending your domain using the UMS template

  The UMS templates are located at *ORACLE_HOME*`/oracle_common/common/templates/wls`. For more information about UMS templates, see *Oracle Fusion Middleware Domain Template Reference*. In the WebLogic Server Configuration Wizard, extend the domain using `oracle.ums.basic_template.jar` to set up the UMS JDBC properties and target the UMS server along with the chosen drivers to your managed servers or clusters. The *Oracle User Messaging Service Basic* template is a quick start template that defines the managed server, `ums_server1`, and targets all UMS components to that server.

> **Note:**
>
> The UMS Client API is packaged in a shared library that an UMS client application must reference. The shared library is available where the UMS Server and the JRF template has been installed. However, when the client application is running on a managed server other than UMS, and uses the UMS Web Services API, and if the JRF template is not used then the client shared library must be deployed explicitly. This can be achieved by extending the domain where the client application is deployed with the UMS Client API template available at `$ORACLE_HOME/common/templates/wls/oracle.ums.client_template.jar`. The UMS Client API shared library is called *oracle.sdp.client*.

Upgrading User Messaging Service

This section describes how to upgrade the User Messaging Service to UMS 14c.

For detailed information about upgrading to UMS 14c, see *Oracle Fusion Middleware Upgrading to the Oracle Fusion Middleware Infrastructure*.

UMS provides a Schema Upgrade plug-in and a Config Upgrade plug-in to the Oracle Fusion Middleware Upgrade Assistant. The config upgrade plug-in handles the change from application level configuration in 12c to domain level configuration in 14.1.2. This includes copying the application configuration from remote managed servers to the Administration Server and merging it into the UMS domain level configuration file. For information about upgrading from 12c to 14.1.2 using Upgrade Assistant, see *Oracle Fusion Middleware Upgrading with the Upgrade Assistant*.

Scalability and High Availability

You can achieve a highly scalable environment for UMS. To achieve high scalability, UMS scales horizontally by adding new identically configured nodes. This means, the same type of drivers and UMS applications are deployed on each node.

This gives UMS linear scalability up to the point where the scalability of JMS or the scalability of data storage becomes the limiting factor.

Since the WS-UMS Server application component is deployed on the UMS node, it scales the same way as UMS does. On the other hand, since the WS-UMS Client application is deployed on separate machines, the scalability is up to the design of that component.

UMS supports the following deployment scenarios for scaling up your environment:

- Instances of UMS deployed in a domain are configured identical and deployed identical. No limitation on the number of servers or how the domain/clusters are set up.
- Instances of UMS deployed in a domain that have different configuration (server and/or driver) must be deployed in separate clusters.

For detailed information about scaling up your environment, see *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

Whole Server Migration is a key High Availability feature that UMS supports. For more information about Whole Server Migration, see *Oracle Fusion Middleware Administering Clusters for Oracle WebLogic Server*.

High availability for UMS can be achieved through the following ways:

- Automatic reconnects to external gateways
- Message resends and failover chains
- Persistence layer recovery handling database connections problems

For detailed information about high availability, see *Oracle Fusion Middleware High Availability Guide*.

Moving from a Test to a Production Environment

This framework eases the moving of Oracle Fusion Middleware components from a test environment to a production environment.

For details about the procedures used for moving Oracle Fusion Middleware from a test environment to a production environment, see *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

Most configuration in a test environment is handled automatically by this framework. Components that require custom actions implement these in the T2P plug-ins. UMS provides a T2P plug-in to this framework. Some User Preferences data is stored in a database and requires the T2P plug-in to move that data from a test database to a production database. Also, the T2P plug-in extracts preselected driver configuration properties, like host names and ports to the `moveplan.xml` file so that the settings can be prepared for the production environment before the production system is started.

For detailed information about moving from a test to a production environment, see *Oracle Fusion Middleware Administering Oracle Fusion Middleware*.

3

Oracle User Messaging Service Drivers

This chapter describes the purpose, features and limitations of Oracle User Messaging Service (UMS) drivers.
UMS drivers contain protocol specific implementation to connect UMS to various messaging gateways, for instance, email servers, or short message service centers (SMSC), and so on. Drivers can be deployed or undeployed, independently of one another, depending on the availability of messaging channels in a given installation.

- Email Driver
  The Email Driver sends and receives messages. It supports all relevant email protocols, SMTP for sending emails, and IMAP and POP3 for receiving emails, to be able to communicate with every standard mail server.
- SMS (SMPP) Driver
  The Short Message Peer-to-Peer (SMPP) protocol is a TCP/IP based industry protocol for exchanging SMS messages between SMS peer entities such as short message service centers (SMS-C) and/or External Short Messaging Entities (ESME). The UMS SMPP Driver is implemented as an ESME. It is based on SMPP protocol v3.4.
- XMPP Driver
  The XMPP Driver provides unidirectional or bidirectional access from Oracle Fusion Middleware to end users for real-time IM through the Extensible Messaging and Presence Protocol (XMPP). This driver enables end users to receive alert notifications or interactively chat with applications through the IM client of their choice.
- Extension Driver
  The Extension Driver sends messages to the configured Endpoint URL that implements the Notification WebServices interface defined by UMS. A messaging gateway, or an adapter to a gateway, can implement this Web Service, thereby, extending UMS with no changes in UMS.
- APNS Driver
  The APNS driver is available in User Messaging Service (UMS) 12.2.1 and onwards. It provides support for sending a UMS message as a notification to Apple iOS devices such as an iPhone or an iPad.

Email Driver

The Email Driver sends and receives messages. It supports all relevant email protocols, SMTP for sending emails, and IMAP and POP3 for receiving emails, to be able to communicate with every standard mail server.

Support is implemented for enabling the security protocols, TLS or SSL, to protect email contents on the wire. The Email Driver uses JavaMail v1.4, which is the standard Java component that implements the required protocols and clients, to create, access, send, and receive emails.

The Email Driver uses a multi-threaded design to be able to poll multiple mail boxes (over IMAP or POP3 protocol). One limitation is that if there is only one mail box to poll, this leads to only one thread working on that box.

> **Note:**
>
> POP3 is deprecated protocol and should be avoided.

- Scalability notes
- High Availability notes
- Compatibility notes
- UMS API Programmer notes

Scalability notes

The Email Driver can be scaled out to multiple nodes.

High Availability notes

When the connection to the Email Server fails, the Email Driver will wait and retry. The wait period and maximum retries are configurable.

Compatibility notes

The Email Driver is compatible with these protocols: POP3, IMAP4, and SMTP. Table 3-1 lists the Email Driver gateway vendors and their versions.

Table 3-1 Email Driver Gateway Vendors and Versions

Vendor	Version
Oracle Beehive	Release 1 (1.4.3)
Oracle Collaboration Suite	10g Release 1 (10.1.2)
Microsoft Exchange	2003
Dovecot (IMAP4/POP3)	0.99.11
sendmail (SMTP)	8.13.1

The UMS Message API Priority Levels are translated into the Email message header "X-Priority" as follows:

```
MessagePriorityType.LOWEST => header value "5"
MessagePriorityType.LOW => header value "4"
MessagePriorityType.NORMAL => header value "3"
MessagePriorityType.HIGH => header value "2"
MessagePriorityType.HIGHEST => header value "1"
```

UMS API Programmer notes

The Email Driver handles transformation of a UMS message to an email message (and vice versa), including headers, content type and charset encoding, MIME multipart and body parts. For an incoming message, the UMS Message recipient is the email address of the email box that received the mail. You can tell if the recipient was in the email To, CC or BCC field, by looking at the Address metadata.

When an outbound email is not deliverable, a Delivery Status Notification (DSN) can be created by the recipient's mail server and sent back to the sender's email address. The Email Driver tries to match a received DSN to the outbound email which the DSN is referring to and create a “failed to deliver" status of that outbound message. In such a case, a successful delivery status may, after some time, be changed to a failed delivery status.

SMS (SMPP) Driver

The Short Message Peer-to-Peer (SMPP) protocol is a TCP/IP based industry protocol for exchanging SMS messages between SMS peer entities such as short message service centers (SMS-C) and/or External Short Messaging Entities (ESME). The UMS SMPP Driver is implemented as an ESME. It is based on SMPP protocol v3.4.

If the sending feature is enabled, the SMPP driver opens one TCP connection to the SMS-C as a transmitter for sending. If the driver's receiving feature is enabled, it opens another connection to the SMS-C as a receiver for receiving. Only two TCP connections (both initiated by the driver) are needed for all communication between the driver and the SMS-C.

Scalability notes

The SMPP Driver can be scaled out to multiple nodes. There would be a limitation only if the SMS-C does not allow multiple login by the same account.

High Availability notes

If the connection to the SMS-C is lost, the SMPP driver will periodically attempt to reconnect.

Compatibility notes

The SMPP driver is based on SMPP Protocol v3.4. Table 3-2 lists the SMPP Driver gateway vendors.

Table 3-2 SMPP Driver Gateway Vendors

Vendor
Syniverse
Clickatell
Logica CMG
OpenSMPP (simulator)

UMS API Programmer notes

The outgoing text message is a concatenation of the UMS Message Subject and Content. The incoming SMS Content is put in the UMS Message Content. The UMS Message Subject will be empty.

The UMS Message API Priority Levels are translated into SMPP Priority as follows:

MessagePriorityType.LOWEST, MessagePriorityType.LOW, MessagePriorityType.NORMAL => Smpp Priority: 0

MessagePriorityType.HIGH => Smpp Priority: 1

MessagePriorityType.HIGHEST => Smpp Priority: 2

However, the Smpp Priority may be limited by the SMPP Driver configuration parameter “PriorityAllowed".

The SMPP Driver honors the UMS API MessageInfo “expiration" data and passes that information with the message to the SMS-C.

XMPP Driver

The XMPP Driver provides unidirectional or bidirectional access from Oracle Fusion Middleware to end users for real-time IM through the Extensible Messaging and Presence Protocol (XMPP). This driver enables end users to receive alert notifications or interactively chat with applications through the IM client of their choice.

XMPP is an open XML-based protocol for IM and consists of a client/server architecture, which resembles the ubiquitous email network. Recipients are addressed by an XMPP ID (or Jabber ID or JID) with the following form: [username]@domain[/resource]. To use the XMPP Driver in UMS, you must have access to a Jabber/XMPP server and an XMPP account for the UMS XMPP Driver instance with which to log in.

An end user of XMPP connects to an XMPP server using an XMPP client to send instant messages to other XMPP users. XMPP, however, is not the only protocol network available for IM. XMPP has an extensible and modular architecture. It integrates with proprietary IM networks, enabling XMPP users to communicate with those on other networks.

- Scalability notes
- High Availability notes
- Compatibility notes
- UMS API Programmer notes

Scalability notes

An XMPP Driver cannot be scaled out to multiple servers since the XMPP server does not allow multiple drivers (i.e. clients) to access the same account. When the second driver logs in, the XMPP server disconnects the first driver. The first driver, then, reconnects after a while and then the second driver is disconnected and so on. In this setup, outgoing messages work only if the driver that handles the message is connected at that particular moment.

High Availability notes

When there are connection problems towards the XMPP server, the XMPP driver keeps attempting to reconnect to the remote server, but, increases the delay to avoid tight loops.

Compatibility notes

lists the XMPP Driver gateway vendors and versions.

Table 3-3 XMPP Driver Gateway Vendors and Versions

Vendor	Version
ejabberd	2.1.3
jabberd2	2.2.14
jabberd14	1.6.1.1-p1
Oracle Beehive	2.0.1.2.1

UMS API Programmer notes

XMPP has a concept of *Subject* and *Body* which corresponds to the UMS Message *Subject* and *Content*.

Extension Driver

The Extension Driver sends messages to the configured Endpoint URL that implements the Notification WebServices interface defined by UMS. A messaging gateway, or an adapter to a gateway, can implement this Web Service, thereby, extending UMS with no changes in UMS.

The Extension driver also supports sending messages to any ADF Business Components web application, for instance, the ATK Popup Service which is one of Oracle's workflow and collaboration products. At runtime, the Extension Driver detects if the configured Endpoint URL is the Notification WebService or the ATK Popup Service and acts accordingly.

The Extension Driver is used for outgoing messages only.

Scalability notes

The Extension Driver can be scaled out to multiple nodes.

High Availability notes

The driver communicates with the external system via WebServices calls and the host part in the Endpoint URL is a virtual IP, that is, the external system is behind a load balancer.

Compatibility notes

Not applicable

UMS API Programmer notes

The UMS Message content must be textual, not binary. To enable a new protocol in UMS using the Extension Driver, perform the following tasks:

1. Implement and deploy a web service listener endpoint based on the `MessagingNotifyService` WSDL (`umsnotify.wsdl`):

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<wsdl:definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
                  xmlns:tns="http://xmlns.oracle.com/ucs/messaging/extension"
                  xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
                  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                  name="MessagingNotifyService"
                  targetNamespace="http://xmlns.oracle.com/ucs/messaging/extension">

   <wsdl:types>

   <xsd:schema targetNamespace="http://xmlns.oracle.com/ucs/messaging/extension">
     <xsd:element name="notification">
       <xsd:complexType>
         <xsd:sequence>
           <xsd:element name="messageId" type="xsd:string" minOccurs="0"
maxOccurs="1">
             <xsd:annotation>
               <xsd:documentation>Unique message identifier from User
Messaging Service.</xsd:documentation>
             </xsd:annotation>
           </xsd:element>
           <xsd:element name="sender" type="xsd:string">
             <xsd:annotation>
               <xsd:documentation>The sender address.</xsd:documentation>
             </xsd:annotation>
           </xsd:element>
           <xsd:element name="recipient" type="xsd:string">
             <xsd:annotation>
               <xsd:documentation>The recipient address (typically
username).</xsd:documentation>
             </xsd:annotation>
           </xsd:element>
           <xsd:element name="subject" type="xsd:string" minOccurs="0"
maxOccurs="1">
             <xsd:annotation>
               <xsd:documentation>The subject of the message, if
available.</xsd:documentation>
             </xsd:annotation>
           </xsd:element>
           <xsd:element name="contentType" type="xsd:string"
default="text/plain">
             <xsd:annotation>
               <xsd:documentation>The MIME type of the message. e.g.
text/plain, text/html, text/xml.</xsd:documentation>
             </xsd:annotation>
           </xsd:element>
           <xsd:element name="content" type="xsd:string">
             <xsd:annotation>
               <xsd:documentation>The main body of the message. Textual
content only (no binary content).</xsd:documentation>
             </xsd:annotation>
           </xsd:element>
           <xsd:element name="parameters" type="tns:parameter" minOccurs="0"
maxOccurs="unbounded">
             <xsd:annotation>
               <xsd:documentation>Additional key-value pairs. This interface
does not define any specific key-value pair meanings. Use of such parameters
is defined on a private basis by particular implementations of this interface.
  </xsd:documentation>
```

```
            </xsd:annotation>
          </xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:complexType name="parameter">
      <xsd:sequence>
        <xsd:element name="name" type="xsd:string">
          <xsd:annotation>
            <xsd:documentation>Parameter name</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
        <xsd:element name="value" type="xsd:string">
          <xsd:annotation>
            <xsd:documentation>Parameter value</xsd:documentation>
          </xsd:annotation>
        </xsd:element>
      </xsd:sequence>
    </xsd:complexType>
    <xsd:element name="notificationResponse">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="messageId" type="xsd:string" minOccurs="0"
maxOccurs="1">
            <xsd:annotation>
              <xsd:documentation>A message identifier returned in response to
successfully accepting the message. If returned, the identifier should be
unique. Note: A fault is raised if the message cannot be
accepted.</xsd:documentation>
            </xsd:annotation></xsd:element>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
    <xsd:element name="notificationFault">
      <xsd:complexType>
        <xsd:sequence>
          <xsd:element name="code" type="xsd:string"/>
          <xsd:element name="message" type="xsd:string"/>
        </xsd:sequence>
      </xsd:complexType>
    </xsd:element>
  </xsd:schema>
 </wsdl:types>
 <wsdl:message name="notifyRequest">
   <wsdl:part element="tns:notification" name="parameters" />
 </wsdl:message>
 <wsdl:message name="notifyResponse">
   <wsdl:part element="tns:notificationResponse" name="parameters"/>
 </wsdl:message>
 <wsdl:message name="notifyException">
   <wsdl:part element="tns:notificationFault" name="parameters"/>
 </wsdl:message>
 <wsdl:portType name="Notify">
   <wsdl:operation name="invoke">
     <wsdl:input message="tns:notifyRequest"/>
     <wsdl:output message="tns:notifyResponse"/>
     <wsdl:fault message="tns:notifyException" name="NotifyException"/>
   </wsdl:operation>
 </wsdl:portType>
 <wsdl:binding name="NotifySOAPBinding" type="tns:Notify">
   <soap:binding style="document"
transport="http://schemas.xmlsoap.org/soap/http" />
```

```
    <wsdl:operation name="invoke">
      <soap:operation
        soapAction="http://www.oracle.com/ucs/messaging/extension" />
      <wsdl:input>
        <soap:body use="literal" />
      </wsdl:input>
      <wsdl:output>
        <soap:body use="literal" />
      </wsdl:output>
      <wsdl:fault name="NotifyException">
        <soap:fault name="NotifyException" use="literal"/>
      </wsdl:fault>
    </wsdl:operation>
  </wsdl:binding>

  <wsdl:service name="NotifyService">
    <wsdl:port binding="tns:NotifySOAPBinding" name="Notify">
      <soap:address location="http://localhost:8001/NotifyService"/>
    </wsdl:port>
  </wsdl:service>
</wsdl:definitions>
```

2. Configure the extension driver by performing the following tasks:

   a. Target the predeployed extension driver called **usermessagingdriver-extension** (or a new deployment) to the appropriate server where UMS (usermessagingserver) is running and start the driver.

   b. In Enterprise Manager Fusion Middleware Control, navigate to the usermessagingserver home page.

   c. Click **User Messaging Service** and navigate to **Driver Properties**.

   d. Select and edit the driver **usermessagingdriver-extension** or create a new driver with the same name as your new driver deployment.

   e. Under Driver-Specific Configuration, add a new extension endpoint configuration group and specify the properties. EndpointURL is the URL to the web service listener endpoint that you created in Step one. Protocol is the value of the new messaging channel for which you want to add notification support (for example, *myProtocol*).

   f. Under Common Configuration, update **Supported Protocols** with a comma-separated list of protocols defined in each Extension Endpoint group.

   g. Click **OK** to save the configuration.

      This completes the configuration and integration of a new messaging channel (protocol) in UMS using the extension driver.

   To send notifications to this new channel (protocol), recipients must be specified for the URI delivery type using the following URI addressing format:

   ```
   URI:scheme:scheme-specific-address-value
   ```

   where, `scheme` is the protocol. The URI delivery type is optional. For example, if the extension driver was configured to support the protocol, *myProtocol*, an application can compose a message to *myProtocol:john.doe@example.com*.

   End users can also declare their messaging preferences by creating a new messaging channel for the new channel type in the User Communication Preferences UI. Note that user preferences are only applied when applications send user-based notifications (that is, to recipients of the form USER:username).

APNS Driver

The APNS driver is available in User Messaging Service (UMS) 12.2.1 and onwards. It provides support for sending a UMS message as a notification to Apple iOS devices such as an iPhone or an iPad.

The following are some features of the APNS driver:

- **Otbound Message**: Messages sent from a UMS client application to an Apple Device via APNs are called outbound messages in the UMS terminology. In Apple terminology, it is a notification. A Message sent to recipient "**uri:apns:token**" is handled by the UMS APNS driver. The UMS APNS driver extracts the token from the recipient address and creates an empty Apple Push Notification Payload. Given the information in the UMS Message, Apple Push Notification Payload is populated and finally sent to APNs.
  - Delivery Type Supported

    Delivery type **URI** is supported. Protocol **apns** is supported.
  - Mime types supported

    Mime types of text/plain and application/json are supported.
  - Delivery status types supported

    The UMS API defines various status types. The APNS driver uses the following:

    * `DELIVER_TO_DRIVER_SUCCESS`

      The message has been received by the driver for further processing.

    * `DELIVERY_TO_GATEWAY_FAILURE`

      The notification could not be sent to APNs.

    * `DELIVERY_TO_GATEWAY_SUCCESS`

      The notification was sent to APNs. However, note that since APNs does not guarantee delivery and UMS does not know if the message was sent to the device or not, the APNS driver can never return DELIVERY_TO_DEVICE_SUCCESS.

    * `DELIVERY_TO_DEVICE_FAILURE`

      The APNS driver has for instance found out that the address used is invalid. Typically the user has unregistered the application on its device.
  - Mapping to external protocol

    Table 3-4 shows how UMS Message properties maps to the APNs protocol.

Table 3-4 GCM Metadata for Outbound Messages

UMS message property	APNs protocol	Value Type	Description
Recipient address e.g. 'uri:apns:token'	device token	UMS address	The token part of the recipient address is used as device token towards the APNs.

Table 3-4 (Cont.) GCM Metadata for Outbound Messages

UMS message property	APNs protocol	Value Type	Description
Content	alert	string	UMS Message text content is used as payload in the alert message. If the content type is text/plain then the content will be used in the alert. Metadata properties (see table below) are added to the payload. If content type is application/json the content is passed as is to APNs and the metadata properties will be ignored.
Expiration (on the MessageInfo) in seconds. Default `zero`.	expiry	integer	UMS property Expiration specifies a time to live in seconds; zero or negative means that the message shall never expire. If Expiration is negative or zero then expiry will be set to MAX_INT otherwise expiry property will be calculated as current time in seconds + Expiration. Push notifications that expire before being delivered are not considered a failed delivery and don't impact the feedback service.
Subject	-	-	Subject on the UMS message will be ignored.

Table Table 3-5 lists the optional UMS Message metadata.

Table 3-5 Optional UMS Message Metadata

Name	Description
badge	The number to display as the badge of the application icon. If this property is absent, the badge is not changed. To remove the badge, set the value of this property to `0`.
sound	The name of a sound file in the application bundle. The sound in this file is played as an alert. If the sound file doesn't exist or `default.aiff` is specified as the value, the default alert sound is played.
action-loc-key	If a string is specified, displays an alert with two buttons. However, iOS uses the string as a key to get a localized string in the current localization to use for the right button's title instead of 'View'. If the value is null, the system displays an alert with a single OK button that simply dismisses the alert when tapped.
loc-key	A key to an alert-message string in a Localizable.strings file for the current localization (which is set by the user's language preference). The key string can be formatted with %@ and %n$@ specifiers to take the variables specified in `loc-args`
launch-image	The filename of an image file in the application bundle; it may include the extension or omit it. The image is used as the launch image when users tap the action button or move the action slider. If this property is not specified, the system either uses the previous snapshot, uses the image identified by the UILaunchImageFile key in the application's Info.plist file, or falls back to `Default.png`.

Table 3-5 (Cont.) Optional UMS Message Metadata

Name	Description
custom	To add custom values to the Apple Push Payload use metadata name 'custom' and a JSON formatted String as value. The JSON object will be passed as is to the device application. Custom values must use the JSON structured and primitive types: dictionary (object), array, string, number and boolean. The JSON object will be appended to the payload.

- Driver specific message validation

  The device token and the message size will be validated. If the device token is not of proper length the message will not be sent and a fail status will be generated. If the message size exceeds 256 bytes the message will not be sent and a failed status will be generated.

  In case of content type application/json or that a custom JSON object is added in the metadata property, a JSON sanity check is done to see if the JSON object is parsable, if not the message will not be sent and a failed status will be generated.

- Send raw JSON

  By setting the content type to application/json the complete APNs message can be set. UMS will handover the JSON data as is to APNs. When the content type is set to application/json all driver specific metadata properties are ignored.

  This will also allow other message types to APNs, like MDM (Mobile Device Management) messages.

- Multiple application support

  The APNS driver supports multi configuration so that multiple independent application can use the APNS driver. In order to select the proper APNS driver for a message the UMS core feature of driver selection is used. The preferred way is to use the SupportedApplicationNames or the SupportedCarriers driver configuration properties.

  The message must in case of SupportedCarriers have the corresponding Carrier property set on the message's info object (oracle.sdp.messaging.MessageInfo).

  ```
  message.getMessageInfo().setCarrier("myapp");
  ```

  In case of SupportedApplicationNames, the default value for the application name property is the application's deployment name so in normal cases that can be used. If for some reason some other value must be used the application name property can be set when the MessagingClient is created, like this:

  ```
  Map<String, Object> parameters = new HashMap<String, Object>();

  parameters.put(ApplicationInfo.APPLICATION_NAME, "myapp");

  messagingClient = MessagingClientFactory.createMessagingClient(parameters);
  ```

- **Inbound Message - Feedback service**: Inbound message consists of reports from the APNs feedback service. The feedback service will report devices that are not reachable any more due to for instance that the user has uninstalled the iOS application. The APNS driver will poll the feedback service for invalid device tokens for this device application and report those in an inbound message to the provider application. In order for the provider application to receive those messages it needs to register an access point with the same values as the driver is configured with e.g. '**uri:apns:myapp**'

The feedback service will return all unreachable device tokens for the device application that the TLS connection indicates. It is not possible to only retrieve a subset of device tokens.

Apple APNs documentation strongly recommends using the feedback service. By using this information to stop sending push notifications that will fail to be delivered, the application will reduce unnecessary message overhead and improve overall system performance. APNs monitors providers for their diligence in checking the feedback service and refraining from sending push notifications to non-existent applications on devices.

The format of the message will be in JSON. The message is an array of JSON objects with device token and expiry date (in Epoch milliseconds as long):

```
[
  {
    "deviceToken" : "ad123e45f6f78c9041dd234e5a6f7890",
    "expiry" : 134567878222112333
  },
  {
    "deviceToken" : "bb223e45f6f78c9041dd234e5a6f1234",
    "expiry" : 133577878222123456
  }
]
```

- **Same Information Provider Application for different customers**: In case the provider application is to be used by several customers (installed on premises or in the cloud) then the client side of the application (the device application) cannot be directly reused. The device application needs to be delivered to the customers using unique bundle identities. This is due to the fact that different TLS certificates needs to be used for each customer when the feedback service is to be polled for unreachable device tokens. The APNS driver needs to use different TLS certificates for each customer, otherwise unreachable device tokens for some other customer will be retrieved.

to add more text.

- Scalability notes
- High Availability notes
- Compatibility notes
- UMS API Programmer notes

Scalability notes

The APNS Driver supports as many messages sending as a resource adapter can handle given the JEE platform.

The scalability of the driver is limited by the scalability of the JMS solution. UMS, and the driver, can be deployed on multiple nodes.

High Availability notes

If the call to APNs fails then the message is marked as delivery to gateway failure.

UMS, and the APNS driver, can be deployed on multiple nodes.

> **Note:**
>
> Delivery of Push Notification is not guaranteed by APNs! Data transfer should not be performed using Push Notifications, only send a notification saying that data is available.

Compatibility notes

The APNS Driver is introduced in 12.2.1.0.0.

UMS API Programmer notes

When the UMS APNS driver is deployed into a mid-tier server, the '`uri:apns`' delivery channel should be available to all UMS clients. For more information, refer to the code samples below:

- Send Push Notification
- Send Push Notification With Additional Data
- Send Push Notification With Additional Custom Data
- Send Push Notification With Raw JSON data
- Send Push Notification With Raw JSON data, MDM payload
- Receive unreachable device tokens
- Send Push Notification With Unicode Characters

Send Push Notification

To send a push notification, just send a message to the device token. Suppose the token is '`1234567890…1234567890`'. The following java snippet will push a notification to that device:

```
String recipient = "uri:apns:1234567890..1234567890";
Message message = MessagingFactory.createTextMessage("New Lenny K clips");
message.addRecipient(MessagingFactory.createAddress(recipient));
String id = mMessagingClient.send(message);
```

Send Push Notification With Additional Data

To send a push notification specifying badge number and sound file, add meta data to the UMS Message as shown in following java snippet:

```
String recipient = "uri:apns:1234567890..1234567890";
Message message = MessagingFactory.createTextMessage("New Lenny K clips");
message.addRecipient(MessagingFactory.createAddress(recipient));
message.setMetaData("UMS-APNS", "badge", "3");
message.setMetaData("UMS-APNS", "sound", "Rocknroll.aiff");
String id = mMessagingClient.send(message);
```

To send a push notification specifying localized alert text with arguments, add meta data to the UMS Message as shown in following java snippet:

```
String recipient = "uri:apns:1234567890..1234567890";
Message message = MessagingFactory.createTextMessage("New Lenny K clips");
message.addRecipient(MessagingFactory.createAddress(recipient));
message.setMetaData("UMS-APNS", "loc-key", "GAME_PLAY_REQUEST_FORMAT");
```

```
message.setMetaData("UMS-APNS", "loc-args", "Jenna,Frank");
String id = mMessagingClient.send(message);
```

The above will result in a notification payload looking like this:

```
{
    "aps" : {
        "alert" : {
            "loc-key" : "GAME_PLAY_REQUEST_FORMAT",
            "loc-args" : ["Jenna", "Frank"]
        }
    }
}
```

Send Push Notification With Additional Custom Data

To send a push notification specifying custom data, create a JSON formatted String representing the custom data and add it as metadata "custom" to the UMS Message as shown in following java snippet:

```
String customDataAsJson = "{\"myCustomDataSection\":
{\"mykey1\":\"value1\",\"mykey2\":\"value2\"}, \"section2\":4711}";
String recipient = "uri:apns:1234567890..1234567890";
Message message = MessagingFactory.createTextMessage("New Lenny K clips");
message.addRecipient(MessagingFactory.createAddress(recipient));
message.setMetaData("UMS-APNS", "custom", customDataAsJson);
String id = mMessagingClient.send(message);
```

The resulting notification payload will look like this:

```
{
    "aps" : {
        "alert" : "New Lenny K clips",
    },
    "myCustomDataSection" : {
        "mykey1" : "value1",
        "mykey2" : "value2",
    },
    "section2" : 4711
}
```

> **Note:**
>
> JSON formatted Strings can be created from various objects, for instance using the com.fasterxml.jackson package already available in WebLogic.

Send Push Notification With Raw JSON data

To send a push notification specifying the APS payload data, create a JSON formatted String representing the payload:

```
String rawJson = "{\"aps\":{\"alert\":\"UMS rocks\"}}";
String recipient = "uri:apns:1234567890..1234567890";
Message message = MessagingFactory.createMessage();
message.addRecipient(MessagingFactory.createAddress(recipient));
message.setContent(rawJson,"application/json; charset=\"UTF-8\"");
String id = mMessagingClient.send(message);
```

Send Push Notification With Raw JSON data, MDM payload

To send a push notification specifying the MDM payload data, create a JSON formatted String representing the payload:

```
String rawJson = "{\"mdm\":\"the_push_magic_token\"}";
String recipient = "uri:apns:1234567890..1234567890";
Message message = MessagingFactory.createMessage();
message.addRecipient(MessagingFactory.createAddress(recipient));
message.setContent(rawJson,"application/json");
String id = mMessagingClient.send(message);
```

Receive unreachable device tokens

The APNS driver will poll the feedback services regularly to get a list of unreachable device tokens. Once the list of device tokens is retrieved the feedback service clears its information about the device tokens. So it is not possible to retrieve the same device token again (unless it has registered again and then again become unreachable).

The application needs to register an UMS access point that corresponds to the access point that the APNS driver is configured with.

```
import com.fasterxml.jackson.databind.*;
String address = "uri:apns:myapp";
AccessPoint accessPoint =
AccessPointFactory.createAccessPoint(MessagingFactory.createAddress(address));
messagingClient.registerAccessPoint(accessPoint);
Message[] messages = messagingClient.receive();
byte[] content = (byte[]) messages[0].getContent();
JsonNode devices = new ObjectMapper().readTree(content);
for (JsonNode device : devices) {
  System.out.println("deviceToken: " + device.get("deviceToken").asText());
  System.out.println("expiry: " + new Date(device.get("expiry").asLong()));
}
```

Send Push Notification With Unicode Characters

If non-ASCII characters are used, then the charset on the UMS message must be set accordingly. For example, to UTF-8. Below is a sample code that illustrates creating messages with the character 'ä' and the emoticon.

```
// message with 'ä'
Message message1 = MessagingFactory.createTextMessage("\u00e4", "UTF-8");
// message with SMILING FACE WITH SUNGLASSES (U+1F60E)
Message message2 = MessagingFactory.createTextMessage("\ud83d\ude0e", "UTF-8");
```

4

Configuring Oracle User Messaging Service

This chapter describes how to configure and secure Oracle User Messaging Server (UMS) in your environment.

- Accessing User Messaging Service Configuration Pages
  You can configure UMS through Oracle Enterprise Manager Fusion Middleware Control.
- Configuring User Messaging Server
  UMS is deployed as one enterprise archive for the server and one enterprise archive per driver type. The configuration can be defined at the managed server level or cluster level, where cluster level overrides domain level. It is possible to configure the server and drivers using WebLogic Scripting Tool (WLST) and Enterprise Manager (EM).
- Configuring User Messaging Service Drivers
  UMS supports multiple configurations. This means that, one deployed driver instance can handle more than one configuration. This makes it possible to have one instance of a particular driver configured differently in a domain without having to deploy several instances of that driver. All the drivers support multiple configuration.
- Configuring User Messaging Service Access to the LDAP User Profile
  As part of the LDAP provider setup in a UMS deployment, you configure the **User Name Attribute** through the WebLogic Remote Console. If you configure that attribute with a value other than the default *cn* or if the user's email address is stored in an LDAP attribute which is different from *mail*, you must make an additional configuration change in Oracle Platform Security Services (OPSS) for UMS to successfully access the user profile to obtain the list of communication channels provisioned in LDAP, such as business email.
- Using Oracle User Messaging Service for Group Messaging
  In addition to supporting bi-directional mutli-channel messaging through a variety of channels, UMS supports group messaging. This feature includes sending a message to a group of users by sending it to a group URI, or sending a message to LDAP groups (or enterprise roles) and application roles.
- Configuring Automatic Message Resend
  In 14*c*, the automatic resend feature can be configured to automate the administrator's resend. This means that when a message send attempt is classified as a complete failure, then the message is automatically scheduled for resend.
- Securing the Oracle User Messaging Service
  The User Communications Preferences User Interface can be secured at the transport-level using Secure Sockets Layer (SSL). By default, all deployed web services are unsecured. Web Service Security should be enabled for any services that are deployed in a production environment.

Accessing User Messaging Service Configuration Pages

You can configure UMS through Oracle Enterprise Manager Fusion Middleware Control.

For more information, see *Administering Oracle Fusion Middleware with Fusion Middleware Control*.

Alternatively, you can also use WebLogic Scripting Tool (WLST) to configure UMS. For more information, see *WLST Command Reference for Infrastructure Components*.

Configuring User Messaging Server

UMS is deployed as one enterprise archive for the server and one enterprise archive per driver type. The configuration can be defined at the managed server level or cluster level, where cluster level overrides domain level. It is possible to configure the server and drivers using WebLogic Scripting Tool (WLST) and Enterprise Manager (EM).

If the User Messaging Server configuration is defined at the cluster level, then the cluster name along with all the following properties must be specified.

Table 4-1 Properties for Configuring User Messaging Server

Name	Description	Mandatory
AppReceivingQueuesInfo	The default set of queues from which the application will dequeue received messages.	Y
DuplicateMessageRetryDelay	The delay period for deferring processing of a possible duplicate message.	Y
EngineCommandQueuesInfo	The set of queues from which the engine will dequeue command messages sent by other messaging components.	Y
EnginePendingReceiveQueueInfo	The queue from which the engine will dequeue pending messages. The format for this value is JNDIQueueConnectionFactoryName:JNDIQueueName.	Y
EngineReceivingQueuesInfo	The set of queues from which the engine will dequeue received messages.	Y
EngineSendingQueuesInfo	The set of queues from which the engine will dequeue sent messages.	Y
JpsContextName	The name of the Java Platform Security (JPS) context to use when getting an Identity Store Service instance. Empty value leads to default JPS context.	Y
ReceivedmessageStatusEnabled	Enable received message status reporting - if false, client library does not return delivery status to engine.	Y
ResendDefault	The default number of automatic resends upon delivery failure. You can override this property programmatically on a per message basis. The upper limit is the value specified in the configuration parameter ResendMax.	Y
ResendDelay	The delay in seconds between automatic resends.	Y
ResendMax	The max number of automatic resends upon delivery failure.	Y
SecurityPrincipal	The default system user used.	Y
SessionTimeout	The duration to wait before a session timeout when the session flag is set by a Driver or Messaging Client Application.	Y

Table 4-1 (Cont.) Properties for Configuring User Messaging Server

Name	Description	Mandatory
SupportedDeliveryTypes	The set of delivery types supported by this server.	Y

Configuring User Messaging Service Drivers

UMS supports multiple configurations. This means that, one deployed driver instance can handle more than one configuration. This makes it possible to have one instance of a particular driver configured differently in a domain without having to deploy several instances of that driver. All the drivers support multiple configuration.

You can create multiple configurations of a single deployment of the drivers using a unique name at each configuration. Though possible, it is recommended *not* to use the same configuration name while creating multiple configurations for a particular driver instance, as this may lead to unintended results.

Since UMS can be deployed in a cluster or a server, the configuration of drivers can be done at the cluster or server level. It is recommended that the configuration be done at the same level as that of the deployment. However, exceptional scenarios might justify creating configuration at a level different from that of the deployment level.

You can configure UMS drivers by using Oracle Enterprise Manager Fusion Middleware Control. Alternatively, you can configure the UMS drivers by using the WLST command `configUserMessagingDriver`. For more information about this command, see *WLST Command Reference for Infrastructure Components*.

> **Note:**
>
> UMS drivers can be configured at the cluster level or server level. For more information, see Configuring User Messaging Server to ensure that you select the appropriate configuration level.

- Configuring a Driver

Configuring a Driver

You can navigate to the driver configuration page from any one of the following:

- Associated Drivers table on the User Messaging Service home page
- Driver Properties menu for the driver target in the Target Navigation pane
- Driver Properties menu on the User Messaging Service home page

To configure a driver, perform the following tasks:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control as an administrator.
2. Navigate to the **User Messaging Service** home page.
3. Click **usermessagingserver(AdminServer)**. The Associated Drivers page appears.

4. Select the **Local** tab to access the drivers collocated with the UMS server instance. These drivers may or may not be registered with the UMS server depending on whether they are properly configured. The **ALL** tab lists all drivers that are deployed in the domain and registered to all the UMS server instances.
5. Choose a driver from the list, and click the corresponding **Configure Driver** icon.

   The configuration page that lists all the configurations applied to this driver deployment will be displayed, and the administrator can create, edit, or delete a configuration. User Messaging drivers are configured differently in the following scenarios:

   a. For the email driver (only email driver supports multiple configuration in 12.1.3), the configuration depends on whether the driver is deployed in a clustered or a non-clustered environment.

      - If the driver is deployed in a cluster, for instance *a_ums_cluster*, then all the email configurations for cluster *a_ums_cluster* and also for the whole domain will be listed. The cluster-level configuration will override the domain-level configuration, if they have the same configuration name.
      - For a driver deployed in a non-clustered managed server, the configuration will be at the server level.

   For information about support for multiple configuration, and the relationship between cluster level and domain level configuration, refer to Configuring User Messaging Server and Configuring User Messaging Service Drivers.
6. Click **Create**, or select a driver configuration from the list and click **Edit**. The Driver Properties page appears. You can create a new configuration or update the existing one.
7. If needed, expand the **Driver-Specific Configuration** section and configure the driver parameters. For more information, see Introduction to Driver Properties.
8. To validate if the configuration properties are in correct format and valid in the deployment environment, you can 'test' the driver configuration parameters that you have entered. Click the **Test** button on the page. Click **OK** to continue.

> **Note:**
>
> Even if the testing does not succeed, you can still save the configuration.

- Introduction to Driver Properties
- Securing Passwords
- Saving Driver Properties
- Configuring the Messaging Extension Driver
- Configuring the Email Driver
- Configuring the SMPP Driver
- Configuring the XMPP Driver
- Configuring the APNS Driver
- Configuring GCM Driver

Introduction to Driver Properties

UMS drivers share common properties (listed in Table 4-2) that are used by the Messaging Engine when routing outbound messages.

Table 4-2 Common Driver Properties

Name	Description	Mandatory Property
Capability	Sets the driver's capability to send or receive messages. The values are SEND, RECEIVE, and BOTH.	Yes
Cost	Only used for driver configuration selection between multiple driver configurations of the same type, and only when required by the client application, The cost level of the driver (from 0 - 10). 0 is least expensive; 10 is most expensive. If the value is not in this range, cost is considered to be 0.	No
DefaultSenderAddress	If the UMS Message has no Sender Address of the specific DeliveryType that the driver supports, then the driver may use the DefaultSenderAddress as the Sender Address. The sample DefaultSenderAddress is EMAIL:alice@example.com.	No
SenderAddresses	The list of sender addresses that the driver is configured to handle. A driver with specified SenderAdresses will be selected only for an outgoing message that has a matching Sender Address. A driver that has not specified any SenderAdresses is considered to be able to handle any outgoing message regardless of the Sender Address of the message. The list should consist of UMS addresses separated by comma, for example `EMAIL:alice@example.com` or `EMAIL:alice@example.com,EMAIL:bob@example.com`. The matching is case insensitive.	No
Speed	Only used for driver configuration selection between multiple driver configurations of the same type, and only when required by the client application. The speed level of the driver (from 0-10, with 10 being the fastest).	No
SupportedCarriers	A comma-delimited list of supported carriers.	No

Table 4-2 (Cont.) Common Driver Properties

Name	Description	Mandatory Property
Configuration Level	Enables driver configuration at the `server` level or at the `cluster` level. If **Server** level is selected, then the server name must be specified. If **Cluster** level is selected, then the cluster name must be specified.	Yes
SupportedContentTypes	The content type supported by the driver.	Yes
SupportedDeliveryTypes	The delivery types supported by the driver.	Yes
SupportedProtocols	A comma-delimited list of supported protocols.	No
SupportedStatusTypes	The status types supported by the driver.	No
Supported Application Names	The application name supported by the driver.	No

Securing Passwords

Sensitive driver properties (namely, passwords) can be stored securely in the credential store using Oracle Enterprise Manager Fusion Middleware Control. Properties are marked with the flag **Encoded Credential** and have a custom entry form field.

To store a sensitive driver property securely, perform the following tasks:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control, and navigate to the driver configuration page of the selected driver.

   The configuration page that lists all the configurations applied to this driver deployment will be displayed, and the administrator can create, edit, or delete a configuration.

2. Click **Create** to create a new configuration or select a configuration and click **Edit** to edit an existing configuration.

   The Driver properties page appears.

3. In the **Driver-Specific Configuration** table, locate the properties with the **Encoded Credential** flag set.

4. Select the credential type from the **Type of Password** drop-down list in the adjoining Value column.

5. Depending on the selected credential type, you are prompted to enter the username and/or password. There are the following three options:

   - Indirect password, create new user (default option): specify the username and real password; the password is stored in the credential store with the username as part of the key. The key and a fixed folder (map name) are stored in the driver deployment's file.
   - Indirect password, use existing user: choose an existing indirect username/key in the credential store (to reference the password you stored previously).
   - User a clear text password: specify the password, and it is stored directly in the driver deployment file.

6. Click **OK** to save changes.
7. Restart the driver application or the container for the changes to take effect.

You can check the password in the driver deployment directory's file. For an indirect password, the format is:

```
value="->mapName:keyName"    (mapName can be any name of the user's choice,
and the key is <parameter_name>.<username>)
```

Saving Driver Properties

You can save the driver properties in the UMS database.

When the UMS driver appears, the properties stored in the `DRIVERPROPERTIES` table for specific fields override the values in the file and the driver instance uses those property values from the database to process requests. Given below are the driver property details:

- Table Name - `DRIVERPROPERTIES`
- Columns - `DRIVERNAME`, `PROPERTYNAME`, and `PROPERTYVALUE`

The following MBean is used to save driver properties in the UMS database:

MBean Name

```
oracle.ucs.messaging:Location=ESS_SOAServer_1,name=UserPrefsAdministration,type=S
DPMessagingRuntime ( Note: Any other instance of SOA server can be used to invoke
this mbean than ESS_SOAServer_1)
```

Operation Name

```
saveDriverProperties
```

Parameters

- P1 - driverName
- P2 - propertyName
- P3 - propertyValue

Example

```
mBean =
ObjectName("oracle.ucs.messaging:Location=ESS_SOAServer_1,name=UserPrefsAdmini
stration,type=SDPMessagingRuntime")
set_param = ['usermessagingdriver-apns-HCM', 'Alias', 'apns-prod-server-
hcm-12122023']
set_type = ["java.lang.String", "java.lang.String", "java.lang.String"]
mbs.invoke(mBean, "saveDriverProperties", set_param, set_type)
```

Configuring the Messaging Extension Driver

The extension driver extends the messaging capability of UMS by enabling support for arbitrary administrator-defined channels (protocols) and delivering the notifications for such channels to an administrator-defined web service listener endpoint.

> **Note:**
>
> An instance of this driver is deployed, but not targeted to any servers in the default. To enable this driver instance, it must be targeted to the appropriate servers where UMS (`usermessagingserver`) is running.

Common Properties

These are common driver properties that are indicative of the capabilities of this driver for use by the messaging engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. Table 4-3 lists the common properties of the Extension driver. For detailed description of these properties, refer to Table 4-2. For the complete list of available values, see *User Messaging Service Java API Reference*.

Table 4-3 Extension Driver Common Properties

Name	Mandatory	Default Value
InstanceName	Yes	Extension-Driver
Capability	Yes	SEND
SupportedDeliveryTypes	Yes	URI
SupportedContentTypes	Yes	text/plain, text/html, text/xml
SupportedStatusTypes	No	DELIVERY_TO_GATEWAY_SUCCESS, DELIVERY_TO_GATEWAY_FAILURE
Cost	No	
Speed	No	
SupportedCarriers	No	
Configuration Level	Yes	Server/Cluster
SupportedProtocols	No	popup
SenderAddresses	No	
DefaultSenderAddress	No	
Supported Application Names	No	Empty

Custom Properties

Table 4-3 lists properties specific to this driver and generally associated with configuring a remote endpoint at which to deliver extension notifications:

Table 4-4 Extension Driver Custom Properties

Name	Description	Mandatory
Group Name	The name of this extension endpoint configuration group.	Yes
Endpoint URL	Remote endpoint listener URL.	Yes
Mapped Domain	The extension endpoint used to deliver messages where the domain part of the recipient URI matches this value.	No
Protocol	The extension endpoint used to deliver messages where the protocol (scheme) part of the recipient URI matches this value.	Yes
Security Policies	Comma-separated list of WS-Security policies to apply to this endpoint.	No
Username	Username to propagate through WS-Security headers.	No
Keystore Alias	Keystore alias to use for looking up WS-Security policy public keys.	No
Credential Store Key	Key to use for looking up the WS-Security username and password from the Oracle Web Services Management credential store map.	No

Extension Driver Security

If the remote extension endpoint is secured using WS-Security, then additional configuration of the extension driver is required. There are two typical WS-Security configurations that are supported. The extension driver can either use SAML tokens or username tokens.

To use extension driver security:

1. To use SAML tokens, the Security Policies configuration property should contain value `oracle/wss11_saml_token_identity_switch_with_message_protection_client_policy`, and the Keystore Alias configuration property should contain a valid alias for keystore entries that is accepted by the remote extension endpoint.
2. To use username tokens, the Security Policies configuration property should contain value `oracle/wss11_username_token_with_message_protection_client_policy`, and the Credential Store Key configuration property should contain a valid alias for a credential store entry that is accepted by the remote extension endpoint.

For more details about using WS-Security policies and configuring OWSM, see *Oracle Fusion Middleware Administering Web Services*.

Configuring the Email Driver

The email driver both sends and receives messages (that is, its **capability** property is set to **both** by default). The email driver sends messages over SMTP and uses either IMAP or POP3 for receiving messages.

- Common Properties
- Configuring Email Driver with OAuth

Common Properties

Table 4-5 lists common driver properties that are indicative of the capabilities of this driver for use by the messaging engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. For detailed description of these properties, refer to Table 4-5. For the complete list of available values, see *User Messaging Service Java API Reference*.

Table 4-5 Common Email Properties

Name	Mandatory	Default Value
InstanceName	Yes	Email-Driver
Capability	Yes	Both
SupportedDeliveryTypes	Yes	Email
SupportedContentTypes	Yes	text/plain, text/html, multipart/mixed, multipart/alternative, multipart/related
SupportedStatusTypes	No	DELIVERY_TO_GATEWAY_SUCCESS, DELIVERY_TO_GATEWAY_FAILURE, USER_REPLY_ACKNOWLEDGEMENT_SUCCESS, USER_REPLY_ACKNOWLEDGEMENT_FAILURE
Cost	No	N/A
Speed	No	N/A
SupportedCarriers	No	N/A
Configuration Level	Yes	Server/Cluster
Supported Protocols	No	N/A
SenderAddresses	No	N/A
DefaultSenderAddress	No	N/A
Supported Application Names	No	Empty

- Email Custom Properties

Email Custom Properties

Table 4-6 lists properties specific to this driver and generally associated with configuring access to the remote gateway and certain protocol or channel-specific behavior.

Table 4-6 Custom Email Properties

Name	Description	Mandatory	Default Value
MailAccessProtocol	Email receiving protocol. The possible values are IMAP and POP3. Required only if email receiving is supported on the driver instance.	No	IMAP
AutoDelete	This value indicates if the driver should mark the messages deleted after they have been processed. The default is Disabled. For the POP3 protocol, the messages are always deleted right after they are processed.	No	Disabled
Debug	This value indicates if the driver is running in Debug mode. When enabled, JavaMail prints out requests and responses between the email driver and the mail server to Fusion Middleware Control. The default is Disabled.	No	Disabled
CheckMailFreq	The frequency with which to retrieve messages from the mail server. The unit is in seconds and the default value is 30 seconds.	No	30
ReceiveFolder	The name of the folder from which the driver is polling messages. The default value is INBOX.	No	INBOX
OutgoingMailServer	The name of the SMTP server. This is mandatory only if email sending is required.	No	N/A
OutgoingMailServerPort	The port number of the SMTP server; typically **25**.	No	25
OutgoingMailServerSecurity	The security setting used by the SMTP server. Possible values are None, TLS, and SSL. The default value is None.	No	None
OutgoingDefaultFromAddr	The default FROM address (if one is not provided in the outgoing message). **Note**: The `OutgoingDefaultFromAddr` property is deprecated, use `DefaultSenderAddress` instead. For more details about the `DefaultSenderAddress` property, see Table 4-5 .	No	N/A
OutgoingUsername	The username used for SMTP authentication. Required only if SMTP authentication is supported by the SMTP server.	No	N/A

Table 4-6 (Cont.) Custom Email Properties

Name	Description	Mandatory	Default Value
OutgoingPassword	The password used for SMTP authentication. This is required only if SMTP authentication is supported by the SMTP server. This includes Type of Password (choose from Indirect Password/Create New User, Indirect Password/Use Existing User, and Use Cleartext Password) and Password.	No	N/A
IncomingMailServer	The hostname of the incoming mail server. Required only if email receiving is supported on the driver instance.	No	N/A
IncomingMailServerPort	Port number of IMAP4 (that is, **143** or **993**) or POP3 (that is, **110** or **995**) server.	No	N/A
IncomingMailServerSSL	Indication to enable SSL when connecting to IMAP4 or POP3 server. The default is Disabled.	No	Disabled
IncomingMailIDs	The email addresses corresponding to the user names. Each email address is separated by a comma and must reside in the same position in the list as their corresponding user name appears on the usernames list. Required only if email receiving is supported on the driver instance.	No	N/A
IncomingUserIDs	The list of user names of the mail accounts from which the driver instance is polling. Each name must be separated by a comma, for example, foo,bar. This is required only if email receiving is supported on the driver instance.	No	N/A
IncomingUserPasswords	The list of passwords corresponding to the user names. Each password is separated by a comma and must reside in the same position in the list as their corresponding user name appears on the usernames list. This is required only if email receiving is supported on the driver instance. This includes Type of Password (choose from Indirect Password/Create New User, Indirect Password/Use Existing User, and Use Cleartext Password) and Password.	No	N/A
ProcessingChunkSize	The number of messages processed during each message polling. The default is 100.	No	100
Disconnect After Poll	Whether or not to disconnect from the email server after message poll. Effective only for IMAP, as POP3 always disconnects.	No	False

Table 4-6 (Cont.) Custom Email Properties

Name	Description	Mandatory	Default Value
ImapAuthPlainDisable	Indication to disable or enable plain text authentication (`AUTHENTICATE PLAIN` command) for IMAP user authentication. The default is Disabled.	No	Disabled. When this property is disabled, that means that plain text is allowed.

> **Note:**
>
> Multiple Incoming Email IDs/User IDs/Passwords will be added through a popup dialog (import from a CSV file or add in table), so that hundreds of ID/Passwords can be added.

For information about saving driver properties, see Saving Driver Properties.

Configuring Email Driver with OAuth

OAuth 2.0 based authentication is provided for Gmail and Microsoft exchange. To configure the Email driver with OAuth, you must have a valid Gmail or Microsoft Exchange email account.

- Configuring WebLogic
- Updating Common Properties
- Enabling OAuth for Gmail Accounts
- Enabling OAuth for Microsoft 365 Accounts

Configuring WebLogic

In environments where OAuth access token generation URLs or Gmail/MS secure IMAP or SMTP ports are inaccessible due to firewall or VPN configurations blocking them, you must update the configuration settings.

Table 4-7 WebLogic Configuration

Issue	Workaround	WebLogic Configuration Update
JavaMail connection to the secure ports of 993 and 587 fail due to firewall or network issues.	Connecting/tunnelling through SOCKS proxy.	• -Dmail.imaps.proxy.host=<Your SOCKS proxy server> • -Dmail.imaps.proxy.port=<Proxy server's port> • -Dmail.smtp.proxy.host=<Your SOCKS proxy server> • -Dmail.smtp.proxy.port=<Proxy server's port>
JavaMail SSL handshake with the IMAP/SMTP servers fail when connecting to secure ports due to certificate validation errors.	Ignoring the certificates or disabling certificate validation.	• -Dweblogic.security.SSL.ignoreHostnameVerification=true • -Dmail.imaps.ssl.trust=imap.gmail.com • -Dmail.smtp.ssl.trust=smtp.gmail.com For MS OAuth, you must configure outlook.office365.com and smtp.office365.com.
Failure in connecting to the OAuth access token generation URLs as follows: *https://login.microsoftonline.com/<tenant id>/oauth2/v2.0/token* *https://oauth2.googleapis.com/token*	Connecting/tunnelling through HTTP proxy.	• -Dhttps.proxyHost=<Your https proxy> • -Dhttps.proxyPort=<https proxy port>

Updating Common Properties

You must update the `usermessagingconfig.xml` file with common properties.

Table 4-8 Common Properties

Property Name	Value	Procedure
`ImapAuthUseOAuth2`	True	Set the value to true to use OAuth for all IMAP mail accesses.
`SMTPAuthUseOAuth2`	True	Set the value to true to use OAuth for sending messages through SMTP.

Table 4-8 (Cont.) Common Properties

Property Name	Value	Procedure
`OutgoingUsername`	Your UMS OAuth mail ID for outgoing e-mails.	Enter your UMS OAuth mail ID. For example, `umsoauth@gmail.com`.
`OutgoingPassword`	<Empty>	Do not set the password, as the communication with servers take place using the OAuth access and refresh tokens.
`IncomingMailIDs`	Your UMS OAuth mail ID for incoming e-mails.	Enter your UMS OAuth mail ID. For example, `umsoauth@gmail.com`.
`IncomingUserIDs`	Your UMS OAuth mail ID for incoming e-mails.	Enter your UMS OAuth mail ID. For example, `umsoauth@gmail.com`.
`IncomingUserPasswords`	<Empty>	Do not set the password, as the communication with servers take place using the OAuth access and refresh tokens.

Example 4-1 Common Properties

```
<ns1:Property name="IncomingMailIDs" value="umsoauth@gmail.com"/>
<ns1:Property name="IncomingUserIDs" value="umsoauth@gmail.com"/>
<ns1:Property name="IncomingUserPasswords" value=" "/>
<ns1:Property name="OutgoingUsername" value="umsoauth@gmail.com"/>
<ns1:Property name="OutgoingPassword" value=""/>
<ns1:Property name="SMTPAuthUseOAuth2" value="true"/>
<ns1:Property name="ImapAuthUseOAuth2" value="true"/>
```

Enabling OAuth for Gmail Accounts

This section describes how to enable OAuth 2.0 based authentication for Gmail accounts and includes the following topics:

- Prerequisites
- Updating Configuration Properties
- Creating OAuth Client ID
- Generating Tokens
- Verifying Tokens
- Configuring SendAs

Prerequisites

You must perform the following prerequisite tasks:

1. Ensure that you have a valid Gmail or Microsoft email account.
2. Install Python.

Updating Configuration Properties

You must update the `usermessagingconfig.xml` file with configuration properties.

Table 4-9 Configuration Properties

Property Name	Value	Other Details
IncomingMailServer	imap.gmail.com	Gmail IMAP server
IncomingMailServerPort	993	Gmail's IMAP port
IncomingMailServerSSL	False	Constant value
OutgoingMailServer	smtp.gmail.com	Gmail SMTP server
OutgoingMailServerPort	587	Use SMTP server's port
OAuth2AccessTokenSupplierFactory	oracle.sdpinternal.messaging.oauth.ums.google.UMSGmail AccessTokenSupplierFactory	Constant value. Set this value whenever you are using gmail ID.
GoogleOAuthClientID	<Cliend id of the form xxx-xxx.apps.googleusercontent.com>	For more information about how to get the value, see Creating OAuth Client ID.
GoogleOAuthClientSecret	Encoded secret value	For more information about how to get the value, see Creating OAuth Client ID.
GoogleOAuthRefreshToken	Encoded and valid refresh token	For more information about how to get the value, see Generating Tokens.

Example 4-2 Configuration Properties

```
<ns1:Property name="IncomingMailServer" value="imap.gmail.com"/>
<ns1:Property name="IncomingMailServerPort" value="993"/>
<ns1:Property name="IncomingMailServerSSL" value="false"/>
<ns1:Property name="OutgoingMailServer" value="smtp.gmail.com"/>
<ns1:Property name="OutgoingMailServerPort" value="587"/>
<ns1:Property name="OAuth2AccessTokenSupplierFactory"
value="oracle.sdpinternal.messaging.oauth.ums.google.UMSGmailAccessTokenSuppli
erFactory"/>
<ns1:Property name="GoogleOAuthClientID"
value="GoogleOAuthClientID.apps.googleusercontent.com"/>
<ns1:Property name="GoogleOAuthClientSecret"
value="GoogleOAuthClientSecret"/>
<ns1:Property name="GoogleOAuthRefreshToken"
value="GoogleOAuthRefreshToken"/>
```

Sample UMS Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:MessagingConfiguration xmlns:ns0="http://www.oracle.com/ucs/messaging/
configtemplate" xmlns:ns1="http://www.oracle.com/ucs/messaging/config"
version="12.2.1.3.0">
   <ns1:Driver name="Test1" type="email" server="AdminServer" enabled="true">
      <ns1:Property name="SupportedDeliveryTypes" value="EMAIL"/>
```

```
      <ns1:Property name="SupportedContentTypes" value="*"/>
      <ns1:Property name="Capability" value="BOTH"/>
      <ns1:Property name="Cost" value=""/>
      <ns1:Property name="Speed" value=""/>
      <ns1:Property name="SupportedCarriers" value=""/>
      <ns1:Property name="SupportedProtocols" value="SMTP"/>
      <ns1:Property name="SupportsCancel" value="false"/>
      <ns1:Property name="SupportsReplace" value="false"/>
      <ns1:Property name="SupportsStatusPolling" value="false"/>
      <ns1:Property name="SupportsTracking" value="false"/>
      <ns1:Property name="SupportedStatusTypes"
value="DELIVERY_TO_GATEWAY_SUCCESS, DELIVERY_TO_GATEWAY_FAILURE,
USER_REPLY_ACKNOWLEDGEMENT_SUCCESS, USER_REPLY_ACKNOWLEDGEMENT_FAILURE"/>
      <ns1:Property name="SenderAddresses" value=""/>
      <ns1:Property name="SupportedApplicationNames" value=""/>
      <ns1:Property name="DefaultSenderAddress" value=""/>
      <ns1:Property name="SendingQueuesInfo" value="OraSDPM/
QueueConnectionFactory:OraSDPM/Queues/OraSDPMDriverDefSndQ1"/>
      <ns1:Property name="MailAccessProtocol" value="IMAP"/>
      <ns1:Property name="AutoDelete" value="false"/>
      <ns1:Property name="Debug" value="true"/>
      <ns1:Property name="CheckMailFreq" value="30"/>
      <ns1:Property name="DisconnectAfterPoll" value="false"/>
      <ns1:Property name="ReceiveFolder" value="INBOX"/>
      <ns1:Property name="OutgoingMailServer" value="smtp.gmail.com"/>
      <ns1:Property name="OutgoingMailServerPort" value="587"/>
      <ns1:Property name="OutgoingMailServerSecurity" value="None"/>
      <ns1:Property name="OutgoingDefaultFromAddr" value=""/>
      <ns1:Property name="OutgoingUsername" value="umstest@gmail.com"/>
      <ns1:Property name="OutgoingPassword" value=""/>
      <ns1:Property name="IncomingMailServer" value="imap.gmail.com"/>
      <ns1:Property name="IncomingMailServerPort" value="993"/>
      <ns1:Property name="IncomingMailServerSSL" value="false"/>
      <ns1:Property name="IncomingMailIDs" value="umstest@gmail.com"/>
      <ns1:Property name="IncomingUserIDs" value="umstest@gmail.com"/>
      <ns1:Property name="IncomingUserPasswords" value="welcome"/>
      <ns1:Property name="ProcessingChunkSize" value="100"/>
      <ns1:Property name="ImapAuthPlainDisable" value="false"/>
      <ns1:Property name="CNSMode" value="false"/>
      <ns1:Property name="SMTPAuthUseOAuth2" value="true"/>
      <ns1:Property name="ImapAuthUseOAuth2" value="true"/>
      <ns1:Property name="OAuth2AccessTokenSupplierFactory"
value="oracle.sdpinternal.messaging.oauth.ums.google.UMSGmailAccessTokenSuppli
erFactory"/>
      <ns1:Property name="GoogleOAuthClientID"
value="GoogleOAuthClientID.apps.googleusercontent.com"/>
      <ns1:Property name="GoogleOAuthClientSecret"
value="GoogleOAuthClientSecret"/>
      <ns1:Property name="GoogleOAuthRefreshToken"
value="GoogleOAuthRefreshToken"/>
   </ns1:Driver>
</ns1:MessagingConfiguration>
```

Creating OAuth Client ID

Perform the following steps to create the OAuth Client ID:

1. Open Google Cloud Console using the following URL:

   https://console.developers.google.com/
2. Enter your Gmail account credentials.

   You can create a project or use any of the existing projects.
3. Select the newly created project from the **Dashboard** option.
4. Select **Credentials** option.
5. Click **Create Credentials** and select **OAuth client ID**.
6. Configure the **OAuth consent screen** when prompted as follows:
   a. Under **User Type**, select **External**.
   b. Click **Create**.
   c. In the Edit app registration page, provide details in the following fields:
      - **App name**
      - **User support email**
      - **Developer contact information**

   You can skip the other fields as they are optional.
7. In the **Create OAuth client ID** page, select **Desktop app** from the **Application type** drop-down list.
8. In the **Name** field, enter **My UMS OAuth Client1**.
9. Click **Create**.

   The **OAuth client created** dialog box appears.
10. Copy **Your Client ID** and **Your Client Secret** from the dialog box.
11. Click **OK**.

    The **Your Client ID** and **Your Client Secret** details are also visible in the **OAuth 2.0 Client IDs table** in the **Credentials** section and can also be copied from here when required.

Generating Tokens

Perform the following steps to generate access and refresh tokens:

1. Download `oauth2.py` python script from GitHub.

   https://github.com/google/gmail-oauth2-tools/blob/master/python/oauth2.py
2. Run the script with the following four parameters:

```
--user=<your UMS OAuth gmail id>
--client_id=<your client id>
--client_secret=<your client secret>
--generate_oauth2_token
```

> **Note:**
>
> Ensure you follow all the instructions and authorize the app when prompted.

3. After you receive the verification code, enter the details in the python script to get the refresh token and access token.

Verifying Tokens

You must verify the validity of the refresh token. You must send a POST request to Google OAuth token generation URL to generate a new access token. The verification is complete once you receive the access token. You must set the same refresh token value in `usermessagingconfig.xml`.

You must send a POST request using curl or POSTMAN or any suitable client to the URL given below:

URL

https://oauth2.googleapis.com/token

Parameters

The following is a list of the POST request parameters.

```
client_id: <your client id>
client_secret: <your client secret>
refresh_token: <refresh token value>
grant_type: refresh_token
```

Curl Command

The following example shows how to verify the validity of the refresh token using curl:

```
curl --location --request POST 'https://oauth2.googleapis.com/token' \
                        --header 'Content-Type: application/x-www-form-
urlencoded' \
                        --data-urlencode 'client_id=734918285672-
hij60r3464hd4gt8ejt9vbpgh11ndhj2.apps.googleusercontent.com' \
                        --data-urlencode
'client_secret=t9JYc9QcwLQzZTI4BEkfW4-M' \
                        --data-urlencode
'refresh_token=1//0guaq1jSMUQO8CgYIARAAGBASNwF-
L9IrFMCFjCIVtYLCR73Dyh84adQDA8fBouxSP8du9Zje7Z9VkE3wV2mJ3oNUzSVDx-vNFjc' \
                        --data-urlencode 'grant_type=refresh_token'
```

Response

The following example shows the details of the access token in the Response body:

```
{
    "access_token": "ya29.a0ARrdaM9ToRMTmg6ghpP3GboRg3JrtDN-
dbqEUAq0PjhyKauR4_olpHiMK2OdR-sHm45C6wmrNkJ-LubZlgd7sxFKaiP65kt3migGJcjAK-
WwbbXkxitr2igqzD441kP2OB1M-BVEyR9RU-uUjELEnOmdhx-kYOk",
    "expires_in": 3599,
    "scope": "https://mail.google.com/",
    "token_type": "Bearer"
}
```

Configuring SendAs

For information about how to configure SendAs feature after the OAuth integration, see How to Send Mail on Behalf of Another Person in Google Mail.

Enabling OAuth for Microsoft 365 Accounts

UMS provides OAuth 2.0 based authentication support in the following two approaches to connect Microsoft IMAP and SMTP:

- Authorization Code Flow - UMS is configured using `O365OAuthClientID`, `O365OAuthTenantID`, `O365OAuthRefreshToken`, and `O365OAuthRefreshTokenScope` tokens. UMS internally generates "Access token" which is used to connect the IMAP and SMTP for both inbound and outbound emails.
- Client Credentials Flow - UMS is configured using `O365OAuthClientID`, `O365OAuthTenantID`, `O365OAuthClientSecret`, and `O365OAuthClientSecretScope` tokens. UMS internally generates "Access token" which is used to connect the IMAP and SMTP for both inbound and outbound emails.

Prerequisites

You must perform the following prerequisite tasks:

1. Ensure you have a valid Gmail or Microsoft email account.
2. Install Python.
3. Obtain the following certificate to create trusted connections with MS hosts while using WebLogic in secured mode:
   DigiCertSHA2SecureServerCA-2.crt

Updating Configuration Properties

You must update the `usermessagingconfig.xml` file with configuration properties.

Table 4-10 Configuration Properties

Property Name	Value	Other Details
IncomingMailServer	outlook.office365.com	MS IMAP server
IncomingMailServerPort	993	MS server's IMAP port
IncomingMailServerSSL	False	Constant value
OutgoingMailServer	smtp.office365.com	MS SMTP server
OutgoingMailServerPort	587	SMTP server's port
OAuth2AccessTokenSupplierFactory	oracle.sdpinternal.messaging.oauth.ums.ms.UMSO365AccessTokenSupplierFactory	Constant value. You must set this value when you use MS O365 mail ID.

Table 4-10 (Cont.) Configuration Properties

Property Name	Value	Other Details
O365OAuthClientID	<Cliend id (UUID format)>	For more information about how to get the value, see *Initiation an App Registration* section.
O365OAuthTenantID	<Tenant id (UUID format)>	For more information about how to get the value, see *Initiation an App Registration* section.
O365OAuthRefreshToken	Encoded and valid refresh token	For more information about how to get the value, see *Generating Tokens* section.

Example 4-3 Configuration Properties

```
      <ns1:Property name="IncomingMailServer" value="outlook.office365.com"/>
      <ns1:Property name="IncomingMailServerPort" value="993"/>
      <ns1:Property name="IncomingMailServerSSL" value="false"/>
      <ns1:Property name="OutgoingMailServer" value="smtp.office365.com"/>
      <ns1:Property name="OutgoingMailServerPort" value="587"/>
      <ns1:Property name="OAuth2AccessTokenSupplierFactory"
value="oracle.sdpinternal.messaging.oauth.ums.ms.UMSO365AccessTokenSupplierFac
tory"/>
      <ns1:Property name="O365OAuthClientID" value="O365OAuthClientID"/>
      <ns1:Property name="O365OAuthTenantID" value="O365OAuthTenantID"/>
      <ns1:Property name="O365OAuthRefreshToken"
value="O365OAuthRefreshToken"/>
```

Sample UMS Configuration File

```
<?xml version="1.0" encoding="UTF-8"?>
<ns1:MessagingConfiguration xmlns:ns0="http://www.oracle.com/ucs/messaging/
configtemplate" xmlns:ns1="http://www.oracle.com/ucs/messaging/config"
version="12.2.1.3.0">
   <ns1:Driver name="Test1" type="email" server="AdminServer" enabled="true">
      <ns1:Property name="SupportedDeliveryTypes" value="EMAIL"/>
      <ns1:Property name="SupportedContentTypes" value="*"/>
      <ns1:Property name="Capability" value="BOTH"/>
      <ns1:Property name="Cost" value=""/>
      <ns1:Property name="Speed" value=""/>
      <ns1:Property name="SupportedCarriers" value=""/>
      <ns1:Property name="SupportedProtocols" value="SMTP"/>
      <ns1:Property name="SupportsCancel" value="false"/>
      <ns1:Property name="SupportsReplace" value="false"/>
      <ns1:Property name="SupportsStatusPolling" value="false"/>
      <ns1:Property name="SupportsTracking" value="false"/>
      <ns1:Property name="SupportedStatusTypes"
value="DELIVERY_TO_GATEWAY_SUCCESS, DELIVERY_TO_GATEWAY_FAILURE,
USER_REPLY_ACKNOWLEDGEMENT_SUCCESS, USER_REPLY_ACKNOWLEDGEMENT_FAILURE"/>
      <ns1:Property name="SenderAddresses" value=""/>
      <ns1:Property name="SupportedApplicationNames" value=""/>
```

```
      <ns1:Property name="DefaultSenderAddress" value=""/>
      <ns1:Property name="SendingQueuesInfo" value="OraSDPM/
QueueConnectionFactory:OraSDPM/Queues/OraSDPMDriverDefSndQ1"/>
      <ns1:Property name="MailAccessProtocol" value="IMAP"/>
      <ns1:Property name="AutoDelete" value="false"/>
      <ns1:Property name="Debug" value="true"/>
      <ns1:Property name="CheckMailFreq" value="30"/>
      <ns1:Property name="DisconnectAfterPoll" value="false"/>
      <ns1:Property name="ReceiveFolder" value="INBOX"/>
      <ns1:Property name="OutgoingMailServer" value="smtp.office365.com"/>
      <ns1:Property name="OutgoingMailServerPort" value="587"/>
      <ns1:Property name="OutgoingMailServerSecurity" value="None"/>
      <ns1:Property name="OutgoingDefaultFromAddr" value=""/>
      <ns1:Property name="OutgoingUsername"
value="umstest@myumsoauth.onmicrosoft.com"/>
      <ns1:Property name="OutgoingPassword" value=""/>
      <ns1:Property name="IncomingMailServer" value="outlook.office365.com"/>
      <ns1:Property name="IncomingMailServerPort" value="993"/>
      <ns1:Property name="IncomingMailServerSSL" value="false"/>
      <ns1:Property name="IncomingMailIDs"
value="umstest@myumsoauth.onmicrosoft.com"/>
      <ns1:Property name="IncomingUserIDs"
value="umstest@myumsoauth.onmicrosoft.com"/>
      <ns1:Property name="IncomingUserPasswords" value=""/>
      <ns1:Property name="ProcessingChunkSize" value="100"/>
      <ns1:Property name="ImapAuthPlainDisable" value="false"/>
      <ns1:Property name="CNSMode" value="false"/>
      <ns1:Property name="SMTPAuthUseOAuth2" value="true"/>
      <ns1:Property name="ImapAuthUseOAuth2" value="true"/>
      <ns1:Property name="OAuth2AccessTokenSupplierFactory"
value="oracle.sdpinternal.messaging.oauth.ums.ms.UMSO365AccessTokenSupplierFac
tory"/>
      <ns1:Property name="O365OAuthClientID" value="O365OAuthClientID"/>
      <ns1:Property name="O365OAuthTenantID" value="O365OAuthTenantID"/>
      <ns1:Property name="O365OAuthRefreshTokenScope" value="https://
graph.microsoft.com/IMAP.AccessAsUser.All https://graph.microsoft.com/
SMTP.Send"/>
      <ns1:Property name="O365OAuthRefreshToken"
value="O365OAuthRefreshToken"/>
   </ns1:Driver>
</ns1:MessagingConfiguration>
```

Generating Tokens Using Authorization Code Flow

This section includes the following topics:

- Initiating an App Registration
- Generating Tokens
- Verifying Tokens

Initiating an App Registration

1. Log into Microsoft Azure using the following URL:

   https://portal.azure.com/

2. Enter your MS O365 account credentials.

3. Click the hamburger menu on top.
4. Navigate to **Azure Active Directory**.

Creating Tenant

5. Click **Create a Tenant**.

> **Note:**
>
> You must create a tenant if no tenant is present in your list.

6. In the **Configuration** page, provide details in the following fields:
   - Organization name
   - Initial domain name
   - Country/Region
7. Click **Review + Create**.
8. In the **Azure Active Directory** page, select **App registrations** from the left side navigation menu.
9. Click **New registration**.
10. In the **Register an application** page, provide details in the **Name** field.
11. Under **Supported account types**, select **Accounts in this organizational directory only (myumsoauth only - Single tenant)**.
12. Click **Register**.
13. In the left side navigation menu, select **Authentication**.
14. Click **Add a platform**.
15. Under **Configure platforms**, select **Mobile and desktop applications**.
16. In **Configure Desktop + devices** page, select **https://login.microsoftonline.com/common/oauth2/nativeclient** under **Redirect URIs**.

Defining API Permissions

17. In the left side navigation menu, select **API permissions**.
18. Click **Add a permission**.
19. In **Request API permission** page, select **Delegated permissions**.
20. In **Select permissions** field, enter the following:
    - IMAP.AccessAsUser.All
    - SMTP.Send
21. Select **IMAP.AccessAsUser.All** and **SMTP.Send** listed under **Permission**.

Generating Client Secret

22. In the left side navigation menu, select **Certificates & Secrets**.
23. Click **New client secret**.
24. In the **Add a client secret**, provide details in the following fields:
    - Description
    - Expires

25. Click **Add**.

Generating Tokens

Perform the following steps to generate access and refresh tokens:

1. Open Microsoft URL with the following details:

   https://login.microsoftonline.com/<your tenant id>/ oauth2/v2.0/authorize?client_id=<your client id>&scope=IMAP.AccessAsUser.All SMTP.Send&response_type=code&redirect_uri=https://login.microsoftonline.com/common/oauth2/nativeclient

   Example:

   https://login.microsoftonline.com/887c9fe4-c2e4-4b73-96e7-3f034cde3332/oauth2/v2.0/authorize?client_id=ce347d51-92a8-47ec-85a8-b71a903affc4&scope=IMAP.AccessAsUser.All SMTP.Send&response_type=code&redirect_uri=https://login.microsoftonline.com/common/oauth2/nativeclient

2. Enter your MSO 365 account credentials when prompted.
3. Select **Consent on behalf of your organisation** and click **Accept**.

   You will be redirected to a URL with the following details:

   https://login.microsoftonline.com/common/oauth2/nativeclient?code=<code>& session_state=<session state>

   Example

   https://login.microsoftonline.com/common/oauth2/nativeclient?code=0.AVYA5J98iOTCc0uW5z8DTN4zMlF9NM6okuxHhai3GpA6_8RWAM8.AQABAAIAAAD--DLA3VO7QrddgJg7WevrBzhwQ-VnAhparfeQA3VEDSXbYwDZ87LvKWkmPUWNZZOTOSa1Ja2rJOuizFnqqnDfFWj4UwryPjBNMeIAeHAgJ_RzhZTPCepmlcV_q9-93V6n0ASfjPbKwzN7A9XGRsZ8HJSJ-vMwGOAZvG3O8ywkMj6m_kOYSkCHEaiRMhSJRrIokpL0b70STHBxeaTXM4u71Mv0C3z-PdJqa4Rqwoboo77CF7hhLMHhtil9tJxSlA6Gvh6JVj0qXH27WvFQO7ZIUqrJZrDae2uCTVwyo70hJJ41trZsm6px8diWUf_zgnxuHAt4YCHNmj5TO-laj-02CKsHWpgzDSk7dduhl9KPnYerxLeE1T8EXjodG289sbsoz939yZe-mMbjyKOIzM-eM7B_WUrpt2zFKYQne0wETbi_o2RAxieMHmV-w2TFv54QeGAyXZ7ECkZuiyVuPPIaYnSRrV67Ps_Rj4LfcJJI2kj3QaRCGAvnUJRoJUb-0m2Bv23DtqPpn0o6tRPSUxiuLOHSWpxLhK7IW5nSfyeRiwzTC9-4YUpAfqf4N0u2yHMEAqrbTKOocJKm7KkadhHb1jyJ_rUYJD-0qHugqEtF0aqFvmvEDB-pOAsPXbtO6rycIjexEZgs26mSKeKv65yZ2lBLtN1XmUXa3Rvoec_GejvYwUeFINICkRa1NCrxMukgAA&session_state=fb6c0bf9-c9ea-43c0-ab07-7523d99e491c

4. Copy the code parameter value from the above URL.
5. Send a POST request with required parameters as described below:

   URL

   https://login.microsoftonline.com/<your tenant id>/oauth2/v2.0/token

   Parameters

   The following is a list of the POST request parameters.

```
client_id: <your client application id>
scope: IMAP.AccessAsUser.All SMTP.Send offline_access
grant_type: authorization_code
```

```
code: The code you got from Step #2
redirect_uri: https://login.microsoftonline.com/common/oauth2/nativeclient
```

Curl Command

The following example shows how to generate access and refresh tokens using curl:

```
curl --location --request POST 'https://login.microsoftonline.com/887c9fe4-
c2e4-4b73-96e7-3f034cde3332/oauth2/v2.0/token' \

                        --header 'Content-Type: application/x-www-form-
urlencoded' \
                        --data-urlencode 'client_id=ce347d51-92a8-47ec-
85a8-b71a903affc4' \
                        --data-urlencode 'scope=IMAP.AccessAsUser.All
SMTP.Send offline_access' \
                        --data-urlencode 'grant_type=authorization_code' \

                        --data-urlencode
'code=0.AVYA5J98iOTCc0uW5z8DTN4zMlF9NM6okuxHhai3GpA6_8RWAM8.AQABAAIAAAD--
DLA3VO7QrddgJg7WevrBzhwQ-
VnAhparfeQA3VEDSXbYwDZ87LvKWkmPUWNZZOTOSa1Ja2rJOuizFnqqnDfFWj4UwryPjBNMeIAe
HAgJ_RzhZTPCepmlcV_q9-93V6n0ASfjPbKwzN7A9XGRsZ8HJSJ-
vMwGOAZvG3O8ywkMj6m_kOYSkCHEaiRMhSJRrIokpL0b70STHBxeaTXM4u71Mv0C3z-
PdJqa4Rqwoboo77CF7hhLMHhtil9tJxSlA6Gvh6JVj0qXH27WvFQO7ZIUqrJZrDae2uCTVwyo70
hJJ41trZsm6px8diWUf_zgnxuHAt4YCHNmj5TO-laj-
02CKsHWpgzDSk7dduhl9KPnYerxLeE1T8EXjodG289sbsoz939yZe-mMbjyKOIzM-
eM7B_WUrpt2zFKYQne0wETbi_o2RAxieMHmV-
w2TFv54QeGAyXZ7ECkZuiyVuPPIaYnSRrV67Ps_Rj4LfcJJI2kj3QaRCGAvnUJRoJUb-
0m2Bv23DtqPpn0o6tRPSUxiuLOHSWpxLhK7IW5nSfyeRiwzTC9-
4YUpAfqf4N0u2yHMEAqrbTKOocJKm7KkadhHb1jyJ_rUYJD-0qHugqEtF0aqFvmvEDB-
pOAsPXbtO6rycIjexEZgs26mSKeKv65yZ2lBLtN1XmUXa3Rvoec_GejvYwUeFINICkRa1NCrxMu
kgAA' \
                        --data-urlencode
'redirect_uri=https://login.microsoftonline.com/common/oauth2/nativeclient'
```

The access and refresh token details will be in the response.

Verifying Tokens

You must verify the validity of the refresh token.You must send a POST request to MS OAuth token generation URL to generate a new access token. The verification is complete once you receive the access token.You must set the same refresh token value in `usermessagingconfig.xml`.

You must send a POST request using curl or POSTMAN or any suitable client to the URL given below:

URL

https://login.microsoftonline.com/<your tenant id>/oauth2/v2.0/token

Parameters

The following is a list of the POST request parameters.

```
client_id: <your client application id>
scope: IMAP.AccessAsUser.All SMTP.Send offline_access
grant_type: refresh_token
refresh_token: The refresh token that you just obtained
redirect_uri: https://login.microsoftonline.com/common/oauth2/nativeclient
```

Curl Command

The following example shows how to verify the validity of the refresh token using curl:

```
curl --location --request POST 'https://login.microsoftonline.com/887c9fe4-
c2e4-
4b73-96e7-3f034cde3332/oauth2/v2.0/token' \
                        --header 'Content-Type: application/x-www-form-
urlencoded'
\

        --data-urlencode 'client_id=eba0d084-3a6b-4520-83b2-
52ed27a15b7b' \

        --data-urlencode 'scope=https://outlook.office.com/
IMAP.AccessAsUser.All
        https://outlook.office.com/SMTP.Send offline_access'
\
        --data-urlencode 'grant_type=refresh_token' \
        --data-urlencode
'refresh_token=0.AVYA5J98iOTCc0uW5z8DTN4zMoTQoOtrOiBFg7JS7SehW3tWAM8.AgABAAEAA
AD--
DLA3VO7QrddgJg7WevrAgDs_wQA9P-m1OnkkabmlTMmSH_G5cOl_6_1Cr-
_NoAjwozM1QItkpuWGtxkLiSmAm-o5jg5zTHhFdcxwknuDwCtQ9bNxT32a8xGmeei-
fo5GycV7V6QqApR0jcZqhoGQx1168WeCHBDk7HcTR9RZagoPIgpYCgArdKwSypBOU5s37s2gJG3e9m
_flC9
GK0VxycmnQHWmCnFr91-QYeQNSsWA-nMQmhpkmFKBBlIm6BhWz5XCJA1m0J7-
ZukuJp_D14OsFhwzzNszH-
wAd9_XEBq62NjTeADkvZ28-
2Ppqiky4hlPi0Go4JtpPh0zsDcOIUJrdJPTIY8GlEy2yJOslksctz4gh_Dg1eq35m_s1EeV8HVPOd3
vXZcu
unHWKgrfPIZE7MslS_iu9xeslBcGvEpJNtpC4kdB8uTHLo2g8U8W7nZgyjrc8r09yt0bv27eVWBIRc
UKuJe
yAs_mOIUij6eYpwMEAbuiPRd6efA9T-
arwB5iDbOrAfHt1FCnpJNXP166A22KcSx83CmLBrIsJu8wOpxfitj27fKo0sIOeE5e06GvDaZMLI6r
02kRW
xJuYdUVSnjosuryREwqdSio735YhM9K0wINMPm6yJTmBL85SuZONN5Nd8DgJCNDfCK_1FGyXeqQvy5
TjHRd
W3IBy0SjjAj2VPqtQ2IyuNXrpHxFWsgyqSpESrwkE6LZp6cXTc5DxNz6arilBojlOTKwX9q4KRnnyz
cSBX7
pgyziNWACxMgIK4_s-l7WJyjk-oNiwZMGnJgDNYaeLdffvIqtly6vmI' \
```

```
        --data-urlencode
'redirect_uri=https://login.microsoftonline.com/common/oauth2/nativeclient' \
```

Response

The following example shows the details of the access and refresh tokens in the Response body:

```
{
    "token_type": "Bearer",
    "scope": "EWS.AccessAsUser.All IMAP.AccessAsUser.All Mail.ReadWrite.All
SMTP.Send User.Read profile openid email",
    "expires_in": 5397,
    "ext_expires_in": 5397,
    "access_token":

"eyJ0eXAiOiJKV1QiLCJub25jZSI6IkpieWVRMkVFbXFMQklhWmk3cXhBdmhxX2I2d3RUTzBpWWlNR
nI1Mz

RzTGMiLCJhbGciOiJSUzI1NiIsIng1dCI6ImpTMVhvMU9XRGpfNTJ2YndHTmd2UU8yVnpNYyIsImtp
ZCI6I

mpTMVhvMU9XRGpfNTJ2YndHTmd2UU8yVnpNYyJ9.eyJhdWQiOiIwMDAwMDAwMy0wMDAwLTAwMDAtYz
AwMC0

wMDAwMDAwMDAwMDAiLCJpc3MiOiJodHRwczovL3N0cy53aW5kb3dzLm5ldC84ODdjOWZlNC1jMmU0L
TRiNz

MtOTZlNy0zZjAzNGNkZTMzMzIvIiwiaWF0IjoxNjU0NzQwMTUwLCJuYmYiOjE2NTQ3NDAxNTAsImV4
cCI6M

TY1NDc0NTg0OCwiYWNjdCI6MCwiYWNyIjoiMSIsImFpbyI6IkUyWmdZUGhXMVpiR1kvYkkySVF6MnM
vaTdt

WlBEL1h5ODF5aW5KZCt4Q3Y4TDJUOTR3d0EiLCJhbXIiOlsicHdkIl0sImFwcF9kaXNwbGF5bmFtZS
I6IlR

lc3QgT0F1dGgiLCJhcHBpZCI6ImViYTBkMDg0LTNhNmItNDUyMC04M2IyLTUyZWQyN2ExNWI3YiIsI
mFwcG

lkYWNyIjoiMCIsImZhbWlseV9uYW1lIjoiQmFidSIsImdpdmVuX25hbWUiOiJWZW5rYXRlc2giLCJp
ZHR5c

CI6InVzZXIiLCJpcGFkZHIiOiIxMjIuMTY3LjIyNi4yNiIsIm5hbWUiOiJWZW5rYXRlc2ggQmFidSI
sIm9p

ZCI6IjAwYWM5M2JlLTY1MGQtNGRhNy1iNDI2LTE3YzJiODI0ZGI5MiIsInBsYXRmIjoiMyIsInB1aW
QiOiI

xMDAzMjAwMEU1NTE2MThGIiwicHdkX2V4cCI6IjAiLCJwd2RfdXJsIjoiaHR0cHM6Ly9wb3J0YWwub
Wljcm

9zb2Z0b25saW5lLmNvbS9DaGFuZ2VQYXNzd29yZC5hc3B4IiwicmgiOiIwLkFWWUE1Sjk4aU9UQ2Mw
dVc1e
```

```
jhEVE40ek1nTUFBQUFBQUFBQXdBQUFBQUFBQUFCV0FNOC4iLCJzY3AiOiJFV1MuQWNjZXNzQXNVc2V
yLkFs

bCBJTUFQLkFjY2Vzc0FzVXNlci5BbGwgTWFpbC5SZWFkV3JpdGUuQWxsIFNNVFAuU2VuZCBVc2VyLl
JlYWQ

gcHJvZmlsZSBvcGVuaWQgZW1haWwiLCJzaWduaW5fc3RhdGUiOlsia21zaSJdLCJzdWIiOiJGSUdjc
nNzQU

ZTYnNTdW9VNy1KdlFySDlkRzNxbkpLcXpuMnV0dU9VdlpFIiwidGVuYW50X3JlZ2lvbl9zY29wZSI6
IkFTI

iwidGlkIjoiODg3YzlmZTQtYzJlNC00YjczLTk2ZTctM2YwMzRjZGUzMzMyIiwidW5pcXVlX25hbWU
iOiJ2

ZW5rYXRiYWJ1a3JAbXl1bXNvYXV0aC5vbm1pY3Jvc29mdC5jb20iLCJ1cG4iOiJ2ZW5rYXRiYWJ1a3
JAbXl

1bXNvYXV0aC5vbm1pY3Jvc29mdC5jb20iLCJ1dGkiOiJVUG9wS2xWLXVVT1B2X3FWVTB0RUFBIiwid
mVyIj

oiMS4wIiwid2lkcyI6WyI2MmU5MDM5NC02OWY1LTQyMzctOTE5MC0wMTIxNzcxNDVlMTAiLCJiNzlm
YmY0Z

C0zZWY5LTQ2ODktODE0My03NmIxOTRlODU1MDkiXSwieG1zX3N0Ijp7InN1YiI6IlVZWTFXS2hQaGl
RUDlf

bkdNcXJDcU8zVmVfZC1tZzlReVhZSEQ2alNOb1EifSwieG1zX3RjZHQiOjE2MDA2Njk1Mjh9.DT9fH
a8_IF
    8bZt6sAUt43ep0slEEKh3ZslSHvLee6cIgbP4ACns_XK6-Xv-f1zqsHHj-2uPU3pLwP0_-
    0sOqILiv4dBEDKqUElzb54EqHQix2-yXqomKCZBspF245kpDX-
    dSbu3hJ3lh_qSTPpTG7jAXvWJdEyST1o1X0mTCt1pn1HAu2GxWhWFf2daVVcmlfuLtjyW82T-
    xKY5NlZvlzx5dxn8-M4Txkg-

GInwYtcRgrsxHG9HyIY1dNfjoRv5k4uU1tRzQeTrNOa62E6hk26LIdZi9zGgrAV0KVGDbsxnkrmKZy
-
    7JkvhGmczTg1PCnpbCdmFnNy1UGf3SjmEOKQ",

    "refresh_token":
    "0.AVYA5J98iOTCc0uW5z8DTN4zMoTQoOtrOiBFg7JS7SehW3tWAM8.AgABAAEAAAD--
    DLA3VO7QrddgJg7WevrAgDs_wQA9P-m1OnkkabmlTMmSH_G5cOl_6_1Cr-
    _NoAjwozM1QItkpuWGtxkLiSmAm-o5jg5zTHhFdcxwknuDwCtQ9bNxT32a8xGmeei-

fo5GycV7V6QqApR0jcZqhoGQx1168WeCHBDk7HcTR9RZagoPIgpYCgArdKwSypBOU5s37s2gJG3e9m
_flC9
    GK0VxycmnQHWmCnFr91-QYeQNSsWA-nMQmhpkmFKBBlIm6BhWz5XCJA1m0J7-
ZukuJp_D14OsFhwzzNszH-
    wAd9_XEBq62NjTeADkvZ28-

2Ppqiky4hlPi0Go4JtpPh0zsDcOIUJrdJPTIY8GlEy2yJOslksctz4gh_Dg1eq35m_s1EeV8HVPOd3
vXZcu

unHWKgrfPIZE7MslS_iu9xeslBcGvEpJNtpC4kdB8uTHLo2g8U8W7nZgyjrc8r09yt0bv27eVWBIRc
UKuJe
    yAs_mOIUij6eYpwMEAbuiPRd6efA9T-
```

```
arwB5iDbOrAfHt1FCnpJNXP166A22KcSx83CmLBrIsJu8wOpxfitj27fKo0sIOeE5e06GvDaZMLI6r
02kRW

xJuYdUVSnjosuryREwqdSio735YhM9K0wINMPm6yJTmBL85SuZONN5Nd8DgJCNDfCK_1FGyXeqQvy5
TjHRd

W3IBy0SjjAj2VPqtQ2IyuNXrpHxFWsgyqSpESrwkE6LZp6cXTc5DxNz6arilBojlOTKwX9q4KRnnyz
cSBX7
    pgyziNWACxMgIK4_s-l7WJyjk-oNiwZMGnJgDNYaeLdffvIqtly6vmI"
}
```

Generating Tokens Using Client Credentials Flow

If O365OAuthClientSecret and O365OAuthClientSecretScope are provided in `usermessagingconfig.xml` file, the UMS uses these two values to generate the access token internally.

Perform the following steps to generate access tokens using O365OAuthClientSecret and O365OAuthClientSecretScope values:

1. Create a new account and configure it.

   For more information about configuring the account, see *Updating UMS Configuration File*.
2. Add the following new permissions:
   - IMAP.AccessAsApp
   - POP.AccessAsApp
   - SMTP.SendAsApp
3. Create a new client secret and copy it.

   > **Note:**
   >
   > The new client secret is visible only once (once you leave the page the secret will not be visible) so you must copy the new secret value and save it.

4. Connect PowerShell and add mailbox permissions as described below:
   a. **Install and Import ExchangeOnlineManagement**

      Run the following two commands:

      ```
      Install-Module -Name ExchangeOnlineManagement -allowprerelease

      Import-module ExchangeOnlineManagement
      ```

      > **Note:**
      >
      > If you are connecting for the first time, you must run the two commands.

   b. **Connect Exchange Online**

      Run the following command:

      ```
      Connect-ExchangeOnline -Organization <Tenant id>
      ```

      Example:

      ```
      Connect-ExchangeOnline -Organization 887c9fe4-c2e4-4b73-96e7-3f034cde3332
      ```

> **Note:**
>
> The Azure portal login window appears. You must log into the portal and close it.

c. **Add Mailbox Permission**

Run the following commands:

```
New-ServicePrincipal -AppId <> -ObjectId <>
```

Example:

```
New-ServicePrincipal -AppId eba0d084-3a6b-4520-83b2-52ed27a15b7b -ObjectId
31cb91f7-84fd-40e6-bb30-7b41371b95a7

Add-MailboxPermission -Identity <user_mail_id> -User <ObjectId> -
AccessRights FullAccess
```

Example:

```
Add-MailboxPermission -Identity venkatbabukr@myumsoauth.onmicrosoft.com -
User 31cb91f7-84fd-40e6-bb30-7b41371b95a7 -AccessRights FullAccess
```

5. Enter client secret in the `usermessagingcofnig.xml` file.
6. Restart the SOA servers.

> **Note:**
>
> If the O365OAuthClientSecret and O365OAuthClientSecretScope values are not provided then the system uses refresh token to generate access token.

Enabling SMTP AUTH

The Microsoft 365 admin center or Exchange Online PowerShell provides pre-mailbox setting to enable and disable SMTP AUTH.

Perform the following steps to enable SMTP AUTH on specific mailboxes:

1. Open the Microsoft 365 admin center using the following URL:

   https://admin.microsoft.com/
2. Enter your domain admin email ID and password.
3. In the left side navigation menu, click **Users**.
4. Select **Active Users**.
5. Click your user display name.

   A flyout appears with user details.
6. Select **Mail**.
7. Under **Email apps**, select **Manage email apps**.
8. In **Manage email apps** page, select the option **Authenticated SMTP**.
9. Click **Save changes**.

   For more information, see https://docs.microsoft.com/en-us/exchange/clients-and-mobile-in-exchange-online/authenticated-client-smtp-submission#enable-smtp-auth-for-specific-mailboxes.

Configuring SendAsDenied

You must configure SendAsDenied in the Microsoft admin portal.

Perform the following steps to configure SendAsDenied in the Microsoft admin portal:

1. Open the Microsoft 365 admin center using the following URL:

   https://admin.microsoft.com/
2. Enter your domain admin email ID and password.
3. In the left side navigation menu, click **Users**.
4. Select **Active Users**.
5. Click your user display name to view the user details.
6. Select **Mail**.
7. Under **Mailbox Permissions**, click **Send on behalf of permissions**.
8. Select **Add permission**.
9. Select the OAuth user email.
10. Restart the SOA service.

    You must wait for one hour to get the cache updated on the Microsoft side if it does not work instantly.

Configuring Multiple Inbound Email IDs

You can configure multiple inbound email IDs and give permission to the OAuth user.

Perform the following steps to configure an inbound email ID:

1. Open the Microsoft 365 admin center using the following URL:

   https://admin.microsoft.com/
2. Enter your domain admin email ID and password.
3. In the left side navigation menu, click **Users**.
4. Select **Active Users**.
5. Select the user that needs to be configured for UMS inbound flow.

   For example, *inbounduser1@domain.com*.
6. Select **Mail**.
7. Under **Mailbox Permissions**, click **Send on behalf of permissions**.
8. Select **Add permission**.
9. Select the user that needs to be configured as the OAuth is UMS email.

   For example, *oauthuser1@domain.com*.
10. Click **Add**.

> **Note:**
>
> You must ensure no password is saved in `usermessagingconfig.xml` for IncomingUserPasswords.

Configuring the SMPP Driver

Short Message Peer-to-Peer (SMPP) is a popular GSM SMS protocols. UMS includes a prebuilt implementation of the SMPP protocol as a driver that can send and receive short messages. If the sending feature is enabled, the SMPP driver opens one TCP connection to the Short Message Service Center (SMS-C) as a transmitter for sending messages. If the driver's receiving feature is enabled, it opens another connection to the SMS-C as a receiver for receiving messages. Only two TCP connections (both initiated by the driver) are needed for all communication between the driver and the SMS-C.

> **Note:**
>
> The SMPP Driver implements version 3.4 of the SMPP protocol and only supports connections to an SMS-C or an SMS gateway that supports this version.

- Common Properties

Common Properties

Table 4-11 lists common driver properties that are indicative of the capabilities of this driver for use by the messaging engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. For detailed description of these properties, refer to Table 4-2. For the complete list of available values, see *User Messaging Service Java API Reference*.

Table 4-11 Common SMPP Properties

Name	Mandatory	Default Value
InstanceName	Yes	SMPP-Driver
Capability	Yes	Both
SupportedDeliveryTypes	Yes	SMS
SupportedContentTypes	Yes	text/plain
SupportedStatusTypes	No	DELIVERY_TO_GATEWAY_SUCCESS, DELIVERY_TO_GATEWAY_FAILURE
Cost	No	N/A
Speed	No	N/A
SupportedCarriers	No	N/A
Configuration Level	Yes	Server/Cluster
Supported Protocols	No	N/A
SenderAddresses	No	N/A
DefaultSenderAddress	No	N/A
Supported Application Names	No	Empty

- SMPP Custom Properties

SMPP Custom Properties

Table 4-12 lists properties specific to this driver and generally associated with configuring access to the remote gateway and certain protocol or channel-specific behavior.

Table 4-12 Custom SMPP Properties

Name	Description	Mandatory	Default Value
SmsAccountId	This value indicates the addresses that the SMPP driver is requesting messages for from the server. The value is specified as a UNIX Regular Expression. For example, "555" would specify a single address, and "^123|^789" would indicate all addresses starting with 123 or 789.	Yes	N/A
SmsServerHost	The name (or IP address) of the SMS-C server.	Yes	N/A
TransmitterSystemId	The account ID that is used to send messages.	Yes	N/A
ReceiverSystemId	The account ID that is used to receive messages.	Yes	N/A
TransmitterSystemType	The type of transmitter system. The default is Logica.	Yes	The default value is Logica.
ReceiverSystemType	The type of receiver system. The default is Logica.	Yes	The default value is Logica.
TransmitterSystemPassword	The password of the transmitter system. This includes Type of Password (choose from Indirect Password/Create New User, Indirect Password/Use Existing User, and Use Cleartext Password) and Password.	Yes	N/A
ReceiverSystemPassword	The password for the receiver system. This includes Type of Password (choose from Indirect Password/Create New User, Indirect Password/Use Existing User, and Use Cleartext Password) and Password.	Yes	N/A
ServerTransmitterPort	The TCP port number of the transmitter server.	Yes	N/A
ServerReceiverPort	The TCP port number of the receiver server.	Yes	N/A

Table 4-12 (Cont.) Custom SMPP Properties

Name	Description	Mandatory	Default Value
`DefaultEncoding`	Used for incoming messages. If the SMS-C specifies the encoding to SMSC Default Alphabet, then this is the encoding that SMPP driver will assume. Choose from the drop-down list among the following: IA5, UCS2, GSM_DEFAULT, ISO-8859-1	No	IA5
`PreferredEncoding`	Used for outgoing messages. If set, the text will be encoded according to the PreferredEncoding parameter. If the encoding fails (i.e. a character cannot be encoded using the specified encoder) then the driver uses the 16-bit encoding UCS2. If not set, the driver will attempt to derive an encoding from the UMS Message Content-Type header. Choose from the drop-down list among the following: IA5, UCS2, GSM_DEFAULT, ISO-8859-1	No	IA5
`LocalSendingPort`	The local TCP port used by the SMPP driver to send messages to the SMS-C.	No	N/A
`LocalReceivingPort`	The local TCP port used by the SMPP driver to receive messages from the SMS-C.	No	N/A
`LocalAddress`	The hostname (or IP address) of the server that hosts the SMPP driver.	No	N/A
`WindowSize`	The window size for SMS. This value must be a positive number. Default is **1**.	No	1
`EnquireInterval`	The interval, in seconds, to send an enquire message to the SMS-C. The default is 30 seconds.	No	30
`ThrottleDelay`	The delay, in seconds, between throttles. The default is 30.	No	30
`BindRetryDelay`	The minimum delay, in seconds, between bind entry attempts. Default is 30.	No	30
`ResponseTimer`	Time lapse allowed between SMPP request and response, in seconds. The default is 30.	No	30

Table 4-12 (Cont.) Custom SMPP Properties

Name	Description	Mandatory	Default Value
RegisteredDeliveryMask	The registered delivery bit mask. The default is 0xFF, which does not change the delivery flag value.	No	0xFF
RangeSetNull	Set to true to set the address range field of BIND_RECEIVER to null. Set to false (the default value) to set the address range field to SmsSystemId. The default is Disabled.	No	Disabled
PriorityAllowed	The highest priority the SMPP Driver will set on a message to the SMS-C. The UMS Message priority set by the client application is translated into SMPP priority, but limited by PriorityAllowed. The range is 0 (normal) to 3 (highest). The default is 0.	No	0
BulkSending	Set this value to enabled (the default) to enable sending messages in bulk to the SMS-C.	No.	Enabled
PayloadSending	If you enable this property, the SMPP driver always uses the message_payload parameter that is defined in the SMPP specification, while sending a message to the SMS-C. The default is Disabled.	No	Disabled
SourceTon	The type of number (TON) for ESME address(es) served through SMPP receiver session. The default is 0.	No	0
SourceNpi	The numbering plan indicator (NPI) for ESME address(es) served through the SMPP receiver session. The default is 0.	No	0
DestinationTon	The TON for destination. The default is 0.	No	0
DestinationNpi	The NPI for destination. The default is 0.	No	0
MaxChunks	The maximum SMS chunks for a message. The default is **-1** (no maximum).	No	-1 (no maximum)
ChunkSize	The maximum size of each SMS message chunk. Default is 160.	No	160

Table 4-12 (Cont.) Custom SMPP Properties

Name	Description	Mandatory	Default Value
`LongMessageSending`	Supports sending long messages by setting the optional SMPP parameters `sar_msg_ref_num`, `sar_total_segments` and `sar_segment_seqnum` for fragmented messages. The default value is Enabled.	No	Enabled
`DatagramMessageMode`	Supports datagram message mode. The default is Disabled.	No	Disabled
`Optional Params`	Supports passing of additional parameters (TLVs) along with SMS.	No	Empty

Configuring the XMPP Driver

The XMPP Driver provides unidirectional or bidirectional access from Oracle Fusion Middleware to end users for real-time IM through the Extensible Messaging and Presence Protocol (XMPP). This driver enables end users to receive alert notifications or interactively chat with applications through their IM client of choice.

Perform the following tasks to configure the XMPP Driver.

Task 1: Setting Up Ejabberd

XMPP driver must be set up with Ejabberd which is an XMPP server (Jabber server), MQTT broker, and SIP gateway built to create real time services such as massive chat and instant communication.

For information about how to set up Ejabberd, see Installing ejabberd.

The admin account details that include SERVERNAME, USERNAME, and PASSWORD are configured to set up the driver.

To disable the SSL, the configuration file for `Ejabberd ejabberd.yml` in `/opt/ejabberd/conf/` must be modified:

```
listen:
  -
    port: 5222
    ip: "::"
    module: ejabberd_c2s
    max_stanza_size: 262144
    shaper: c2s_shaper
    access: c2s
    starttls_required: false
```

Task 2: Configure XMPP Driver in UMS

If you use EM to configure the driver in UMS, you must create or update the driver properties.

Perform the following steps to update the Drivers Properties:

1. Log into **EM**.
2. Click **EM**, select **User Messaging Service**, click **usermessagingdriver-xmpp**, select **Driver Properties**, and then click **Create**.
3. In the Driver Properties page, update the following fields:

Table 4-13 Driver Properties

Field	Update
Name	APPNAME
IM Server Host	SERVERNAME
IM Server Port	5222
IM Server Username	USERNAME@SERVERNAME
IM Server Password	IM Server Password section.
Type of Password	Use Cleartext Password.
Password	Enter the password configured in Ejabberd.
Security Mode	None
Enable SASL Authentication	Disabled

If you are not using EM to configure the driver in UMS, add the XMPP driver configuration to the configuration file as follows:

```
<ns1:Property name="IMServerHost" value="SERVERNAME"/>
<ns1:Property name="IMServerPort" value="5222"/>
<ns1:Property name="IMServerUsername" value="USERNAME@SERVERNAME"/>
<ns1:Property name="IMServerPassword" value="PASSWORD"/>
<ns1:Property name="SecurityMode" value="None"/>
<ns1:Property name="SASLAuthenticationEnabled" value="false"/>
```

Task 3: Validate and Test the Driver

You must validate the driver to ensure the configuration is correct. After validating, test the driver using the Ejabberd supported clients such as Coccinella.

- Common Properties

Common Properties

Table 4-14 lists the common driver properties that are indicative of the capabilities of this driver for use by the messaging engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. For detailed description of these properties, see Table 4-2. For the complete list of available values, see *User Messaging Service Java API Reference*.

Table 4-14 Common XMPP Properties

Name	Mandatory	Default Value
`InstanceName`	Yes	usermessagingdriver-xmpp

Table 4-14 (Cont.) Common XMPP Properties

Name	Mandatory	Default Value
Capability	Yes	SEND, RECEIVE
SupportedDeliveryTypes	Yes	IM
SupportedContentTypes	Yes	text/plain
SupportedStatusTypes	No	DELIVERY_TO_GATEWAY_SUCCESS, DELIVERY_TO_GATEWAY_FAILURE
Cost	No	N/A
Speed	No	N/A
SupportedCarriers	No	N/A
Configuration Level	Yes	Server/Cluster
Supported Protocols	No	XMPP
Supported Application Names	No	Empty
Driver Type	Yes	User Messaging XMPP Driver
SenderAddresses	No	N/A
DefaultSenderAddress	No	N/A

- XMPP Custom Properties

XMPP Custom Properties

Table 4-15.lists the custom properties included in the XMPP Driver.

Table 4-15 Custom XMPP Properties

Name	Description	Mandatory	Default Values
IM Server Host	Jabber/XMPP server hostname.	Yes	
IM Server Port	Corresponding Jabber/XMPP server port.	Yes	5222
IM Server Username	Enter Jabber/XMPP user name to log in. You can also enter a complete Jabber ID if its domain name is different from the Jabber/XMPP server hostname.	Yes	
IM Server Password	Corresponding password for the username.	No	Indirect Password, Create New User
SecurityMode	Security mode to use when establishing connection to the server. Available options include the following: • None (Security is disabled and only un-encrypted connections are used) • TLS (Security via TLS encryption is used whenever it is available) • SSL (Security via SSL encryption is used)	No	TLS

Table 4-15 (Cont.) Custom XMPP Properties

Name	Description	Mandatory	Default Values
Enable SASL Authentication	Whether or not to use SASL authentication when logging into the server. If SASL authentication fails, then the driver uses non-SASL authentication.	No	Enabled

Configuring the APNS Driver

The Apple Push Notification Service (APNS) driver is a UMS driver that communicates with the APNS API server. The certificates that you get from Apple for your application needs to be saved in the WLS server, in the OPSS subsystem. For more information on pre-requisites for configuring the APNS driver, see Prerequisites for Configuring APNS Driver.

- Prerequisites for Configuring APNS Driver
- Common Properties
- APNS Custom Properties

Prerequisites for Configuring APNS Driver

To send push notification using the APNS driver, the driver needs access to the iOS application-specific certificates. The certificates (public and private keys) are obtained from Apple's developer portal.

Task 1: Installing a Trust Certificate from the Entrust

You must obtain the APNS keys (`.p12`) and certificate for the app from the mobile app team. To carry out any development tasks, use the sandbox keys and certificate provided for UMS.

Task 2: Importing the Certificate in the Domain

Run the following command to extract the alias name for the certificate:

```
keytool -list -v -keystore ent1_ums_demo.p12 -storetype PKCS12
```

The certificates are packaged in a PKCS #12 file (file extension p12 or pfx). Before the certificates can be imported into the Keystore service, the archive must be converted to a JavaKeyStore file (file extension jks).

Run the following command to convert the archive using the `keytool` command (which is part of the JDK):

```
keytool -importkeystore -destkeystore apns-prod-server-testapns-21112023.jks -
srckeystore ent1_ums_demo.p12 -srcstoretype PKCS12 -destalias apns-prod-server-
testapns-21112023 -deststorepass welcome1 -destkeypass welcome1 -alias
ent1_ums_demo
```

UMS comes with a predefined keystore called `apns` must be used for the imported certificates.

The following sample describes how the certificate can be imported into the Keystore service using WSLT:

```
getOpssService(name='KeyStoreService').importKeyStore(appStripe='ums',
name='apns', password='welcome1', aliases='apns-prod-server-testapns-21112023',
```

```
keypasswords='welcome1', type='JKS', permission=true, filepath="/scratch/
anegupta/keysncerts/apns-prod-server-testapns-21112023.jks")
```

> **Note:**
>
> The value aliases parameter (in the above command) must match both the alias property in the APNS driver configuration and the keystore name.

If a different keystore than `apns` is used, then UMS must be granted additional permission using the same `appstripe` value used to create the new keystore.

The UMS shared library `oracle.sdp.client` must be granted the `oracle.security.jps.service.keystore.KeyStoreAccessPermission` permission using the very same appstripe.

Run the following command to import the apns certificate to WebLogic's configured keystore (select the keystore configured in the environment):

```
keytool -importcert -keystore "..fmwhome12/wlserver/server/lib/DemoTrust.jks" -
storepass DemoTrustKeyStorePassPhrase -file apns.cer -alias "apns.cer"
```

Task 3: Update the New Alias in UMS DB in the Table driverproperties:

Updating driver properties in UMS can be done using MBean as described below:

MBean Name

```
oracle.ucs.messaging:Location=ESS_SOAServer_1,name=UserPrefsAdministration,type=S
DPMessagingRuntime
```

> **Note:**
>
> Any instance of SOA server can be used to invoke this MBean other than `ESS_SOAServer_1`.

Operation Name

```
saveDriverProperties
```

Parameters

P1 - driverName

P2 - propertyName

P3 - propertyValue

For example, saveDriverProperties ("usermessagingdriver-apns-HCM", "Alias","apns-prod-server-hcm-12122023").

Example

```
mBean =
ObjectName("oracle.ucs.messaging:Location=ESS_SOAServer_1,name=UserPrefsAdmini
stration,type=SDPMessagingRuntime")
```

```
set_param = ['usermessagingdriver-apns-HCM', 'Alias', 'apns-prod-server-
hcm-12122023']
set_type = ["java.lang.String", "java.lang.String", "java.lang.String"]
mbs.invoke(mBean, "saveDriverProperties", set_param, set_type)
```

Task 4: Import AAA Certificate for APNS Remote Notification Server

Establish a trusted connection with APNs to set up a remote notification server. To create a trusted connection, you must install the AAA Certificate Services root certificate on each server.

Download the certificate from the following link:

Setting up a remote notification server

After downloading the certificate, run the following command to import the certificate to the WebLogic configured keystore:

```
keytool -importcert -keystore "../fmwhome12/wlserver/server/lib/DemoTrust.jks" -
storepass DemoTrustKeyStorePassPhrase -file AAACertificateServices.crt -alias
"AAACertificateServices.crt"
```

> **Note:**
>
> Select the keystore configured in the environment.

Task 5: Configure APNS Driver in UMS

If you use EM to configure the driver in UMS, you must create or update the driver properties.

Perform the following steps to update the Drivers Properties:

1. Log into **EM**.
2. Click **EM**, select **User Messaging Service**, click **usermessagingdriver-xmpp**, select **Driver Properties**, and then click **Create**.
3. In the Drivers Properties page, update the following fields:

Table 4-16 Driver Properties

Field	Value
Name	APPNAME
Sender Address	URI:APNS:UMS_SENDERNAME
Service Mode	Sandbox
Keystore Name	apns
Alias	apns-sand-server
Mobile App Topic	com.oraclecorp.ums.demo

If you are not using EM to configure the driver in UMS, you must add the following APNS driver configuration to the configuration file:

```
<ns1:Property name="SenderAddresses" value="URI:APNS:UMS_SENDERNAME"/>
<ns1:Property name="ServiceMode" value="sandbox"/>
```

```
<ns1:Property name="KeyStoreName" value="apns"/>

#This Alias should be the same which is provided while importing CRM keys to
the weblogic's keystore
<ns1:Property name="Alias" value="apns-sand-server"/>

#Name of the mobile app topic to which push notification will be send. APNs
uses the app's bundle ID as the default topic, e.g.
'com.oraclecorp.example.myapp'.
<ns1:Property name="AppTopic" value="com.oraclecorp.ums.demo"/>
```

Task 6: Validate and Test the Driver

You must validate the driver to ensure the configuration is correct. After completing validation, the driver must be tested.

For information about saving driver properties, see Saving Driver Properties.

Common Properties

Table 4-17 shows common driver properties that are indicative of the capabilities of this driver for use by the messaging engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. For detailed description of these properties, refer to Table 4-2. For the complete list of available values, see *User Messaging Service Java API Reference*.

Table 4-17 Common Properties of the APNS Driver

Name	Mandatory	Default Values
InstanceName	Yes	usermessagingdriver-apns
Capability	Yes	SEND, RECEIVE
SupportedDeliveryTypes	Yes	URI
SupportedContentTypes	Yes	text/plain, application/json
SupportedStatusTypes	Yes	DELIVERY_TO_GATEWAY_FAILURE,DELIVERY_TO_GATEWAY_SUCCESS,DELIVERY_TO_DEVICE_FAILURE
Cost	No	N/A
Speed	No	N/A
SupportedCarriers	Yes	N/A
Configuration Level	Yes	Server/Cluster
Supported Protocols	No	apns
Supported Application Names	No	Empty
Driver Type	No	User Messaging APNS Driver
SenderAddresses	No	N/A
DefaultSenderAddress	No	N/A

APNS Custom Properties

Table 4-18 lists configurable properties specific to the APNS driver.

Table 4-18 Custom Properties of the APNS Driver

Name	Description	Mandatory	Default Values
Service Mode	Determines the APNs production environment	Yes	
Keystore Name	Name of the keystore in KSS which holds the private key and certificated used for communication with APNs. UMS must be granted read permission to the keystore if a non-default keystore is used.	Yes	apns
Alias	Alias for the private key certificate pair in the keystore.	Yes	
Mobile App Topic	Name of the mobile app topic (or apple mobile app ID) to which push notification is sent. APNs uses the app's bundle ID as the default topic.	Yes	

Configuring GCM Driver

Google Cloud Messaging (GCM) driver is a UMS driver for mobile push notification service. It can send mobile push notifications to Android applications.

You must perform the following tasks to configure the GCM Driver.

Task 1: Obtaining Private Key File for App Service Account

GCM driver must be set up with the service account which is done by obtaining a private key file (JSON file). Perform the following steps to obtain the private key file:

1. Open Firebase Console using the following URL:
   https://console.firebase.google.com
2. Click the **Project Settings** icon.
   The **Project Settings** page is displayed.
3. Select **Service Accounts**.
4. In the **Firebase Admin SDK** dialog box, click **Generate new private key**.
   A JSON file is generated with details of the private key.
5. Save the JSON file.

Task 2: Configuring GCM Driver in UMS

If you use EM to configure the driver in UMS, you must create or update the driver properties.

Perform the following steps to update the Drivers Properties:

1. Navigate to the Drivers Properties page as shown below:
   EM, and then **User Messaging Service**, and then **usermessagingdriver-apns**, and then **Driver Properties**, and then **Create**
2. In the Drivers Properties page, update the following fields:

Table 4-19 Driver Properties

Field	Update
Name	GCM_APPNAME
Sender Address	URI:GCM:APPNAME
Service Mode	Production
FCM Service Account JSON	
Type of Password	Indirect Password, Create new User
Indirect Username/Key	APPNAMEKEY
Password	Enter details of the Private key file (JSON file) obtained in Task 1. The file is added to WebLogic's credential store.

If you are not using EM to configure the driver in UMS, you must add the private key to the credential store.

Run the following WLST command to add the private key to the credential store:

```
createCred(map="UCS", key="UMSDriver.GCM_APPNAME.ApiKey.APPNAMEKEY",
user="UMSDriver.GCM_APPNAME.ApiKey.APPNAMEKEY", password="<content of private key
file obtained in Task 1>", desc="")
```

For more information, see OPSS Security Store WLST Commands.

Add the GCM driver configuration to the configuration file to configure the driver in UMS as follows:

```
<ns1:Property name="SenderAddresses" value="URI:GCM:APPNAME"/>
<ns1:Property name="ApiKey" value="-
>UCS:UMSDriver.GCM_APPNAME.ApiKey.APPNAMEKEY"/>
<ns1:Property name="ServiceMode" value="production"/>
```

Task 3: Validate and Test the Driver

You must validate the driver to ensure the configuration is correct. After completing validation, the driver must be tested.

- Common Properties

Common Properties

Table 4-20 lists common driver properties that are indicative of the capabilities of this driver for use by the messaging engine when routing outbound messages. Some properties are set by the driver developer and do not normally require modification, while others can be modified by the administrator to change the routing behavior. For detailed description of these properties, refer to Table 4-2. For the complete list of available values, see *User Messaging Service Java API Reference*.

Table 4-20 Common Properties GCM Driver

Name	Mandatory	Default Values
InstanceName	Yes	usermessagingdriver-gcm
Capability	Yes	SEND

Table 4-20 (Cont.) Common Properties GCM Driver

Name	Mandatory	Default Values
SupportedDeliveryTypes	Yes	URI
SupportedContentTypes	Yes	text/plain
SupportedStatusTypes	Yes	DELIVERY_TO_GATEWAY_FAILURE,DELIVERY_TO_GATEWAY_SUCCESS,DELIVERY_TO_DEVICE_FAILURE
Cost	No	N/A
Speed	No	N/A
SupportedCarriers	Yes	N/A
Configuration Level	Yes	Server/Cluster
Supported Protocols	No	gcm
Supported Application Names	No	Empty
Driver Type	No	User Messaging GCM Driver
SenderAddresses	No	N/A
DefaultSenderAddress	No	N/A

- GCM Custom Properties

GCM Custom Properties

Table 4-21 lists properties specific to this driver and generally associated with configuring access to the remote gateway and certain protocol or channel-specific behavior.

Table 4-21 Custom Properties GCM Driver

Name	Description	Mandatory	Default Values
FCM Service Account JSON	Firebase service account JSON is used to authenticate Google Firebase API. This provides the driver authorized access to Google Firebase services.	Yes	Indirect Password, Create New User
Service Mode	Determines which environment the GCM driver sends notifications. Production means that notifications are sent to the URL Google's Firebase API. Local means that notifications are sent to the URL specified in parameter LocalEndpointURL.	Yes	Production
Local Endpoint URL	URL for the GCM service. It is used only if the Service Mode is set to the value local. The parameter is mandatory when the Service Mode is set to local.	No	

Configuring User Messaging Service Access to the LDAP User Profile

As part of the LDAP provider setup in a UMS deployment, you configure the **User Name Attribute** through the WebLogic Remote Console. If you configure that attribute with a value other than the default *cn* or if the user's email address is stored in an LDAP attribute which is different from *mail*, you must make an additional configuration change in Oracle Platform Security Services (OPSS) for UMS to successfully access the user profile to obtain the list of communication channels provisioned in LDAP, such as business email.

For more information about Oracle Platform Security Services (OPSS), see *Securing Applications with Oracle Platform Security Services*.

To configure access to the LDAP user profile:

1. Configure the Identity Store to use LDAP by following instructions in Fusion Middleware Enterprise Deployment Guide for Oracle Business Intelligence.

   > **Note:**
   >
   > You may have other properties defined in the Backing Up Configuration Files section.

2. To use the value of the User Name Attribute while searching the back-end LDAP server for user profile, add the following element:

   ```
   <property name="username.attr" value="username_attribute_value"/>
   ```

   where `username_attribute_value` is the value of the User Name Attribute property in the LDAP provider configuration. For instance, if the value of the User Name Attribute is `mail`, add the following line:

   ```
   <property name="username.attr" value="mail"/>
   ```

   The following sample code shows the above line inserted in the `jps-config.xml` file:

   ```
   <!-- JPS WLS LDAP Identity Store Service Instance -->

   <serviceInstance name="idstore.ldap" provider="idstore.ldap.provider">

     <property name="idstore.config.provider"
   value="oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvide
   r"/>

     <property name="CONNECTION_POOL_CLASS"
   value="oracle.security.idm.providers.stdldap.JNDIPool"/>

     <property name="username.attr" value="mail"/>

   </serviceInstance>
   ```

If the LDAP attribute containing the user's business email addresses is something other than the `mail` attribute, add the following element:

```
<property name="PROPERTY_ATTRIBUTE_MAPPING"
value="BUSINESS_EMAIL=attr_containing_email"/>
```

where `attr_containing_email` is the attribute name in the LDAP provider that contains the user's email address. For instance, if the user attribute containing the email address is `externalEmail`, add the following line:

```
<property name="PROPERTY_ATTRIBUTE_MAPPING" value="BUSINESS_EMAIL=externalEmail"/>
```

The following sample code shows the above line inserted in the `jps-config.xml` file:

```
<!-- JPS WLS LDAP Identity Store Service Instance -->

<serviceInstance name="idstore.ldap" provider="idstore.ldap.provider">

  <property name="idstore.config.provider"
value="oracle.security.jps.wls.internal.idstore.WlsLdapIdStoreConfigProvide
r"/>

  <property name="CONNECTION_POOL_CLASS"
 value="oracle.security.idm.providers.stdldap.JNDIPool"/>

  <property name="PROPERTY_ATTRIBUTE_MAPPING" value="BUSINESS_
EMAIL=externalEmail"/>

</serviceInstance>
```

3. Restart your domain.

Using Oracle User Messaging Service for Group Messaging

In addition to supporting bi-directional mutli-channel messaging through a variety of channels, UMS supports group messaging. This feature includes sending a message to a group of users by sending it to a group URI, or sending a message to LDAP groups (or enterprise roles) and application roles.

The group messaging feature enhances the capability of UMS by providing support for the following:

- Sending messages to a group
- Sending messages to a group through a specific channel
- Sending messages to an application role
- Sending messages to an application role through a specific channel

For more information about sending messages to groups and application roles, see "Sending Group Messages" in *Developing Applications with Oracle User Messaging Service*.

The group messaging feature does not require any new configuration of UMS. It reuses the UMS utility to access the User Role API. Since the User Role API configuration is not possible in UMS, any such configuration is done outside UMS. The User Role API is automatically configured to use the first Oracle WebLogic Server authenticator and does not require any special configuration.

Note:

For UMS to be able to resolve an application role, specific security grants are required. The application deployer must configure these security grants using WLST commands as shown in the following example:

```
connect('weblogic','welcome1','t3://host.example.com:7601')

grantPermission(codeBaseURL="file:MW_HOME/user_projects/domains/
DOMAIN_NAME/servers/SERVER_NAME/tmp/_WL_user/
usermessagingserver/-",permClass="oracle.security.jps.service.policyst
ore.PolicyStoreAccessPermission",permTarget="context=APPLICATION,name=
<appStripe>",permActions="getApplicationPolicy"
)
```

For more information about the security commands, see *Infrastructure Security WLST Command Reference*.

Configuring Automatic Message Resend

In 14*c*, the automatic resend feature can be configured to automate the administrator's resend. This means that when a message send attempt is classified as a complete failure, then the message is automatically scheduled for resend.

This is repeated until the message is successfully sent or the configured number of resends is achieved. The delay time and the maximum number of resends can be configured. Functionally, this is the same as an administrator manually resending the messages when the delay time has expired. The purpose of the automatic resend is to resolve temporary network problems or temporary unavailability of backend services.

The UMS server configuration parameters, `ResendDefault`, `ResendDelay`, and `ResendMax` have been introduced for configuring this feature. For more information about these parameters, see Table 4-1.

The number of resend attempts is configured for the server, but may be overridden programmatically per message by the client. The client can specify the number of resends to be used per message to override the `ResendDefault` server configuration parameter. Note that although overridden, it is limited by the `ResendMax` configuration parameter.

For more information about setting the number of resend attempts programmatically, see sections "Using UMS Java API to Specify Message Resends" and "Using UMS Web Service API to Specify Message Resends" in *Developing Applications with Oracle User Messaging Service*.

> **Note:**
>
> If message resend fails even after automatically trying to resend the message the maximum number of times, then the administrator can send it manually from the Enterprise Manager. The resend counter will be reset. If the maximum number of resends is configured to 0, then the behaviour will be identical to that in 12*c*, that is an administrator will have to manually select the failed message and resend it using the Enterprise Manager.

Securing the Oracle User Messaging Service

The User Communications Preferences User Interface can be secured at the transport-level using Secure Sockets Layer (SSL). By default, all deployed web services are unsecured. Web Service Security should be enabled for any services that are deployed in a production environment.

To enable SSL in the Oracle WebLogic Server, see "Configure SSL for Oracle WebLogic Server" in the *Administering Oracle Fusion Middleware*. This step is sufficient to secure the User Communication Preferences User Interface.

UMS supports the use of Oracle Web Services Manager WS-Security policies to protect UMS web services. For more information about Oracle Web Services Manager, see "Using Oracle Web Services Manager Security Policies", in *Securing WebLogic Web Services for Oracle WebLogic Server*.

The recommended security configuration for web services uses Security Assertion Markup Language (SAML) tokens to pass identities between web service clients and UMS. With SAML tokens, instead of the web service client passing a username and password to UMS, a trust relationship is established between the client and UMS because of exchanging certificates. Once this keystore configuration is in place, the web service client passes only the user identity, and vouches for the fact that it has authenticated the user appropriately.

The recommended policies to use for UMS web services are:

- `oracle/wss11_saml_token_with_message_protection_service_policy (server-side)`
- `oracle/wss11_saml_token_with_message_protection_client_policy (client-side)`
- `oracle/wss11_saml_token_identity_switch_with_message_protection_client_policy (client-side)`

> **Note:**
>
> The choice of client-side policy depends on the security context in which your application is executing.
>
> – If the thread that is making the web service call has the intended Subject associated with it (for example, from a web application that performs user authentication, or a Jakarta EE module with a *run-as* identity defined), then use the policy `oracle/wss11_saml_token_with_message_protection_client_policy`.
>
> The current thread Subject is passed through using the SAML Policy WS-Security headers. In this case you should not specify the parameter `javax.xml.ws.BindingProvider.USERNAME_PROPERTY` when creating your web service client instance.
>
> – If the thread that is making the web service call has an undefined Subject associated with it, or if you must programmatically supply a different identity, then use the policy `oracle/wss11_saml_token_identity_switch_with_message_protection_client_policy`, and specify the parameter `javax.xml.ws.BindingProvider.USERNAME_PROPERTY` when creating your web service client instance. If you want to perform dynamic identity switching, you must grant additional code permissions to your application. For more information, see *Administering Web Services*.

Web Service Security on Notification

The different web services include corresponding notification web services (`MessageNotification`) that run on the client side and receive notifications (message delivery status, message receipt, presence status change) when the appropriate event occurs.

Enabling UMS Web Service Security

To enable a policy for a UMS web service, see *Securing WebLogic Web Services for Oracle WebLogic Server*. You must select policy `oracle/wss11_saml_token_with_message_protection_service_policy`. This configuration must be repeated for each service that you want to secure.

Enabling Client Security

Web service client security must be enabled programmatically. When using the client libraries described in *Developing Applications with Oracle User Messaging Service*, WS-Security policy configuration is provided when a client object is constructed. The client constructors take an

argument of type `Map<String, Object>`. In general when using SAML authentication, the key/value pairs () should be added to the configuration map in addition to other required properties such as the endpoint address.

Table 4-22 Client Security Keys

Key	Typical Value
`oracle.ucs.messaging.ws.ClientConstants.POLICIES`	`oracle/wss11_saml_token_with_message_protection_client_policy`
`javax.xml.ws.BindingProvider.ENDPOINT_ADDRESS_PROPERTY`	Endpoint URL for the remote UMS WS. This is typically "http://<host>:<port>/ucs/messaging/webservice".
`javax.xml.ws.BindingProvider.USERNAME_PROPERTY`	(Optional) `<valid username>` Note: Do not specify this key while using `oracle/wss11_saml_token_with_message_protection_client_policy`.
`oracle.wsm.security.util.SecurityConstants.Config.KEYSTORE_RECIPIENT_ALIAS_PROPERTY`	(optional) keystore alias for target service. See Client Aliases.
`oracle.wsm.security.util.SecurityConstants.ClientConstants.WSS_CSF_KEY`	Used for OWSM policy attachment. Specifies a credential store key to use for looking up remote username/password information from the Oracle Web Services Management credential store map.

Example 4-4 Web Service Client Security

```
HashMap<String, Object> config = new HashMap<String, Object>();
config.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
 "http://example.com:8001/ucs/messaging/webservice");
config.put(oracle.ucs.messaging.ws.ClientConstants.POLICIES, new String[]
 {"oracle/wss11_saml_token_with_message_protection_client_policy"});

mClient = new MessagingClient(config);
```

Keystore Configuration

To use the recommended WS-Security policy, you must configure a keystore containing the public and private key information required by OWSM. Refer to "Configuring the Credential Store" in *Securing Web Services and Managing Policies with Oracle Web Services Manager* for information on how to configure the keystore and corresponding credential store entries.

- If both your web service client and UMS server are in the same domain, then they share a keystore and credential store.
- If your web service client and UMS server are in different domains, then you must import the UMS public key into your client domain's keystore, and must import your client domain's public key into the UMS keystore.

Client Aliases

When using certain WS-Security policies such as the SAML policy recommended here, the client must use the server's public key to encrypt the web service request. However, there is

generally only one keystore configured per domain. Therefore, if you have a domain in which there are web service clients that communicate with web services in multiple other domains, then you may be required to override the default keystore entry used by OWSM.

For example, if you have a domain in which application "A" is a web service client to a UMS web service, and application "B" is a web service client to a web service in another domain, then A's requests must be encrypted using the public key of the UMS domain, and B's requests must be encrypted using the public key of the other domain. You can accomplish this goal by overriding the keystore alias used by OWSM for each request:

- Import (for example) the UMS public key with alias "ums_public_key", and the other public key with alias "other_public_key".
- When creating an UMS Web Service client, specify the recipient keystore alias parameter, setting the key to `oracle.wsm.security.util.SecurityConstants.Config.KEYSTORE_RECIPIENT_ALIAS_PROPERTY` and the value to "ums_public_key" as shown in Example 4-5.
- The other web service client similarly must override the keystore alias, but the exact mechanism may differ. For example if using a JAX-WS client stub directly, then you can add the override property to the JAX-WS request context. See "Overriding the Policy Configuration for the Web Service Client" in *Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server* for more details.

Example 4-5 Client Aliases

```
HashMap<String, Object> config = new HashMap<String, Object>();
config.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY,
 "http://example.com:8001/ucs/messaging/webservice");
config.put(ClientConstants.POLICIES, new String[] {"oracle/wss11_saml_token_
identity_switch_with_message_protection_client_policy"});
config.put(BindingProvider.USERNAME_PROPERTY, "user1");
config.put(oracle.wsm.security.util.SecurityConstants.Config.CLIENT_CREDS_
LOCATION, oracle.wsm.security.util.SecurityConstants.Config.CLIENT_CREDS_LOC_
SUBJECT);
config.put(oracle.wsm.security.util.SecurityConstants.Config.KEYSTORE_RECIPIENT_
ALIAS_PROPERTY, "ums_public_key");
config.put(MessagingConstants.APPLICATION_NAME, "MyUMSWSApp");
mClient = new MessagingClient(config);
```

Securing JMS Resources

This (optional) procedure enables administrators to restrict access to the UMS' JMS resources (such as queues) for enhanced security.

To secure the JMS system resources, lock all JMS sub-deployments that start with the name *UMSJMSSystemResource* (there may be multiple automatically-created resources for UMS in a multi-server or cluster deployment) with the role *OracleSystemRole*. Do this using the WebLogic Remote Console, or you may run a WLST script (available at *MIDDLEWARE_HOME*`/oracle_common/communications/bin/secure_jms_system_resource.py`) as follows:

```
MIDDLEWARE_HOME/oracle_common/common/bin/wlst.sh
./secure_jms_system_resource.py
-userConfigFile=<UserConfigFile>, -userKeyFile=<UserKeyFile>
-url=<AdminServer_t3_url> -jmsSystemResource=<JMSSystemResourceName> -
role=<SecurityRoleToUse>
```

The `UserConfigFile` shall contain encrypted username and password for the AdminUser. The key for the encrypted data shall be in `UserKeyFile`.

By default, the UMS system runs as the user *OracleSystemUser* for accessing JMS resources. If the user *OracleSystemUser* does not exist, or you secure the UMS JMS resources with any other role that some other user has been granted, you must override the default user identity used by the UMS system by specifying an alternate username.

5

Monitoring Oracle User Messaging Service

This chapter describes how to monitor Oracle User Messaging Service (UMS) by using Oracle Enterprise Manager Fusion Middleware Control.

- Monitoring Oracle User Messaging Service
  You can monitor UMS logs and metrics using Oracle Enterprise Manager Fusion Middleware Control.
- Viewing Log Files
  You can view log files.
- Viewing Metrics and Statistics
  The performance of your applications is reflected in metrics and statistics.

Monitoring Oracle User Messaging Service

You can monitor UMS logs and metrics using Oracle Enterprise Manager Fusion Middleware Control.

To monitor UMS:

1. Log in to Oracle Enterprise Manager Fusion Middleware Control as an administrator.
2. Expand the **User Messaging Service** folder. You will see a User Messaging server, and a list of User Messaging drivers.
3. Select the server or driver of your choice.

   If you select a driver, quick statistics are displayed that indicate the state and performance of the driver.

   If you select a server, you see a list of associated drivers, in addition to the quick statistics. You can select one of the drivers to view its statistics, or you can click the Configure Driver icon to configure it. For more information on configuring drivers, see Configuring Oracle User Messaging Service.
4. You can perform a series of actions on the server. Right-click the server to select any of the actions. Table 5-1 lists the selection and their resultant actions.

Table 5-1 Server Selection and Actions

Selection	Action
Home	The home page lists the quick statistics for the selected driver
Control	Start Up or Shut Down driver
Logs	View and configure message logs for the selected driver
Performance Summary	Displays performance statistics on a customizable metrics page. Use this page to view statistics for this driver. Customize this page using the Metric Palette. The Metric Palette enables you to choose from all of the available metrics so that you see only the information that is most valuable to you

Table 5-1 (Cont.) Server Selection and Actions

Selection	Action
Message Status	Check the delivery status of messages sent and received, and resend selected messages. You can filter the search by adding more search fields and setting the desired operator and search value. Some fields can be added multiple times to use them with different and complementary operators, or with the *Contains* operator
Messaging Client Applications	Messaging client applications registered with UMS can be manually deregistered in cases where the applications have been undeployed and are holding onto access points that must be made available to other applications
Server Properties	Configure message storage method and business terms for message filter creation. For more information, see Configuring Oracle User Messaging Service.
System MBean Browser	System MBean Browser and its configuration settings
Target Information	Target Information displays the version, Middleware Home, Domain Home, Host and Deployed On details for the selected driver

- Using Message Status
- Deregistering Messaging Client Applications

Using Message Status

You can check the delivery status of messages sent and received, delete messages, and resend selected messages.

Checking message status

To check message status, perform the following tasks:

1. From the navigation tree, navigate to the server page. On the server page, select **Message Status** from the drop-down list that appears at the top of the page.

   The *Message Status* page appears.

2. Click **Search** to search the messages using the default criteria. The search returns a listing for the messages.

Customizing the Search

You can customize the search by adding more search fields and setting the desired operator and search value. Some fields can be added multiple times to use them with different and complementary operators, or with the Contains operator. To customize the search, perform the following tasks:

1. Click **Add Fields**.
2. Select the field(s) on which you want to search.
3. Choose operators and fill in variables as needed.
4. Click **Search**. The customized search is done and results returned.
5. If you want to resend a message, select the message in the list and click **Resend**.

Deleting Messages

You can delete selected messages or delete messages in bulk by setting the option for deleting all messages older than a specific date.

- To delete a selected message, select the message in the list and click **Delete Selected**.
- To delete all messages older than a specific date, click **Delete with Options**. In the pop-up window that appears, you must specify a date that is older than 7 days, and click **OK**. All messages before the specified date will be deleted.

> **Note:**
>
> If you choose to delete messages using the date feature in the EM UI, ensure that you do not have more than 2000 messages to be deleted at any given time. If there are more than 2000 messages to be deleted, you will see the following error message:
>
> ```
> The specified options result in the deletion of more than 2000 messages.
> Please narrow your query and try again.
> ```
>
> To delete more than 2000 messages, you must use the DB purge script for Oracle database.

Deregistering Messaging Client Applications

You can manually deregister Messaging Client Applications after the applications have been undeployed and are holding onto access points that must be made available to other applications. To deregister Messaging Client Applications, perform the following tasks:

1. Right-click a target in the navigation tree, and select **Messaging Client Applications**. The Messaging Client page appears.
2. Select the message to deregister and click **De-register**.

   A confirmation box appears asking you to confirm. Confirm your choice.

Viewing Log Files

You can view log files.

To view log files:

1. Right-click the driver (or server) for which you want to view log information, then choose **Logs** > **View Log Files**.

   The Log Messages page appears.

   Use this page to query for log information about a driver (or server). Fields and lists are used to customize the query.
2. After entering your search criteria, click **Log Files**. The Log Files page appears.
3. View log information or download the log.

- Configuring Logging

Configuring Logging

Use Oracle Enterprise Manager Fusion Middleware Control to configure log levels.

For each logger, set the notification level.

Viewing Metrics and Statistics

The performance of your applications is reflected in metrics and statistics.

To view metrics and statistics:

1. Select the Performance Summary for a driver (or server).

   The Performance Summary page appears.

   Many metrics are available for capture and display. To get the most valuable, focused information, use Metric Palette.
2. Click **Show Metric Palette** to display the Metric Palette.
3. Choose the metrics in which you are most interested. As you select or deselect metrics from the palette, the metrics display is automatically updated.

6

Managing Oracle User Messaging Service

This chapter describes how to manage Oracle User Messaging Service (UMS). It discusses how to deploy, undeploy and register UMS drivers by using Oracle Enterprise Manager Fusion Middleware Control using the configuration wizard. It also describes the procedure used to purge database records that are no longer need by the UMS DB schema.

- Deploying Drivers
  When you install Oracle UMS, preinstalled drivers are included (Email, XMPP, and SMPP). Among these drivers, only one or a few drivers are deployed to the WebLogic Server, depending on the template that is used when creating the domain.
- Using UMS Schema Purge Script
  `ums_cleanup.purge` is a PL/SQL procedure used to purge records that are no longer needed by the UMS DB schema. The procedure purges DB records from Oracle databases based on their age. It is highly recommend that the first two or three runs of the procedure be performed by a certified DBA.

Deploying Drivers

When you install Oracle UMS, preinstalled drivers are included (Email, XMPP, and SMPP). Among these drivers, only one or a few drivers are deployed to the WebLogic Server, depending on the template that is used when creating the domain.

You can deploy additional drivers by using the expandable server groups in the Fusion Middleware Configuration Wizard while updating your domain.

- Deploying Drivers Using the Fusion Middleware Configuration Wizard

Deploying Drivers Using the Fusion Middleware Configuration Wizard

Follow the instructions on this section to deploy drivers using the Configuration Wizard.

- Starting the Configuration Wizard
- Selecting a Configuration Type
- Updating an Existing Domain Using Product Templates
- Assigning User-Expandable Server Groups to Managed Servers
- Completing the Configuration

Task 1 Starting the Configuration Wizard
Start the Configuration wizard as described in "Starting the Configuration Wizard" in *Oracle Fusion Middleware Creating WebLogic Domains Using the Configuration Wizard*. The Configuration Type screen is displayed.

Task 2 Selecting a Configuration Type
On the Configuration Type screen, select **Update an Existing Domain**.

Select the domain directory from the **Domain Location** drop-down list, or click **Browse** to navigate to and select the domain directory. Click **Next** to continue. The Templates screen appears.

Task 3 Updating an Existing Domain Using Product Templates
On the Templates screen, select **Update Domain Using Product Templates** and then select the check box for JRF template to add to your domain.
Click **Next** and follow the configuration wizard screens till the Managed Servers screen appears, as described in "Updating WebLogic Domains" in *Oracle Fusion Middleware Creating WebLogic Domains Using the Configuration Wizard*.

Task 4 Assigning User-Expandable Server Groups to Managed Servers
On the Managed Servers screen, for each managed server, select the check box for the server group that corresponds to the driver that shall be targeted to that managed server. It is possible to select multiple drivers.
For more information about user-expandable server groups, see "Configuration Wizard Screens" in *Oracle Fusion Middleware Creating WebLogic Domains Using the Configuration Wizard*.

> **Note:**
>
> A new driver called GCM (Google Cloud Messaging) Driver is included in UMS in release 12.2.1. This driver is included as a preview feature in the release and, is not generally available.

Task 5 Completing the Configuration
Complete the configuration by following the configuration wizard screens described in "Updating WebLogic Domains" in *Oracle Fusion Middleware Creating WebLogic Domains Using the Configuration Wizard*.

Using UMS Schema Purge Script

`ums_cleanup.purge` is a PL/SQL procedure used to purge records that are no longer needed by the UMS DB schema. The procedure purges DB records from Oracle databases based on their age. It is highly recommend that the first two or three runs of the procedure be performed by a certified DBA.

> **WARNING:**
>
> This procedure will delete DB records from the UMS DB tables and commit instantly. It is impossible to rollback. The deleted records cannot be recovered. It is recommended to backup these tables before purging.

The purge reduces inbound and outbound message entries from UMS tables. Table 6-1 shows the tables in which records are purged:

Table 6-1 Records purged in UMS DB tables

Name of the Table	Records Purged
MESSAGE	Outbound/Inbound messages and their attributes

Table 6-1 (Cont.) Records purged in UMS DB tables

Name of the Table	Records Purged
ADDRESS	Sender and recipient addresses
DELIVERY_ATTEMPT	Records of deliveries
STATUS	Statuses of message deliveries
DELIVERY_CONTEXT	Records of deliveries

- Purging UMS DB Schema Records

Purging UMS DB Schema Records

`ums_cleanup.purge()` takes a single parameter, that is, **days_of_retention**.The value of this parameter is a positive integer or a float number that signifies the number of days. The procedure deletes any records that are older than the specified number of days from the UMS DB schema. For example, `ums_cleanup.purge(30.5)` will delete all records that are older than 30 days and 12 hours.

> **Note:**
>
> The value of **days_of_retention** must be greater than or equal to seven days.

To purge records in UMS DB schema, perform the following tasks:

1. Shut down all Mid-tier servers that are using the UMS schema.
2. Set SQL Plus Options as shown below:

```
set serveroutput on
set autocommit off
```

3. Invoke the procedure with the desired **days_of_retention** parameter from SQL Plus to purge records that are no longer needed. To delete entries older than 100 days, run the following command:

```
SQL> call ums_cleanup.purge(100);
```

> **Note:**
>
> If the UMS schema has not been cleaned up for a long time and there are many rows in the tables, it is recommended that you purge it in multiple small steps. For example, if the instance has been heavily used for six months and you want to keep records from the last 30 days, purge the schema gradually in small steps: 170, 160, …, 30.

4. Start all Mid-tier servers that were shut down during the purging process.

7

Troubleshooting Oracle User Messaging Service

To debug Oracle User Messaging Service (UMS), first check the server diagnostic logs. The logs may contain exception, error, or warning messages that provide details about incorrect behavior along with actions to remedy the problem. Table 7-1 describes additional methods for debugging common UMS problems.

Table 7-1 Troubleshooting UMS

Symptom	Possible Causes	Solutions
SSL handshake error.	The default keystore (`DemoTrust.jks`) for WebLogic Server may cause this error.	Change the default keystore for WebLogic Server. Configure the **Custom Identity and Java Standard Trust** keystore for the WebLogic Server. For more information, see Oracle WebLogic Remote Console Online Help.
Email notification is not being sent.	The Outgoing (SMTP) Mail Server settings in the UMS Email Driver are incorrect.	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: • `OutgoingMailServer` • `OutgoingMailServerPort` **Note**: Validate the values by using them in any email client for connecting to the SMTP server.
	The SMTP server requires authentication or a secure connection (TLS or SSL).	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: • `OutgoingUsername` • `OutgoingPassword` • `OutgoingMailServerSecurity`

Table 7-1 (Cont.) Troubleshooting UMS

Symptom	Possible Causes	Solutions

Notifications are not being sent because of error message: `No matching drivers found for sender address = <address>`	The UMS Driver for the appropriate channel is configured with a specific list of *SenderAddresses*, and the message sent by the application has set a non-matching Sender Address.
	Note: UMS Server matches the outbound message's sender address, if set, against the available drivers' *SenderAddresses* to find a matching driver to use for delivering the message. If a driver has set one or more SenderAddresses, then the UMS Server only sends messages with the matching sender address to it.
		• Check the following settings in the appropriate UMS Driver using Oracle Fusion Middleware Control:
		`SenderAddresses`
		Note: The format for SenderAddresses is a comma-delimited list of `<DeliveryType>:<Address>`.
		For example:
		`EMAIL:sender@example.com, EMAIL:sender@example2.com`
		• Leave this property blank, if you want this driver to service outbound messages for all sender addresses for this channel (delivery type).
		• If there are multiple driver instances deployed for the same channel (delivery type) with different configurations, use the *SenderAddresses* to differentiate the driver instances. For example, one instance can be set with a value in SenderAddresses to only service outbound messages with that matching sender address, while the other instance can keep the SenderAddresses blank to service all outbound messages that do not specify any sender address or one that does not match that of the first driver instance.
		• SenderAddresses that are configured with the incorrect syntax (such as missing `<DeliveryType>:`) are ignored by the UMS Server for driver selection.

The email client inconsistently receives notifications.	The Incoming Mail Server settings in the UMS Email Driver are configured with the same email account to which notifications are being sent. If the notification is sent to the same account, the UMS Email Driver may download and process the email before the email client can display it.	Use an exclusive email account for Incoming Mail Server settings. Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: • `IncomingMailIDs` • `IncomingUserIDs`

Table 7-1 (Cont.) Troubleshooting UMS

Symptom	Possible Causes	Solutions
The application does not receive emails.	The Incoming Mail Server settings in the UMS Email Driver are incorrect.	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: • `MailAccessProtocol` (*IMAP* or *POP3*, in uppercase) • `ReceiveFolder` • `IncomingMailServer` • `IncomingMailServerPort` • `IncomingMailServerSSL` • `IncomingMailServerSSL` • `IncomingUserIDs` • `IncomingUserPasswords` • `ImapAuthPlainDisable` **Note**: Validate the values by using them in any email client for connecting to an IMAP or POP3 server.
	The mail access protocol is incorrect.	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: • `MailAccessProtocol` (*IMAP* or *POP3*, in uppercase)
	The email server is SSL-enabled.	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: • `IncomingMailServerSS`
	The receive folder name is incorrect.	Check the following settings in the UMS Email Driver using Oracle Fusion Middleware Control: • `ReceiveFolder` **Note**: Some email servers may expect the value INBOX to be inbox or Inbox (that is, case-sensitive). Based on your email server, use an appropriate value.
	The application did not register the corresponding AccessPoint.	Register an AccessPoint using the UMS API. For more information, see *Oracle Fusion Middleware Developing Applications with Oracle User Messaging Service*.
The connection succeeds but gets closed immediately since there is a mismatch in the subject	The certificate used is incorrect for the environment.	Each environment (dev, test, production, etc.) needs to be provisioned with its own certificate. Verify that the correct certification is used for the remote environment.
APNS notifications are not delivered	A device token retrieved for production environment is likely being used for non-production.	Use the correct device token retrieved for each environment. For more information, see `https://developer.apple.com/library/ios/technotes/tn2265/_index.html`

Table 7-1 (Cont.) Troubleshooting UMS

Symptom	Possible Causes	Solutions
PKIX path building is failing and is unable to find a valid certification path to the requested target for the APNS driver in Secure mode environment	The relevant APNS certificates are not imported.	To setup a remote notification server, you must establish a trusted connection to APNS. To create this connection, you must install the AAA Certificate Services root certificate on each of your servers. Download the certificate from Setting up a remote notification server. You must import the downloaded certificate to the WebLogic's configured keystore by using the following command: `keytool -importcert -keystore "../fmwhome12/wlserver/server/lib/DemoTrust.jks" -storepass DemoTrustKeyStorePassPhrase -file AAACertificateServices.crt -alias "AAACertificateServices.crt"` **Note:** Select the keystore configured in the environment.
GCM workflow throws an error while sending messages in secure mode environment after successful test connection.	Missing trusted certificates The hostnameverifier configuration in WebLogic is not accepting wild characters.	You must import the global root certificate with keytool to the custom cert store configured in WebLogic. Global sign root certificates can be downloaded at GlobalSign Root Certificates.
Runtime throws an error after configuring GCM driver.	Default "BEA Hostname verifier" does not support wildcards.	You must verify the hostname for the Managed Server. The following setting works for the hostname verification: 1. None. 2. Wildcard Hostname Verifier. The following setting does not work for the hostname verification: 1. BEA Hostname verifier

A

Configuring User Messaging Service with AQ JMS

This appendix describes how to configure UMS to use AQ JMS instead of WLS JMS. This can be achieved through the WebLogic Remote Console.
User Messaging Service (UMS) can be configured to use Oracle Streams Advanced Queuing (AQ) JMS. AQ JMS uses a database connection and stores JMS messages in a database that is accessible to an entire WebLogic Server cluster, thus enabling the use of database features and tooling for data manipulation and backup. A typical use case would be enhanced high availability (HA), where the standard whole server migration support is not sufficient. If one UMS node in a cluster fails, then the other nodes will pick messages from the database for the failing UMS node, causing no loss of messages.

The following tasks describe how to configure UMS with AQ JMS through the WebLogic Remote Console.

1. Log into Oracle WebLogic Remote Console.

   Shutdown all Managed servers in the domain.
2. Click **Edit Tree** option in the left panel. Expand **Environments** and select **Domains** for configuration changes.
3. Expand the **Services** node and navigate to JMS system resource. A page listing the JMS system modules created for this domain is displayed.
4. Select **UMSJMSSystemResource** and navigate to the **Subdeployments** tab. Navigate to the UMSJMSServer running on the cluster you want to reconfigure and un-target all the *UMSJMSServer_auto_x* servers. Click **Save**.
5. Navigate to the **Targets** tab for UMSJMSSystemResource. Un-target the UMSJMSSystemResource cluster(s) by deselecting the cluster check box. Click **Save**.
6. Navigate to **UMSAQJMSSystemResource** from the JMS Modules table.
7. Navigate to the **Targets** tab for UMSAQJMSSystemResource and target the UMSAQJMSSystemResource to the same cluster from which you un-targeted the UMSJMSSYstemResource.

   Save the settings.
8. Click **shopping cart** on top right corner and select **commit changes**. Start all Managed servers in the domain.