Oracle® Cloud
Using Oracle Java Cloud Service - SaaS Extension

Release 19.2.2
E41172-50
May 2019
Contents

Preface

Audience ix
Documentation Accessibility ix
Related Resources ix
Conventions x

1 Getting Started with Oracle Java Cloud Service - SaaS Extension

About Oracle Java Cloud Service - SaaS Extension 1-1
Understanding the Oracle Java Cloud Service - SaaS Extension Architecture 1-2
Understanding the PaaS Infrastructure and Java Environment 1-3
About Supported Java EE, Oracle WebLogic Server, and Oracle ADF Applications 1-3
About Supported Interfaces to Oracle Java Cloud Service - SaaS Extension 1-4
About the Oracle Java Cloud Service - SaaS Extension SDK 1-4
About Using Integrated Development Environments 1-5
Using Oracle JDeveloper with Oracle Java Cloud Service - SaaS Extension 1-6
Using NetBeans with Oracle Java Cloud Service - SaaS Extension 1-6
Using Oracle Enterprise Pack for Eclipse with Oracle Java Cloud Service - SaaS Extension 1-6
About Managing Application Security 1-7
Default User Authentication 1-7
Securing Web Services 1-7
About Third-Party Framework Support 1-8
Considerations When Developing Applications on Oracle Java Cloud Service - SaaS Extension 1-8
About Underlying Oracle Technologies 1-8
About Supported Applications, Standards, and APIs 1-9
Using Third-Party Frameworks with Oracle Java Cloud Service - SaaS Extension 1-11
Supported Third-Party Frameworks for Oracle Java Cloud Service - SaaS Extension 1-12
Omitting Checks for Updates to Quartz Job Scheduler 1-13
Using Non-Listed Frameworks 1-13
About the Application Deployment Validation Process and Run-time Security 1-14
Application and Library Deployment Validation Flow 1-14
Oracle Java Cloud Service - SaaS Extension Whitelist Validation 1-15
Prerequisites for Using Oracle Java Cloud Service - SaaS Extension 1-15
Sizing and Deployment Recommendations 1-16
About Oracle Java Cloud Service - SaaS Extension Roles and Users 1-16
Getting Started with Your JCS-SaaS Extension Subscription 1-18
  Accessing Your Service for the First Time 1-18
  Provisioning a Database Instance 1-19
  Provisioning a JCS-SaaS Extension Instance 1-20
Accessing Oracle Java Cloud Service - SaaS Extension 1-22
  Accessing Oracle Java Cloud Service - SaaS Extension Control from a URL 1-22
  Accessing JCS-SaaS Extension Administration Console from the My Services Dashboard 1-23
  Accessing the JCS-SaaS Extension Administration Console from the Service Details Page 1-24
Using the Welcome App 1-25

2 Developing Applications for Oracle Java Cloud Service - SaaS Extension

Typical Workflow for Using Oracle Java Cloud Service - SaaS Extension 2-1
Preparing Applications for Oracle Java Cloud Service - SaaS Extension Deployment 2-4
  Understanding Application Library Behavior Changes on Oracle Cloud 2-4
  Guidelines for Applications That Use a JDBC Data Source 2-4
  Using a JNDI Alias for a JDBC Data Source 2-5
Guidelines for ADF Applications 2-6
Guidelines for Applications That Use Java EE or ADF Application Security 2-7
  Required Changes to ADF Applications Using Role-based Security 2-7
  Required Changes to Java EE Applications Using Role-based Security 2-8
Guidelines for Applications When Accessing System Properties 2-9
Guidelines for Applications When Using Log4j Appenders 2-9
Guidelines for Applications When Accessing a Local File System 2-10
Accessing Applications Deployed on Oracle Java Cloud Service - SaaS Extension 2-11
Messaging Support in Oracle Java Cloud Service - SaaS Extension 2-12
Using JMS in Oracle Java Cloud Service - SaaS Extension 2-12
Developing RESTful Web Services 2-12
  Using the Jersey JAX-RS Reference Implementation 2-13
    Summary of the Jersey JAX-RS RI Shared Library 2-13
    Using the Jersey JAX-RS RI Shared Library 2-13
    Configuring the Web Application to Use the Jersey JAX-RS RI 2-14
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating JAX-RS Web Services and Clients</td>
<td>2-16</td>
</tr>
<tr>
<td>Securing Applications in Oracle Java Cloud Service - SaaS Extension</td>
<td>2-17</td>
</tr>
<tr>
<td>Securing Java EE and ADF Applications – Authentication</td>
<td>2-18</td>
</tr>
<tr>
<td>Internet Public Pages</td>
<td>2-18</td>
</tr>
<tr>
<td>Oracle Public Pages</td>
<td>2-18</td>
</tr>
<tr>
<td>Tenant Restricted Pages</td>
<td>2-19</td>
</tr>
<tr>
<td>Securing JAX-WS Web Services</td>
<td>2-20</td>
</tr>
<tr>
<td>Securing Java EE Applications – Roles and Constraints</td>
<td>2-22</td>
</tr>
<tr>
<td>Updating the web.xml Deployment Descriptor</td>
<td>2-22</td>
</tr>
<tr>
<td>Updating the weblogic.xml Deployment Descriptor</td>
<td>2-24</td>
</tr>
<tr>
<td>Special Considerations When Accessing Secured Oracle Cloud Pages</td>
<td>2-24</td>
</tr>
<tr>
<td>Securing ADF Applications – Roles and Constraints</td>
<td>2-27</td>
</tr>
<tr>
<td>Updating the jazn-data.xml File</td>
<td>2-27</td>
</tr>
<tr>
<td>Configuring JPS Policy Migration Settings</td>
<td>2-27</td>
</tr>
<tr>
<td>Creating an On-premises WebLogic Server Environment</td>
<td>2-28</td>
</tr>
</tbody>
</table>

### 3 PaaS-SaaS Association

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prerequisites and Restrictions for Association Between Services</td>
<td>3-1</td>
</tr>
<tr>
<td>The Benefits of Association</td>
<td>3-2</td>
</tr>
<tr>
<td>Understanding Identity Propagation</td>
<td>3-3</td>
</tr>
<tr>
<td>Identity Propagation with SAML</td>
<td>3-4</td>
</tr>
<tr>
<td>Identity Propagation with OAuth</td>
<td>3-6</td>
</tr>
<tr>
<td>Identity Propagation Use Cases</td>
<td>3-6</td>
</tr>
<tr>
<td>Writing a Client That Can Access an Oracle Sales Cloud Application</td>
<td>3-6</td>
</tr>
<tr>
<td>Writing a Web Service that an Oracle Sales Cloud Application Can Access</td>
<td>3-8</td>
</tr>
<tr>
<td>Propagating ID with OAuth</td>
<td>3-8</td>
</tr>
<tr>
<td>Verify the Client Configuration</td>
<td>3-10</td>
</tr>
<tr>
<td>Creating a Report of Oracle Sales Cloud User Accounts</td>
<td>3-10</td>
</tr>
<tr>
<td>PaaS-SaaS Association Sample Applications</td>
<td>3-17</td>
</tr>
</tbody>
</table>

### 4 Setting Up Trust Between WebLogic Domains and JCS-SaaS Extension

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>About the setup-wss-trust Tool</td>
<td>4-1</td>
</tr>
<tr>
<td>Guidelines for Using setup-wss-trust</td>
<td>4-3</td>
</tr>
<tr>
<td>Getting More Information</td>
<td>4-4</td>
</tr>
</tbody>
</table>

### 5 Managing Instances

<table>
<thead>
<tr>
<th>Subsection</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relocating a Service to a Different Identity Domain</td>
<td>5-1</td>
</tr>
</tbody>
</table>

ORACLE
Relocating the Service Instance 5-1
Completing Post-relocation Tasks 5-3
Upgrading an FMW 11.1.1.7 Instance to–and Downgrading it from–FMW 11.1.1.9 5-4
Upgrading an Instance from FMW 11.1.1.7 to FMW 11.1.1.9 5-5
Downgrading an Upgraded Instance 5-10

6 Administering Instances with JCS-SaaS Extension Control
Understanding Oracle Java Cloud Service - SaaS Extension Control 6-2
  Understanding the Regions of the Oracle Java Cloud Service - SaaS Extension Control Home Page 6-3
Restarting a Java Service Instance from JCS-SaaS Extension Control 6-5
Managing Applications with JCS-SaaS Extension Control 6-7
  Deploying an Application with JCS-SaaS Extension Control 6-8
  Deleting an Application with JCS-SaaS Extension Control 6-9
  Redeploying an Application with JCS-SaaS Extension Control 6-9
  Starting and Stopping Applications with JCS-SaaS Extension Control 6-9
Managing Shared Libraries with JCS-SaaS Extension Control 6-9
  About Shared Java EE Libraries and Optional Packages 6-10
  Creating Shared Java EE Libraries and Optional Packages 6-10
  Deploying, Redeploying, and Deleting Libraries 6-11
    Deploying a Library 6-11
    Redeploying a Library 6-11
    Deleting a Library 6-11
Viewing Application-Specific Statistics on JCS-SaaS Extension Control 6-12
Viewing the Activity Logs on JCs-SaaS Extension Control 6-12
Viewing Service and Application Logs on JCS-SaaS Extension Control 6-13

7 Administering Instances with the JCS-SaaS Extension Administration Console
Administration Console Keyboard Shortcuts 7-2
Working with the Java Cloud Service - SaaS Extension Administration Console 7-3
  Viewing Basic Service Information on the Home Page 7-5
  Viewing and Managing Application Information on the Applications Page 7-7
  Viewing and Managing Shared Library Information on the Libraries Page 7-9
  Managing System Properties on the System Properties Page 7-11
  Managing Data Source Alias on the Data Source Alias Page 7-12
  Managing Certificates on the SSL Certificates Page 7-13
  Managing Certificates on the WSS Certificates Page 7-16
  Managing Credentials on the Credential Stores Page 7-18
  Viewing Metrics on the Metrics Page 7-20
Preface

Using Oracle Java Cloud Service — SaaS Extension explains how to develop, monitor, and manage applications using Oracle Java Cloud Service - SaaS Extension.

Topics:
- Audience
- Document Accessibility
- Related Resources
- Conventions

Audience

This document is intended for administrators or application developers who are using Oracle Java Cloud Service - SaaS Extension. This guide assumes you are familiar with web technologies and have a general understanding of Java development environments.

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.

Related Resources

For more information, see these Oracle resources:
- Oracle Public Cloud
  http://cloud.oracle.com
- About Oracle Cloud
- About Oracle Database Cloud - Database Schema Service
- About Oracle Storage Cloud Service
• About Oracle Messaging Cloud Service

Conventions

The following text conventions are used in this document:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>boldface</strong></td>
<td>Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.</td>
</tr>
<tr>
<td><em>italic</em></td>
<td>Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.</td>
</tr>
<tr>
<td>monospace</td>
<td>Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.</td>
</tr>
</tbody>
</table>
Getting Started with Oracle Java Cloud Service - SaaS Extension

Using Oracle Java Cloud Service - SaaS Extension provides documentation on using Oracle Java Cloud Service - SaaS Extension for Oracle Cloud developers and administrators. This section provides information to help you get started with this product.

Topics:
• About Oracle Java Cloud Service - SaaS Extension
• Considerations When Developing Applications on Oracle Java Cloud Service - SaaS Extension
• Prerequisites for Using Oracle Java Cloud Service - SaaS Extension
• Sizing and Deployment Recommendations
• About Oracle Java Cloud Service - SaaS Extension Roles and Users
• Getting Started with Your JCS-SaaS Extension Subscription
• Accessing Oracle Java Cloud Service - SaaS Extension
• Using the Welcome App

See Oracle Cloud Terminology in Getting Started with Oracle Cloud for definitions of terms found in this and other documents in the Oracle Cloud library.

About Oracle Java Cloud Service - SaaS Extension

Oracle Java Cloud Service - SaaS Extension reduces the complexity associated with the deployment and maintenance of enterprise Java applications. It enables you to create Oracle Java Cloud Service - SaaS Extension instances quickly. You can deploy your applications to a service instance, then secure and manage them without worrying about the underlying infrastructure.

Topics
• Understanding the Oracle Java Cloud Service - SaaS Extension Architecture
• Understanding the PaaS Infrastructure and Java Environment
• About Supported Java EE_ Oracle WebLogic Server_ and Oracle ADF Applications
• About Supported Interfaces to Oracle Java Cloud Service - SaaS Extension
• About the Oracle Java Cloud Service - SaaS Extension SDK
• About Using Integrated Development Environments
• About Managing Application Security
• About Third-Party Framework Support
Understanding the Oracle Java Cloud Service - SaaS Extension Architecture

Oracle Java Cloud Service - SaaS Extension provides a platform to develop and deploy business applications in the cloud. With Oracle Java Cloud Service - SaaS Extension, businesses can maximize productivity with instant access to a cloud environment powered by Oracle WebLogic Server, complete with integrated security and database access.

The following graphic illustrates the relationship of Oracle Java Cloud Service - SaaS Extension with Oracle Cloud. The Oracle Cloud hosted at the Oracle Cloud data center includes the Cloud Portal (My Services and My Account), an Oracle Java Cloud Service - SaaS Extension instance, an Identity Domain, and a Database Service.

As shown in the previous illustration, Oracle Java Cloud Service - SaaS Extension instances are hosted within the Oracle Cloud in a data center operated by Oracle. This architecture provides a great deal of power and flexibility. Advantages include:

- Quick provisioning of new Oracle Java Cloud Service - SaaS Extension instances in a self-service fashion on the cloud.oracle.com page. See Requesting a Trial Subscription to an Oracle Cloud Service in Getting Started with Oracle Cloud.
- The ability to choose the service's capacity, throughput, and high availability at provisioning time.
- A fully-managed and operated environment from Oracle with high availability, scalability, and built-in disaster recovery.
- Allows for the portability of applications between Oracle Cloud and on-premise environments.
Understanding the PaaS Infrastructure and Java Environment

As a Platform as a Service (PaaS) solution, the focus of Oracle Java Cloud Service - SaaS Extension is to automate the back-end infrastructure (that is, the operating system, virtual machine, Java EE container, and Oracle Java Cloud Service - SaaS Extension settings), as well as the provisioning and configuration process. The infrastructure of the Oracle Java Cloud Service - SaaS Extension runtime is not directly exposed to its service users. In other words, Oracle Java Cloud Service - SaaS Extension is not an Infrastructure as a Service (IaaS) solution. Despite this limitation, certain aspects of the infrastructure can be managed through the My Services interface of the Oracle Java Cloud Service - SaaS Extension as follows:

- Type of Oracle Java Cloud Service - SaaS Extension instance (that is, basic, standard, enterprise). The type of Oracle Java Cloud Service - SaaS Extension determines the number of Java EE server processes, memory storage, and file system capacity for the service instance.
- Identity domain to which the Oracle Java Cloud Service - SaaS Extension belongs. The identity domain determines the identity store and single-sign-on realm of the instance.
- The association of an Oracle Java Cloud Service - SaaS Extension instance with a Database Cloud Service instance. This association makes the database instance available to deployed applications as a JDBC data source.

**Note:**

When you request an Oracle Java Cloud Service - SaaS Extension trial, Oracle automatically includes a Database Cloud Service trial because Java requires Oracle Database to function. You receive two trials in a single request: one Java and one database.

**Note:** You upload and manage data for Database Cloud Service instance using the Oracle Cloud Data Loading utility, the Oracle Application Express Data Load utility, or a SQL script in SQL Workshop. See Developing Applications for the Database Cloud Service in *Using Oracle Database Cloud Service.*

About Supported Java EE, Oracle WebLogic Server, and Oracle ADF Applications

Use Oracle Java Cloud Service - SaaS Extension to instantly create Java EE environments within the Oracle Cloud and deploy your applications to them. You can create these kinds of environments:

- Standard Java EE WAR (Web Application Archive) or EAR (Enterprise Archive) formats.
- Applications that make use of Oracle WebLogic Server-specific extensions in release 10.3.6. See:
  - Other Supported Public WebLogic Server 10.3.6 APIs and Capabilities
About Supported Interfaces to Oracle Java Cloud Service - SaaS Extension

Five different interfaces to Oracle Java Cloud Service - SaaS Extension will assist you in developing, deploying, and managing applications.

<table>
<thead>
<tr>
<th>Interface</th>
<th>Description</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Java Cloud Service - SaaS Extension Control</td>
<td>A web-based management console that enables you to deploy and monitor your hosted applications.</td>
<td>Using the Oracle Java Cloud Service - SaaS Extension Control</td>
</tr>
<tr>
<td>Oracle Java Cloud Service - SaaS Extension SDK</td>
<td>Provides utilities that facilitate the management of Oracle Java Cloud Service - SaaS Extension instances and the development of applications for the Oracle Java Cloud Service - SaaS Extension.</td>
<td>About the Oracle Java Cloud Service - SaaS Extension SDK</td>
</tr>
<tr>
<td>Oracle Java Cloud Service - SaaS Extension interface in Oracle JDeveloper IDE</td>
<td>Provides tooling so developers can directly interact with target service instances as part of the development process.</td>
<td>Using Oracle JDeveloper with Oracle Java Cloud Service - SaaS Extension</td>
</tr>
<tr>
<td>Oracle Java Cloud Service - SaaS Extension interface in Oracle Enterprise Platform for Eclipse IDE</td>
<td>Provides tooling so developers can directly interact with target service instances as part of the development process.</td>
<td>Using Oracle Enterprise Pack for Eclipse with Oracle Java Cloud Service - SaaS Extension</td>
</tr>
<tr>
<td>Oracle Java Cloud Service - SaaS Extension interface in NetBeans IDE</td>
<td>Provides tooling so developers can directly interact with target service instances as part of the development process.</td>
<td>Using NetBeans with Oracle Java Cloud Service - SaaS Extension</td>
</tr>
</tbody>
</table>

About the Oracle Java Cloud Service - SaaS Extension SDK

The Oracle Java Cloud Service - SaaS Extension SDK (software development kit) is a downloadable package that provides tools that facilitate the management of Oracle Java Cloud Service - SaaS Extension instances and the development of applications for the Oracle Java Cloud Service - SaaS Extension in Oracle Cloud. These same tools can also be used in your development environment against a local WebLogic Server domain.
The Oracle Java Cloud Service - SaaS Extension SDK is required if you want to integrate your service instance with one of the supported IDEs described in About Using Integrated Development Environments.

The Oracle Java Cloud Service - SaaS Extension SDK contains:

- Command-line interfaces (CLI):
  - javacloud.jar – general application management tasks
  - File System Access Shell – local file system management
  - Configuration Shell – application and domain configuration
- Apache Ant tasks
- Apache Maven plug-in
- Whitelist validation
- Documentation
  
  **Note:** See the index.html file under the /doc directory for all SDK usage instructions.

- Sample applications
  
  **Note:** See the sample.html file under the SDK_HOME/doc directory for all sample installation and usage instructions.

You can download the Oracle Java Cloud Service - SaaS Extension SDK and use its CLI-based utilities from Oracle. See:

- Downloading the Oracle Java Cloud Service - SaaS Extension SDK
- Using the Command-Line Interface to Monitor Oracle Java Cloud Service - SaaS Extension
- Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension

About Using Integrated Development Environments

The Oracle Java Cloud Service - SaaS Extension provides tooling within the Oracle JDeveloper, Oracle Enterprise Platform for Eclipse, and NetBeans IDEs that enables developers to directly interact with target service instances as part of the development process.

**Topics:**

- Using Oracle JDeveloper with Oracle Java Cloud Service - SaaS Extension
- Using NetBeans with Oracle Java Cloud Service - SaaS Extension
- Using Oracle Enterprise Pack for Eclipse with Oracle Java Cloud Service - SaaS Extension

Use the Oracle Cloud **Resources** menu to access additional tools that enable you to directly interact with your Oracle Java Cloud Service - SaaS Extension instance.

You can download these tools and the Oracle Java Cloud Service - SaaS Extension SDK from: [http://www.oracle.com/technetwork/topics/cloud/downloads/](http://www.oracle.com/technetwork/topics/cloud/downloads/)
Using Oracle JDeveloper with Oracle Java Cloud Service - SaaS Extension

Oracle JDeveloper is a free integrated development environment that simplifies the development of Java-based SOA and Java EE applications. JDeveloper offers complete end-to-end development for Oracle Fusion Middleware and Oracle Fusion Applications with support for the full development life cycle.

**Supported Versions:**


**Documentation:**


Using NetBeans with Oracle Java Cloud Service - SaaS Extension

NetBeans is a free, open-source Integrated Development Environment (IDE) for software developers. All the tools needed to create professional desktop, enterprise, web, and mobile applications with the Java platform, as well as with C/C++, PHP, JavaScript and Groovy.

**Supported Versions:**


**Documentation:**

The official NetBeans documentation contains information on using the IDE's Oracle Java Cloud Service - SaaS Extension integration capabilities. See [http://netbeans.org/kb/docs/web/oracle-cloud.html](http://netbeans.org/kb/docs/web/oracle-cloud.html).

Using Oracle Enterprise Pack for Eclipse with Oracle Java Cloud Service - SaaS Extension

Oracle Enterprise Pack for Eclipse (OEPE) provides tools that make it easier to develop applications using specific Oracle Fusion Middleware technologies and Oracle Database. For Oracle Cloud, OEPE provides direct deployment to Oracle Java Cloud Service - SaaS Extension, integrated whitelist scanning to check for errors before deployment, integration into the Oracle Java Cloud Service - SaaS Extension Control, and log viewers to check on the status of the application.

**Supported Versions:**


**Documentation:**

About Managing Application Security

Special instructions are required for managing the security of the Java EE and ADF applications that have been deployed to an Oracle Java Cloud Service - SaaS Extension instance.

Topics

• Default User Authentication
• Securing Web Services

Default User Authentication

All Java EE and ADF web applications deployed to an Oracle Java Cloud Service - SaaS Extension instance are automatically secured. When users access an application deployed on Oracle Cloud the default authentication mechanism requests their user ID, password, and the name of the identity domain.

Once logged in, users are authenticated for applications. By default (that is, if no specific configurations are defined), only users who have been authenticated through Single Sign-On (SSO) can access a deployed application, but this includes users from any identity domain.

To provide finer-grained secure access to your Java EE or ADF applications, you can specify role-based authentication that can vary from being publicly accessible to restricted to only users within the same identity domain. See Securing Applications in Oracle Java Cloud Service - SaaS Extension and Managing Users and Roles in Getting Started with Oracle Cloud.

Securing Web Services

Applications deployed on Oracle Java Cloud Service - SaaS Extension can invoke externally exposed web services (Outbound WS Client), as well as host web services (Inbound Web Service), that can be either non-secured or secured (for example, using WS-Security).

For guidelines on building secure JAX-WS web services, see Securing JAX-WS Web Services.

For more information on supported OWSM policies, see Predefined Policies in Oracle Fusion Middleware Security and Administrator's Guide for Web Services.

To use OWSM policies, you must attach them at design time:

• For plain Java EE JAX-WS web services clients, see Policy Configuration Overrides for the Web Service Client in Oracle Fusion Middleware Securing WebLogic Web Services for Oracle WebLogic Server.

• For ADF web services clients, see How to Attach Oracle WSM Policies to Web Service Clients in Oracle Fusion Middleware User's Guide for Oracle JDeveloper.
About Third-Party Framework Support

Oracle makes no specific claims about a definite list of third-party libraries that should work within an Oracle Java Cloud Service - SaaS Extension environment.

In general, an application’s use of most third-party frameworks should work within Oracle Java Cloud Service - SaaS Extension, so long as:

- All dependencies can be embedded within the deployment archives.
- All third-party JARS and their dependencies pass the Oracle Java Cloud Service - SaaS Extension whitelist. See Oracle Java Cloud Service - SaaS Extension Whitelist Validation.

See Supported Third Party Frameworks for Oracle Java Cloud Service - SaaS Extension for a list of supported third-party frameworks.

Considerations When Developing Applications on Oracle Java Cloud Service - SaaS Extension

When developing applications to deploy to an Oracle Java Cloud Service - SaaS Extension instance, you need to consider the supported application standards and APIs available to ensure successful deployment.

Topics

- About Underlying Oracle Technologies
- About Supported Applications_ Standards_ and APIs
- Using Third-Party Frameworks with Oracle Java Cloud Service - SaaS Extension
- About the Application Deployment Validation Process and Run-time Security

About Underlying Oracle Technologies

Think of each Oracle Java Cloud Service - SaaS Extension instance as a deployment target for applications using a set of Java EE release 5, Java EE release 6, and Oracle WebLogic Server capabilities.

Oracle Java Cloud Service - SaaS Extension is built on the following Oracle technologies:

- Oracle WebLogic Server (WebLogic Server) release 10.3.6
- Oracle Application Development Framework (ADF) release 11.1.1.9.0 (11.1.1.7.1 for Service instances created before February 3, 2017)

Note:

All references in this document to WebLogic Server capabilities and ADF specific capabilities refer only to the releases specified in the previous list.
About Supported Applications, Standards, and APIs

Oracle Java Cloud Service - SaaS Extension supports the deployment of many types of applications and technology standards:

These applications and standards are:

**WAR or EAR Deployment**

Web Application Archive (WAR) or Enterprise Archive (EAR) deployment. All supported applications must be deployed through a WAR file or an EAR file (which can contain multiple WAR or JAR files).

**Tip:**
The class loader behavior of EAR archives when deployed to an Oracle Java Cloud Service - SaaS Extension is the same as that of WebLogic Server. See Understanding WebLogic Server Application Classloading in Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server.

**ADF Applications**

Oracle Application Development Framework (ADF) 11.1.1.9.0 applications are supported.

**Web Applications**

- Applications using Servlet 2.5, JavaServer Pages (JSP) 2.1, Java Server Faces (JSF) release 1.2 and release 2.0.
- Use of `web.xml` and `weblogic.xml` deployment descriptors, and related annotations are supported.

**Web Services Applications**

- Applications using Java API for XML Web Services (JAX-WS) 2.1 based web services. Use of `webservices.xml`, `weblogic-webservices.xml` deployment descriptors and related annotations is supported.
- Applications providing REST-based APIs through Java API for RESTful web services (JAX-RS) 1.1 and Jersey 1.9 annotations are supported.

**Enterprise Java Beans (EJB) Containers**

Applications using EJB 2.1 and EJB 3.0 specifications. Use of `ejb-jar.xml`, `weblogic-ejb-jar.xml`, and related annotations are supported with the following exceptions:

- Only local EJB invocations are supported, specifically, the client code invoking an EJB application's interface must be either within the same deployment archive as the EJB implementation code itself or within a deployment archive that is deployed to the same Oracle Java Cloud Service - SaaS Extension instance.
- EJB 2.x Entity Beans are not supported.
JDBC Services

- Applications using Java Persistence API (JPA) 2.0 specifications and use of JPA persistence.xml elements with EclipseLink 2.1.3 specific extensions.
- Direct use of Java Database Connectivity (JDBC) 4.0 APIs. See WebLogic Server 12.1.1 Compatibility with Previous Releases in Oracle Fusion Middleware Upgrade Guide for Oracle WebLogic Server.
- Use of Oracle Database 11g compatible SQL statements.
- JDBC Data Sources provisioned within an Oracle Java Cloud Service - SaaS Extension instance upon association with a Database Cloud Service instance will be XA-enabled JDBC data sources.

Java Platform, Standard Edition (SE) 1.6 or 1.7 APIs

Applications can use the set of Java SE 1.6 or 1.7 public APIs, as long as they pass the Oracle Java Cloud Service - SaaS Extension whitelist tool, and that their use is in-line with Java EE best practices. See Oracle Java Cloud Service - SaaS Extension Whitelist Validation.

Other Supported Java EE 5 and 6 Specifications

This section describes other Java EE 5 and Java EE 6 specifications supported by the Oracle Java Cloud Service - SaaS Extension.

Tip:

Some Java EE specifications in this section relate purely to the underlying Java EE container environment and are irrelevant to the actual Java EE deployment archives and how they are developed (for example, Java Authentication and Authorization Service (JaaS)). Although these specifications are supported, they are not listed here.

<table>
<thead>
<tr>
<th>Supported Specification</th>
<th>Supported Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaServer Pages Standard Tag Library (JSTL)</td>
<td>1.2</td>
</tr>
<tr>
<td>Java Data Base Connectivity (JDBC)</td>
<td>4.0</td>
</tr>
<tr>
<td>Java Persistence API</td>
<td>2.0</td>
</tr>
<tr>
<td>Web Services Metadata for the Java Platform</td>
<td>2.0</td>
</tr>
<tr>
<td>Java Naming and Directory Interface Specification (JNDI)</td>
<td>1.2</td>
</tr>
<tr>
<td>Java Transaction API (JTA)</td>
<td>1.1</td>
</tr>
<tr>
<td>Streaming API for XML (StAX)</td>
<td>1.0</td>
</tr>
<tr>
<td>SOAP with Attachments API for Java (SAAJ)</td>
<td>1.3</td>
</tr>
<tr>
<td>JavaBeans Activation Framework Specification (JAF)</td>
<td>1.1</td>
</tr>
<tr>
<td>Java API for XML Processing (JAXP)</td>
<td>1.3</td>
</tr>
</tbody>
</table>
Supported Specification | Supported Version
---|---
Java Management Extensions (JMX) | 1.2
**Note:** JMX is only supported for exposure of MBeans within a deployment archive and access to these MBeans from the deployment archive itself or other archives deployed to the same Oracle Java Cloud Service - SaaS Extension instance.

Java API for XML-based Web Services (JAX-WS) | 2.1
Java API for RESTful Web Services (JAX-RS) | 1.1
Java Architecture for XML Binding (JAXB) | 2.0

**Other Supported Public WebLogic Server 10.3.6 APIs and Capabilities**

This section describes additional public WebLogic Server 10.3.6 APIs and capabilities supported by the Oracle Java Cloud Service - SaaS Extension.

**Note:**

As a best practice, Oracle recommends that you always use standard Java APIs for your Oracle Java Cloud Service - SaaS Extension and avoid using WebLogic Server APIs to ensure that your applications are portable to other environments. This way your applications will not get locked into running only on the Oracle Oracle Java Cloud Service - SaaS Extension.

<table>
<thead>
<tr>
<th>API</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>weblogic.logging.*</td>
<td>Used for internal (non-catalogue) WebLogic server logging</td>
</tr>
<tr>
<td>weblogic.jsp.*</td>
<td>For applications using custom WebLogic Server specific tags</td>
</tr>
<tr>
<td>weblogic.cache.*</td>
<td>Response caching servlet filter</td>
</tr>
<tr>
<td>weblogic.application.*</td>
<td>Used for implementation of application life-cycle listeners</td>
</tr>
<tr>
<td>weblogic.i18n.*</td>
<td>Public I18N APIs and logging</td>
</tr>
<tr>
<td>weblogic.i18ntools.*</td>
<td>Public I18N APIs and logging</td>
</tr>
<tr>
<td>weblogic.jndi.*</td>
<td>For Java Naming and Directory Interface (JNDI) lookup within WLS JNDI tree</td>
</tr>
<tr>
<td>weblogic.jws.*</td>
<td>WebLogic specific extensions to JAX-WS for supporting WS-*</td>
</tr>
<tr>
<td>weblogic.servlet.*</td>
<td>For annotations based servlet descriptions</td>
</tr>
<tr>
<td>weblogic.transaction.*</td>
<td>API used for direct JTA interaction</td>
</tr>
</tbody>
</table>

**Using Third-Party Frameworks with Oracle Java Cloud Service - SaaS Extension**

You can use third-party frameworks to extend the functionality of Oracle Java Cloud Service - SaaS Extension.

You can use each of these frameworks with Oracle Java Cloud Service - SaaS Extension in one of the following ways:
• Packaging the framework with applications that use it
• Deploying the framework as a shared library
  For more information, see Deploying, Redeploying, and Deleting Libraries.

If multiple applications use a framework, or if you want to simplify updates by minimizing the size of applications that use the framework, deploy the framework as a shared library.

Topics
• Supported Third-Party Frameworks for Oracle Java Cloud Service - SaaS Extension
• Omitting Checks for Updates to Quartz Job Scheduler
• Using Non-Listed Frameworks

Supported Third-Party Frameworks for Oracle Java Cloud Service - SaaS Extension

Oracle Java Cloud Service - SaaS Extension supports several third-party frameworks.

<table>
<thead>
<tr>
<th>Framework</th>
<th>Release</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Commons component BeanUtils</td>
<td>1.9.2</td>
<td>Simplify the use of the Java reflection and introspection APIs.</td>
</tr>
<tr>
<td>Apache Commons component Collections</td>
<td>3.2.1</td>
<td>Extend or augment the Java Collections Framework.</td>
</tr>
<tr>
<td>Apache Commons component Digester</td>
<td>3.2</td>
<td>Map XML configuration data to Java objects.</td>
</tr>
<tr>
<td>Apache Commons component IO</td>
<td>2.4</td>
<td>Help develop functionality for input and output through data streams.</td>
</tr>
<tr>
<td>Apache Commons component Logging</td>
<td>1.2</td>
<td>Enable a library to be used with a chosen logging implementation at runtime.</td>
</tr>
<tr>
<td>Apache log4j</td>
<td>1.2.17</td>
<td>Enables logging at runtime without modifying the application binary and allows these statements to remain in shipped code without incurring a heavy performance cost.</td>
</tr>
<tr>
<td></td>
<td>2.0</td>
<td></td>
</tr>
<tr>
<td>Apache Struts</td>
<td>2.3.3</td>
<td>Simplify the development of Java web applications that use a Model-View-Controller (MVC) architecture.</td>
</tr>
<tr>
<td>Apache Tapestry</td>
<td>5.3.6</td>
<td>Used for creating dynamic, robust, highly scalable web applications in Java.</td>
</tr>
<tr>
<td>Apache Wicket</td>
<td>6.18.0</td>
<td>Simplify the development of Java web applications by:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Properly separating markup and logic</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Using a Plain Old Java Object (POJO) data model</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Limiting the use of Extensible Markup Language (XML) configuration files</td>
</tr>
</tbody>
</table>

Chapter 1
Considerations When Developing Applications on Oracle Java Cloud Service - SaaS Extension
<table>
<thead>
<tr>
<th>Framework</th>
<th>Release</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>FreeMarker</td>
<td>2.3.19</td>
<td>A Java package or class library used by Java programmers to generate template-based text output (anything from HTML to auto-generated source code).</td>
</tr>
<tr>
<td>Google Guice</td>
<td>3.0</td>
<td>An open source software Java framework that provides support for dependency injection by using annotations to configure Java objects.</td>
</tr>
<tr>
<td>Google Web Toolkit (GWT Web Toolkit)</td>
<td>2.5.1</td>
<td>Provides a framework for creating and maintaining complex JavaScript front-end applications in Java.</td>
</tr>
<tr>
<td>Hibernate ORM</td>
<td>4.2.4</td>
<td>Provide a framework for mapping an object-oriented domain model to a traditional relational database.</td>
</tr>
<tr>
<td>JodaTime</td>
<td>2.1</td>
<td>Serves as the <em>de facto</em> standard date and time library for Java applications.</td>
</tr>
<tr>
<td>JQuery</td>
<td>2.0.3</td>
<td>Provide a JavaScript library to simplify HTML document traversal and manipulation, event handling, animation, and Ajax.</td>
</tr>
<tr>
<td>Play</td>
<td>2.1.0</td>
<td>Optimizes your productivity by using convention over configuration, hot code reloading and browser display of errors.</td>
</tr>
<tr>
<td>Quartz Job Scheduler</td>
<td>2.1.5</td>
<td>Create simple or complex schedules for executing jobs whose tasks are defined as standard Java components.</td>
</tr>
<tr>
<td>SLF4J (Simple Logging Facade for Java)</td>
<td>1.7.7</td>
<td>Provides a simple facade or abstraction for various logging frameworks (<code>java.util.logging</code>, <code>logback</code>, <code>log4j</code>) which allows end users to plug in the desired logging framework at deployment time.</td>
</tr>
<tr>
<td>Spring</td>
<td>3.0</td>
<td>Provides a comprehensive programming and configuration model for modern Java-based enterprise applications, regardless of their deployment platform.</td>
</tr>
</tbody>
</table>

Omitting Checks for Updates to Quartz Job Scheduler

By default, Quartz Job Scheduler checks for updates when it starts.

The check for updates involves connecting to a remote server. If the server cannot be reached, the check fails and an exception is written to log file. The failure does not prevent Quartz Job Scheduler from starting and does not affect the functionality of Quartz Job Scheduler in any way. However, you can prevent this exception by omitting checks for updates to Quartz Job Scheduler.

To omit checks for updates to Quartz Job Scheduler, use the Oracle Java Cloud Service - SaaS Extension SDK to set the Quartz configuration property `org.quartz.scheduler.skipUpdateCheck` to `true`.

See the following Quartz Job Scheduler documentation:

- **Skip Update Check** in *Best Practices*
- **Configure Main Scheduler Settings** in *Configuration Reference*

Using Non-Listed Frameworks

You can use third-party frameworks *not* included on the approved list but you need to be careful that these frameworks are valid.
While Oracle strongly recommends that you limit your use of third-party frameworks to only those that have been tested and approved (see Supported Third-Party Frameworks for Oracle Java Cloud Service - SaaS Extension), that doesn't mean that non-listed libraries won't work. If you chose to use a non-listed framework, you should use the Whitelist Tool to validate the framework. The Whitelist tool performs a type of compatibility test on every application installed or updated in a Java Cloud Service - SaaS Extension instance by validating deployment descriptors and other application configuration files. If you use the tool during runtime, you will need to address any warning that are generated.

Run the Whitelist Tool by issuing this command:

```
./whitelist [-argument ...] [-help] [file1 file2 dir1 dir2 ...]
```

For example:

```
./whitelist -log /home/log/newlog.log /home/apps/myapp.war
```

The valid arguments for `whitelist` are described in Use the Whitelist Tool.

About the Application Deployment Validation Process and Run-time Security

During the Oracle Java Cloud Service - SaaS Extension deployment process, every application or library undergoes a series of security checks before that application or library is actually deployed. For technical and security reasons, a small number of specific APIs are prevented from executing in Oracle Cloud.

**Note:**

During application deployment and at run-time, the Oracle Java Cloud Service - SaaS Extension utilizes both the Java Security Manager and a whitelisting tool to enforce certain API restrictions. However, these API validations are not the primary security defense mechanisms for Oracle Java Cloud Service - SaaS Extension. Oracle Cloud has extensive primary security defense mechanisms at the VM, OS and network layers.

The Java Security Manager performs additional security validation during application run-time. For example, an application that has packaged some third-party JAR files that have API violations are permitted to be deployed as long as the violated usages are not exercised during run-time. Security exceptions will be raised when those APIs are exercised.

For additional information, see Unsupported Features and APIs.

Application and Library Deployment Validation Flow

Every application or library that is being deployed undergoes background security checks before that application is attempted to deployed.

The background and security checks follow this sequence:
1. Virus scan
2. Whitelist validation
3. WLS compile
4. Cloud compile
5. Deploy

For a typical deployment, Oracle Java Cloud Service - SaaS Extension generates five logs, one for each of these security checks. These logs are the result of background jobs that ran against the application and determined whether the application contains a virus or could otherwise cause problems. See Viewing the Activity Logs.

Oracle Java Cloud Service - SaaS Extension Whitelist Validation

The Whitelist tool validates deployment descriptors and other application configuration files, such as the log4j.properties file, as part of the Java API validation.

If an application contains any Java API validations, the Whitelist tool might not reject it from being deployed. Instead, it would create a warning report against the violations. A security exception will be raised only during runtime, should those exceptions be exercised when the application is running. For example, it is common for third-party libraries to raise warnings during Whitelist validation; however, they are rarely exercised during runtime.

For instructions on downloading the SDK, see Downloading the Oracle Java Cloud Service - SaaS Extension SDK.

Note:
The Oracle Java Cloud Service - SaaS Extension "whitelist" is actually the result of what are sometimes called blacklist and whitelist checks. It may be helpful to think of Oracle Java Cloud Service - SaaS Extension whitelist validation as simply a compatibility check.

After automatic whitelist validation, if you are encountering additional deployment problems, you can locally validate an application by using the Whitelist Tool (whitelist.jar), which is available in the Java Cloud Service - SaaS Extension SDK. See Use the Whitelist Tool.

Prerequisites for Using Oracle Java Cloud Service - SaaS Extension

Prior to using Oracle Java Cloud Service - SaaS Extension, ensure you are familiar with the prerequisites described in this topic.

- Oracle Cloud
  Create and configure your account on Oracle Cloud. See Subscribing to an Oracle Cloud Service Trial or Buying a Non-metered Subscription to an Oracle Cloud Service in Getting Started with Oracle Cloud.
• Oracle Java Cloud Service - SaaS Extension SDK Provides utilities that facilitate the management of Oracle Java Cloud Service - SaaS Extension instances and the development of applications for the Oracle Java Cloud Service - SaaS Extension, such as a CLI, Apache Ant tasks, and a Maven plug-in. To download the Java SDK from the Oracle Technology Network, see Downloading the Oracle Java Cloud Service - SaaS Extension SDK.

The Oracle Java Cloud Service - SaaS Extension SDK is required if you want to integrate your service instance with one of the supported IDEs. See Downloading the Oracle Java Cloud Service - SaaS Extension SDK.

• Supported IDEs

If you prefer to use an IDE for developing WebLogic Server applications, you can download one the supported IDEs from the Oracle Technology Network. These IDEs have embedded tooling that enables to directly interact with an Oracle Java Cloud Service - SaaS Extension instance.

– Oracle JDeveloper
– Oracle Enterprise Platform for Eclipse
– NetBeans

Sizing and Deployment Recommendations

Oracle recommends the following sizing and deployment values for JCS-SaaS Extension implementations:

• Default maximum PermGen: 512 MB (adjustable up to 1024 MB)

• RAM allocated is 1.5GB per node:

  – S1 (1 node): 1.5GB
  – S2 (2 nodes): 3GB
  – S4 (4 nodes): 6GB

For a production environment, an S2 or S4 configuration is recommended. These configurations provide high-availability protection in case of loss of a single compute node.

• Maximum 10 database connections per node:

  – Trial instances: 5
  – S1 (1 node): 10
  – S2 (2 nodes): 20
  – S4 (4 nodes): 40

About Oracle Java Cloud Service - SaaS Extension Roles and Users

Predefined roles and users determine who can access, deploy, and administer tasks Oracle Java Cloud Service - SaaS Extension and applications.

In addition to the roles and privileges described in Adding Users and Assigning Roles in Getting Started with Oracle Cloud for Identity Domain Administrators and Service
Administrators, the following table summarizes Oracle Java Cloud Service - SaaS Extension roles and users.

<table>
<thead>
<tr>
<th>Role</th>
<th>Description</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oracle Java Cloud Service - SaaS Extension Administrator</td>
<td>Access Web-based Oracle Java Cloud Service - SaaS Extension Control UI to manage and monitor a service instance. Use the downloaded Oracle Java Cloud Service - SaaS Extension SDK to manage and monitor a service instance.</td>
<td>Understanding Oracle Java Cloud Service - SaaS Extension Control</td>
</tr>
</tbody>
</table>


- Oracle JDeveloper
- Oracle Enterprise Platform for Eclipse
- NetBeans

Use the downloaded Oracle Java Cloud Service - SaaS Extension SDK to interact with a service instance. Use Web-based Oracle Java Cloud Service - SaaS Extension Control to interact with a service instance.
Getting Started with Your JCS-SaaS Extension Subscription

Once your subscription is purchased and processed, you receive a Welcome email. From here, you can begin your JCS-SaaS Extension experience.

Topics:
• Accessing Your Service for the First Time
• Provisioning a Database Instance
• Provisioning a JCS-SaaS Extension Instance

Accessing Your Service for the First Time

Use the Get Started link in your Welcome email to access your Oracle Cloud dashboard provision the necessary services to launch your JCS-SaaS Extension experience.

Note:
The Welcome email contains your username, a temporary password, and the identify domain for your service. Before accessing the service, it is recommended that you record or otherwise make note of this information as it is required for that access.

1. In the Access Details section of the Welcome email, click Get Started With Oracle Cloud.
2. If you are logging in to this identity domain for the first time, enter the identity domain provided in the Welcome email and click Go.
3. On the ensuing log-in page, enter your username and the temporary password and click Sign In.
4. Follow the prompts to enter and confirm a new password and register your challenge questions. Then click Submit.
5. If necessary, on the Set Preferences dialog box, select your language and timezone preferences and click OK.
7. On the Develop and Deploy page, for:
   • What kind of application: select Java
   • Any particular type: select Java EE
   Then click Go to My Dashboard.

From My Dashboard, you will then create both an Oracle Database Cloud Service instance and your first JCS-SaaS Extension instance.
Provisioning a Database Instance

Once you’ve gained access to your subscription, you need to create—or “provision”—an instance of Oracle Database Cloud Schema Service. This is required before you can create a JCS-SaaS Extension instance.

1. From the dashboard, click **Create Instance**.
2. In the Database Schema section of the Create Instance dialog box, click **Create** to open the Create New Oracle Database Cloud Schema Service Instance page.
3. On the Create… page, enter these Instance Details:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>The name of the Schema Service instance. This name must begin with a letter and contain only numbers and lowercase letters.</td>
</tr>
<tr>
<td>Shape</td>
<td>The shape of the Schema Service instance. “Shape” determines the hardware resources provided to your cloud database, particularly the maximum storage available.</td>
</tr>
</tbody>
</table>

4. Enter these Administrator Details:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>Enter the email address for the person who will be the administrator for the instance.</td>
</tr>
<tr>
<td>Use email as user name</td>
<td>Select this checkbox if you want to use the administrator’s email address as the username for this account.</td>
</tr>
<tr>
<td>User Name</td>
<td>If you aren’t using the administrator email address as the user name for this instance, enter a user name here (if you selected Use email as user name, this field will not appear).</td>
</tr>
<tr>
<td>First Name</td>
<td>Enter your first (given) name or the given name of the person who will own this service instance.</td>
</tr>
<tr>
<td>Last Name</td>
<td>Enter your surname or the surname of the person who will own this service instance.</td>
</tr>
</tbody>
</table>

5. Click **Create** and then, on the Confirmation message, click **Create** again. This will launch the provisioning process and open the My Services page for Oracle Database Cloud Schema Service. Click to refresh your screen until details of the new database instance appear on the page.

After the instance is provisioned, not only will the instance appear on your My Services page, but Oracle Cloud will send a confirming email to the administrator address specified in step 4.

See:

- About Oracle Database Cloud - Database Schema Service in *Using Oracle Database Cloud — Database Schema Service*
- Creating a Service Instance in *Using Oracle Database Cloud — Database Schema Service*
Provisioning a JCS-SaaS Extension Instance

Once you’ve provisioned an Oracle Database Cloud Schema Service instance, you can provision your JCS-SaaS Extension instance.

1. From the dashboard, click **Create Instance**.
2. In the JCS-SaaS Extension section of the Create Instance dialog box, click **Create Service Instance** to open the Create New Oracle Java Cloud Service - SaaS Extension Instance page.
3. On the Create... page, enter these Instance Details:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Enter a name for the Oracle Java Cloud Service — SaaS Extention instance. This name must start with a letter and can contain up to 25 alphanumeric characters. Letters must be lowercase and you cannot use spaces or special characters.</td>
</tr>
<tr>
<td>Related Deployments</td>
<td>If a Fusion Application instance is detected in any of the domains configured within your account, then the instance creation wizard shows an Related Deployments selection box.</td>
</tr>
<tr>
<td></td>
<td>Select the target domain where you want to create the instance. If you create the instance in the identity domain where the Fusion Application is located, then you can use Single Sign-On (SSO).</td>
</tr>
<tr>
<td></td>
<td>If you do not select an identity domain, a new identity domain is created.</td>
</tr>
<tr>
<td></td>
<td>If you select the wrong identity domain, then you can delete the instance and create a new one.</td>
</tr>
<tr>
<td>Associations</td>
<td>Select the Database Cloud Schema Service Instance you created in Provisioning a Database Instance. You can also select HCM, CRM, and ERP, provided you have any of these applications in the same account.</td>
</tr>
<tr>
<td>Recommended Associations</td>
<td>If you have multiple SaaS instances (for example, HCM and ERP) in your account, they will be listed here. Select the application you intend to modify with custom ADF applications and deploy on JCS-SaaS Extension. If you aren’t planning to customize any listed instance, then you can leave this field blank.</td>
</tr>
</tbody>
</table>
Select the JCS-SaaS Extension plan to which you’ve subscribed. The “Plan” determines the cluster size of the Oracle Java Cloud Service SaaS Extension. Available Plans are:

- **S1 (Basic):**
  - Managed servers: 1 Java EE Server
  - Memory: 1.5GB /Instance
  - Storage: 5 GB
  - Data Transfer: 50GB / Month

- **S2:**
  - Managed servers: 2 Java EE Servers
  - Memory: 3GB /Instance
  - Storage: 10GB
  - Data Transfer: 250GB / Month

- **S4:**
  - Managed servers: 4 Java EE Servers
  - Memory: 6GB /Instance
  - Storage: 25GB
  - Data Transfer: 500GB / Month

4. Enter these Administrator Details:

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Email</td>
<td>Enter the email address for the person who will be the administrator for the instance.</td>
</tr>
<tr>
<td>Use email as user name</td>
<td>Select this checkbox if you want to use the administrator’s email address as the username for this account.</td>
</tr>
<tr>
<td>User Name</td>
<td>If you aren’t using the administrator email address as the user name for this instance, enter a user name here (if you selected Use email as user name, this field will not appear).</td>
</tr>
<tr>
<td>First Name</td>
<td>Enter your first (given) name or the given name of the person who will own this service instance.</td>
</tr>
<tr>
<td>Last Name</td>
<td>Enter your surname or the surname of the person who will own this service instance.</td>
</tr>
</tbody>
</table>

5. Click **Create** and then, on the Confirmation message, click **Create** again. This will launch the provisioning process and open the My Services page for Oracle Database Cloud Schema Service. Click to refresh your screen until details of the new database instance appear on the page.

Once the instance is provisioned, not only will the instance appear on your My Services page, but Oracle Cloud will send a confirming email to the administrator address specified in step 4.
Accessing Oracle Java Cloud Service - SaaS Extension

Access your Oracle Java Cloud Service - SaaS Extension Control environment by using the credentials you received by e-mail when you signed up for your trial service or purchased your new service.

Note:
Before you attempt to access JCS-SaaS Extension, you must have:
• Your data center name
• Your identity domain
• the user name and password for your Cloud Account

This information is provided with the Welcome email you receive when you subscribe to the service. If you don't have this information, click Account Details on the right side of the Cloud Account sign-in page and enter the email address associated with the Cloud Account. Oracle will send you an email with a summary of your account information.

If you are logging in for the first time, use the temporary password you received in the Welcome email or from your administrator. You will be prompted to change your password immediately.

Topics:
• Accessing JCS-SaaS Extension Administration Console from a URL
• Accessing JCS-SaaS Extension Administration Console from the My Services Dashboard
• Accessing the JCS-SaaS Extension Administration Console from the Service Details Page

Accessing Oracle Java Cloud Service - SaaS Extension Control from a URL

You can access your Oracle Java Cloud Service - SaaS Extension Control by using the URL provided by email or by your administrator.

1. Open your web browser and enter the Oracle Java Cloud Service - SaaS Extension Control URL.
   The Sign In page appears.

2. Enter your Sign In credentials and click Sign In. (Note that you need Service Owner credentials to access this console.)
   The Oracle Java Cloud Service - SaaS Extension Control home page appears.
Accessing JCS-SaaS Extension Administration Console from the My Services Dashboard

You can view and access all of your PaaS services, for example JCS-SaaS Extension, from the My Services Dashboard. From here, you can also create new instances of your services and access their administration consoles.

Signing In to Your Account and Accessing the Administration Console

Note:

If this is the first time you are accessing this account, see the procedures for first-time login in Accessing Your Service for the First Time.

To access your Oracle Java Cloud Service - SaaS Extension Administration Console from the My Services Dashboard:

1. Access the My Services Dashboard:
   a. Navigate to cloud.oracle.com and click Sign In.
   b. On the Cloud Account log-in page, open the account type drop-down and select Traditional Cloud Account.
   c. Open the Select Data Center and select your data center.
   d. Click My Services.

The Traditional Cloud Account sign-in page appears. If this is the first time you’ve ever accessed this account, the first page will request the identity domain provided in the Welcome email, otherwise your identity domain will appear in the edit box.

1. If necessary, enter your identity domain then click Go.
f. On the subsequent log-in page, enter your user name and password and click **Sign In**.
   The My Services Dashboard for your identity domain appears.

2. Depending on your subscription type, do one of the following:
   - **If you are running a production version** of JCS-SaaS Extension, click and select **Open Service Console**.
   - **If you are running a trial version** of JCS-SaaS Extension:
     a. Click the name of your JCS-SaaS Extension service.
     b. On the My Services Overview page, scroll to the Service Instances section and locate the service you want to access. Click **Open Service Console**.

**Accessing the JCS-SaaS Extension Administration Console from the Service Details Page**

The service details page allows you to view status history, availability history, usage metrics, and additional information for a specific service.

To access your JCS-SaaS Extension Control from the My Services Details page:

1. Access the My Services Dashboard:
   a. Navigate to `cloud.oracle.com` and click **Sign In**.
   b. On the Cloud Account log-in page, open the account type drop-down and select Traditional Cloud Account.

   ![Cloud Account](image)

   Selecting Traditional Cloud Account causes a new drop-down control, labeled Select Data Center, to appear below the Account Type drop-down.

c. Open the Select Data Center and select your data center.

d. Click My Services.

The Traditional Cloud Account sign-in page appears. If this is the first time you've ever accessed this account, the first page will request the identity domain provided in the Welcome email, otherwise your identity domain will appear in the edit box.
e. If necessary, enter your identity domain then click **Go**.

f. On the subsequent log-in page, enter your user name and password and click **Sign In**.

   The My Services Dashboard for your identity domain appears.

2. Click the name of your JCS-SaaS Extension service.

3. On the My Services Overview page, scroll to the Service Instances section and locate the service you want to access. Click **Open Service Console**.

---

**Using the Welcome App**

The Welcome app provides access to important supporting information and resources for your JCS-SaaS Extension implementation.

The Welcome app is the default application deployed on a JCS-SaaS Extension instance. There are two ways of launching the Welcome app:

- Link on the email you receive to confirm that your ready
- Link in the Java Console in MyServices

The Welcome app provides access to the following:

- A link to the JCS-SX Administration Console
- Documentation, tutorials and videos
- The JCS-SaaS Extension SDK and CLI along with its documentation
- Blogs covering JCS-SaaS Extension and other Oracle Cloud services
- Frequently asked questions

The following table describes the content of each link target.

<table>
<thead>
<tr>
<th>Link</th>
<th>Description of target content</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Java Cloud Service - SaaS Extension Administration Console</strong></td>
<td>This is a link to the new JCS-SaaS Extension Resource UI, where you can manage your instance.</td>
</tr>
<tr>
<td><strong>Documentation</strong></td>
<td>This link forwards you to the Oracle Cloud Service Help Center for JCS-SaaS Extension. The Help Center provides access to the most current user assistance (UA) assets available for this service, including user procedures, SDK details, instructions for managing and monitoring instances, and extending SaaS applications by propagating user identity across application and the requisite platform services.</td>
</tr>
<tr>
<td><strong>SDK Documentation</strong></td>
<td>In addition to the documentation available in Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension, you can access online documentation for all SDK command-line tools by clicking SDK Documentation. This documentation is included with the SDK download and is installed in your [SDKHOME]/doc/ directory. The landing page is in index.html.</td>
</tr>
<tr>
<td>Link</td>
<td>Description of target content</td>
</tr>
<tr>
<td>----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
</tbody>
</table>
| **Download SDK**     | This link forwards you to the Oracle Cloud Downloads page, where you can download the JCS-SaaS Extension SDK. This SDK provides tools that help you manage service instances and develop applications for the Oracle for the service. These same tools can also be used in your development environment against a local WebLogic Server domain.  
  - For more information on the SDK, see [About the Oracle Java Cloud Service - SaaS Extension SDK](#).  
  - For additional download instructions, see [Downloading the Oracle Java Cloud Service - SaaS Extension SDK](#). |
| **Related Links**    | These links point to additional UA assets:                                                                                                                                 |
|                      |  - Oracle by Example tutorials to assist in learning to use JCS-SaaS Extension.  
  - Videos containing instructions and tips for using JCS-SaaS Extension.  
  - Books related to JCS-SaaS Extension, downloadable in multiple formats, including HTML, PDF, and MOBI. |
| **FAQ**              | This link points to general, frequently asked questions about JCS-SaaS Extension, including how it differs from Java Cloud Service. The FAQ also contains information about other JCS-SaaS Extension and JCS features. Note that not all features supported by one service are supported by the other. This is a generalized product FAQ; you can find a more JCS-SaaS Extension usage-specific FAQ available in [Frequently Asked Questions for Oracle Java Cloud Service - SaaS Extension](#). |
Developing Applications for Oracle Java Cloud Service - SaaS Extension

This section provides documentation about the application development tasks for the Oracle Java Cloud Service - SaaS Extension.

Topics:
- Typical Workflow for Using Oracle Java Cloud Service - SaaS Extension
- Downloading the Oracle Java Cloud Service - SaaS Extension SDK
- Preparing Applications for Oracle Java Cloud Service - SaaS Extension Deployment
- Accessing Applications Deployed on Oracle Java Cloud Service - SaaS Extension
- Messaging Support in Oracle Java Cloud Service - SaaS Extension
- Developing RESTful Web Services
- Securing Applications in Oracle Java Cloud Service - SaaS Extension
- Creating an On-premise WebLogic Server Environment

Typical Workflow for Using Oracle Java Cloud Service - SaaS Extension

Using the Oracle Java Cloud Service - SaaS Extension should follow a series of tasks described in a typical workflow.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request a trial or purchase a subscription to an Oracle Cloud service</td>
<td>Provide your information, and sign up for a free trial or purchase a subscription.</td>
<td>Subscribing to an Oracle Cloud Service Trial or Ordering an Oracle Cloud Service, in Getting Started with Oracle Cloud</td>
</tr>
<tr>
<td>Activate a service</td>
<td>After Oracle processes your trial request or purchase order, sign in to My Account and activate the service.</td>
<td>Activating Your Trial Subscription or Activating Your Order section in Getting Started with Oracle Cloud</td>
</tr>
<tr>
<td>Verify the service is activated</td>
<td>Once the activation process is complete, sign in to My Services and confirm that your service is now up and available for use.</td>
<td>Verifying a Service Is Running section in Getting Started with Oracle Cloud</td>
</tr>
<tr>
<td>Task</td>
<td>Description</td>
<td>More Information</td>
</tr>
<tr>
<td>----------------------------------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Add and manage users and user roles</td>
<td>Create accounts for your users and assign them appropriate privileges.</td>
<td>Adding Users and Assigning Roles in Getting Started with Oracle Cloud</td>
</tr>
<tr>
<td>Access the My Services Dashboard and provision a Database Cloud Service instance</td>
<td>A DBCS instance is required before you can create a JCS-SaaS Extension instance. This tasks enables you to meet that requirement. This tasks enables you to meet that requirement.</td>
<td>Provisioning a Database Instance</td>
</tr>
<tr>
<td>Remain on the My Services Dashboard and provision a JCS-SaaS Extension instance</td>
<td>You require a JCS-SaaS Extension instance in order to deploy and manage applications.</td>
<td>Provisioning a JCS-SaaS Extension Instance</td>
</tr>
<tr>
<td>Monitor and manage Oracle Cloud services performance and usage</td>
<td>Monitor Oracle Cloud services performance and usage by observing the available service metrics and utilization</td>
<td>Viewing Service Details in Managing and Monitoring Oracle Cloud</td>
</tr>
<tr>
<td>Download the Oracle Java Cloud Service - SaaS Extension SDK</td>
<td>The Oracle Java Cloud Service - SaaS Extension SDK provides utilities that facilitate the management of Oracle Java Cloud Service - SaaS Extension instances and the development of applications for the Oracle Java Cloud Service - SaaS Extension</td>
<td>Downloading the Oracle Java Cloud Service - SaaS Extension SDK</td>
</tr>
<tr>
<td>Prepare applications for deployment to the service</td>
<td>Review guidelines before developing applications to deploy on an Oracle Java Cloud Service - SaaS Extension instance</td>
<td>Preparing Applications for Oracle Java Cloud Service - SaaS Extension Deployment</td>
</tr>
<tr>
<td>Access applications deployed on the service</td>
<td>From a web-browser, use an URL to access applications that have been deployed on an Oracle Java Cloud Service - SaaS Extension instance</td>
<td>Accessing Applications Deployed on Oracle Java Cloud Service - SaaS Extension</td>
</tr>
</tbody>
</table>
## Typical Workflow for Using Oracle Java Cloud Service - SaaS Extension

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
<th>More Information</th>
</tr>
</thead>
</table>
| Monitor and manage the service using Oracle Java Cloud Service - SaaS Extension Control. | The Oracle Java Cloud Service - SaaS Extension Control is a web-based management console that enables you to deploy and monitor your hosted applications | For instances created before January, 2018, see Understanding Oracle Java Cloud Service - SaaS Extension Control  
For instances provisioned after January 2018, see Working with the Oracle Java Cloud Service - SaaS Extension Administration Console |
| Monitor and manage the service using the command-line interface. | The Oracle Java Cloud Service - SaaS Extension SDK provides access to a command-line interface to monitor and manage the service | Using the Command-Line Interface to Monitor Oracle Java Cloud Service - SaaS Extension |
| Develop RESTful web services              | Build RESTful web services using the pre-built, shared Jersey JAX-RS RI library | Developing RESTful Web Services                                                   |
| Secure your applications                  | Provide secure access to your Java EE or ADF applications, such as specifying role-based authentication | Securing Applications in Oracle Java Cloud Service - SaaS Extension               |
| Optionally, create an on-premise environment | Create an on-premise Java EE environment that is comparable to an Oracle Java Cloud Service - SaaS Extension instance | Creating an On-premise WebLogic Server Environment                                |
| Change your paid subscription             | Upsize or update your service to a higher subscription level.                | Updating Your Paid Subscription from Oracle Cloud in Managing and Monitoring Oracle Cloud |

**Note:**

The JCS-SaaS Extension Administration Console is a beta product. It is in preproduction status and is intended for demonstration and preliminary use only. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to this product and will not be responsible for any loss, costs, or damages incurred due to the use of this documentation.
Preparing Applications for Oracle Java Cloud Service - SaaS Extension Deployment

This section provides guidelines and considerations for developing applications, such as applications that use a JDBC data source, in order to deploy them on an Oracle Java Cloud Service - SaaS Extension instance.

Topics:

- Understanding Application Library Behavior Changes on Oracle Cloud
- Guidelines for Applications That Use a JDBC Data Source
- Guidelines for ADF Applications
- Guidelines for Applications That Use Java EE or ADF Application Security
- Guidelines for Applications When Accessing a Local File System
- Guidelines for Applications When Accessing System Properties
- Guidelines for Applications When Using Log4j Appenders

You must follow these guidelines either when developing new applications targeted to an Oracle Java Cloud Service - SaaS Extension instance or when modifying existing applications targeted to an Oracle Java Cloud Service - SaaS Extension instance.

Understanding Application Library Behavior Changes on Oracle Cloud

Unless the Oracle Java Cloud Service - SaaS Extension documentation explicitly states changes to the original behavior of application libraries in order to be compatible with Oracle Java Cloud Service - SaaS Extension, then the application library's default behavior is not changed.

An example of a default application library behavioral change is the Log4j Console Appender, which has changed on Oracle Cloud. See Guidelines for Applications When Using Log4j Appenders.

An example of unchanged behavior is the setting of Log4j or JDK log levels using application code. These application libraries have not changed on Oracle Cloud, so they should work as they do in on-premise environments.

Frequently Asked Questions for Oracle Java Cloud Service - SaaS Extension is the only section of Using Oracle Java Cloud Service that provides examples for application libraries whose behavior does not change on Oracle Cloud.

Guidelines for Applications That Use a JDBC Data Source

The association of an Oracle Java Cloud Service - SaaS Extension instance with a Database Cloud Service instance makes the database instance available to deployed applications as a JDBC data source. By default, the JNDI name of the data source is the same name given to the Database Cloud Service instance at provisioning time. For example, if the name of the Database Cloud Service service instance is javatrial1801db, then the JNDI name of the Database Cloud Service instance will also be javatrial1801db.
If your application is using a JDBC data source for database access, then all references within the application to the assigned data source must be configured or modified to use either:

- The JNDI name of the data source assigned within the Oracle Java Cloud Service - SaaS Extension instance.
- A used-defined alias for the JNDI name of data source assigned within the Oracle Java Cloud Service - SaaS Extension instance.

Using a JNDI Alias for a JDBC Data Source

You can create an alias for the JNDI name of the JDBC data source assigned to your Oracle Java Cloud Service - SaaS Extension instance. Therefore, if your application is using a data source for database access, then all references to the data source can use the alias JNDI name instead of having to use the actual JNDI name of the data source.

For example, if the JNDI name of the Database Cloud Service instance is javatrial1801db, you can create an JNDI name alias for the data source named mycustomalias, and then just use mycustomalias in all references to the data source within the application.

Using an JNDI alias for a data source eliminates the need to update your applications if they need to be redeployed to another Oracle Java Cloud Service - SaaS Extension instance that is associated with a different Database Cloud Service. Instead, you only need to add same the JNDI alias referenced in your applications to the JDBC data source in the new Oracle Java Cloud Service - SaaS Extension instance.

Adding a JNDI Name Alias

You can create and manage JNDI name aliases for data sources using the javacloud.jar CLI in the Oracle Java Cloud Service - SaaS Extension SDK.

Here is an example of using the CLI to add a JNDI name alias for a data source:

```
$ ./javacloud -u username@oracle.com -id usoracletrial08411 -si javatrial5334 -add-datasource-jndiname -jndiname mycustomalias

JNDI Name "mycustomalias" added to the data source : javatrial5334db
```

Here is an example of using the CLI to list your data source JNDI names:

```
$ ./javacloud -u username@oracle.com -id usoracletrial08411 -si javatrial5334 -list-datasource-jndinames

#=================================#
| Listing 2 DS JNDI Alias(es) |
| [Identity | Domain=usoracletrial08411, |
| Service   | Instance=javatrial5334]     |
#=================================#
| S.NO  | Jndi Alias | ReadOnly |
|=======|============|==========|
| 1     | javatrial5334db | true    |
| 2     | mycustomalias    | false   |
```
Guidelines for ADF Applications

Follow the guidelines in this topic whenever you use Oracle Java Cloud Service — SaaS Extension to deploy an ADF application.

If you are deploying an ADF application, you must configure or modify its weblogic.xml deployment descriptor to use the <exact-match> element, as described in the following example.

```xml
<library-ref>
  <library-name>jsf</library-name>
  <specification-version>1.2</specification-version>
  <exact-match>true</exact-match>
</library-ref>
```

Additionally, to ensure an adequate pool of database connections, set the following properties:

- jbo.doconnectionpooling=true
- jbo.txn.disconnect_level=1
- jbo.ampool.doampooling=true

**Note:**

This library reference is added automatically if an ADF application is deployed from JDeveloper to a Cloud server, providing that the deployment profile is set to use “Oracle Cloud” as the platform, and as long as this library reference do not already exist in the weblogic.xml file.

If the application needs to be deployed to both on-premise environments as well as an Oracle Java Cloud Service - SaaS Extension instance, and if the on-premise environments use different role names and JDBC data source JNDI names, then use the WebLogic Server release 10.3.6 deployment plan feature when deploying the application to the on-premise environments. This approach will support the configuration differences between the on-premise environment and the Oracle Java Cloud Service - SaaS Extension instance.

To learn more about deployment descriptors, JDBC data sources, and deployment plans, see:

**Note:**

The following references pertain only to on-premises customers who need to use a deployment plan when their enterprise role names are different. Java Cloud Service - SaaS Extension **does not** support these deployment plans.

- How to Create and Edit Deployment Descriptors in *Oracle Fusion Middleware Fusion Developer’s Guide for Oracle Application Development Framework*
Guidelines for Applications That Use Java EE or ADF Application Security

If an application uses Java EE or ADF application security for securing part or all of its pages (either programmatically or through its deployment descriptors), you must configure or modify the application to refer to the appropriate application roles.

The appropriate application roles are described in Securing Java EE Applications – Roles and Constraints and Securing ADF Applications – Roles and Constraints.

The current release of Oracle Java Cloud Service - SaaS Extension uses the second generation of the Identity Management System in Oracle Cloud. Therefore, applications that are deployed to the Oracle Java Cloud Service - SaaS Extension no longer need to prefix the principal name with the `identity-domain-name` when defining enterprise role policies in ADF or Java EE applications.

Required Changes to ADF Applications Using Role-based Security

For currently deployed ADF applications that use role-based security, you no longer need to prefix the `principal-name` with the `identity-domain-name` when defining enterprise roles in the `jazn-data.xml` deployment descriptor.

For role-based security with deployed ADF applications, you must either:

- Modify the `jazn-data.xml` deployment descriptors by removing all `identity-domain-name` prefixing from the principal names.
- If you cannot modify your deployment descriptors, use the Security page to append the identity domain name to the enterprise role so that it exactly matches the principal name in the `jazn-data.xml` file.

The following example shows how the `jazn-data.xml` deployment descriptor was typically configured for role-based security in previous releases of Oracle Java Cloud Service - SaaS Extension. Note that the identity domain name `myidentitygroupfoo` prefixes the `eastcoastsales` principle name.

```
...  
  <app-role>  
    <name>customer</name>  
    <class>oracle.security.jps.service.policystore.ApplicationRole</class>  
    <members>  
      <member>  
        <name>myidentitygroupfoo.eastcoastsales</name>  
        <class>weblogic.security.principal.WLSUserImpl</class>  
      </member>  
    </members>  
  </app-role>  
...
```

The following example shows how the `jazn-data.xml` deployment descriptor should be configured for role-based security in the current release of Oracle Java Cloud Service - SaaS Extension.
Service - SaaS Extension. Note that the eastcoastsales principle name is no longer prefixed by an identity-domain-name.

```
...<app-role>
  <name>customer</name>
  <class>oracle.security.jps.service.policystore.ApplicationRole</class>
  <members>
    <member>
      <name>eastcoastsales</name>
      <class>weblogic.security.principal.WLSUserImpl</class>
    </member>
  </members>
...</app-role>
```

For more information, see Securing ADF Applications – Roles and Constraints.

### Required Changes to Java EE Applications Using Role-based Security

For currently deployed Java EE applications that use role-based security, you no longer need to prefix the principal-name with the identity-domain-name when defining enterprise roles in the weblogic.xml deployment descriptor.

To use role-based security with currently deployed Java EE applications, you must either:

- Modify the weblogic.xml deployment descriptors by removing all identity-domain-name prefixing from the principal names.
- If you cannot modify your deployment descriptors, use the Security page to append the identity domain name to the enterprise role so that it exactly matches the principal name in the weblogic.xml file.

The following example shows how the weblogic.xml deployment descriptor was typically configured for role-based security in previous releases of Oracle Java Cloud Service - SaaS Extension. Note that the identity domain name myidentitygroupfoo prefixes the WestCoastSales principle name.

```
...<wls:security-role-assignment>
  <wls:role-name>sales</wls:role-name>
  <wls:principal-name>myidentitygroupfoo.WestCoastSales</wls:principal-name>
</wls:security-role-assignment>
```

The following example shows how the weblogic.xml deployment descriptor should be configured for role-based security in the current release of Oracle Java Cloud Service - SaaS Extension. Note that the WestCoastSales principle name is no longer prefixed by an identity-domain-name.

```
...<wls:security-role-assignment>
  <wls:role-name>sales</wls:role-name>
  <wls:principal-name>WestCoastSales</wls:principal-name>
</wls:security-role-assignment>
```

For more information, see Securing Java EE Applications – Roles and Constraints.
Guidelines for Applications When Accessing System Properties

You can access most system properties from application code by using getters and setters.

Examples of the system properties you can access (get or set) by using application code are:

- Any custom properties. (That is, any properties not defined by WebLogic Server or Java EE.)
- The following HTTP proxy properties are getable:
  - http.proxyHost
  - http.proxyPort
  - https.proxyHost
  - https.proxyPort
- The following categories of system properties are not available to be either get or set. Attempting to do so will result in an access control exception.
  - Java system properties related to the Java Security Manager.
  - Java system properties related to WebLogic Server security.
  - Java system properties related to the JVM specification.

**Note:**
All system properties that are set using application code in Oracle Java Cloud Service - SaaS Extension are not persisted when the service is restarted. Therefore, it is the responsibility of the application setting and getting them to persist them. A security exception will be raised if the application is trying access a property that is not allowed.

Guidelines for Applications When Using Log4j Appenders

The Oracle Java Cloud Service - SaaS Extension supports Log4j so deployed applications can use packaged Log4j libraries to log their messages. There are no limitations on the Log4j appender that is configured for applications, provided those appenders do not violate any security requirements.

Oracle recommends the two most commonly-used appenders:

- **ConsoleAppender** – Logs that are written through this appender get redirected to the Oracle Java Cloud Service - SaaS Extension logs, which already contain JDK logs, along with certain WebLogic Server log messages. By using the ConsoleAppender, you can use the Oracle Java Cloud Service - SaaS Extension's CLI-based tools or the Oracle Java Cloud Service - SaaS Extension Control to query the service logs to find the Log4j logs as well.
- **FileAppender or any of its subclasses** – The log files need to be written to a unique volume that can be read and written to, such as `/customer/scratch/${weblogic.Name}/log4j.log`, so that different managed servers do not end up
using the same path. See Guidelines for Applications When Accessing a Local File System. The whitelist tool validates when the log4j.properties or log4j.xml configuration files are used to initialize Log4j. The whitelist will provide a warning if it cannot find the expected path for writing the logs.

This sample log4j.properties file shows the path where log files are written as /customer/scratch/${weblogic.Name}/log4j.log, which is inside the supported read/write area. It also shows how the dynamic portion ${weblogic.Name} ensures that each managed server finds a different path.

```java
# Copyright (c) 2011, 2013, Oracle and/or its affiliates. All rights reserved.
# Root logger option
log4j.rootLogger=ALL, file, out

# Direct log messages to a log file
log4j.appender.file=org.apache.log4j.RollingFileAppender
log4j.appender.file.file=/customer/scratch/${weblogic.Name}/log4j.log
log4j.appender.file.append=true
log4j.appender.file.layout=org.apache.log4j.PatternLayout
log4j.appender.file.layout.ConversionPattern=%d{yyyy-MM-dd} %-5p %c{1}:%L - %m%n

# Direct log messages to stdout
log4j.appender.out=org.apache.log4j.ConsoleAppender
log4j.appender.out.layout=org.apache.log4j.PatternLayout
log4j.appender.out.layout.ConversionPattern=%d{yyyy-MM-dd} %-5p %c{1}:%L - %m%n
```

**Note:**

Oracle Cloud does not affect Log4j’s mechanism for locating its own configuration files; therefore, the location of Log4j properties files should be by default found on the system CLASSPATH. Please refer to the Log4j documentation for configuration file information.

Guidelines for Applications When Accessing a Local File System

Oracle Java Cloud Service - SaaS Extension allows local file system access to deployed applications through standard java.io.File APIs. The root path on the local file system is /customer/scratch/, and under this directory deployed applications can freely read from and write to any necessary files, such as Log4j configuration files.

For guidelines, see Guidelines for Applications When Using Log4j Appenders.

You can get the value of your /customer/scratch/ directory by using the system property java.scratch.dir, as follows:

```java
System.getProperty("java.scratch.dir")
```

To facilitate portability, your on-premise environment can use a different path for the same java.scratch.dir property. (See Guidelines for Applications When Accessing System Properties.)

The /customer/scratch volume is shared among all the managed servers in an Oracle Java Cloud Service - SaaS Extension instance; therefore, these managed server can see the same physical file with the same path. Therefore, the application
code running from different managed servers can read the same file using the same path. However, if the application needs to write data to this volume, the application instances running in different managed servers need to find a unique path under the volume to write to.

To manage this, applications can read a dynamic \{weblogic.Name\} system property to get the name of the managed server and append this name with the volume to get a unique path for writing. For example, applications running on a managed server name myMS1, would be configured to write to a /customer/scratch/${weblogic.Name}/myMS1/ directory.

**Note:**
The Oracle Java Cloud Service - SaaS Extension SDK has a sample application named “File Browser” that can be used to manage all the files in your /customer/scratch/ directory. The sample application also shows how java.io.* APIs can be used to read and write files. See Accessing the Local File System.

---

Accessing Applications Deployed on Oracle Java Cloud Service - SaaS Extension

Once you deploy an application to an Oracle Java Cloud Service - SaaS Extension instance, users can access it using a web browser.

**Note:**
The process of accessing your applications deployed on Oracle Java Cloud Service - SaaS Extension has not changed from the first generation of the Identity Management system in previous releases of Oracle Cloud to the second generation of Identity Management in the current release.

All applications deployed to an Oracle Java Cloud Service - SaaS Extension can be accessed using the following URL:

https://servicename-identitydomain.java.dc.oraclecloudapps.com/context-path

Where:

- **servicename-identitydomain** is the name of the Oracle Java Cloud Service - SaaS Extension instance chosen at creation time, a hyphen, and the name of the Oracle Java Cloud Service - SaaS Extension instance identity domain.
- **dc** is the short name of the data center. For example, us1 refers to the US Commercial 1 data center.
- **context-path** is typically the application name.
For information about accessing and managing your applications using the Oracle Java Cloud Service - SaaS Extension Control, see Accessing Oracle Java Cloud Service - SaaS Extension.

Messaging Support in Oracle Java Cloud Service - SaaS Extension

This section describes how Oracle Java Cloud Service - SaaS Extension supports messaging.

Topics

• Using JMS in Oracle Java Cloud Service - SaaS Extension

Using JMS in Oracle Java Cloud Service - SaaS Extension

When an Oracle Java Cloud Service - SaaS Extension instance is deployed, WebLogic JMS servers and destinations are created by default.

When the instance is deployed, one set of a connection factory, queue, and topic are created on each managed server. This allows client applications to use WebLogic JMS out-of-the-box, without having to configure it. Applications can look-up and use the following JNDI names in their applications:

• local-connection-factory
• local-queue
• local-topic

Note that you cannot change or update the local JMS configuration.

The Oracle Java Cloud Service - SaaS Extension SDK includes a Maven plug-in project that allows you to deploy message-driven beans (MDBs) samples on your Oracle Java Cloud Service - SaaS Extension instance. The deployed MDBs will listen on the local destinations (queue and topic) that are automatically created as part of the local JMS configuration. The connection factory is registered in JNDI as local-connection-factory while the queue and topic are registered as local-queue and local-topic, respectively. For more information, navigate to theSDK_HOME/samples/maven/messagedrivenbean directory (where SDK_HOME is the directory containing your Oracle Java Cloud installation) and open mdb-instructions.html in your web browser.

Developing RESTful Web Services

You can develop RESTful Web Service by using techniques supported by Oracle Java Cloud Service - SaaS Extension.

REST describes any simple interface that transmits data over a standardized interface (such as HTTP) without an additional messaging layer, such as Simple Object Access Protocol (SOAP). REST provides a set of design rules for creating stateless services that are viewed as resources, or sources of specific information, and can be identified by their unique URIs. A client accesses the resource using the URI, a standardized fixed set of methods, and a representation of the resource is returned. The client is said to transfer state with each new resource representation.
Oracle Java Cloud Service - SaaS Extension supports the following methods to enable the development of RESTful web services:

- Reference and use the pre-built shared library, Jersey JAX-RS RI Version 1.9, delivered with Oracle Java Cloud Service - SaaS Extension, that is required to run Jersey JAX-RS Reference Implementation (RI).

- You can build and deploy a more recent version of the Jersey JAX-RS RI shared libraries. Just package the recent version you want to use with your application archive.

Using the Jersey JAX-RS Reference Implementation

Oracle Java Cloud Service - SaaS Extension ships with a pre-built shared library, Jersey JAX-RS RI Version 1.9, packaged as a web application, that is required to run applications that are based on the Jersey JAX-RS RI.

Topics

- Summary of the Jersey JAX-RS RI Shared Library
- Using the Jersey JAX-RS RI Shared Library
- Configuring the Web Application to Use the Jersey JAX-RS RI
- Creating JAX-RS Web Services and Clients

Summary of the Jersey JAX-RS RI Shared Library

The Jersey JAX-RS RI shared library is pre-deployed for your convenience, so you only need to reference it.

The following table describes the pre-built shared library that supports Jersey JAX-RS RI Version 1.9 web services.

<table>
<thead>
<tr>
<th>Functionality</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jersey</td>
<td>Shared Library Name: jax-rs</td>
</tr>
<tr>
<td>JAX-RS API</td>
<td>JAR Filename: jersey-bundle-1.9.jar</td>
</tr>
<tr>
<td>JSON processing and streaming</td>
<td>WAR Filename: jersey-bundle-1.9.war</td>
</tr>
<tr>
<td>ATOM processing</td>
<td>Version: 1.9</td>
</tr>
<tr>
<td></td>
<td>License: SUN CDDL+GPL</td>
</tr>
</tbody>
</table>

Using the Jersey JAX-RS RI Shared Library

The Jersey JAX-RS RI shared library is pre-deployed for your convenience; using it is a two-step process.

To use the Jersey JAX-RS RI:

As required, you can build and deploy a more recent version of the Jersey JAX-RS RI shared libraries. Just package the library with your application archive.

1. Configure the application that contains the RESTful web service to use the Jersey JAX-RS RI shared libraries. See Configuring the Web Application to Use the Jersey JAX-RS RI.
2. Create the JAX-RS web services and clients. See Creating JAX-RS Web Services and Clients.

Configuring the Web Application to Use the Jersey JAX-RS RI

You need to configure the web application that contains the RESTful web services to use the Jersey shared libraries.

To configure the Web Application to use the Jersey JAX-RS RI, you need to update the following two deployment descriptor files that are associated with your application:

- **web.xml**—Update to delegate web requests to the Jersey servlet. See Updating web.xml to Delegate Web Requests to the Jersey Servlet.
- **weblogic.xml**—Update to reference the shared library that is required by your application. See Summary of the Jersey JAX-RS RI Shared Library and Updating weblogic.xml to Reference the Shared Libraries.

Updating web.xml to Delegate Web Requests to the Jersey Servlet

Update the web.xml file to delegate all web requests to the Jersey Servlet, com.sun.jersey.spi.container.servlet.ServletContainer. The web.xml file is located in the WEB-INF directory in the root directory of your application archive.

The following provides an example of how to update the web.xml file:

```xml
<web-app>
  <servlet>
    <display-name>My Jersey Application</display-name>
    <servlet-name>MyJerseyApp</servlet-name>
    <servlet-class>com.sun.jersey.spi.container.servlet.ServletContainer</servlet-class>
    <init-param>
      <param-name>javax.ws.rs.Application</param-name>
      <param-value>myPackage.myJerseyApplication</param-value>
    </init-param>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyJerseyApp</servlet-name>
    <url-pattern>/*</url-pattern>
  </servlet-mapping>
</web-app>
```

As shown in the previous example, you need to define the following elements:

- **<servlet-class>** element defines the servlet that is the entry point into the Jersey JAX-RS RI. This value should always be set to com.sun.jersey.spi.container.servlet.ServletContainer.

- **<init-param>** element defines the class that extends the javax.ws.rs.Application.

- **<servlet-mapping>** element defines the base URL pattern that gets mapped to the MyJerseyApp servlet. The portion of the URL after the http://<host>[:<port>] +<webAppName> is compared to the <url-pattern> by Oracle Java Cloud Service - SaaS Extension. If the patterns match, the servlet mapped in this element will be called.
For more information about the `web.xml` deployment descriptor, see `web.xml` Deployment Descriptor Elements in *Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server*.

**Updating web.xml to Set Authentication**

HTTP Basic Authentication forces the server to request a user name and password from the web client and then verify that the user name and password are valid by comparing them against a database of authorized users. You can set basic authentication in `web.xml` as shown here:

```xml
<login-config>
    <auth-method>BASIC</auth-method>
</login-config>
```

When you use basic authentication, passwords are not protected, which means that passwords sent between a client and a server on an unprotected session can be viewed and intercepted by third parties. If you want to prevent hijacking of data when BASIC is your chosen authentication method, use a *user data constraint*. A user data constraint (`<user-data-constraint>` in the deployment descriptor) forces all URL patterns and HTTP methods specified in the security constraint to be received over a protected connection, such as HTTPS. A user data constraint specifies a transport guarantee (`<transport-guarantee>` in the deployment descriptor). The choices for transport guarantee include CONFIDENTIAL, INTEGRAL, or NONE. If you specify CONFIDENTIAL or INTEGRAL as a security constraint, that type of security constraint applies to all requests that match the URL patterns in the web resource collection and not just to the login dialog box. If you don't want a user data constraint, you can set it to NONE, as in this example:

```xml
<security-constraint>
    <display-name>index</display-name>
    <web-resource-collection>
        <web-resource-name>index</web-resource-name>
        <url-pattern>/index.jsp</url-pattern>
    </web-resource-collection>
    <user-data-constraint>
        <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
</security-constraint>
```

**Updating weblogic.xml to Reference the Shared Libraries**

Update the `weblogic.xml` file to reference the shared library that is required by your application. The `weblogic.xml` file is located in the WEB-INF directory in the root directory of your application archive.

The `<exact-match>` directive enables you to control whether the latest version of the deployed shared library will be used. In Oracle Java Cloud Service - SaaS Extension, only implementation version 1.9 is deployed as shared library. You can set the `<exact-match>` element in the following ways:

- You can skip adding this element all together. In this case, do not add the `<specification-version>` as well. The service runtime will pick the latest implementation version, which is 1.9.
- You can add the element and set it to `false`. The service runtime will pick the latest version deployed to the Oracle Java Cloud Service - SaaS Extension,
regardless of what is specified in the <specification-version> of the weblogic.xml file.

- You can add the element and set it to true. In this case, you have to set the <specification-version> to 1.9, otherwise deployment will fail.

The following example shows how to update the weblogic.xml file to use the Jersey JAX-RS RI Version 1.9.

```xml
<library-ref>
  <library-name>jax-rs</library-name>
  <specification-version>1.1</specification-version>
  <implementation-version>1.9</implementation-version>
  <exact-match>false</exact-match>
</library-ref>
```

For more information about the weblogic.xml deployment descriptor, see weblogic.xml Deployment Descriptor Elements in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.

Creating JAX-RS Web Services and Clients

After you have configured your web application, you can start creating JAX-RS web services and clients.

The following sections show a simple web service and client.

**A Simple RESTful Web Service**

The following provides a very simple example of a RESTful web service:

```java
package samples.helloworld;
import javax.ws.rs.GET;
import javax.ws.rs.Path;
import javax.ws.rs.Produces;
// Specifies the path to the RESTful service
@Path("/helloworld")
public class helloworld {

  // Specifies that the method processes HTTP GET requests
  @GET
  @Path("sayHello")
  @Produces("text/plain")
  public String sayHello() {
    return "Hello World!";
  }
}
```

**A Simple RESTful Client**

The following example provides a simple RESTful client that demonstrates basic authorization, adds a header and query parameters. This sample uses classes that are provided by the Jersey JAX-RS RI specifically; they are not part of the JAX-RS standard.

```java
import com.sun.jersey.api.client.Client;
import com.sun.jersey.api.client.ClientResponse;
import com.sun.jersey.api.client.WebResource;
import com.sun.jersey.api.client.filter.HTTPBasicAuthFilter;
```
import com.sun.jersey.api.client.filter.LoggingFilter;

public static void main(String s[]) throws Exception {
    Client client = Client.create();
    WebResource resource = client.resource("https://javas2-
jcscdc.java.us2.oraclecloudapps.com/secapp/");
    // Adds query parameter
    resource = resource.queryParam("param1", "value");

    // Handles Basic Authentication
    client.addFilter(new HTTPBasicAuthFilter("user", "pwd");

    // Logs request and response
    // This is for debugging only; do not use it
    // in the production instance.
    client.addFilter(new LoggingFilter());
    Response response = resource.header("X-custom-
header","value").get(Response.class);
    System.out.println(response.getStatus());
}

Securing Applications in Oracle Java Cloud Service - SaaS Extension

All Java EE and ADF web applications deployed to an Oracle Java Cloud Service -
SaaS Extension instance are automatically secured because only users who have
been authenticated through SSO can access a deployed application.

Tutorial

Topics:

- Securing Java EE and ADF Applications – Authentication
- Securing Java EE Applications – Roles and Constraints
- Securing ADF Applications – Roles and Constraints
- Configuring JPS Policy Migration Settings

The default authentication includes users from any identity domain. To provide finer-grained secure access to your Java EE or ADF applications, you can specify role-based authentication that can vary from being publicly accessible to restricted to only users within the same identity domain.

This section describes how to specify Java EE and ADF application roles and security constraints within the Oracle Java Cloud Service - SaaS Extension instance’s identity domain.
Securing Java EE and ADF Applications – Authentication

This section describes the levels of secure authentication for your Java EE and ADF applications.

Topics:

- Internet Public Pages
- Oracle Public Pages
- Tenant Restricted Pages
- Securing JAX-WS Web Services

Internet Public Pages

Pages that anyone on the internet can access are referred to as internet public; for example, www.oracle.com/index.html. A user is not required to login to access such pages.

To configure your application to be in internet public mode, it requires an empty security element called <login-config/> in the web.xml deployment descriptor, as shown in this example:

```xml
  ...
  <login-config/>
  ...
</web-app>
```

Oracle Public Pages

Pages that only valid Oracle Cloud users can access are referred to as Oracle public. Any user that can log into Oracle Cloud can access these pages.

Oracle public mode prevents you from accidentally making your applications internet public pages. Note that this mode is different from internet public because a user has to be authenticated to access Oracle Cloud, while internet public pages can be accessed without any login. This is the default access mode. However, it is not the Oracle recommended mode of securing your pages. Instead, Oracle strongly recommends explicitly setting your choice of authentication mode from the options discussed in Tenant Restricted Pages.

The absence of a <login-config/> element in the web.xml deployment descriptor configures the application in the default mode. Another key difference is that the application code is always accessed as an anonymous user. The authenticated user is not passed to the application; instead, the application is made to believe that the user is anonymous.
Tenant Restricted Pages

Pages that can only be accessed by users within a tenant's identity domain are referred as tenant restricted.

Oracle recommends using this mode when you want to protect your application from unauthorized use. To protect your application in this mode, you need to add a `<login-config>` security element in the `web.xml` deployment descriptor. There are three modes of authentication that you can use:

- **CLIENT-CERT** – Oracle's recommended mode of authentication, it enables the tenant-specific SSO authentication mode for an application. Any user accessing pages secured under this mode will be prompted to login to Oracle Cloud, if the user has not already done so in the current browser session. The login will persist to any other application the user navigates to within the same tenant. See Migrating Applications from FORM or BASIC Authentication Mode to CLIENT-CERT Mode.

- **BASIC** – Enables the HTTP BASIC mode of authentication.

- **FORM** – Enables the HTTP FORM mode of authentication.

Here's an example of using the `<login-config>` security element in a web application:

```xml
  ...
  <login-config>
    <auth-method>CLIENT-CERT</auth-method>
    <realm-name>default</realm-name>
  </login-config>
  ...
</web-app>
```

**Important!** Users should also add the `<security-constraint>` element to specify what part of the application is protected. Without this element, the application will be internet public when using the FORM or BASIC mode, and Oracle public when using the CLIENT-CERT mode. Oracle strongly recommends adding the `<security-constraint>` element, as shown in this example:

```xml
  ...
  <security-constraint>
    <display-name>name</display-name>
    <web-resource-collection>
      <web-resource-name>name</web-resource-name>
      <url-pattern>/*</url-pattern>
    </web-resource-collection>
  </security-constraint>
  ...
</web-app>
```
To fully log off, embed either of the following two patterns in your web applications. When your users click these links, their complete SSO session will be terminated; that is, not just the user applications, but also artifacts like MyServices.

- In this example, the final page is delivered by Oracle after logout:

  `<h3>1. Just log out</h3>
  <a href="/oamsso/logout.html">logout</a>`

- In this example, `thankyou.jsp` is inside the application to which the request will be redirected upon successful termination of SSO.

  `<h3>2. Log me out and take me to the given page</h3>
  <a href="/oamsso/logout.html?end_url=thankyou.jsp">logout</a>`

**Note:**

If `thankyou.jsp` is a secured page, a fresh SSL challenge is thrown after the previous session is terminated. If it is an unsecured page, the page content is displayed.

## Securing JAX-WS Web Services

Certain Oracle Web Service Manager (OWSM) security policies allow you to secure your JAX-WS web services.

### Supported OWSM Policies

**Note:**

Oracle Java Cloud Service - SaaS Extension only supports SSL-based policies.

**NOT_SUPPORTED:**

OWSM policies are not supported by third-party web service libraries.

You can use OWSM Security policies to protect WebLogic Server JAX-WS Web services and Web service clients. Oracle Java Cloud Service - SaaS Extension supports a limited number of these policies, which are listed here:
### Client Policy

<table>
<thead>
<tr>
<th>Client Policy</th>
<th>Service Policy</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oracle/wss_username_token_over_ssl_client_policy</td>
<td>oracle/wss_username_token_over_ssl_service_policy</td>
<td>This policy uses the credentials in the UsernameToken WS-Security SOAP header to authenticate users against the configured identity store. Both plain text and digest mechanisms are supported. The policy verifies that the transport protocol provides SSL message protection. This policy can be attached to any SOAP-based endpoint.</td>
</tr>
<tr>
<td>oracle/wss_saml20_token_bearer_over_ssl_client_policy</td>
<td>oracle/wss_saml20_token_bearer_over_ssl_service_policy</td>
<td>This policy authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header. The credentials in the SAML token are authenticated against a SAML login module. The policy verifies that the transport protocol provides SSL message protection. This policy can be applied to any SOAP-based endpoint.</td>
</tr>
<tr>
<td>oracle/wss_saml_token_over_ssl_client_policy</td>
<td>oracle/wss_saml_token_over_ssl_service_policy</td>
<td>This policy authenticates users using credentials provided in SAML tokens in the WS-Security SOAP header. The credentials in the SAML token are authenticated against a SAML login module. The policy verifies that the transport protocol provides SSL message protection. This policy can be applied to any SOAP-based endpoint.</td>
</tr>
</tbody>
</table>

When building a secure web service, these policies can be attached to the JAX-WS web service code in the following way (this example uses the policy oracle/wss_username_token_over_ssl_client_policy):

```java
import weblogic.wsee.jws.jaxws.owsm.SecurityPolicy;
@WebService
@SecurityPolicy(uri = "oracle/wss_username_token_over_ssl_service_policy")
public class HelloWorld { public HelloWorld() {
```

In Oracle Java Cloud Service - SaaS Extension, the default security posture is "Secured by Default" so any web application, including a SOAP or REST web service application, is secured upon deployment. This also means any web application will have Single Sign-On (SSO) security enabled by default unless you specify otherwise in the web.xml deployment descriptor. See [Updating the web.xml Deployment Descriptor](#).
In order for "non-browser" web services clients to talk to a web service that is deployed in Oracle Java Cloud Service - SaaS Extension, the web service end point and the WSDL must be made available to the public internet. See Internet Public Pages.

Securing Java EE Applications – Roles and Constraints

When securing a Java EE web application, you can specify application roles and security constraints within the application deployment descriptors or the application code.

Topics:
• Updating the web.xml Deployment Descriptor
• Updating the weblogic.xml Deployment Descriptor
• Special Considerations When Accessing Secured Oracle Cloud Pages

Application roles are mapped to enterprise roles defined within the Oracle Java Cloud Service - SaaS Extension's identity domain using the Security page. Implicit mapping is based on the role name. See Managing Uses and Roles in Getting Started with Oracle Cloud.

Updating the web.xml Deployment Descriptor

Applications targeted to an Oracle Java Cloud Service - SaaS Extension instance can choose to participate in a Single Sign-On (SSO) with other applications deployed to services within the same identity domain.

In order to enable SSO participation, applications must use a CLIENT-CERT authentication method as specified through their deployment descriptor's <auth-method> element and illustrated through the following web.xml deployment descriptor snippet:

```xml
<login-config>
  <auth-method>CLIENT-CERT</auth-method>
</login-config>
<security-role>
  <role-name>sales</role-name>
</security-role>
```

Applications using a BASIC or FORMS based authentication can also be deployed to an Oracle Java Cloud Service - SaaS Extension instance. However, such applications will not participate in SSO. Instead, their authentication will be local to the application.

All deployed applications without any explicit security elements in the web.xml file are set with default protection that allows anonymous access to any Oracle Cloud user. To prevent undesired access to your applications, you must set a proper user authentication method in the web.xml file. See Securing Java EE and ADF Applications – Authentication.

The following are supported authentication configurations:
Fully Secured Application
This method is highly recommended. This configuration allows choosing which portion of the application is protected by SSO. In the web.xml file the auth-method element must be set to the value CLIENT-CERT as noted in this example:

```
<login-config>
  <auth-method>CLIENT-CERT</auth-method>
  <realm-name>default</realm-name>
</login-config>
```

You must also define the section of the application that needs to be protected by SSO. In the following configuration, all the URL patterns for the application are protected:

```
<security-constraint>
  <display-name>name</display-name>
  <web-resource-collection>
    <web-resource-name>name</web-resource-name>
    <url-pattern>/*</url-pattern>
  </web-resource-collection>
</security-constraint>
```

Only the URL patterns covered by a security constraint in a web-resource-collection element will prevent users from different identity domains, and external non-authenticated users from accessing the application. All the directories that are not specified as a URL pattern will become internet public. Multiple security-constraint elements are allowed in web.xml.

> Note:
Specific application security configuration for SSO is highly recommended to enhance the security of your applications and prevent unwanted user access.

Internet Public Application
An application that requires complete public access without any login challenge needs to include an empty `<login-config/>` element in web.xml.

Partially Secured Application
A partially secured application is one that has login-config and auth-method specified but the security collection elements do not cover all the sections of the application. This means the portions of the URL patterns that are not covered by any security-collection element are public. Here is an example of a partially secured application:

```
<security-constraint>
  <display-name>My-Constraint-0</display-name>
  <web-resource-collection>
    <web-resource-name>My-Constraint-0</web-resource-name>
    <url-pattern>/westsalesmgrs/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>sales</role-name>
  </auth-constraint>
</security-constraint>
```
Note that all pages under /westsalesmgrs require authentication and the correct privileges. All pages under /secured only require authentication. All other pages become internet public.

**Updating the weblogic.xml Deployment Descriptor**

ADF applications that use role-based security no longer need to prefix the `principal-name` with the `identity-domain-name` when defining enterprise roles in the `weblogic.xml` deployment descriptor.

This sample `weblogic.xml` file snippet assumes that application roles have been defined through the Security page and mapped to the enterprise role.

```
...<wls:security-role-assignment>
  <wls:role-name>sales</wls:role-name>
  <wls:principal-name>WestCoastSales</wls:principal-name>
</wls:security-role-assignment>
...
```

**Tip for Migrating Java EE Applications:**

For Java EE applications deployed to a previous release of Oracle Java Cloud Service - SaaS Extension, if you are unable to edit the deployment descriptors to remove the `identity-domain-name` prefix from the `principal-name`, then use the Security page to append the identity domain name to the enterprise role so that it exactly matches the principal name in the `jazn-data.xml` file.

Additionally, all applications participating in SSO must have a unique value specified for `cookie-path` in `weblogic.xml`, as follows:

```
<session-descriptor>
  <cookie-path>myapp</cookie-path>
</session-descriptor>
```

**Special Considerations When Accessing Secured Oracle Cloud Pages**

An unexpected Oracle Cloud page access limitation could occur when using the `<auth-constraint>` element in the `web.xml` deployment descriptor to protect a page with role-based access control.
In certain situations, if a user navigates from a page that is public (that is, no authentication was required to reach the page), to a page that is protected with role-based access control using `<auth-constraint>` in the `web.xml` file, then the user may encounter a "403 Forbidden" HTTP status code.

**How It Occurs**

For example, if the user had already authenticated with Oracle Cloud in the same browser session, the user's identity is active inside the session context and so will be used by Oracle Java Cloud Service - SaaS Extension to authorize access to the protected page. If the user has the appropriate privileges, the user will be able to access the protected page. However, if the user only accessed the public page and was never authenticated before navigating to the protected page, Oracle Java Cloud Service - SaaS Extension will not automatically prompt the user for authentication. Instead, Oracle Java Cloud Service - SaaS Extension expects the user's authenticated identity to be active in the session context when navigating to the protected page. In absence of that identity, the user will encounter the 403 Forbidden error.

This `web.xml` snippet illustrates how unexpected page access behavior occurs:

```xml
<security-constraint>
    <display-name>My-Constraint-0</display-name>
    <web-resource-collection>
        <web-resource-name>My-Constraint-0</web-resource-name>
        <url-pattern>/westsalesmgrs/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>sales</role-name>
    </auth-constraint>
</security-constraint>

<login-config>
    <auth-method>CLIENT-CERT</auth-method>
    <realm-name>default</realm-name>
</login-config>
```

In this example, all pages under the `/westsalesmgrs` sub-context are protected with the `sales` role. The authentication method is `CLIENT-CERT`, which means Single Sign-On. All other pages are unprotected and publicly accessible to anyone; therefore, users will not be asked to log in. In which case, if a user accesses the public welcome page, then clicks a link that directs the user to a page inside `/westsalesmgrs`, the user will encounter a 403 error. This is because the user was not asked to login in when accessing the public welcome page. On navigating to the protected page, Oracle Java Cloud Service - SaaS Extension expected the user to already be logged in, but since that was not the case, Oracle Java Cloud Service - SaaS Extension returned a 403 error.

**Typical Solution**

A typical solution is first redirecting users from the public page to an intermediate page that only requires a user to be authenticated. This intermediate page should not be protected with `<auth-constraint>` in `web.xml`. In other words, this intermediate page should only be accessible to any valid user in the tenant's identity domain. Once successfully logged in, the user can be redirected to the page protected with `<auth-constraint>` in `web.xml` (that is, a page protected with role-based access control). The intermediate page will force user to provide a valid user name and password. A successful login will insert the user's identity in the current session along with all the associated roles. Upon a redirect to the protected page, Oracle Java Cloud Service -
SaaS Extension will enforce the access control rules of that page by verifying that the current user has the right privileges to access the protected page.

This web.xml snippet illustrates how an intermediated page can prevent unexpected page access behavior from occurring:

```xml
<security-constraint>
    <display-name>My-Constraint-0</display-name>
    <web-resource-collection>
        <web-resource-name>My-Constraint-0</web-resource-name>
        <url-pattern>/westsalesmgrs/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
        <role-name>sales</role-name>
    </auth-constraint>
</security-constraint>

<security-constraint>
    <display-name>My-Constraint-1</display-name>
    <web-resource-collection>
        <web-resource-name>My-Constraint-1</web-resource-name>
        <url-pattern>/protected/*</url-pattern>
    </web-resource-collection>
</security-constraint>

<login-config>
    <auth-method>CLIENT-CERT</auth-method>
    <realm-name>default</realm-name>
</login-config>
```

In this example, note that the `<security-constraint>` for the intermediate page is under `/protected`, and that the main welcome page needs to redirect the user to a page under `/protected`. That page in turn, can redirect the user to any other page protected with role-based access control. Since there is no `<auth-constraint>` element under the second `<security-constraint>` for `/protected/*`, access to any page under that sub-context will force user to login using SSO. Once the user is logged in, the identity of the user is stored in the session context. Therefore, if the user gets redirected to protected pages under `/westsalesmgrs`, the user's identity is now known to Oracle Java Cloud Service - SaaS Extension. If the user belongs to the sales role, the user will be allowed access to the page. If not, the user will encounter a 403 Forbidden error page.

**Migrating Applications from FORM or BASIC Authentication Mode to CLIENT-CERT Mode**

The special consideration discussed in this section will most likely be observed when you migrate an application that uses the FORM or BASIC authentication modes to CLIENT-CERT. With BASIC or FORM, a switch from a public page to a protected one with role-based access control results in a prompt for the user to login if the user has not already logged in during the same browser session. Therefore, the unexpected page access behavior is not observed with FORM or BASIC authentication mode.

However, if you migrate your application to use Oracle Cloud's SSO capabilities, when using the CLIENT-CERT mode any redirect from a public unauthenticated page to a page protected with role-based access control will result in the unexpected page access behavior. That is, the user will not be prompted to login, but instead will immediately encounter a 403 Forbidden HTTP Code.
Securing ADF Applications – Roles and Constraints

When securing a Java ADF application, you can specify application roles and security permissions within ADF application's `jazn-data.xml` file.

Application roles are mapped to enterprise roles defined within the Oracle Java Cloud Service - SaaS Extension's identity domain. Implicit mapping is based on the role name. To learn more, see Managing Uses and Roles in *Getting Started with Oracle Cloud*.

Updating the jazn-data.xml File

In the current release of Oracle Java Cloud Service - SaaS Extension, ADF applications that use role-based security no longer need to prefix the `principal-name` with the `identity-domain-name` when defining enterprise roles in the `jazn-data.xml` deployment descriptor.

This sample `jazn-data.xml` file snippet assumes that application roles have been defined through the Security page and mapped to the enterprise role. Note that the names of all identity domains, users, and roles must be spelled in lowercase in the `jazn-data.xml` file...

```
...<app-role>
  <name>customer</name>
  <class>oracle.security.jps.service.policystore.ApplicationRole</class>
  <members>
    <member>
      <name>westcoastsales</name>
      <class>weblogic.security.principal.WLSUserImpl</class>
    </member>
  </members>
</app-role>
...```

Tip for Migrating ADF Applications:

For ADF applications deployed to a previous release of Oracle Java Cloud Service - SaaS Extension, if you are unable to edit the deployment descriptors to remove the `identity-domain-name` prefix from the `principal-name`, then use the Security page to append the identity domain name to the enterprise role so that it exactly matches the principal name in the `jazn-data.xml` file.

Configuring JPS Policy Migration Settings

The JPS policy migration parameter in the `META-INF/weblogic-application.xml` file specifies whether the migration should take place, and, when it does, whether it should merge with or overwrite matching security policies present in the target policy store.

Oracle Java Cloud Service - SaaS Extension currently only supports the `MERGE` value while restricting the use of the `OFF` and `OVERWRITE` values. As a result, a deployed
application cannot be updated to change its security policies because the existing policies cannot be overwritten. That is, any policies that were part of the original application deployment operation will still be attached to the deployed application.

Therefore, if a set of policies needs to be changed for a deployed application, instead of updating the application, you must first undeploy the application, then you can reinstall the application along with the new set of policies. You can then use MERGE to ensure the policies are seeded only once.

This XML snippet shows an example of the correct MERGE value usage for the `jps.credstore.migration` and `jps.policystore.migration` parameters in a `weblogic-application.xml` file:

```xml
<application-param>
    <param-name>jps.credstore.migration</param-name>
    <param-value>MERGE</param-value>
</application-param>

<application-param>
    <param-name>jps.policystore.migration</param-name>
    <param-value>MERGE</param-value>
</application-param>
```

Creating an On-premises WebLogic Server Environment

An on-premises environment is a local WebLogic Server /Java EE environment that is comparable to an Oracle Java Cloud Service - SaaS Extension instance. It is useful for both developing and troubleshooting applications deployed to Oracle Java Cloud Service - SaaS Extension.

**Tip:**

Review the topics in Preparing Applications for Oracle Java Cloud Service - SaaS Extension Deployment to verify whether your existing on-premises applications need to be updated to utilize the latest Oracle Java Cloud Service - SaaS Extension features. For example, there are Guidelines for Applications When Accessing a Local File System and Guidelines for Applications When Using Log4j Appenders.

To create an on-premises WebLogic Server environment, do the following:

1. Install WebLogic Server 10.3.6. No other version of WebLogic Server is supported as an on-premises environment for Oracle Java Cloud Service - SaaS Extension.

2. Create a domain as follows:
   - If the application you are deploying is not an ADF application and does not use any web services security (using OWSM), use the plain `wls.jar` WebLogic Server domain configuration template.
   - If the application you are deploying is using ADF or web services (by either exposing or invoking them) that must be protected through OWSM security policies, use the plain `wls.jar` as well as the JRF and OWSM domain configuration templates.
3. Deploy the deployment archives listed in the following table as shared libraries to the domain. Note that \texttt{MW\_HOME} refers to the Middleware Home directory you used when you installed WebLogic Server.

\textbf{Tip:}
During deployment, you must use the exact deployment names specified in this table.

<table>
<thead>
<tr>
<th>Application Type</th>
<th>Deployment Archive Path</th>
<th>Deployment Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>Uses JAX-\texttt{RS 1.9 REST interfaces}</td>
<td>\texttt{MW_HOME/wlserver_10.3/common/deployable-libraries/jersey-bundle-1.9.war}</td>
<td>\texttt{jax-rs}</td>
</tr>
<tr>
<td>Uses \texttt{JSF 2.0 for web application components}</td>
<td>\texttt{MW_HOME/wlserver_10.3/common/deployable-libraries/jsf-2.0.war}</td>
<td>\texttt{jsf}</td>
</tr>
</tbody>
</table>

4. Create a single XA enabled JDBC data source using the Oracle JDBC Thin driver connected to an on-premises Oracle Database 11g release 1 (11.1) data source. Give the data source the same name as the Database Cloud Service instance associated with your target Oracle Java Cloud Service - SaaS Extension.

To learn more about JDBC data sources, see How to Create a JDBC Data Source for Oracle WebLogic Server in \textit{Oracle Fusion Middleware Administrator's Guide for Oracle Application Development Framework}.

\textbf{Moving an Application between an on-premises Environment and an Oracle Java Cloud Service - SaaS Extension Instance}

To move an application (represented by a set of EAR/WAR files) between an on-premises environment and Oracle Java Cloud Service - SaaS Extension instances (or vice-versa), you must ensure that their database and Identity Management user repository content are also moved appropriately.

\textbf{Moving Data from an on-premises Environment to Database Cloud Service Instances}

To move database data from an on-premises environment to Database Cloud Service instances associated with Oracle Java Cloud Service - SaaS Extension instances:
1. Ensure that your schema tables are created within the target Database Cloud Service instance. To do this you can use either the database service’s SQL Workshop interface (see the Using SQL Workshop Data Upload Utility section in Using Oracle Database Cloud Service) or SQL Developer (see the Using SQL Developer for Data Loading section in Using Oracle Database Cloud Service).

When using EclipseLink JPA, you can also have the schemas created upon application deployment by using the following snippet within your application’s persistence.xml descriptor:

```xml
<property name="eclipselink.ddl-generation" value="create-tables"/>
```

2. Make sure that the data source name in the persistence.xml is configured to match the Database Cloud Service instance name that is associated with your Oracle Java Cloud Service - SaaS Extension instance.

3. Import your bulk data from on-premises to the target Database Cloud Service instance by using the SQL Workshop Data Upload Utility.

Moving Data from Database Cloud Service Instances Associated with Oracle Java Cloud Service - SaaS Extension Instances to on-premises Schemas

To import your bulk data from on-premises to the target Database Cloud Service instance, see Exporting Data in Using Oracle Database Cloud Service.

Moving Repository Data from on-premises User Repositories to the Identity Domain Associated with an Oracle Java Cloud Service - SaaS Extension Instance

Export data from your on-premises identity repository into a single file in CSV format (for instructions, see Managing Users and Roles in Using Oracle Database Cloud Service).
If you are running an Oracle Software as a Service (SaaS) application, for example, Oracle Sales Cloud, you can write extensions to that service and deploy them on Oracle Java Cloud Service - SaaS Extension. The topics in this section provide necessary background for associating an Oracle SaaS application with Java Cloud Service - SaaS Extension.

Topics

• Prerequisites and Restrictions
• The Benefits of Integration
• Understanding Identity Propagation
• Security Policy Use Cases
  – Writing a Client That Can Access an Oracle Sales Cloud Application
  – Writing a Web Service that an Oracle Sales Cloud Application Can Access
• Creating a Fusion Application User List
• PaaS-SaaS Association Sample Applications

Prerequisites and Restrictions for Association Between Services

Associating services such as Oracle Sales Cloud and Oracle Java Cloud Service - SaaS Extension enable single sign-on between them, thus allowing one service to act as the identity provider for both. There are certain prerequisites and restrictions that govern association.

What is Association?

Association is the process of enabling authentication across an Oracle Application Cloud Services, in this case, Oracle Sales Cloud, and Oracle Java Cloud Service - SaaS Extension (or other Platform as a Service application). Association is necessary if you want to integrate your Sales Cloud application with Oracle Java Cloud Services - SaaS Extension and unify the authentication mechanisms by enabling single sign-on (SSO) across the two services. Association is automatic when the services are provisioned in the same identity domain. Although association is required for SSO, they are not inclusive: two services can be associated but still not have SSO set up.

By enabling single sign-on across multiple services, users and applications are not required to sign-on each time they change the application context. Also, the application or web services that you develop in Java Cloud Services - SaaS Extension will be able to switch context from one environment to the other without having to provide credentials each time a switch occurs.
Additionally, association enables Security Assertion Markup Language (SAML)-based identity propagation for Oracle Sales Cloud and Java Cloud Service - SaaS Extension web service interactions. For example, if you are logged into Oracle Sales Cloud and invoke a web service running on Java Cloud Service - SaaS Extension, you can use SAML-based security policies that will automatically use the current logged-in user in Oracle Sales Cloud to invoke the web service in Java Cloud Service - SaaS Extension. Similar behavior can be achieved when Java Cloud Service - SaaS Extension invokes Oracle Sales Cloud web services. Associated services have this SAML trust pre-established by Oracle.

What are the Prerequisites?

Customers who already have Oracle Sales Cloud Services and would like to purchase Oracle Java Cloud Services - SaaS Extension can enable SaaS – PaaS association when Java Cloud Services - SaaS Extension is being provisioned. Customers who plan to purchase Oracle Sales Cloud and Java Cloud Services - SaaS Extension services together newly can also enable the association. Currently the SaaS - PaaS association can be enabled only when both the service instances of the tenant are provisioned in the same identity domain.

Association between services is required before you can enable SSO.

Single Sign-On requires user accounts to be synchronized. The user synchronization is a manual procedure. To do this, you must export user accounts from your Oracle Sales Cloud application and then import them into your Oracle Cloud identity domain. You must re-import accounts whenever there are changes with accounts, such as when a new user is added or an existing user is removed. See Creating a Fusion User Account Report.

What are the Restrictions?

To associate instances, they must be provisioned in the same identity domain.

You cannot readily associate two instances with each other if they were provisioned in different identity domains. When this occurs, you should contact your Oracle representative and raise a service request to evaluate the feasibility of such an association.

The Benefits of Association

By integrating a SaaS application such as Oracle Sales Cloud with an Oracle Platform Service, in this case Java Cloud Service - SaaS Extension, you can extend its capabilities by adding powerful WebLogic Server-based features.

Why Associating Java Cloud Service - SaaS Extension with Oracle Sales Cloud is Valuable

The key features that make a PaaS-SaaS association a useful and often necessary tool lie in the flexibility it provides SaaS developers for extending their applications’ capabilities beyond those available out-of-the-box. These include:

• Enhanced control over your interface:
  – As a developer, you have control over the interfaces you create and host on Java Cloud Service - SaaS Extension, enabling you to embed items and create mashups, as desired.
– You can present SaaS content within your Java Cloud Service - SaaS Extension application's UI.

• External pageflows.
• Complex integration, including:
  – External data access.
  – Integrated Oracle Database Cloud Service - Database as a Service (DBaaS) instance included with every Java Cloud Service - SaaS Extension subscription.
  – External or multiple web services access multiple application flows.
  – A UI or web service that is shared by multiple applications, including other Oracle SaaS offerings.

Why You Might Want to Associate Java Cloud Service - SaaS Extension with a SaaS Application

Extending a SaaS application by using Java Cloud Service - SaaS Extension is useful when you need to implement power or features not readily available from your SaaS application's tooling; for example:

• When you want Oracle Sales Cloud to display data that is not directly related to the product deliverables. For example, a bank that uses Oracle Sales Cloud and wants to see customer profile information from within that application could develop an application that loads the data into Oracle Database Cloud Service, display it on an application built on Java Cloud Service - SaaS Extension, and embed or link out to this application from Oracle Sales Cloud.

• To provide a user interface that you cannot create with your SaaS application. For example, you want to graphically display accounts and related contacts by using UI widgets different from those provided by Oracle Sales Cloud. You could build the widgets on Java Cloud Service - SaaS Extension and then embed them in the portal.

• To create a common UI or web service you can reuse across different applications. For example, you want to display a comprehensive view of a customer's records in a UI that captures that information multiple applications. You could build this UI on Java Cloud Service - SaaS Extension and then link it between each application in the enterprise.

Note:

To associate instances, they must be provisioned in the same identity domain. You cannot readily associate two instances with each other if they were provisioned in different identity domains. When this occurs, you should contact your Oracle representative and raise a service request to evaluate the feasibility of such an association.

Understanding Identity Propagation

Oracle Java Cloud Service - SaaS Extension/SaaS association relies on a shared identity domain wherein an individual user’s identity credentials are passed—or “propagated”—by using trusted security tokens between the services.
Note:

To associate instances, they must be provisioned in the same identity domain. You cannot readily associate two instances with each other if they were provisioned in different identity domains. When this occurs, you should contact your Oracle representative and raise a service request to evaluate the feasibility of such an association.

Identity Propagation is the replication of authenticated identities and can happen through multiple business systems and processes. Identity Propagation is used by the client application to send a user assertion on behalf of the user. When Java Cloud Service - SaaS Extension is established as the Identity Provider, it authenticates the requests from associated Service Providers and establishes the user identity; that identity is then used as the basis for authorization. A user assertion is a user token that contains identity and security information about the user and can be used to authenticate the user. An assertion can be used instead of a username and password as it contains information that will be useful to validate the client. The intent of the client assertion is to provide an alternative client authentication mechanism (one that doesn't send client secrets). Oracle Cloud supports two protocols for propagating identity:

- Security Assertion Markup Language (SAML)
- OAuth

Identity Propagation with SAML

While you can use SAML tokens, Username Tokens (UNT), or JSON Web Tokens (JWT) to establish trust between services, Oracle recommends using SAML-based client policies. SAML is an XML-based, open-standard data format for exchanging authentication and authorization data between parties, in particular, between an identity provider and a service provider.

Why Use SAML?

SAML-based authentication provides these advantages:

- When you use a SAML token, the identity of the user who is signed in to an extension service hosted application is propagated automatically to the SaaS application.
- When you use a SAML token, the SaaS application applies the authorization rules for the signed-in user when processing web service calls.

When Java Cloud Service - SaaS Extension is established as the Identity Provider, it authenticates the requests from associated Service Providers and establishes the user identity; that identity is then used as the basis for authorization. SAML is typically used with web service messaging between associated services. Associated Oracle Cloud services can use Oracle Web Service Manager (OWSM) for SAML authentication. OWSM is shipped with the Oracle SaaS application and provides a menu of security policies, including SAML-based policies, for developers to leverage when making web service calls between services. See Securing JAX-WS Web Services.

To a large extent, SAML automates token-building. These client policies are an effective alternative to building tokens that usually contain user name and password attributes formatted to some specification. With SAML, a preconfigured SAML
infrastructure is presumed. On the client side, SAML tokens are included in outbound web service requests automatically, and a SAML login module knows how to deconstruct the token for authentication purposes. In many respects, SAML token-based policies can be the easiest to implement, as Oracle provides a working SAML infrastructure.

To successfully propagate identities, Java Cloud Service - SaaS Extension and the SaaS application must exist in the same identity domain. When service instances are provisioned in the same identity domain, they are usually automatically associated, which enables SAML-based identity propagation between Java Cloud Service - SaaS Extension and the SaaS application and enables SSO capability with the SaaS application acting as the identity provider. Developers can leverage SAML-based security policies that will automatically use the current logged-in user of the SaaS application to invoke the web service in Java Cloud Service - SaaS Extension. Associated services have this SAML trust pre-established by Oracle. The association is automatic when you purchase a new Java Cloud Service - SaaS Extension instance to be used with an existing SaaS instance or when you purchase a SaaS application and Java Cloud Service - SaaS Extension instance at the same time.

**Supported Policies**

Oracle Java Cloud Service - SaaS Extension and Oracle SaaS integration supports these SAML policies:

- **Client Policies:**
  - `oracle/wss11_saml_token_with_message_protection_client_policy`
    This policy enables message protection (integrity and confidentiality) and SAML token population for outbound SOAP requests using mechanisms described in WS-Security 1.1. A SAML token is included in the SOAP message for use in SAML based authentication with sender vouches confirmation.
  
  - `oracle/wss_saml_token_bearer_over_ssl_client_policy`
    This policy includes SAML tokens in outbound SOAP request messages. The SAML token with confirmation method Bearer is created automatically. The policy also verifies that the transport protocol provides SSL message protection. This policy can be attached to any SOAP-based client.

- **Service Policy:** `oracle/wss11_saml_or_username_token_with_message_protection_service_policy`
  This policy enforces message protection (integrity and confidentiality) and one of the following authentication policies:
  
  - SAML-based authentication for inbound SOAP requests in accordance with the WS-Security 1.1 standard.
  
  - SAML-based authentication using credentials provided in SAML tokens with confirmation method 'Bearer' in the WS-Security SOAP header. Verifies that the transport protocol provides SSL message protection.

**Sample Use Cases**

You can find two sample use cases for propagating ID with SAML in Writing a Client That Can Access an Oracle Sales Cloud Application and Writing a Web Service that an Oracle Sales Cloud Application Can Access.
Identity Propagation with OAuth

Oracle Cloud also supports OAuth 2.0, an open standard for authorization. This protocol allows Internet users to authorize websites or applications to access their information on other websites but without sharing passwords, making it easy for users to share information about their accounts with third-party applications or websites.

Why Use OAuth?

Use OAuth 2.0 to define authorization in JCS–SaaS Extension for your custom applications. OAuth 2.0 has an authorization framework, commonly used for third-party authorization requests with consent. Custom applications can implement two-legged OAuth flows only. OAuth 2.0 provides the following benefits:

- It increases security by eliminating the use of passwords in service-to-service REST interactions.
- It reduces the lifecycle costs by centralizing trust management between clients and servers. OAuth reduces the number of configuration steps to secure service-to-service communication.

Both JCS-SaaS Extension and the SaaS application instances to which it will be propagating identity should be provisioned in the same identity domain. This way, the resources and clients needed for communicating using OAuth are automatically configured along with an OAuth server, which is used for obtaining the tokens.

Sample Use Case

You can find a sample use case that shows you how to associate a JCS-SaaS Extension client with a SaaS resource in Propagating ID with OAuth.

Identity Propagation Use Cases

To establish trust when integrating Oracle Java Cloud Service - SaaS Extension and Oracle Sales Cloud, you need to include a supported security policy for the type of integration you want to establish.

Three use cases demonstrate how to implement security policies to ensure successful identity propagation:

- A client that can access a SaaS application (SAML).
- A web service that a SaaS application can access (SAML).
- Identity propagation with OAuth

Writing a Client That Can Access an Oracle Sales Cloud Application

This client exposes a SAML policy that enables message-level protection and SAML token population for outbound web service requests, which allows it to access the Oracle Sales Cloud application with which it is associated.

To create the client and expose the proper security policy, you need to:

1. Obtain the web service descriptor (WSDL) from the service you want your Java Cloud Service - SaaS Extension application to access by issuing this command:

   `service_end_point ? wsdl`
To find the `service_end_point`, either use the Service Catalog Service or:

a. Look up the web service in the Oracle Enterprise Repository to get the service path by following the instructions in Searching for Public External Services.

b. Derive the end point by following the instructions in Deriving the Business Object Service Endpoint and WSDL.

2. Create a client `.java` application; for example, `helloWorld.java`.

3. Attach the security policy you will be invoking; for example, the following snippet, uses the SAML security policy `oracle/wss11_saml_token_with_message_protection_client_policy`, which is supported by Oracle Sales Cloud and enables message-level protection and SAML token population for outbound web service requests by using mechanisms described in WS-Security 1.1:

   ```java
   package oracle.jcs.ws.sample.saml.proxy;
   import java.net.MalformedURLException;
   import java.net.URL;
   import java.util.Map;
   import javax.xml.namespace.QName;
   import javax.xml.ws.BindingProvider;
   import weblogic.wsee.jws.jaxws.owsm.SecurityPolicyFeature;
   public class HelloWorldPortClient {
       public static String callHelloWorld(String wsdl, String address, String issuername, String name, String username) throws MalformedURLException {
           QName serviceName = new QName("http://saml.sample.ws.jcs.oracle/", "HelloWorldService");
           HelloWorldService helloWorldService = new HelloWorldService(new URL(wsdl), serviceName);
           HelloWorld helloWorld = null;
           helloWorld = helloWorldService.getHelloWorldPort(new SecurityPolicyFeature("oracle/wss11_saml_token_with_message_protection_client_policy"));
           Map<String, Object> ctxt = ((BindingProvider)helloWorld).getRequestContext();
           ctxt.put("oracle.webservices.security.saml.issuer.name", issuername);
           ctxt.put("oracle.webservices.security.recipient.key.alias", "orakey");
           ctxt.put(BindingProvider.ENDPOINT_ADDRESS_PROPERTY, address);
           ctxt.put(BindingProvider.USERNAME_PROPERTY, username);
           System.out.println("Testing inbound to the Cloud");
           return helloWorld.hello(name);
       }
   }
   
   For additional information on security policies, see Securing JAX-WS Web Services.

4. Package your application into a WAR file and deploy it to Java Cloud Service - SaaS Extension by using your specific IDE’s deployment process or as described in Deploying an Application.
Writing a Web Service that an Oracle Sales Cloud Application Can Access

A service that enforces message protection and SAML-based authentication for inbound web service requests on Java Cloud Service - SaaS Extension will publish a Web Service Descriptor Language (WSDL) document that an Oracle Sales Cloud application can consume.

To write a Web Service that an Oracle Sales Cloud application can access, you need to:

1. Create a client .java application; for example, helloWorld.java.

2. Use the @SecurityPolicy annotation to attach the security policy you want to invoke; for example, the following snippet uses the SAML security policy oracle/wss11_saml_or_username_token_with_message_protection_service_policy, supported by Oracle Sales Cloud that enforces message protection (integrity and confidentiality) and SAML-based authentication for inbound web service requests in accordance with the WS-Security 1.1 standard:

   ```java
   package oracle.jcs.ws.sample.saml;

   import javax.jws.WebService;

   import weblogic.wsee.jws.jaxws.owsm.SecurityPolicy;

   @WebService
   @SecurityPolicy(uri = "oracle/wss11_saml_or_username_token_with_message_protection_service_policy")
   public class HelloWorld {

       public HelloWorld() {
           super();
       }

       public String hello(String name) {
           return ("Hello " + name + ". The saml authentication worked!");
       }
   }
   ```

   For additional information on security policies, see Securing JAX-WS Web Services.

3. Package your application into a WAR file and deploy it to Java Cloud Service - SaaS Extension:

   a. In the Oracle Java Cloud Service - SaaS Extension Control page, click Deploy. The Deploy Application page appears.

   b. Enter a name for the application you are deploying, and then click Browse to search your local file steam for the application archive to be deployed. After locating the archive, click Deploy Application.

Propagating ID with OAuth

To use OAuth ID propagation with JCS-SaaS Extension, you need to associate a JCS-SaaS Extension client with the desired SaaS resource or resources (that is, the SaaS
application). Then you need to verify that the client configuration is defined in the REST-Client.

To associate a JCS-SaaS Extension client with the desired SaaS resource or resources:

**Note:**

This procedure assumes that both the JCS-SaaS Extension and the SaaS resource instances have been provisioned in the same identity domain. When provisioned in the same identity domain, the resources and clients needed for communicating using OAuth are automatically configured along with an OAuth server which is used for obtaining the tokens.

1. Log in to Oracle Cloud and access the Service Details page for your JCS-SaaS Extension account.
2. Click **Users** to open the User page and then click the **OAuth Administration** tab.
   - The OAuth page appears. This page will be populated with resources and clients registered for each cloud service in the specific identity domain.

3. For the selected client, click and, from the menu, select **Modify**.
   - The Modify Client dialog box appears.
4. Select the checkbox for the resource or resources to which you want to grant the client access.
5. Click **Save**.

To complete this process, you should verify the client configuration.

**Verify the Client Configuration**

OAuth ID propagation requires that the client configuration be set in your REST-Client file.

Verify that the client configuration is defined in your REST-Client file, as illustrated in the highlighted lines in this example:

```java
public class HelloWorldJAXRSClient extends HttpServlet {
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response) throws Exception {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        ClientResponse clientResponse = null;
        ClientConfig cc = new DefaultClientConfig();
        Client client = Client.create(cc);
        ClientFilter filter = new RESTClientFilter();
        client.addFilter(filter);
        WebResource webResource = client
            .resource("https://example.java.us.oraclecloud.com/rest/resources/helloworld");
        clientResponse = webResource.accept("text/plain").get(
            ClientResponse.class);
        String res = clientResponse.getEntity(String.class);
    }
}
```

**Creating a Report of Oracle Sales Cloud User Accounts**

To effectively enable Oracle Sales Cloud users' access Oracle Java Cloud Service - SaaS Extension, you should periodically synchronize those users with Oracle Java Cloud Service - SaaS Extension. This synchronization is a manual, two-step process:
Creating the list of users from the Oracle Sales Cloud service and importing the users into Oracle Java Cloud Service - SaaS Extension instance.

The following procedure creates a report containing a comma-separated values (CSV) list of the Fusion Application user account information required to set up each user account. Once you create the CSV file, you can import these accounts to Oracle Cloud. To extract the Fusion Applications user account information:

1. From the Fusion Applications Home page, click **Navigator**.

2. Under Tools, select **Reports and Analytics**.

3. Click the **Browse Catalog** icon to open the Oracle BI Catalog, then select **New**, then **Data Model**.
4. Under the Diagram tab, click New, then SQL Query to create a new SQL Query data set.

5. In the New Data Set - SQL Query dialog, enter a Name, select ApplicationDB_HCM as the Data Source and select Standard SQL as the Type of SQL. Enter the following SQL query in the SQL Query section, then click OK.

```sql
--HCM
SELECT e.email_address AS email
FROM fusion.per_users u, fusion.per_roles_dn r, fusion.per_user_roles ur,
fusion.per_all_people_f f
JOIN fusion.per_email_addresses e ON e.person_id = f.person_id
AND e.email_address_id = f.primary_email_id AND e.email_type = 'W1'
WHERE TRUNC(SYSDATE) BETWEEN f.effective_start_date AND f.effective_end_date
AND u.person_id = f.person_id AND u.active_flag = 'Y'
AND r.role_common_name = :Bind
AND r.role_guid = ur.role_guid AND ur.active_flag = 'Y'
AND ur.terminated_flag != 'Y'

--TCA
UNION SELECT c.email_address AS email
FROM fusion.per_users u, fusion.per_roles_dn r, fusion.per_user_roles ur,
fusion.hz_person_profiles p
JOIN fusion.hz_contact_points c ON c.owner_table_id = p.party_id
AND c.owner_table_name = 'HZ_PARTIES'
AND c.overall_primary_flag = 'Y'
AND c.contact_point_type = 'EMAIL'
AND c.status = 'A'
AND TRUNC(SYSDATE) BETWEEN c.start_date AND c.end_date
WHERE u.party_id = p.party_id
AND c.owner_table_id = p.party_id
AND TRUNC(SYSDATE) BETWEEN p.effective_start_date AND p.effective_end_date
AND p.status = 'A'
AND u.active_flag = 'Y'
AND r.role_common_name = :Bind
AND r.role_guid = ur.role_guid
AND ur.active_flag = 'Y'
AND ur.terminated_flag != 'Y'
```

6. In the Add Parameter dialog, select the first Bind and click OK. This parameter is used as the input to the report for getting all users for a Role.
7. Enter a name and display label for the bind parameter.

8. Click View Data to display the Data tab, then enter a value for role name parameter, for example, FUSION_APPS_HCM_ADF_APPID.
   a. Click View.
   b. Once the data appears, click Save As Sample Data.
   c. Click Save to save the data model in the Drafts folder (under My Folders).
9. At the top of the window, click New, then select Report.

10. In the Select Data dialog, for Data Model, select the Data Model you created, accept the other default selections, then click Next.
11. In the Select Layout dialog, accept the default report layout selections, then click 
Next.

12. In the Create Table dialog, drag and drop the EMAIL column.

13. Deselect the Show Grand Totals Row check box, and then click Next.

15. Save the report.
   a. In the Save As dialog, name the report *Users*, and save it in the Drafts folder (under My Folders), then click **OK**
      
      The Layout Editor automatically displays the Users report.

   b. From the output drop-down list, select your desired spreadsheet application; for example, **Excel (*.xlsx)**.

   c. In the Open dialog, open the report with the default application, Microsoft Excel.

   d. In Microsoft Excel, from the Save As menu, select Save as type **CSV**, select where you want to store the file, and then click **Save**.
16. In the CSV file, rename the column to **Email** to comply with the Oracle Cloud requirements for importing this file.

17. Import the CSV file with the Fusion Applications account users to Oracle Cloud. For instructions, see Importing a Batch of User Accounts in *Getting Started with Oracle Cloud*.

**PaaS-SaaS Association Sample Applications**

A set of sample applications have been developed to help you get started associating your SaaS extensions with Oracle Java Cloud Service - SaaS Extension.

The goal of PaaS-SaaS integration is to enable extensions to SaaS applications deployed on JCS-SaaS Extension to seamlessly authenticate and authorize users of the associated services without requiring additional sign-on or other credential verification. The Oracle Developer Cloud Service portal provides a number of sample applications that you can use to help you get started with your integration tasks.

**Note:**

While these sample applications were developed for Oracle Sales Cloud implementations, they do provide a useful conceptual foundation for other SaaS applications.

**Invoke Sales Cloud SOAP Web Services by Using Pre-configured SAML**

With this sample application, you can build a custom application in JCS - SaaS Extension that invokes Sales Cloud SOAP web services by using pre-configured SAML as the security mechanism.

**Sales Merchandise Tracker** is an application used by sales representatives to record the company-branded merchandise given to customers and prospects. Sales representatives enter the merchandise value for each of their customers and can query historical merchandise issuance data. Although not implemented here, the application could be extended to track distribution patterns and inventory.
To obtain this sample, see the Sample application: Sales Merchandise Tracker.

Invoke a Web Service Deployed to JCS - SaaS Extension by Using Pre-configured SAML

If you have Oracle Sales Cloud, you can use its Application Composer to invoke a web service deployed to JCS - SaaS Extension by using pre-configured SAML as the security mechanism.

Credit Health Score allows a company to attach a Credit Health Score to every account/opportunity. This score is calculated based on a complex logic that integrates data from both internal and external systems. Then, for a given account, Sales, Service, and Order Management teams can each access these credit scores to facilitate their decision-making process.

To obtain this sample, see the Sample application: Credit Health Score.

Embed a Custom Sales Cloud Application and Access it Through an Embedded UI by Using Pre-configured Single Sign-On

With this sample, you can embed a custom-built Java Application in Sales Cloud and seamlessly access the embedded UI by using pre-configured SSO.

Sales Preparation Insight enhances account pages in Sales Cloud with information about critical or long standing Service Requests that could potentially impede additional sales opportunities. You might also configure the application to add news concerning leadership changes, recent announcements, or other information relevant to the sales process.

To obtain this sample, see the Sample application: Sales Preparation Insight.

Launch a SaaS Application Based on User Privileges and Pre-configured Single Sign-On

With this sample, you can build a custom application in JCS - SaaS Extension that allows launching a SaaS application—in this case Sales Cloud—based on user privileges and pre-configured SSO.

Lead Capture System captures certain pieces of information related to sales leads and maintains them through an ADF application deployed to JCS - SaaS Extension. External users can access and update leads and contact information through a standalone interface, without access to other Sales Cloud functionality.

To obtain this sample, see the Sample application: Lead Capture System.

Embed a Custom-built JavaScript Application UI in Sales Cloud and Access it by Using Pre-configured Single Sign-On

Use this sample, Dealer Feedback System, to embed a custom-built JavaScript-based application in Sales Cloud and seamlessly access its embedded UI by using pre-configured SSO. This sample supports both federated and non-federated login mechanisms.

To obtain this sample, see the Sample application: Dealer Feedback System.
Setting Up Trust Between WebLogic Domains and JCS-SaaS Extension

You can enable Web Service Security trust from a local WebLogic domain to a JCS-SaaS Extension instance in the cloud by using the command-line tool `setup-wss-trust`.

Topics

- About the setup-wss-trust Tool
- Guidelines for Using setup-wss-trust
- Getting More Information

About the setup-wss-trust Tool

`setup-wss-trust` is a command-line tool that automates the process of setting up Web Service Security (WSS) trust from a local WebLogic Server domain to a JCS-SaaS Extension instance in the cloud.

This command is supported by the same command-line tool, javacloud, available with JCS-SaaS Extension SDK, version 16.4.1. You can use this command to set up trust from an on-premises domain (that is, any environment from which you have access to your local WebLogic Server domain) to an instance in the cloud deployed on JCS-SaaS Extension so that you can propagate IDs and protect messages from that domain to the instance. Note, however, that while this command allows you to propagate IDs and protect messages from your on-premises domain to a JCS-SaaS Extension instance, it does not provide similar functionality in the other direction; that is, you must use other techniques to establish similar trust between the instance and your on-premises domain. You can also run this command if you need to set up point-to-point WSS trust between two JCS-SaaS Extension instances running in separate identity domains.

Using the Command

The following syntax describes typical usage of the command. Required commands are in **bold**. Line breaks have been added for clarity; do not include them when entering the command..

```
$ javacloud -setup-wss-trust -user|-u userName
        -password|-p password
        -identitydomain|-id identityDomain
        -serviceinstance|-si serviceInstance
        -alias certAlias [-path|-p //pathToCert]
        -issuer|-is SAMLIssuer
        [-httpproxy|-hp proxyhost:port@user/password]
        [-certfiletype|-cft certFileType]
        [-output|-o //pathToCertDownload]
```


For example:

```
$javacloud -setup-wss-trust -identitydomain myiddomain -serviceinstance myinstance -user user.com -password **** -alias myorg -path myorg.jks -issuer myorgname
```

Response:

```
[SETUP TRUST] [INFO] - Checking if the alias already exists in the Web service security store.
[SETUP TRUST] [INFO] - The certificate with the alias myorg does not exist already.
[SETUP TRUST] [INFO] - Importing certificate with command-line:add-wss-certificates
password ********
-serviceinstance "myinstnace" -adminurl "https://javaservices.us2.cloud.oracle.com" -path myorg.jks" -alias "myorg"
[SETUP TRUST] [INFO] - 1 certificate(s) added.
[SETUP TRUST] [INFO] - Establishing trust with DN: CN=MyOrgName Serial Number: -167863760719642507519543905148448728112
[SETUP TRUST] [INFO] - Creating required Trust configuration using -config-shell
[SETUP TRUST] [INFO] - Checking if the config-shell is already open...
[SETUP TRUST] [INFO] - Ending existing config-shell session.
[SETUP TRUST] [INFO] - Entering into config-shell in the auto-mode. This would not require any manual operation until the shell exits. Please be patient as you observe slight delays.
[SETUP TRUST] [INFO] - Running config-shell with the command-line:config-shell
identitydomain "myiddomain" -user "user.com" -password ********
```

Chapter 4
About the setup-wss-trust Tool
Please exit and re-enter the shell if the prompt does not appear within a few seconds. You can type "exit" to exit the shell.

Config-shell: the trusted DN lists are successfully set
Config-shell: the trusted DN lists are successfully set
Config-shell: JWT trusted issuers successfully set
Config-shell: Please exit and re-enter the shell if the prompt does not appear within a few seconds. You can type "exit" to exit the shell.

[SETUP TRUST]
[INFO] - Config-shell finished successfully!

[SETUP TRUST] [INFO] - Exporting cloud instance certificate...
[SETUP TRUST] [INFO] - If the trust from the cloud instance to the local weblogic downloaded configuration at the local weblogic domain.
[SETUP TRUST] [INFO] - 2 certificates downloaded.
[SETUP TRUST] [INFO] - Downloaded at: /Users/velsubra/Downloads/work/downloaded_certificates.jks
[SETUP TRUST] [TIP] - Success: This completes one way trust setup from the local weblogic domain to the cloud instance.

Guidelines for Using setup-wss-trust

Understanding how to use certain key parameters, including -alias and -path, is critical to establishing trust from an on-premises environment to an instance deployed on JCS-SaaS Extension.

Specifying the Alias and Path

If a certificate has already been uploaded to the JCS-SaaS Extension instance in the cloud, it would have been uploaded against an alias. To set up trust, the instance needs to know this alias. The mandatory -alias parameter identifies the certificate issued for your local WebLogic Server domain. To be identified by its alias, the certificate needs to be uploaded to the JCS-SaaS Extension instance in the cloud. If this certificate is not already imported to the cloud instance, you’ll also need to specify the argument -path:

```
$ javacloud -setup-wss-trust -identitydomain myiddomain -serviceinstance myinstance -user user.com -password **** -alias myorg -path myorg.jks -issuer myorgname
```
By specifying the path, when setting up the trust the certificate will be imported automatically against the alias value. If the certificate is already imported, just specify the existing alias:

```bash
$ javacloud -setup-wss-trust -identitydomain myiddomain -serviceinstance myinstance -user user.com -password **** -alias myorg -issuer myorgname
```

and JCS-SaaS Extension will know where to find the certificate based on that alias.

**Specifying the Certificate Filetype**

The JCS-SaaS Extension instance needs to know the certificate's filetype. If you specify this value as part of the path (-path), the instance can derive the filetype from there; for instance, in the preceding command example, the value for -path is myorg.jks so the JCS-SaaS Extension instance would use this filetype .jks as the certificate filetype. If you don't include a filetype with the -path parameter, you need to specify it by using the -certfiletype parameter:

```bash
$ javacloud -setup-wss-trust -identitydomain myiddomain -serviceinstance myinstance -user user.com -password **** -alias myorg -path myorg -certfiletype JKS -issuer myorgname
```

**Listing Available Certificates**

An instance might already have a number of certificates uploaded. To see if you can list all the trusted certificates using the command -list-wss-certificates:

```bash
$ javacloud -list-wss-certificates -identitydomain myiddomain -serviceinstance myinstance -user user.com -password ****
```

For information on -list-wss-certificates, see Managing Web Services Security Truststore.

**Setting Up Trust from the Instance to the WebLogic Server Environment**

setup-wss-trust only establishes trust in one direction: from your on-premises environment to the JCS-SaaS Extension instance in the cloud. If you want to set up trust in the other direction, you will have to follow the steps required for your specific SaaS application. However you set up this “reverse” trust, you will need to use the -output flag with setup-wss-trust to specify the location where the certificate will be downloaded. For example:

```bash
$javacloud -setup-wss-trust -identitydomain myiddomain -serviceinstance myinstance -user user.com -password **** -alias myorg -path myorg.jks -issuer myorgname -output c:/mycerthome/trustcert/
```

**Getting More Information**

As a component of the JCS-SaaS Extension SDK, more comprehensive documentation for setup-wss-trust and its associated CLI commands is available.
through the Oracle Help Center and directly from the SDK documentation shipped with the product.

In the Oracle Help Center, you can find more information about `setup-wss-trust` in CLI Commands in the SDK or navigate to the `$SDK_HOME/doc/index.html` file. You can also access all the SDK documentation via the Welcome App. To do so:

1. In the Applications region of the Oracle Java Cloud Service - SaaS Extension Control, click `welcome-app`.
   The Application: welcome-app page appears.

2. In the Application URLs table, click the URL.
   The Oracle Java Cloud Service - SaaS Extension home page appears.

3. Click Oracle Java Cloud Service - SaaS Extension SDK.
   The Oracle Java Cloud Service - SaaS Extension SDK Home page appears. From here, you can select the desired CLI documentation; for example, CLI-Javacloud.jar

Also see Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension.
Managing Instances

You can perform certain management activities on specific instances, such as relocating the instance and upgrading and downgrading an instance between versions.

Topics

• Relocating a Service to a Different Identity Domain
• Upgrading an FMW 11.1.1.7 Instance to–and Downgrading it from–FMW 11.1.1.9

Relocating a Service to a Different Identity Domain

You can easily move your service instance from one identity domain to another so long as both identity domains are in the same data center.

Occasionally, whether due to errors made when originally subscribing to a JCS-SaaS Extension service instance or, in the case of a SaaS application, when that application and the service instance were provisioned in different identity domains, you might need to move your service instance from one identity domain to another. If you are the service administrator, you can do this easily with the Identity Domain Administration's Relocate Service feature.

Note:

These procedure for relocating an instance to a different identity domain apply only to instances provisioned prior to August 2017.

In the new Entitlement model, instances are not created by default. Instead, you provision APEX in the Dashboard first, then associate the APEX instance with JCS-SaaS Extension.

Topics

• Relocating the Service Instance
• Completing Post-relocation Tasks

Relocating the Service Instance

Relocate the JCS-SaaS Extension instance by using the Oracle Cloud Console.

Before You Begin

Before you relocate a service instance, ensure that the target identity domain (that is, the identity domain to which you are moving the service instance):

• Is provisioned as part the same customer account ID.
• Resides in the same data center as the source identity domain.

To Relocate the Service

1. Navigate to your Oracle Cloud My Accounts page and click the name of the service you want to relocate.
   The Service Details page for that service appears.

2. Click Identity Domain Administration

   ![Service Details](image)

   **Overview**
   (for December 2016)
   100% uptime

   **Business Metrics**
   (as of 3 minutes ago)
   0.43% cpu usage

   **Associations**
   1 associations

   **Identity Domain Administration**

   ![Identity Domain Administration](image)

The Identity Domain Administration page appears.

3. Click the Relocate Service tab.
4. Click the **Identity Domain** dropdown control and select the identify domain to which you want to relocate your service instance.

   The relocation will not delete service data, so after the relocation all services will be before in the new identity domain.

   ![Identity Domain dropdown control](image)

<table>
<thead>
<tr>
<th>Identity Domain</th>
<th>Select Identity Domain</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Name</td>
<td>jcssxdecimal</td>
</tr>
</tbody>
</table>

   **Note:**

   Do not change the **Service Name** unless the default name is already in use in the selected identity domain. In that case, enter a new name.

5. Click **Submit** and then respond affirmatively to the confirmation message.

   After a few moments, you'll receive a confirmation email that will include the new My Services URL, the new identity domain, and the old identity domain.

   You now have relocated your service instance; however, before you can proceed, you need to complete some post-relocation tasks.

### Completing Post-relocation Tasks

Before you can proceed with a relocated service instance, the service administrator will need to create other users and administrators in the new identity domain and associate and relocate the associated database service instance.

#### Updating User Information

Upon relocation, the service administrator will be added to the target identity domain but other users and administrators will not. The service administrator will need to create other users and administrators in the new identity domain. If applicable, the bulk user import can be used for this task. The service administrator then will need to assign the service roles to all the required users. Bulk role assignment can be used. See Managing Users and Roles in *Getting Started with Oracle Cloud*.

#### Reassociating and Relocation the Database Instance

After relocating a JCS-SaaS Extension service instance, the database service associated with it needs to be reassociated and then moved to the same identity domain.

1. Reassociate the database instance:

   a. Click the link for the service with which you want to associate a non-associated service; for example, if you want to associate database service "db1", with Java service "java1", click the "java1" service link.

   The Service Details page appears.
b. Open the Associations tab and click Manage Associations. The Manage Associations dialog box appears.

c. Under Non Associated Services, select the service you want to associate and click the left arrow. The selected service moves to the Currently Associate Services list.

d. Click OK and then, on the confirmation message, click OK again. The Service Details page reappears with a message showing that the selected service is being associated.

2. Relocate the database instance to the same identity domain to which you relocated the JCS-SaaS Extension service instance. To do this, follow the same steps you used to relocated the JCS-SaaS Extension service instance, except do so for a Database Cloud Service service instance:

   a. Navigate to your Oracle Cloud My Accounts page and click the name of the Database Cloud Service service instance you want to relocate.

   b. Click Identity Domain Administration.

   c. Click the Relocate Service tab.

   d. Click the Identity Domain dropdown control and select the identify domain to which you want to relocate your service instance.

   e. Click Submit and then respond affirmatively to the confirmation message.

After a few moments, you'll receive a confirmation email that will include the new My Services URL, the new identity domain, and the old identity domain.

Upgrading an FMW 11.1.1.7 Instance to–and Downgrading it from–FMW 11.1.1.9

You can use the CLI to migrate your JCS-SaaS Extension instances provision with the 16.4.5 (or earlier) release from FMW 11.1.1.7 binaries to FMW 11.1.1.9 and, if necessary, revert those specific instances back from FMW 11.1.1.7 to FMW 11.1.1.9.

JCS-SaaS Extension supports the FMW 11.1.1.9 version of the JRF binaries in instances deployed on it. Since the 16.4.5 release (December, 2016), JCS-SaaS Extension has used a hybrid approach to support both FMW 11.1.1.7 and FMW 11.1.1.9 JRF components wherein components have both a FMW 11.1.1.7-based oracle_common home and one based on FMW 11.1.1.9. This topology facilitates upgrading the current instances from FMW 11.1.1.7 to FMW 11.1.1.9 during patching by addressing significant changes in the JRF Components (ADF, OWSM, OPSS, and so on) but creates incompatibilities with the current configuration and applications running PS6-based instances, which introduces a degree of risk during the migration process. To provide a safer means of moving instances between FMW 11.1.1.7 and FMW 11.1.1.9, the JCS-SaaS Extension SDK provides these two commands:
- **upgrade-service-instance** upgrades instances provisioned in release 16.4.5 and earlier and running with FMW 11.1.1.7 JRFs to FMW 11.1.1.9. See [Upgrading an Instance from PS6 to PS7](#).

- **downgrade-service-instance** reverts the upgrade of instances previously upgraded with **upgrade-service-instance**. See [Downgrading an Upgraded Instance](#).

**Note:**

**upgrade-service-instance** and **downgrade-service-instance** replace **jrf-migrate-to-ps7** and **jrf-revert-to-ps6**—introduced in JCS-SaaS Extension version 17.2.3—respectively. While these former commands will still work, we recommend that you use the new commands going forward and reserve the former commands only for when you need to ensure backward compatibility.

### Upgrading an Instance from FMW 11.1.1.7 to FMW 11.1.1.9

To facilitate safe migration of your current instance to FMW 11.1.1.7-based binaries, use the **upgrade-service-instance** CLI command.

**Note:**

**upgrade-service-instance** replaces **jrf-migrate-to-ps7** introduced in JCS-SaaS Extension version 17.2.3. While the former command will still work, we recommend that you use the new commands going forward and reserve the former commands only for when you need to ensure backward compatibility.

### Using the Command

**Note:**

This command only works with instances created with JCS-SaaS Extension 16.4.1 and earlier.

```
./javacloud -user userName -id identityDomain -si serviceInstance -upgrade-service-instance
```

<table>
<thead>
<tr>
<th>Parameter (Alias)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user (-u)</td>
<td>The name used to authenticate the user.</td>
</tr>
<tr>
<td>identitydomain (-id)</td>
<td>The name of the identity domain in which the service instance exists.</td>
</tr>
<tr>
<td>serviceinstance (-si)</td>
<td>The name of the service instance you want to migrate.</td>
</tr>
</tbody>
</table>
For a list of optional parameters:

1. Navigate to the $SDK_HOME/doc/index.html file (where $SDK_HOME is the directory containing your JCS - SaaS Extension installation) or go to the SDK documentation via the "Welcome App".
2. Click CLI-Javacloud.jar.
3. In the command list, search for jrf-migrate-to-ps7 and click it to retrieve details about the command.

For example:

```bash
$ ./javacloud -dc em2 -user joe.user@myco.com -id migration1234 -si migrationinst1234 -upgrade-service-instance
```

How Migration Works

**Note:**

If you have an FMW 11.1.1.7-based instance and have deployed ADF-based applications, your applications are working with ADF 11.1.1.7. In this case, before migrating to FMW 11.1.1.9, you must ensure that your ADF-based applications work with the ADF 11.1.1.9 (FMW 11.1.1.9) in a local Weblogic environment.

upgrade-service-instance switches the instance to work with FMW 11.1.1.9 binaries, restarts the domain to let the change take effect, sets the configuration required to migrate the OWSM component and then restarts the domain again.

Migration Use Case

This use case demonstrates a typical instance migration from FMW 11.1.1.7 to FMW 11.1.1.9.

1. First, we'll use list-config to verify that the instance we plan to migrate is in FMW 11.1.1.7; that is, uses JRF 11.1.1.7.0:

```bash
$ ./javacloud.jar -dc em2 -user joe.user@myco.com -id migration1234 -si migrationinst1234 -list-config -sv -v -search jrf
```

The system responds:

```bash
#=======================================================================
#Listing one Simple Config
#=======================================================================
Listing one Simple Config
```
Chapter 5

Upgrading an FMW 11.1.1.7 Instance to–and Downgrading it from–FMW 11.1.1.9

... [Identity Domain=migrationtestid9, Service Instance=migrationtests19]

| #=#=========================#==========#========#=======#========#======#
|========================================================================
| ===================#==========#=====#
| |                         |          | Value  |Value  |Restart |
| |     |     |     |
| #|          Name           |Value Type|Readable|Writabl| Required |
| |                     |          |        |  e    |        |
| |                         |          |        |       |        |It shows the current version of JRF which the instance is using. The possible values for this config property are:
|1|oracle.common.jrf.version|STRING    |   Y    |       |        |
| |                   |          |        |       |        |
| |========================================================================
| =|-------------------------|----------|--------|-------|--------|
|+-----------------------------------------------------------------------
|+-----------------------------------------------------------------------
|Note that the Value is 11.1.1.7.0, indicating the JRF version is FMW 11.1.1.7.

2. Next, migrate the instance:

$ ./javacloud -dc em2 -user joe.user@myco.com -id migration1234  -si migrationinst1234 -upgrade-service-instance

The system responds:

1:Job Id - 9513
----------- - -----------------------
-> - Properties
----------- - -----------------------
Status - NEW
Identity Domain - migrationtestid9
Service Instance - migrationtests19
Application -
Start Time - Wednesday, March 29, 2017 11:48:40 PM PDT
Operation - Migrate JRF to PS7
You can see that the process returned a Job ID (9513). We’ll use this number with -list-job-logs -jobid to check see which logs have been completed for this migration. These logs are useful in tracking the activity within the process:

```
$ ./javacloud -dc em2 -user joe.user@myco.com -id migration1234  -si migrationinst1234 -list-job-logs -jobid 9513
```

The system responds with this list of the three job logs produced for this migration:

```
<table>
<thead>
<tr>
<th>Log Name</th>
<th>Last Updated</th>
<th>Content Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>validate-instance-migration</td>
<td>14 minutes and 38 seconds ago</td>
<td>text/plain</td>
</tr>
<tr>
<td>switch-oracle-home-to-ps7</td>
<td>14 minutes and 31 seconds ago</td>
<td>text/plain</td>
</tr>
<tr>
<td>exec-migration-script</td>
<td>7 minutes and 30 seconds ago</td>
<td>text/plain</td>
</tr>
</tbody>
</table>
```

**Note:**
These are the only job logs produced for a migration operation.

Next, we’ll run list-config again to verify that the migration has completed and the instance is using JRF 11.1.1.9.0:

```
$ ./javacloud -dc em2 -user joe.user@myco.com -id migration1234  -si migrationinst1234 -list-config -sv -v -search jrf
```

The system responds:
--------------------------
| Listing one Simple
Config
---

Identity
Domain=migrationtestid9, Service
Instance=migrationtestsi9
---

#=#=========================#==========#========#=======#========#======
========================================================================

### Name | Value Type | Readable | Writable | Required | Description
---

It shows the current version of JRF which the instance is using. The possible values for this config property are:

- 11.1.1.7.0, 11.1.1.9.0.

Note that the Value is 11.1.1.9.0, indicating the migration to FMW 11.1.1.9 is successful.

5. Finally, we’ll open the configuration shell and use the list-token-issuer-trust command to verify that the list of trusted token issuers hasn’t changed, thus indicating that migration was successful:

```
$ ./javacloud.jar -dc em2 -user joe.user@myco.com -id migration1234 -si migrationinst1234 -config-shell
```

The system responds:

```
[INFO]    - Java Cloud Service - SaaS Extension config shell.
Initializing ...
```

Please exit and re-enter the shell if the prompt does not appear within a few seconds. You can type "exit" to exit the shell.
Then, enter the `list-token-issuer-trust` command:

```
Config-shell:> list-token-issuer-trust
```

The system responds:

List of trusted issuers for this type: migrationtestid9

List of trusted key(s) for this issuer:
  Key Identifier:
  cn=migration1234_javasvc,dc=#1605636c6f7564,dc=#16066f7261636c65,dc=#1603636f6d
    Key Type           : x509certificate
    Value Type         : dn
    Key Identifier:
  cn=cloud9ca-2,dc=#1605636c6f7564,dc=#16066f7261636c65,dc=#1603636f6d
    Key Type           : x509certificate
    Value Type         : dn
    Key Identifier:
  cn=migration1234_idm,dc=#1605636c6f7564,dc=#16066f7261636c65,dc=#1603636f6d
    Key Type           : x509certificate
    Value Type         : dn

www.oracle.com

List of trusted key(s) for this issuer:
  Key Identifier:
  cn=migration1234_javasvc,dc=#1605636c6f7564,dc=#16066f7261636c65,dc=#1603636f6d
    Key Type           : x509certificate
    Value Type         : dn
    Key Identifier:
  cn=cloud9ca-2,dc=#1605636c6f7564,dc=#16066f7261636c65,dc=#1603636f6d
    Key Type           : x509certificate
    Value Type         : dn
    Key Identifier:
  cn=migration1234_idm,dc=#1605636c6f7564,dc=#16066f7261636c65,dc=#1603636f6d
    Key Type           : x509certificate
    Value Type         : dn

**Downgrading an Upgraded Instance**

If, upon migration, you discover that your instance is unstable because FMW 11.1.1.9 binaries are incompatible your applications, you can use the command `downgrade-service-instance` to restore it to the FMW 11.1.1.7-based binaries.
Be aware of the following:

- `downgrade-service-instance` replaces `jrf-revert-to-ps6`, introduced in JCS-SaaS Extension version 17.2.3. While the former command will still work, we recommend that you use the new commands going forward and reserve the former commands only for when you need to ensure backward compatibility.

- This command only works with instances that have already been migrated to FMW 11.1.1.9 by using the `upgrade-service-instance` command. You cannot revert an instance that was originally provisioned in FMW 11.1.1.9.

Using the Command

```
./javacloud -u userName -id identityDomain -si serviceInstance -downgrade-service-instance
```

<table>
<thead>
<tr>
<th>Parameter (Alias)</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>user (-u)</td>
<td>The name used to authenticate the user.</td>
</tr>
<tr>
<td>identitydomain (-id)</td>
<td>The name of the identity domain in which the service instance exists.</td>
</tr>
<tr>
<td>serviceinstance (-si)</td>
<td>The name of the service instance you want to migrate.</td>
</tr>
</tbody>
</table>

Note:

For a list of optional parameters:

1. Navigate to the `$SDK_HOME/doc/index.html` file (where `SDK_HOME` is the directory containing your JCS - SaaS Extension installation) or go to the SDK documentation via the “Welcome App”.

2. Click CLI-Javacloud.jar.

3. In the command list, search for `downgrade-service-instance` and click it to retrieve details about the command.

How Revert Works

downgrade-service-instance removes the configuration added during the migration process and then restarts the domain.

Use Case 1: Reverting an Instance

Assuming an instance was migrated to FMW 11.1.1.9 by using `upgrade-service-instance`, you can revert it to FMW 11.1.1.7, as demonstrated in this use case.
1. First, we'll use the `list-config` command to verify that the instance we plan to migrate is in FMW 11.1.1.9; that is, uses JRF 11.1.1.9.0:

```
$ ./javacloud -dc em2 -user joe.user@myco.com -id migration1234 -si migrationinst1234 -list-config -sv -v -search jrf
```

The system responds:

```
#==============================================
# Listing one Simple Config
# Domain=migrationtestid9, Service Instance=migrationtests19]
#=#=========================#==========#========#=======#========#======
# | Value |Value | Restart |
# Name |Value Type|Readable|Writabl| Required|
| Description | Value |Label| | | | | |
| | | | | | | | |
|-------------------------|----------|--------|-------|--------|
| It shows the current version of JRF which the instance is using. The possible values for this config property are: (11.1.1.7.0,11.1.1.9.0).
```

| 11.1.1.9.0 | | | | |
|-----------|-----------|-------------|-----------------------|
| 11.1.1.9.0 | | | | |
| ++-------------------------------+---|
```
2. Note that the **Value** is 11.1.1.9.0, indicating the JRF version is FMW 11.1.1.9, so next we'll use `downgrade-service-instance` to revert the instance to FMW 11.1.1.7:

```
$ ./javacloud -dc em2 -user joe.user@myco.com -id migration1234 -si migrationinst1234 -downgrade-service-instance
```

The system responds:

```
[INFO]    - The revert to PS6 has been performed.
```

<table>
<thead>
<tr>
<th>Job Id</th>
<th>9515</th>
</tr>
</thead>
<tbody>
<tr>
<td>Status</td>
<td>NEW</td>
</tr>
<tr>
<td>Identity Domain</td>
<td>migrationtestid9</td>
</tr>
<tr>
<td>Service Instance</td>
<td>migrationtests19</td>
</tr>
<tr>
<td>Application</td>
<td>-</td>
</tr>
<tr>
<td>Start Time</td>
<td>Thursday, March 30, 2017 12:08:14 AM PDT</td>
</tr>
<tr>
<td>Operation</td>
<td>Revert JRF to PS6</td>
</tr>
</tbody>
</table>

[TIP] - You can use the command "job-status" to monitor a job.

3. You can see that the process returned a **Job ID** (9515). We'll use this number with the `list-job-logs -jobid` command to check see which logs have been completed for this migration. These logs are useful in tracking the activity within the process:

```
$ ./javacloud -dc em2 -user joe.user@myco.com -id migration1234 -si migrationinst1234 -list-job-logs -jobid 9515
```

The system responds with this list of the two job logs produced for this migration:

```
### Listing 2 job(id=9515) logs
```

<table>
<thead>
<tr>
<th>Log Name</th>
<th>Last Updated Description</th>
<th>Content Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>validate-instance-migration</td>
<td>4 minutes and 55 seconds ago</td>
<td>text/plain</td>
</tr>
<tr>
<td>exec-revert-script</td>
<td>4 minutes and 10 seconds ago</td>
<td>text/plain</td>
</tr>
</tbody>
</table>
Next, we’ll run `list-config` again to verify that the revert has completed and the instance is again using JRF 11.1.1.7.0:

```bash
$ ./javacloud -dc em2 -user joe.user@myco.com -id migration1234 -si migrationinst1234 -list-config -sv -v -search jrf
```

The system responds:

```
#----------------------------------------------------------------------
#----------------------------------------------------------------------
# Listing one Simple Config
#----------------------------------------------------------------------
| Identity Domain=migrationtestid9, Service Instance=migrationtests19 |
#----------------------------------------------------------------------
#----------------------------------------------------------------------
<table>
<thead>
<tr>
<th>Name</th>
<th>Value Type</th>
<th>Readable</th>
<th>Writable</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>------</td>
<td>------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td>------</td>
<td>------------</td>
<td>----------</td>
<td>----------</td>
<td>----------</td>
</tr>
</tbody>
</table>
```

It shows the current version of JRF which the instance is using. The possible values for this config property are: (11.1.1.7.0, 11.1.1.9.0).

The **Value** is 11.1.1.7.0, indicating the revert to FMW 11.1.1.7 is successful.
5. Finally, we'll open the configuration shell and use the `list-token-issuer-trust` command to verify the revert:

```bash
$ ./javacloud.jar -dc em2 -user joe.user@myco.com -id migration1234 -si migrationinst1234 -config-shell
```

The system responds:

```
[INFO]    - Java Cloud Service - SaaS Extension config shell.
    Initializing ...

Please exit and re-enter the shell if the prompt does not appear within a few seconds. You can type "exit" to exit the shell.
```

Then, enter the `config-shell` command:

```
Config-shell:>list-token-issuer-trust
```

The system responds:

```
migration1234
www.oracle.com
```

### Use Case 2: Reverting an Instance Originally Provisioned in FMW 11.1.1.9 to FMW 11.1.1.7

In this use case, we try take an instance that was provisioned in FMW 11.1.1.9--that is, it never ran on FMW 11.1.1.7--and revert it to FMW 11.1.1.7. Since JCS-SaaS Extension does not allow this type of revert, this case should fail.

1. First, we'll use the `list-config` command to verify that the instance we plan to migrate is in FMW 11.1.1.9; that is, uses JRF 11.1.1.9.0:

```bash
$ ./javacloud -dc em2 -user joe.user@myco.com -id migration1234 -si migrationinst5678 -list-config -sv -v -search jrf
```

The system responds:

```
#===============================================================
#== Configuration for the Simple Config----------------------------
#== Identity Domain=migrationtestid9, Service migrationtests19    #
| Listing one Simple Config                                      |
|--[Identity Domain=migrationtestid9, Service migrationtests19]  |
| #=================================================================
| #=================================================================
| |                           |                           | Value   | Value   | Restart |
|=================================================================
```
Note that the Value is 11.1.1.9.0, indicating the JRF version is FMW 11.1.1.9.

2. Now we’ll use -jrf-revert-to-ps6 and try to revert the instance to FMW 11.1.1.7:

```
$ ./javacloud -dc em2 -user joe.user@myco.com -id migration1234  -si migrationinst5678 -downgrade-service-instance
```

The system responds:

It is not possible to perform the operation JRF Revert To PS6 over this instance, as it has been created originally with the JRF Version 11.1.1.9.0

3. To further verify that the revert failed, we’ll try to migrate the instance from FMW 11.1.1.7 to FMW 11.1.1.9:

```
$ ./javacloud -dc em2 -user joe.user@myco.com -id migration1234  -si migrationinst1234 -upgrade-service-instance
```

The system responds:

[ERROR] - The instance migration1234.migrationinst5678 already is running with JRF 11.1.1.9.0 version.

This indicates that the instance migrationinst5678 was never reverted.
Administering Instances with JCS-SaaS Extension Control

Oracle Java Cloud Service - SaaS Extension Control is the management console that enables you to deploy and monitor hosted applications.

Note:
JCS-SaaS Extension Control is used with instances created before 2018. You should manage JCS-SaaS Extension instances created after the January 2018 update with the JCS-SaaS Extension Administration Console. See Administering Instances with JCS-SaaS Extension Administration Console. Instances created before 2018 can also use JCS-SaaS Extension Administration Console, but the Java Console button for each instance in the My Services dashboard will point to JCS-SaaS Extension Control.

Topics:
- Understanding Oracle Java Cloud Service - SaaS Extension Control
- Starting and Stopping Applications with JCS-SaaS Extension Control
- Managing Applications with JCS-SaaS Extension Control
- Managing Shared Libraries with the JCS-SaaS Extension Control
- Viewing Application-Specific Statistics on JCS-SaaS Extension Control
- Viewing Activity Logs on JCS-SaaS Extension Control
- Viewing Service and Application Logs on JCS-SaaS Extension Control

Use Oracle Java Cloud Service - SaaS Extension Control to:
- View the list of services you own, including the metrics and availability of each service
- Deploy and redeploy applications
- Start and stop applications on any of the services you own
- View job logs to see recent activity

The primary Oracle Java Cloud Service - SaaS Extension Control user is the service administrator. The service administrator has privileges to access Oracle Java Cloud Service - SaaS Extension Control and can deploy and redeploy applications, as well as start and stop deployed applications.
Understanding Oracle Java Cloud Service - SaaS Extension Control

The Oracle Java Cloud Service - SaaS Extension Control displays usage metrics for a single Oracle Java Cloud Service - SaaS Extension instance.

The metrics shown on Oracle Java Cloud Service — SaaS Extension Control include performance (availability and metrics), CPU/heap usage for the service, and response and load charts. It also shows available applications and libraries. You use this tool to deploy/redeploy and start/stop applications. As shown in the next illustration, the Oracle Java Cloud Service - SaaS Extension Control is divided into regions.
Understanding the Regions of the Oracle Java Cloud Service - SaaS Extension Control Home Page

The Oracle Java Cloud Service - SaaS Extension Control home page provides statistics on performance, data sources, response and load, using Oracle Java Cloud Service - SaaS Extension jobs, applications, and resource usage.

The metrics shown represent the most recent values available. For all "per minute" statistics, they are per minute in the last five minutes.

The Oracle Java Cloud Service - SaaS Extension Control page is made up of the following regions:

<table>
<thead>
<tr>
<th>Region</th>
<th>Description</th>
</tr>
</thead>
</table>
| Restart Service         | If the Oracle Java Cloud Service - SaaS Extension instance somehow reaches an inconsistent state, for example, due to a network error, you can use the Restart Service option to restart the service, which will restart all the managed nodes in the service's domain.  
                          | See Restarting a Java Service Instance.                                                                                                                                                                    |
| Performance Summary     | Displays usage metrics for a single Oracle Java Cloud Service - SaaS Extension. All metrics represent the most recent values available. The metric descriptions are:  
                          | General  
                          | • Service Version - the version number of the running Oracle Java Cloud Service - SaaS Extension.  
                          | • Customer Disk Usage - the amount of disk space, in megabytes, the customer has consumed.  
                          | Servlets and JSPs  
                          | • Active Sessions - Number of active sessions  
                          | • Request Processing Time (ms) - Time spent processing each request.  
                          | • Requests (per minute) - Average number of invocations per minute of the selected servlet or JSP in the last five minutes  
                          | JDBC Usage  
                          | • Open JDBC Connections - Number of JDBC (Java Database Connectivity) connections currently in use  
                          | • JDBC Connection Creates (per minute) - Number of database connections created per minute in the last five minutes  
                          | You can also download your service-specific log files using the View Log Messages link. Clicking this link takes you to the Log Messages page. This page display the log messages based on the search criteria which you provided.  
                          | See Viewing the Service and Application Logs.                                                                                                                                                           |
| Data Sources            | Identifies the data sources accessed by the service.                                                                                                                                                     |
| Response and Load       | Use this region to review the response and load charts associated with each server in the Java service. Using this data, you can determine if the servers are behaving as expected, if any server is overloaded, or if a server is taking too long to respond and load.  
<pre><code>                      | If a server is overloaded or taking too long to respond and load, request an upgrade (add more servers) or ask the Cloud administrator to restart your servers if they are performing poorly. |
</code></pre>
<table>
<thead>
<tr>
<th>Region</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jobs</td>
<td>The Jobs region shows the jobs submitted for this service and the status for each of these jobs. You can also get additional details (view logs) for each of these jobs. Click an Activity ID and select View Activity Logs. For a typical deploy job, there are five logs: Virus Scan, Application Whitelist Validation, WLS Compile, Cloud Compile, and Deploy Application. These logs are text documents that you can either open or save. These logs are the result of background jobs that ran against the application and determined whether the application contains a virus or could otherwise cause problems. If a deploy job fails, the best course of action is to view the logs for the job. Depending on the different job logs available, you can determine where the failure occurred. If there are Java API validations, the Whitelist Validation may not reject the application from being deployed. Instead, it would create a warning report against the violations. You can refresh the Jobs region using the Refresh control located at the top right of the region. This is useful when you have just submitted a job and you want to see if the status has changed. By default, the refresh control is set to manual. You can change it to auto-refresh using the available time intervals. See Viewing the Activity Logs.</td>
</tr>
<tr>
<td>Applications</td>
<td>Use this region to see the list of applications available for this service. To study the statistics associated with a particular application and to view its log files, click the name of the application. • View By using the View list, you can conveniently choose the columns to be displayed in the table. To test an application, click the Test Application icon associated with the application. This launches a dialog containing a list of URLs from which you can access the application. Click a given URL and the application appears in a new window. • Deploy New Once your application is ready for use, you can upload the application for deployment. The application goes through a number of checks to ensure compliance with Oracle standards. These checks include a virus scan and a whitelist validation. After the application passes these tests, deployment of the application begins. Note that deployment is asynchronous. You will be prompted to verify the status of the job submitted for deployment. You can check the status of the deployment by manually refreshing the Jobs table at the bottom of the page. You can also use the control on the top right corner of the table to set the auto-refresh interval for the table. Note: The welcome-app is pre-deployed to each service instance. It provides links to the SDK documentation, samples, blogs, and white papers, and more. • Delete Application When an application is no longer of value, delete it from the Oracle Java Cloud Service - SaaS Extension Control. • Redeploy Use this option when you upgrade or make changes to an application. • Start and Stop You can easily start and stop an application. Use these options after you have deployed the application. Note: If these applications are exposed to your external users, you need to inform these users of potential downtime when you stop the application. See Managing Applications.</td>
</tr>
</tbody>
</table>
Restoring a Java Service Instance from JCS-SaaS Extension Control

When you need to restart an instance, you can use the Restart Service option to restart your service. This restarts all managed servers in service’s domain.

One example of when you might need to restart an instance might be when adding an SSL certificate to the ssl trust-store requires a service restart for the changes to take effect. See Managing SSL Truststores.

To restart a Using Oracle Java Cloud Service - SaaS Extension instance:

1. Click Restart Service Instance.
2. From the Service Features menu, select Restart Service.
3. Click Yes on the Confirmation dialog if you are sure you want to restart service.

   **Note:** In some cases, it may be necessary to force the service to restart even if there are active jobs running. In that case, select the **Force the service restart, even if there are active jobs?** check box on the Confirmation dialog to force the restart.

4. The service restart Confirmation dialog explains how a service restart works in single-server and multi-server environments. Note that there is a default minimum time-out of 10 minutes to allow active jobs to complete.
The **Confirmation** dialog also has the following options:

- In some situations, it may be necessary to force the service to restart even if there are active jobs running. Therefore, you can select the **Force the service restart, even if there are active jobs?** check box to force a service restart.

- You can allow more time for active jobs to complete by increasing the **Restart Timeout** value longer than the 10-minute default value.

5. Click **Yes** when you are ready to restart your service instance.

6. When the service restart begins, there will be a "Restart Service Instance" log entry in the activity logs in the Jobs region and it will show a Status of Running. When the restart is complete, the Status column for the restart will indicate that, as shown in this illustration:

7. When the service has restarted, the **View Job Logs** becomes active. To save or view the log for the restart process:
   a. Click **View Job Logs**.
   b. From the drop-down menu, select Restart Service.
      The download dialog associated with your specific browser appears.
   c. You can either save the log to disk or view it immediately in a text editor.

   This is a sample log for restarting single managed-server service:

   2014-07-08 15:15:07 PDT: Starting action "Restart Service"
   2014-07-08 15:15:07 PDT: Restart Service started
   2014-07-08 15:15:07 PDT: Restarting service: server restart timeout = 600000 ms ...
   2014-07-08 15:15:07 PDT:
   Stopping server m0 : current state = RUNNING ...
   2014-07-08 15:15:09 PDT: Server m0 stopped
   2014-07-08 15:15:09 PDT:
   Starting server m0 : current state = SHUTDOWN ...
   2014-07-08 15:16:16 PDT: Server m0 started
   2014-07-08 15:16:16 PDT: "Restart Service" complete: status SUCCESS
   Server m0 started
Restarting a Single Managed Server Service

For single managed-server services, there may be some service downtime until the managed-server is restarted. Therefore, options like Application Deploy, Start, and Stop on the Oracle Java Cloud Service - SaaS Extension Control are disabled while a service restart is in progress.

Restarting a Multiple Managed Server Service

For multiple managed-server services, there should not be any service downtime because the managed servers are restarted sequentially, as follows:

- Restarts one server at a time.
- Waits for each server to boot up before executing restart on the next server.
- Restarts in sequence of m0, m1, m2, m3, etc.
- Time-outs after waiting for three minutes for a server to restart, and then triggers a restart on the next server.

Managing Applications with JCS-SaaS Extension Control

You can deploy, redeploy, and delete an application using the Applications pane of the Home page. Use the View menu to add, remove, or reorder the columns in the Applications table on the Java Cloud Service — SaaS Extension Control home page.

This illustration shows the Applications pane on the Java Cloud Service Control home page with a single application, welcome-app, deployed. This application is pre-deployed to each service instance. It provides links to the SDK documentation, samples, blogs, and white papers, and more.

Application Management Tasks

Key tasks you can manage with JCS-SaaS Extension Control are:

- Deploying an Application
- Deleting an Application
- Redeploying an Application
- Starting and Stopping Applications
Deploying an Application with JCS-SaaS Extension Control

Deploy applications from the Deploy Applications page, which you can access from the Oracle Java Cloud Service - SaaS Extension Control home page. Upon deployment, the application undergoes a series of security checks before it is actually deployed.

To deploy an application:

1. Locate the Applications region in the Oracle Java Cloud Service - SaaS Extension Control page and click Deploy. The Deploy Application page appears.

2. Enter a name for the application you are deploying, and then click Browse to search your local file stream for the application archive to be deployed. After locating the archive, click Deploy Application.

Oracle Java Cloud Service - SaaS Extension Control uploads and deploys the selected WAR (Web Application Archive) file or EAR (Enterprise Archive) file.

Validating Deployed Applications

Once deployed, each application undergoes a series of security checks before it is actually deployed. These checks include a virus scan and a whitelist validation. For technical and security reasons, a small number of specific APIs are prevented from executing in Oracle Cloud. Once the application passes these tests, deployment of the application begins. See About the Application Deployment Validation Process and Run-time Security.

Verifying Deployment Status

Deployment is asynchronous so you will be prompted to verify the status of the job submitted for deployment. You can check the status of the deployment by manually refreshing the Recent Activity table at the bottom of the home page. You can also use the control on the top right corner of the table to set the auto-refresh interval for the table.

If a deploy job fails, the best course of action is to view the logs for the job. Depending on the different job logs available, you can determine where the failure occurred. See Viewing the Activity Logs.

If whitelist validation fails, deploy is never run and so there will be no deploy log.
Deleting an Application with JCS-SaaS Extension Control

When an application is no longer needed, you can remove it from JCS -SaaS Extension Control.

To delete an application:

1. Locate the Applications region in JCS-SaaS Extension Control
2. Highlight the application you want to delete and click **Delete Application**.

Redeploying an Application with JCS-SaaS Extension Control

Redeploy an application whenever you upgrade or otherwise make changes to an application. In most situations you need to stop the application, make the necessary changes, and then redeploy the application.

To redeploy an application:

1. Locate the Applications region in the Oracle Java Cloud Service - SaaS Extension Control.
2. Highlight the application of interest and click **Redeploy**.
3. Validate the redeployed application and verify the deployment status as directed in Deploying an Application with JCS-SaaS Extension Control.

Starting and Stopping Applications with JCS-SaaS Extension Control

Once an application is deployed, you can start and stop the application as needed. If the application is exposed to external users, inform them of potential downtime before stopping the application.

To start or stop an application:

**Note:** If this application is exposed to your external users, you need to inform these users of potential downtime when you stop the application.

1. Locate the Applications region in the Oracle Java Cloud Service - SaaS Extension Control.
2. Highlight the application of interest and click either **Start** or **Stop**.

Managing Shared Libraries with JCS-SaaS Extension Control

Oracle Java Cloud Service - SaaS Extension Administration Console enables you to deploy and manage any shared Java EE library and optional package supported Oracle WebLogic Server. This include both OOTB libraries and optional packages provided by Oracle Cloud, as well as any user-defined custom shared libraries or
optional packages that are packaged using the standard process as supported by WebLogic Server

**Shared Library Management Tasks**

Key tasks you can perform with JCS-SaaS Extension Control are:

- Understand Shared Java EE Libraries and Optional Packages
- Create Shared Java EE Libraries and Optional Packages
- Deploy, Redeploy, and Delete Libraries

**About Shared Java EE Libraries and Optional Packages**

The shared Java EE library feature in WebLogic Server provides an easy way to share one or more different types of Java EE modules among multiple Enterprise applications.

A shared Java EE library is a single module or collection of modules that is registered with the Java EE application container upon deployment. It can be any of the following:

- Standalone EJB module
- Standalone web application module
- Multiple EJB modules packaged in an enterprise application
- Multiple web application modules package in an enterprise application
- Single plain JAR file

WebLogic Server also supports optional packages, which provide similar functionality to Java EE libraries, allowing you to easily share a single JAR file among multiple applications. As with Java EE libraries, optional packages must first be registered with WebLogic Server by deploying the associated JAR file as an optional package. After registering the package, you can deploy Java EE modules that reference the package in their manifest files.

The shared Java EE libraries and optional packages supported in WebLogic Server are described in detail in Supported Deployment Units in *Oracle Fusion Middleware Deploying Applications to Oracle WebLogic Server*.

**Creating Shared Java EE Libraries and Optional Packages**

Oracle Java Cloud Service - SaaS Extension provides a number of OOTB libraries and optional packages. However, you can package your own content into a shared library or an optional package using the standard process as supported by WebLogic Server. You can then deploy, and then use the Oracle Java Cloud Service - SaaS Extension Control to deploy to them your Oracle Java Cloud Service - SaaS Extension instance.

Detailed instructions for creating these components are included in Creating Shared Java EE Libraries and Optional Packages in *Oracle Fusion Middleware Developing Applications for Oracle WebLogic Server*.
Deploying, Redeploying, and Deleting Libraries

To deploy, redeploy, and delete a library in Oracle Java Cloud Service — SaaS Extension, use the Libraries page.

These tasks are described in:
- Deploying a Library
- Redeploying a Library
- Deleting a Library

Deploying a Library

Use the Libraries page of Oracle Java Cloud Service - SaaS Extension Control to deploy libraries.

Once deployed, a library goes through a number of checks to ensure compliance with Oracle standards. To deploy a library, do the following:

2. Enter a Name for the library.
3. If necessary, enter an Implementation Version and a Specification Version number for the library.
4. Click Choose File to search your local file system for the library archive to be deployed. After locating the archive, click Deploy.

JCS - SaaS Extension Control uploads and deploys the selected library archive. Once deployed, a library goes through a number of checks to ensure compliance with Oracle standards.

Redeploying a Library

Use the Library page of Oracle Java Cloud Service - SaaS Extension Control to redeploy libraries.

After you have upgraded or otherwise make changes to a library, you need to redeploy it. Click Deploy New or Redeploy as necessary.

2. Click Choose File to search your local file system for the library archive to be deployed. After locating the archive, click Deploy.

Oracle Java Cloud Service - SaaS Extension Control uploads and deploys the selected library archive. Once deployed, a library goes through a number of checks to ensure compliance with Oracle standards.

Deleting a Library

Use the Deploy Library page of Oracle Java Cloud Service - SaaS Extension Control to delete libraries.
1. Access the Libraries page in the Oracle Java Cloud Service - SaaS Extension Control.

2. Highlight the library you want to delete and click **Delete**.

3. On the deletion confirmation dialog, click **OK**.

### Viewing Application-Specific Statistics on JCS-SaaS Extension Control

The Application Home page shows how well the application is working. Data shown includes availability and performance metrics, as well as version, state of the application, and application URLs for testing the application.

From the Application menu, you can: start, stop, redeploy, and delete the application. There is also an option for monitoring the performance of ADF applications.

To view the Application Home page:

1. Locate the Applications region on the Oracle Java Cloud Service - SaaS Extension Control
2. Click the Name of the application of interest. The application's home page appears.

### Viewing the Activity Logs on JCs-SaaS Extension Control

Activity logs show the status and progress of activities, such as application deployment and service restarts.

You can access activity logs from the Oracle Java Cloud Service - SaaS Extension Control Jobs region, which is at the bottom of Oracle Java Cloud Service - SaaS Extension Control.
When you choose a Job ID in the Oracle Java Cloud Service - SaaS Extension Control Jobs section of the Oracle Java Cloud Service - SaaS Extension Control home page, the View Job Logs drop-down menu becomes active, and you can choose from the following list of activity logs:

- Virus scan
- Application Whitelist Validation
- WLS Compile
- Cloud Compile

These logs are text documents that you can either open or save, and they are the result of background jobs that ran against the application to check relevant information, for example, to determine whether the file contains a virus or could otherwise cause problems. These background jobs run asynchronously.

### Viewing Service and Application Logs on JCS-SaaS Extension Control

All application logs, along with certain service logs, are available for troubleshooting purposes by using either the Oracle Java Cloud Service - SaaS Extension Control or with the command-line interface provided through Oracle Java Cloud Service - SaaS Extension SDK.

To view service/application log messages with Oracle Java Cloud Service - SaaS Extension Control, click the View Log Messages link in the Servlets and JSPs area of the Performance Summary pane. That link displays the Log Messages - `<service-name>` page, where you can select a date range, message type, and so on, and can view or export messages to a file, as shown in the following illustration.

To view log messages:

1. From Oracle Java Cloud Service - SaaS Extension Control, click the View Log Messages link in the Servlets and JSPs area of the Performance Summary pane.

2. If necessary, Expand Search and specify the date range, message types, message text, and application name.
   
   The Log Messages table appears.
3. To view the details of a message, select the message.

By default, the messages are sorted by time, in ascending order. You can sort the messages by the any of the columns, such as Message Type, by clicking the column name.
Administering Instances with the JCS-SaaS Extension Administration Console

The Oracle Java Cloud Service - SaaS Extension Administration Console is the management console that enables you to deploy and monitor hosted applications.

Use Oracle Java Cloud Service - SaaS Extension Administration Console to:

- View the list of services you own, including the metrics and availability of each service
- Deploy and redeploy applications and libraries
- Start and stop applications on any of the services you own
- View job logs to see recent activity
- Restart your service instance,
- Manage system properties
- Manage data source alias
- Manage SSL and WSS certificates
- Manage credential stores
and perform many other management tasks.

The primary JCS-SaaS Extension Administration Console user is the service administrator. The service administrator has privileges to access Oracle Java Cloud Service - SaaS Extension Administration Console and can deploy and redeploy applications, as well as start and stop deployed applications.

The metrics shown on JCS–SaaS Extension Administration Console include performance (availability and metrics), CPU/heap usage for the service, and response and load charts. It also shows available applications and libraries. You use this tool to deploy/redeploy and start/stop applications. As shown in the next illustration, the Oracle Java Cloud Service - SaaS Extension Control is divided into regions. To access any page, click its name in the left navigation panel.

Topics:

- Administration Console Keyboard Shortcuts
- Working with the Java Cloud Service - SaaS Extension Administration Console
- Restarting a Java Service Instance
- Managing Applications with the JCS-SaaS Extension Administration Console
- Managing Shared Libraries with the JCS-SaaS Extension Administration Console
Administration Console Keyboard Shortcuts

You can run functions on the JCS-SaaS Extension Administration Console by using specific keyboard shortcuts.

**How do I edit settings by using the keyboard?**

Once you reached the lists of settings, from the keyboard, do the following:

1. Use Up and Down Arrow keys to select the desired setting.
2. Press F2 to activate the control to expand the row.
3. Press Enter or Ctrl + Right Arrow key to expand the row of the setting. **Note:**

   - If a screen-reader is running, the key combination used to expand the row might be different; for example, for some screen-readers, you have to press Alt + Enter to expand the row.
4. Press Esc to exit the mode that activates the control to expand the row.
5. Use Down Arrow to move to the row that contains the setting details.
6. Press F2 to activate the **Change Value** button.
7. Press the space bar to activate the button. The value of the setting will become an edit control. Move to the edit control by pressing Shift + Tab.
8. Change the value of the setting and press Tab to move to the **Set Value** button.
9. Press space bar to activate the button, then press Esc to exit the edit mode.

**How do I perform operations (start, stop, redeploy, etc.) on applications by using the keyboard?**

Once you've displayed the lists of applications, follow these steps to perform actions on an application:

1. Use Up and Down Arrow keys to select the desired setting.
2. Press F2 to enable the controls (that is, buttons to start, stop, redeploy, and links for application’s URL).
3. Use tabs to move to the desired control.
4. Activate the control pressing Enter (for links and buttons) or space bar (for buttons).
5. Press F2 to exit the edit mode.

**How do I review the job log details by using the keyboard?**

Once you reached the lists of job logs, follow the next steps to review the details using the keyboard:

1. Use UP and DOWN arrow keys to select the log entry.
2. Press F2 to enable the Details link.
3. Press ENTER to activate the link.
4. Press ESC to exit the edit mode.

Working with the Java Cloud Service - SaaS Extension Administration Console

Oracle Java Cloud Service - SaaS Extension Administration Console is a multi-page console that provides statistics on performance, data sources, response and load, JCS–SaaS Extension jobs, applications, and resource usage. It also allows you to deploy applications and libraries.
To access any page, click its name in the left navigation panel of the JCS–SaaS Extension Administration Console.

The metrics shown represent the most recent values available. For all "per minute" statistics, they are per minute in the last five minutes.

See the following topics:
Viewing Basic Service Information on the Home Page

Viewing and Managing Application Information on the Applications Page

Viewing and Managing Shared Library Information on the Libraries Page

Managing System Properties on the System Properties Page

Managing Data Source Alias on the Data Source Alias Page

Managing Certificates on the SSL Certificates Page

Managing Certificates on the WSS Certificates Page

Managing Credentials on the Credential Stores Page

Viewing Metrics on the Metrics Page

Viewing System Configurations on the Settings Page

Viewing HTTP Transactions on the Access Logs Page

Viewing Application Logging on the Application Logs Page

Viewing Service Activity on the Job Logs Page

Resolving Performance Issues

If you notice performance issues with your application, upgrading the underlying JCS - SaaS Extension instance may help resolve the problem. Some indications of performance issues include:

• Response times are slow or request throughput is high
• Server CPU or Java Heap memory usage is high.

Additional information is available in the service logs listed above.

Viewing Basic Service Information on the Home Page

When you open the JCS - SaaS Extension Administration Console, the first page you see is the Home page, which shows general information about your service instance.
This page displays basic information about a service along with usage metrics for a single JCS - SaaS Extension. All metrics represent the most recent values available. The page also contain the **Restart Service Instance** button.

**Service Identification Information**

The service identification information shown on the Home page is:

- Service instance name
- Identity domain
- State
- Size
- Associations

**General Information**

General Information includes:

- Service Version; the version number of the running JCS-SaaS Extension.
- Customer Disk Usage; the amount of disk space, in megabytes, the customer has consumed.

**JDBC Usage**

JDBC usage statistics include:

- Open JDBC Connections; Number of JDBC (Java Database Connectivity) connections currently in use
- JDBC Connection Creates (per minute); Number of database connections created per minute in the last five minutes

Servlets and JSPs

Servlet and JSP statistics include:
- Active Sessions; Number of active sessions
- Request Processing Time (ms); Average number of servlet or JSP (Java Service Pages) invocations per minute in the last five minutes
- Requests (per minute); Time spent processing request

Restart Service Instance

This page also contains the Restart Service Instance function. If the JCS - SaaS Extension instance somehow reaches an inconsistent state, for example, due to a network error, you can use the Restart Service option to restart the service, which will restart all the managed nodes in the service's domain.

See Restarting a Java Service Instance.

Viewing and Managing Application Information on the Applications Page

Use this page to analyze and manage applications available for this service.
Applications

forwarding-app-rsh
Status: Active
URLs:
/jcsadmin

wls-management-services1
Status: Active
URLs:
/wsunt-logger

forwarding-app
Status: Prepared

welcome-app
Status: Active
URLs:
/ (root)

Showing 4 applications
This page shows important statistics about applications deployed to your service and provides functionality to manage these applications. Applications might or might not be running, states that you control from this page. You can also deploy and delete applications from this page.

**Application Management Functions**

You can perform these page-scoped functions (that is, other than deployment, functions that are not specific to a particular application):

- **Deploy New**: deploy a new application to the service instance. See Deploying an Application.
- **Refresh**: reloads the application list with the latest data from the server.
- **Change your display setting**: change the display to sort applications by usage and to exclude stopped applications.
- **Filter Content**: show just those applications that meet a specified filtering criteria.

**Application Statistics**

This page shows the following application statistics:

- **Status**: the status of the application, Active or Prepared.
- **URLs**: links for the application module’s different URLs. For example, if this was a simple web application, this would be the link to access it. In the case of enterprise applications, more than one URL might be displayed.
- **Active Sessions Count**: the number of active sessions on that application.
- **Request Processing Time**: average amount of time to process a request.
- **Request Count**: number of requests generated by the application.

**Application Functionality**

For each application deployed on your account, you can perform the following function (select the application line item to reveal the function controls).

- **Redeploy**: use this option when you upgrade or make changes to an application.
- **Start and Stop**: you can easily start and stop an application. Use these options after you have deployed the application.
  
  **Note**: If these applications are exposed to your external users, you need to inform these users of potential downtime when you stop the application.

- **Delete Application**: when an application is no longer of value, delete it from the JCS - SaaS Extension Administration Console.
- **Download**: downloads the application archive to your file system.

See Managing Applications.

**Viewing and Managing Shared Library Information on the Libraries Page**

Use this page to see the list of libraries available for this service.
This page shows important statistics about libraries deployed to your service and provides functionality to manage these libraries. You can also deploy and delete libraries from this page.

**Application Management Functions**

You can perform these page-scoped functions (that is, other than deployment, functions that are not specific to a particular library):

- **Deploy New**: deploy a new library to the service instance. See Deploying a Library.
- **Redeploy**: refresh the currently deployed library with a new version of the library file (that is, an updated binary file).
- **Delete**: remove a library from your service instance. A deleted library can be redeployed. See Deleting a Library.
- **Refresh**: Refreshes—or reloads—the page.
- **Filter Content**: show just those applications that meet a specified filtering criteria.
Library Data

Libraries deployed (or deployed and undeployed) are presented on a table in the main body of the page. Data displayed for each library includes:

- Name; the library name.
- Type; the archive type.
- Status; the current status of the shared library, either active or inactive.
- Spec Version; the specification on which the shared library is based.
- Impl Version; Implementation version of the shared library.
- Deployment Type; whether or not the shared library is read-only or writable. The accepted values are Read Only and Custom. You can modify only custom libraries.
- Referencing Applications; the names of all applications on your service instance that reference the shared library.

You can sort column data in either ascending or descending order. To do so, right-click anywhere on the table and select either Ascending or Descending, per your requirements.

Using the Libraries Page

For instructions on working with libraries, see Managing Shared Libraries.

Managing System Properties on the System Properties Page

This page allows you to manage the system properties applicable to your service instance.

System Properties

Main page functionality includes:
• **Add New**, adds a new property:
  1. Click **Add New** to open the Add New system property dialog.
  2. Enter the system property Name and Value and click **OK**.

• **Delete**, removes the selected property:
  1. Select the selected property you want to delete and click **Delete**.
  2. Click **OK** to confirm the deletion.

Other controls on the page also allow you to refresh the page and list properties based on specific filtering criteria. System properties are displayed in a table as a key:value pair.

**Managing Data Source Alias on the Data Source Alias Page**

Use this page to manage the data sources for your service instances.
Data source alias are shown by name on a table, along with their read-only status (true or false). Page functionality includes:

- **Data Sources;** use the drop-down control to select the data source you want to manage.
- **Add New;** adds a new data source.
  1. Click **Add New** to open the Add New data source dialog.

  ![Add New data source dialog](image)

  2. Enter the Alias name and click **OK**.

- **Delete;** removes the selected data source:
  1. Select the data source alias you want to delete and click **Delete**.
  2. Click **OK** to confirm the deletion.

Other controls on the page also allow you to refresh the page and list aliases based on specific filtering criteria. Data source aliases are displayed in a table as a key: value pair.

### Managing Certificates on the SSL Certificates Page

Use this page to view and manage the SSL certificates.
This page allows you to add and remove SSL truststore certificates to a service instance. Once added, you can view important information about each certificate. Adding an SSL certificate to the SSL truststore requires a service restart for the changes to take effect. Oracle does not support the use of special characters in the alias name of WebLogic SSL truststore certificates.

### Page Functionality

You can perform the following tasks on this page:

<table>
<thead>
<tr>
<th>Alias</th>
<th>Expiry</th>
<th>Serial Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>digicertassuredidrootca</td>
<td>Mon Nov 10 00:00:00 GMT 2031</td>
<td>1715471793412058786</td>
</tr>
<tr>
<td>comodorsaca</td>
<td>Mon Jan 18 23:59:59 GMT 2038</td>
<td>1019090845375820333</td>
</tr>
<tr>
<td>thawtepremiumserverca</td>
<td>Fri Jan 01 23:59:59 GMT 2021</td>
<td>7187247466323322592</td>
</tr>
<tr>
<td>swisssignplatinumg2ca</td>
<td>Sat Oct 25 08:36:00 GMT 2036</td>
<td>567059532396054351</td>
</tr>
<tr>
<td>swisssignsilverg2ca</td>
<td>Sat Oct 25 08:32:46 GMT 2035</td>
<td>5700383053117599563</td>
</tr>
<tr>
<td>identrustdtx3</td>
<td>Thu Sep 30 14:01:15 GMT 2021</td>
<td>912997355753395335</td>
</tr>
<tr>
<td>equifaxsecurebusinessca1</td>
<td>Mon Jun 22 04:00:00 GMT 2020</td>
<td>23011</td>
</tr>
<tr>
<td>securetrustca</td>
<td>Mon Dec 31 19:40:55 GMT 2029</td>
<td>171997745891252778</td>
</tr>
<tr>
<td>digicertassuredidg3</td>
<td>Fri Jan 15 12:00:00 GMT 2038</td>
<td>1545931298100855373</td>
</tr>
<tr>
<td>utnuserfirstclientauthemailca</td>
<td>Tue Jul 09 17:36:58 GMT 2019</td>
<td>9137429454288470402</td>
</tr>
</tbody>
</table>
• **Add New**, add a new certificate:
  1. Click **Add New** to open the Add New dialog for SSL certificates.
  2. Enter an Alias then click **Choose File**.
  3. Navigate to the file location for the desired certificate, select the certificate, and click **OK** (or whatever dismisses the file system's Find dialog).

    The selected certificate name appears in the Add New dialog's **Certificate** field.
  4. Click **OK**.

    The certificate is added and appears on the certificates table.

• **Delete**, delete a certificate:
  1. Select the certificate you want to delete and click **Delete**.
  2. Click **OK** to confirm the deletion.

• **Download**, downloads a selected certificate to your local file system.

Other controls on the page also allow you to refresh the page and list aliases based on specific filtering criteria (Filter Content). Data source aliases are displayed in a table as a key:value pair.

**Certificate Details**

The body of this page shows the details of the SSL truststore certificates applied to the service instance.

- **Alias**, the certificate’s alias.
- **Expiry**, the date the certificate expires.
- **Serial Number**, the serial number that identifies the certificate.
- **Distinguished Name**, a complex name for the certificate. This name identifies such the certificate authority, the certificate organization, that organization’s URL, and the certificate authority’s location.

**Supported Certificate File Formats**

Java Cloud Service - SaaS Extension supports three certificate file formats:
• Single certificates appear in DER binary format.
• One or more certificates can appear in PEM format. The base64-encoded content of certificates is printed in base64-encoded format with the alias for each certificate shown in custom headers in the .pem file itself; for example:

  alias: orakey

  -----BEGIN CERTIFICATE-----
  MIICXjCCAcegAwIBAgI1H1Lb185PqPEwDQYJKoZIhvcNAQEFBQAwVzETMBEGCgmS
  JomT8ixkARKA2WNvTEVMBCQgCSmJS0mT8ixkARKWBM9yYWNsZTEVMBMGCSmJS0mT
  8ixkARKWBNs3bVbMR4wDwYDVQDEQwDdBG912DIQTAeFw0xNExx1XeNz15bD1jdBw
  Fw0yNExxMTgxFNhZ5NDNaWgKxEzARGboJkiaJk/I5ZAEZFqNjbj20xFJABgoJkiaJk
  k/IsZAEZFg2vcmFjBGUXTATBoGkiaJk/IsZAEZFgVjbG91IZDejMCEGAwawa
  dXNvcmFsY2V0cmhbeXWnjA4XKphdmmFzdumMgwZ8wDQYJKoZIhvcNAQEBBQADAgQO
  M1GJaG0BAIldnsUASWc309fjpsLT04XPIHToK9Hy6wn1WydX8hP3K+epNs77rd
  ICvHz1m5v1wOKSyD0CNrByf4pJFDDuO2Ff5PGy+aAdoQkhdIpwhWzQRSkyuDFeh
  8FCyQROJUSbP80dxLbLyGBbiuODz2bfL2fSuZiu/gu/pkIuTVxUpAgMAAgaJITAf
  MBGSAldqWQBBcRdY513M3s1LBDK14yyrmW0SjTANBqkhkiG9w0BAUQFAMBQg
  Q3A3kPcj2voos93UD3w7J7QkS3FRTYksWcBA81XQLXp050WCJ0y/NV/P
  GYDAXDzmS8H7HvW+jI/CpAYNiSeGD0Rzg8f6uub76Ny5A97ULckLHTCNJNIKWi
  0P1ScXS93ka7X6zUmLjgWVOURl/oT/1iqy/PR7z5fttL7==
  -----END CERTIFICATE-----
  alias: entrustpremium

  -----BEGIN CERTIFICATE-----
  MIICXjCCAcegAwIBAgI1H1Lb185PqPEwDQYJKoZIhvcNAQEFBQAwVzETMBEGCgmS
  JomT8ixkARKA2WNvTEVMBCQgCSmJS0mT8ixkARKWBM9yYWNsZTEVMBMGCSmJS0mT
  8ixkARKWBNs3bVbMR4wDwYDVQDEQwDdBG912DIQTAeFw0xNExx1XeNz15bD1jdBw
  Fw0yNExxMTgxFNhZ5NDNaWgKxEzARGboJkiaJk/I5ZAEZFqNjbj20xFJABgoJkiaJk
  k/IsZAEZFg2vcmFjBGUXTATBoGkiaJk/IsZAEZFgVjbG91IZDejMCEGAwawa
  dXNvcmFsY2V0cmhbeXWnjA4XKphdmmFzdumMgwZ8wDQYJKoZIhvcNAQEBBQADAgQO
  M1GJaG0BAIldnsUASWc309fjpsLT04XPIHToK9Hy6wn1WydX8hP3K+epNs77rd
  ICvHz1m5v1wOKSyD0CNrByf4pJFDDuO2Ff5PGy+aAdoQkhdIpwhWzQRSkyuDFeh
  8FCyQROJUSbP80dxLbLyGBbiuODz2bfL2fSuZiu/gu/pkIuTVxUpAgMAAgaJITAf
  MBGSAldqWQBBcRdY513M3s1LBDK14yyrmW0SjTANBqkhkiG9w0BAUQFAMBQg
  Q3A3kPcj2voos93UD3w7J7QkS3FRTYksWcBA81XQLXp050WCJ0y/NV/P
  GYDAXDzmS8H7HvW+jI/CpAYNiSeGD0Rzg8f6uub76Ny5A97ULckLHTCNJNIKWi
  0P1ScXS93ka7X6zUmLjgWVOURl/oT/1iqy/PR7z5fttL7==
  -----END CERTIFICATE-----

• One or more certificates can appear in JKS format. Multiple certificates appear in the JKS file, listed by their respective aliases, but are otherwise in binary format.

Managing Certificates on the WSS Certificates Page

Manage Oracle Web Services Management (OWSM) security policies (WSS) for an OWSM truststore used for web services with OWSM policies from the WSS Certificates page.
This page allows you to add and remove WSS truststore certificates to a service instance. Once added, you can view important information about each certificate. Adding a WSS certificate to the WSS truststore requires a service restart for the changes to take effect. Oracle does not support the use of special characters in the alias name of WebLogic WSS truststore certificates.

**Page Functionality**

You can perform the following tasks on this page:

- **Add New**: add a new certificate:
  1. Click **Add New** to open the Add New dialog for WSS certificates.
2. Enter an Alias then click Choose File.

3. Navigate to the file location for the desired certificate, select the certificate, and click OK (or whatever dismisses the file system Find dialog).

   The selected certificate name appears in the Add New dialog.

4. Click OK.

   The certificate is added and appears on the certificates table.

   - **Delete**: delete a certificate:
     1. Select the certificate you want to delete and click Delete.
     2. Click OK to confirm the deletion.

   - **Download**: downloads a selected certificate to your local file system.

Other controls on the page also allow you to refresh the page and list certificates based on specific filtering criteria.

**Certificate Details**

The body of this page shows the details of the WSS truststore certificates applied to the service instance.

- **Alias**: the certificate's alias.
- **Expiry**: the date the certificate expires.
- **Serial Number**: the serial number that identifies the certificate.
- **Distinguished Name**: a complex name for the certificate. This name identifies such the certificate authority, the certificate organization, that organization's URL, and the certificate authority's location.

**Managing Credentials on the Credential Stores Page**

Use this page to manage the credential stores for your service instances.
Credential Stores are shown by name on a table, along with their read-only status (true or false) Page functionality includes:

- **Credential Map**: use the drop-down control to select the credential map you want to manage. User credentials are stored in any of three maps:
  - **user.custom.map**: for storing credentials to which user applications have access. All the authenticated users assigned the role UserMapAccessRole will be able to read credentials from this map.
  - **oracle.wsm.security**: for storing credentials that web service clients can use.
  - **user.public.map**: for storing credentials to which user applications can have anonymous access. Users will be able to read credentials from this map without authentication.

- **Add New**: adds a new credential to the credential map:
  1. Click **Add New** to open the Add New credential map dialog.
2. Enter the following and click **OK**.
   - **Key**: descriptive name to use when retrieving the credential.
   - **User and Password**: users and passwords stored in the credential map that a customer application can retrieve by using the specified Key to authenticate to a service.
   - **Description**: an optional description of the credential map.

3. Click **OK**.

   • **Delete**: removes the selected credential:
   1. Select the credential you want to delete and click **Delete**.
   2. Click **OK** to confirm the deletion.

Other controls on the page also allow you to refresh the page and list credentials based on specific filtering criteria.

**Viewing Metrics on the Metrics Page**

The Metrics page is a view-only page that shows storage, application, and data source statistics for a service instance.
The Metric page is divided into two sections, one side of the page contains links to the three informational components while the other side shows the actual metrics for the selected component:

- **Storage**: select Storage to see a pie chart depicting how the service instance is using customer disk storage (as shown in the preceding example).

- **Applications**: select Applications to open a list of all applications used by the service instance. Next, select the desired application. A table appears containing the name, description, and value that application.
Data presented is:

- Active Sessions Count; that is, the number of current active sessions.
- Request Processing Time; that is, the average amount of time it takes a request to process.
- Request Count; that is, the number of requests.

You can also select Servlet Metrics to see the same data for each servlet running on the application.

- Data Source Metrics; select Data Sources to open a list of all data sources used by the service instance. Next, select the desired data source. A table appears containing the name, description, and value for the data source:
Open JDBC Connections Count; that is, the current number of open JDBC connections.

Connections Total Count; that is, the current total number of connections, both open and closed.

JDBC Connection Create Count; that is, number of JDBC connections created.

Viewing System Configurations on the Settings Page

Use this page to view information about the system configurations in your JCS-SaaS Extension service instance.
This page lists all configuration available to your service instance. Clicking any setting name will display these details for that setting:
• Description; a description of the configuration.
• Type; the data type (string, integer, boolean, etc.)
• Label; the unit of the data, (seconds, MB, etc.)
• Restart Required; true or false value indicating whether or not the service needs to be restarted for any update to take effect.
• Readable; true or false value indicating whether or not the configuration can be read by other applications.
• Writable; true or false value indicating whether or not the configuration can be updated by other applications.
• Current Value; the current value for the configuration. For example, if this setting was a client alias, as in the example, this would be the name used for that alias.

This is the only value you can change. To do so, click Change value, enter the new value, and click Set new value.

---

Viewing HTTP Transactions on the Access Logs Page

WebLogic Server keeps a log of all HTTP transactions in a text file called access.log. Use this page to see a list of these logs.
The Access Logs page shows all access.log updates for your service in list format. You can either see all updates at once or restrict the list by specifying filtering criteria. To filter, do the following:

1. Click Filters.
2. Add whatever filtering criteria you want to use:
   - **Http Method**: the type of operation for the request to complete. Acceptable values are POST, GET, PUT, and DELETE.
   - **Request URL Path**: the fully-qualified path for the request.
   - **Auth User**: the username associated with the requested log.
   - **Source IP**: the source IP address against which to filter.
   - **Status**: the Request Status Code against which you want to filter.
   - **Start Time**: the beginning date and time (mm/dd/yy hh:mm a/p) for a specified time range in which you want to search logs. Click within the edit box to open a calendar, from which you can select the date.
   - **End Time**: the end date and time (mm/dd/yy hh:mm a/p) for a specified time range in which you want to search logs. Click within the edit box to open a calendar, from which you can select the date.

3. Click **Apply Filters**.

**Viewing Application Logging on the Application Logs Page**

The Application Logs page displays the logged information of all deployed applications. Anything the deployed application logs will be displayed in this section. You can restrict results on this page by filtering by log severity level, application, and time range.
This page shows this logging information:

- **Time:** the time the message was logged.
- **Severity:** the severity level of the message. These levels are:
– Notification; the item logged is notification that doesn’t indicate an error condition.
– Warning; the item logged indicates a possible error condition or a condition that is anomalous to the status quo.
– Error; the item logged indicates that an error condition was encountered.
– Trace; chronological recordings of a service’s general activities. The trace log is created by configuring the service to make an explicit call using the trace logging custom XPath or Java API.
– Incident Error: errors that occur during a service’s activities.
– Unknown; the severity is not specified.

• Message Text; the actual message logged.

Restricting Results
You can restrict results by filtering the applications by certain criteria. Do the following:

1. Click Filters to display the filtering criteria.

   ![Application Logs](image)

   2. Enter the criteria by which you want to filter. You can use any combination of the following criteria:
    - Before; select the date before which you want to see message. Click the edit box to open a calendar and select the date there.
• After; select the date before which you want to see message. Click the edit box to open a calendar and select the date there.
• Contains; enter an string of text that the messages returned must contain.
• Application; enter the name of the application for which you want to see messages.
• Severity; click the drop-down control and select the severity level of the messages you want to see.

3. Click **Apply Filters**.
   The page refreshes and shows just those logged messages that meet the filtering criteria.

4. Click **Clear Filters** to remove existing filtering criteria.

**Viewing Service Activity on the Job Logs Page**

Use this page to track the activity of jobs run on your service instance.
Whenever you launch any job on your service instance, for example, an application deployment or service restart, a log is created that tracks the progress of that job. The Job Logs page displays all jobs you have run, including those in process. The page shows the jobs by job number, name, status, and start and end time. You can also click a link that opens a window containing the details of the log. From this window you can open a copy of the log and, if necessary, download it to your own file system for
later review (the log is downloaded as a .zip file). You can also download job logs from the Job Log page.

### Viewing and Saving Job Log Details

To see details about a specific job, do the following:

1. Locate the job on the jobs list and, in the Details column for that job, click **View Details**.

   The details window for the selected job appears.

---

#### Deploy Application: Benefits

<table>
<thead>
<tr>
<th>Name</th>
<th>Status</th>
<th>Start Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>app-validation</td>
<td>Success</td>
<td>Aug 29, 2017, 3:39:20 PM</td>
</tr>
<tr>
<td>whitelist-compile</td>
<td>Success</td>
<td>Aug 29, 2017, 3:39:20 PM</td>
</tr>
</tbody>
</table>

This window is labeled with a description of the job—for example, Deploy Application: Benefits—and shows, for each job process:

- **Name**: the generic name of the action performed by the job.
- **Status**: the status of the job, such as success, failure, or processing.
- **Start Time**: the time the job started.
- **End Time**: the time the job stopped.
- **Elapsed Time**: the time it took for the job to complete (hh:mm:ss)
- **Log**: contains the **View Log** link.

2. Click **View Log**.

   A window opens containing the content of the log for the process you selected; for example:
3. If you want to
   - Download a copy of the job log, click **Download**. A .zip file containing the log will download to your file system.
   - View the information in your browser, click **View log**. A window opens containing the log.

**Other Page Management Tasks**

You can perform these additional page management tasks:

- Select a job and click **Download** to download a .zip file copy of the job log to your file system.
- Restrict the number of results per page by changing the value in Logs to display.
- Click **Refresh Automatically** to have the page refresh at a regular interval.
- Click to refresh the page.

**Restarting a JCS-SaaS Extension Instance**

Use the Restart Service option to restart all managed servers in service's domain.

One example of when you might need to restart an instance might be when adding an SSL certificate to the *ssl trust-store* requires a service restart for the changes to take effect. See [Managing SSL Truststores](#).

When you restart a service, all servers in the service instance will be stopped and restarted. If the service instance only has a single server, this could result in downtime. If it has multiple servers, these servers will be restarted sequentially. Other operations are not permitted during a service instance restart.

To restart a JCS-SaaS Extension instance:

1. Navigate to the **Home** page.
2. Click **Restart Service Instance**.
   The Restart service instance dialog appears.
3. Click **OK** when you are ready to restart your service instance.

4. Go to the **Job Logs** page to verify that you successfully restarted the job. When the service restart begins, there will be a Restart Service Instance log entry in the Jobs log page and it will show a Status of Running. When the restart is complete, the Status column for the restart will indicate Completed.

5. To see the content of the log, locate it on the job list and click **View Details**.

6. You can either save the log to disk or view it immediately in your browser.
   - To download it to your file system, click **Download**.
   - To view it in your browser, click **View Log** (in the Log column of the dialog)

This is a sample log for restarting single managed-server service:

```
2014-07-08 15:15:07 PDT: Starting action "Restart Service"
2014-07-08 15:15:07 PDT: Restart Service started
2014-07-08 15:15:07 PDT: Restarting service: server restart timeout = 600000 ms ...
2014-07-08 15:15:07 PDT: Stopping server m0 : current state = RUNNING ...
2014-07-08 15:15:09 PDT: Server m0 stopped
2014-07-08 15:15:09 PDT: Starting server m0 : current state = SHUTDOWN ...
2014-07-08 15:16:16 PDT: Server m0 started
2014-07-08 15:16:16 PDT: "Restart Service" complete: status SUCCESS
```

**Restarting a Single Managed Server Service**

For single managed-server services, there may be some service downtime until the managed-server is restarted. Therefore, options like Application Deploy, Start, and Stop on the JCS - SaaS Extension Administration Console are disabled while a service restart is in progress.

**Restarting a Multiple Managed Server Service**

For multiple managed-server services, there should not be any service downtime because the managed servers are restarted sequentially, as follows:

- Restarts one server at a time.
• Waits for each server to boot up before executing restart on the next server.
• Restarts in sequence of m0, m1, m2, m3, etc.
• Time-outs after waiting for three minutes for a server to restart, and then triggers a restart on the next server.

Managing Applications with the JCS-SaaS Extension Administration Console

You can deploy, redeploy, and delete an application using the Applications page.

This illustration shows the Applications Page with a number of applications deployed. Note that the Welcome application (welcome-app) is pre-deployed to each service instance. It provides links to the SDK documentation, samples, blogs, and white papers, and more.

See the following topics for procedures for managing applications from the Application page.

• Deploying an Application
• Deleting an Application
• Redeploying an Application
• Starting and Stopping Applications

Deploying an Application

Deploy applications from the Applications page, which you can access from the Oracle Java Cloud Service - SaaS Extension Administration Console. Upon deployment, the application undergoes a series of security checks before it is actually deployed.

To deploy an application:

1. Open the Applications page in the Oracle Java Cloud Service - SaaS Extension Administration Console and click Deploy New. The Deploy Application page appears.
2. Enter a name for the application you are deploying, and then click **Choose File** to search your local file system for the application archive to be deployed. After locating the archive, click **OK**.

Oracle Java Cloud Service - SaaS Extension Control uploads and deploys the selected WAR (Web Application Archive) file or EAR (Enterprise Archive) file.

**Validating Deployed Applications**

Once deployed, each application undergoes a series of security checks before it is actually deployed. These checks include a virus scan and a whitelist validation. For technical and security reasons, a small number of specific APIs are prevented from executing in Oracle Cloud. Once the application passes these tests, deployment of the application begins. See [About the Application Deployment Validation Process and Run-time Security](#).

**Verifying Deployment Status**

When deployment is successful, the application appears on the list of applications with a Status of Active.

If a deploy job fails, the best course of action is to view the logs for the job. Depending on the different job logs available, you can determine where the failure occurred. See [Viewing the Activity Logs](#).

If whitelist validation fails, deploy is never run and so there will be no deploy log.

**Deleting an Application**

When an application is no longer needed, you can remove it from the Oracle Java Cloud Service - SaaS Extension Administration Console.

To delete an application:

1. Access the Applications page in the Oracle Java Cloud Service - SaaS Extension Administration Console.

2. Highlight the application you want to delete and click **Delete**.

3. On the deletion confirmation dialog, click **OK**.

**Redeploying an Application**

Redeploy an application whenever you upgrade or otherwise make changes to an application. In most situations you need to stop the application, make the necessary changes, and then redeploy the application.

To redeploy an application:

1. Access the Applications page in the Oracle Java Cloud Service - SaaS Extension Administration Console.

2. Highlight the application of interest and click **Redeploy**.
   
   The Deploy Application dialog appears.

3. Click **Choose File** to search your local file system for the application archive to be deployed.
4. After locating the archive, click **OK**.

Oracle Java Cloud Service - SaaS Extension uploads and deploys the selected WAR (Web Application Archive) file or EAR (Enterprise Archive) file.

Starting and Stopping Applications

Once an application is deployed, you can start and stop the application as needed. If the application is exposed to external users, inform them of potential downtime before stopping the application.

To start or stop an application:

**Note:** If this application is exposed to your external users, you need to inform these users of potential downtime when you stop the application.

1. Access the Applications page in the Oracle Java Cloud Service - SaaS Extension Administration Console.

2. Highlight the application of interest and click either **Start** or **Stop**.
Administering Instances with the JCS - SaaS Extension SDK

This section provides documentation about monitoring and managing applications deployed on Oracle Java Cloud Service - SaaS Extension by using the command line interface provided with the JCS-SaaS Extension SDK.

Topics:

- Downloading the Oracle Java Cloud Service - SaaS Extension SDK
- Using the Command-Line Interface to Monitor Oracle Java Cloud Service - SaaS Extension
- Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension

Downloading the Oracle Java Cloud Service - SaaS Extension SDK

The Oracle Java Cloud Service - SaaS Extension SDK (software development kit) is a downloadable package that provides command-line-based utilities that facilitate the management of Oracle Java Cloud Service - SaaS Extension instances and the development of applications for the Oracle Java Cloud Service - SaaS Extension.

The Oracle Java Cloud Service - SaaS Extension SDK is required if you want to integrate your service instance with one of the supported IDEs described in About Using Integrated Development Environments.

To learn more about the command-line interface available in the Oracle Java Cloud Service - SaaS Extension SDK, see Using the Command-Line Interface to Monitor Oracle Java Cloud Service - SaaS Extension and Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension.

Download the Oracle Java Cloud Service - SaaS Extension SDK.
Note:

Alternately, you can download the SDK from the Oracle Cloud home page by doing the following:

1. Go to the Oracle Cloud home page at https://cloud.oracle.com or to any Oracle Cloud page that has the Resources menu.
2. Click the Resources menu and, in the Support column, select for Developers

3. In the Downloads area, click Oracle Cloud Downloads. If prompted, sign in using your Oracle.com account credentials.

   The Oracle Cloud Downloads page appears. All relevant Oracle Cloud downloads will be accessible on the page.

4. Under Java Cloud Services, click Oracle Java Cloud Service - SaaS Extension SDK.

   This will take you to the Oracle Java Cloud Service - SaaS Extension SDK panel.

5. Click the Download Oracle Java Cloud Service - SaaS Extension SDK link.

Downloading software will require you to accept the license agreement, so click Accept License Agreement. Then, under Oracle Java Cloud Service - SaaS Extension SDK (release 16.3.3.0), start the download by clicking All Supported Platforms. Extract the downloaded zip file to your local system, preferably into its own directory. This directory will be referred to as the SDK_HOME.

- For installation and usage instructions, see index.html in the /doc directory
- For sample application installation and usage instructions, see samples.html in the /doc directory
Using the Command-Line Interface to Monitor Oracle Java Cloud Service - SaaS Extension

The Oracle Java Cloud Service - SaaS Extension Software Development Kit (SDK) provides a command-line interface (CLI) that exposes monitoring commands so you can monitor applications deployed on an Oracle Java Cloud Service - SaaS Extension instance.

The monitoring commands exposed through the CLI allow you to monitor applications deployed on an Oracle Java Cloud Service - SaaS Extension instance. Each command in the CLI initiates an asynchronously executed job within the Oracle Cloud for a specific Oracle Java Cloud Service - SaaS Extension instance. You can view the existing jobs for a specific Oracle Java Cloud Service - SaaS Extension, view status, and review associated log files.

Using the Command-line Interface

You can use the CLI to monitor an Oracle Java Cloud Service - SaaS Extension instance as follows:

- **Job Monitoring**. View the existing jobs for a service, view job status, and review associated log files. For more information on the job monitoring commands, see the SDK documentation for `list-jobs`, `list-job-logs`, `job-status` and the job creation commands such as `install`, `delete` and so on.

- **Server Monitoring**. List and describe the underlying servers. Each server is a dedicated Java Virtual Machine (JVM) for the Oracle Java Cloud Service - SaaS Extension instance and executes the applications that are deployed to that service instance. For more information on the server monitoring commands, see the SDK documentation for `query-service-metrics`.

- **Application Monitoring**. List the deployed applications and view their status within a service instance. For more information on the application monitoring commands, see the SDK documentation for `query-service-metrics` and `list-applications`.

- **Service Monitoring**. List the service instances within an identity domain and view their status. For more information on the service monitoring commands, see the SDK documentation on `describe-service-instance` and `list-service-instances`.

SDK Documentation

To learn more about the commands available in the CLI, see CLI Commands in the SDK or navigate to the `$SDK_HOME/doc/index.html` file. You can also access all the SDK documentation via the Welcome App. To do so:

1. In the Applications region of the Oracle Java Cloud Service - SaaS Extension Control, click `welcome-app`.
   
   The Application: welcome-app page appears.

2. In the Application URLs table, click the URL.
   
   The Oracle Java Cloud Service - SaaS Extension home page appears.

3. Click Oracle Java Cloud Service - SaaS Extension SDK.
The Oracle Java Cloud Service - SaaS Extension SDK Home page appears. From here, you can select the desired CLI documentation; for example, CLI-Javacloud.jar

Also see Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension.

Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension

The management commands exposed through the CLI allow you to perform various application management, file management, and service instance management tasks. Each command in the CLI initiates an asynchronously executed job within the Oracle Cloud for a specific Oracle Java Cloud Service - SaaS Extension instance.

Topics

- Streamlining Command Entry by Using javacloud.properties
- Enabling Email Notifications in JCS-SaaS Extension
- Managing Shared Libraries
- Managing Configurations
- Managing Credentials
- Managing Web Services Security Truststore
- Managing SSL Truststores
- Setting Up WSS Trust Between Two Instances or Between an On-premises WLS Domain and One Instance
- Managing System Properties
- Viewing Access Logs
- Viewing Service Logs
- Viewing Service Metrics
- Refreshing ADF Applications
- Synchronizing UI and SDK Data
- Accessing the Local File System
- Using the Application and Domain Configuration Shell
- CLI Commands in the SDK

Obtaining the Command Line Interface

By installing the Oracle Java Cloud Service - SaaS Extension SDK (software development kit), you have access to a command-line interface (CLI). Click here to download the Oracle Java Cloud Service - SaaS Extension SDK.

Alternately, you can download the SDK from the Oracle Cloud home page by doing the following:

1. Go to the Oracle Cloud home page or to any Oracle Cloud page that has the Resources menu.
2. Click the **Resources** menu and, in the Support column, select **for Developers**

3. In the Downloads area, click **Oracle Cloud Downloads**. If prompted, sign in using your Oracle.com account credentials.

   The Oracle Cloud Downloads page appears. All relevant Oracle Cloud downloads will be accessible on the page.

4. Under Java Cloud Services, click **Oracle Java Cloud Service - SaaS Extension SDK**.

   This will take you to the Oracle Java Cloud Service - SaaS Extension SDK panel.

5. Click the **Download Oracle Java Cloud Service - SaaS Extension SDK** link.

**Using the Command Line Interface**

You can use the CLI to manage an Oracle Java Cloud Service - SaaS Extension instance as follows:

- **javacloud.jar** – General application management tasks, such as install, remove, update, start, and stop.
- File System Access Shell – Basic file management commands to manage your local `/customer/scratch/` directory. See *Using the File System Access Shell*.
- Configuration Shell – General web service and WebLogic domain configuration tasks. See *Using the Application and Domain Configuration Shell*.

To learn more about the commands available in the CLI, navigate to the `$SDK_HOME/doc/index.html` file (where `$SDK_HOME` is the directory containing your Oracle Java Cloud Service - SaaS Extension installation). You can also access all of the SDK documentation via the "Welcome App". See *Using the Command-Line Interface to Monitor Oracle Java Cloud Service - SaaS Extension*.

**Streamlining Command Entry by Using javacloud.properties**

Setting command-line arguments in the `javacloud.properties` file streamlines command entry because, once the property file is defined, the arguments set in it do not need to be specified at the command line. Additionally, the property file path itself is implicit and does not need to be passed.

**Where to Store javacloud.properties**

You can store `javacloud.properties` in either
• **Your home directory**, which is returned by the System.getProperty( "user.home" ) call in Java. For example:
  - Windows: C:\Documents and Settings\<username>
  - Linux: /home/<username>
  - Mac: /Users/<username>

• The **current working directory**; that is the same directory where you execute java -jar javacloud.jar.

The version of the property file located in the current working directory takes the precedence over the one available in your home directory.

javacloud.properties Keys and Values

The keys are simply the argument names. These rules apply:

- You cannot specify argument shortcuts here.
- Arguments of multiple commands can be specified here.
- Unrecognized keys will be ignored.

The value specified is the value of the argument.

The keys can be specified in two forms:

- **Simple form**—The name of an argument is specified as it is; for example, identitydomain=mydomain. This is applicable to all the commands that takes the argument identitydomain.

- **Full form**—The argument name is specified along with the command name; for example, list-jobs.sorton=STATUS, where list-jobs is a valid command name and sorton is a valid argument supported by the command. This is applicable only to the command specified. The full form takes precedence over the simple form.

Specifying Passwords

You cannot specify password type arguments with javacloud.properties because the value for a password type argument is never read from this file.

Resolving Argument Values

The value of an argument is resolved in the following order:

1. Argument value, if specified in the command line.
2. Argument value, if specified in the property file.
3. Default value of the argument, if available. Use ./javacloud -<command> -help to list the default values.

If the value is not resolved for a mandatory argument, the command line will result in validation error.

Sample javacloud.properties file

```properties
#Note: Values specified here can be overridden on the command line.
user=your-user-name
```
Enabling Email Support in JCS-SaaS Extension

JCS-SaaS Extension's email notification feature allows you to use the JavaMail APIs in your application to send out emails. This feature uses an Oracle managed outbound mail gateway. Your application should not use or configure any SMTP mail gateway. The Oracle managed email gateway has quota limitations as described below.

Process for Enabling the Email Feature

To enable applications to trigger notification emails, complete the four tasks described in the following topics:

- **Task 1:** Create the User and Assign the Java_Notification_User Role
- **Task 2:** Set Credentials in the Application
- **Task 3:** Add Notification Triggering Code to the Application
- **Task 4:** Update weblogic.xml to Reference the Jersey JARs

Email Quota

Email quota limits the number of emails you can send and is determined by your service type. Before proceeding, determine which of the following service types you have:

<table>
<thead>
<tr>
<th>Service Type</th>
<th>Daily Message Quota</th>
<th>Usage Warning</th>
</tr>
</thead>
<tbody>
<tr>
<td>JAVA-BASIC-TRIAL</td>
<td>200</td>
<td>At 90%</td>
</tr>
<tr>
<td>JAVA-BASIC-PRODUCTION</td>
<td>1000</td>
<td>At 90%</td>
</tr>
<tr>
<td>JAVA-STANDARD-PRODUCTION</td>
<td>2000</td>
<td>At 90%</td>
</tr>
<tr>
<td>JAVA-ENTERPRISE-PRODUCTION</td>
<td>6000</td>
<td>At 90%</td>
</tr>
</tbody>
</table>

When you exhaust your email quota, you can no longer send emails. If you want to send more, increase your quota by upgrading your service.

Email Size Limit

The maximum size for an entire email is 1 MB, including both the body of the email and the attachments. If your email size exceeds this limit, then the email is not sent and an error message is added to the log file.
Prerequisites for Enabling the Email Feature

Before you can enable email notifications in your application, you need to meet certain prerequisites.

1. Enable applications to send Java Mail-based emails by using the set-config command against your instance to set the java.mail.enabler to true:

   ```
   ./javacloud -id myIDDomain62337 -si javatrial1870 -u jesse.essex@mycompany.com -set-config -name java.mail.enabler -value true
   ```

2. The Java Security Manager is disabled by default. If, for some reason, it is enabled, disable it by using the set-config command to set the jvm.standard.security.manager.enabled value to false:

   ```
   ./javacloud -id myIDDomain62337 -si javatrial1870 -u jesse.essex@mycompany.com -set-config -name jvm.standard.security.manager.enabled -value false
   ```

Task 1: Create the User and Assign the Java_Notification_User Role

To enable email notifications you’ll need to create a new user and the custom role Java_Notification_User and then assign that role to the new user.

To create a user and custom role and then assign the role, do the following:

> **Note:**

If you want to use an existing user (not recommended), start with step 4.

1. Login to My Services and click **Users** to open the Users tab.
2. On the Users tab, click **Add**.
   
   The Add User dialog appears.
3. Enter the specific identity data for the user and click **Add**.
   
   The Add User dialog closes and the new user appears on the Users tab.
4. Click Custom Roles to open the Custom Roles tab and click **Add**.
   
   The Add Custom Role dialog appears.
5. In Role Name, enter Java_Notification_User (optionally, give the role a display name—for example Java Notification User—and a short description). Then click **Add**.
   
   The Add Custom Role dialog closes.
6. Click Users to return to the Users tab and then click the menu icon for the new user and select **Manage Roles**.
   
   The Manage Roles dialog for the specific user appears.
7. In the Roles section, under Available Roles, select Java_Notification_User and click the right arrow to move it to the Assigned Roles list. Then click Save.

**Task 2: Set Credentials in the Application**

Next, you need to set the user credentials for the email user in the credential store and then configure the triggering application to retrieve them from the store.

**To set the user credentials in the application:**

1. **Use the** set-credential **CLI command to store the credentials in your credential store:**

   ```
   ./javacloud -set-credential -map user.custom.map -user userName -id identityDomain -si serviceInstance -keyuser keyUsername -keypassword keyPassword -key csfKey
   ```

   For example:

   ```
   ./javacloud -set-credential -map user.custom.map -user system -id myDomain123 -si myInstance -keyuser foo -keypassword foobar -key myCSFKey
   ```

   Storing the credentials in the credentials stores is optional (but recommended), but you can hardcode your user name and password, for example.

2. **Configure the Credential Store API in the triggering application to use keys associated with the credentials to get these credentials from the credential store.**

   **Use the following code snippet as an example:**

   ```java
   import oracle.security.jps.service.credstore.CredentialStore;
   import oracle.security.jps.service.credstore.CredentialFactory;
   import oracle.security.jps.service.credstore.Credential;
   import oracle.security.jps.service.credstore.PasswordCredential;
   
   public class CredentialStoreClassTest {

   public void CredentialStoreTest() {
   
   try {
   
   CredentialStore credentialStore =
   oracle.security.jps.service.JpsServiceLocator.getServiceLocator().lookup(CredentialStore.class);

   String map = "user.custom.map";
   String mykey = "mykey";
   try {
   
   System.out.println("Creating map "+map);
   credentialStore.setCredential(map, mykey,
   CredentialFactory.newPasswordCredential("user", "pwd".toCharArray()));
   System.out.println("Password set");
   
   } catch (Exception e) {
   e.printStackTrace();
   }
   
   try {
   System.out.println("Accessing credential");
   Credential cred = credentialStore.getCredential(map, mykey);
   
   }
   
   } catch (Exception e) {
   e.printStackTrace();
   }
   
   
   }
   
   }
   ```
System.out.println("Password got:" + cred);
if (cred != null) {
    System.out.println("Password got:" + new
    String(((PasswordCredential)cred).getPassword()));
} catch (Exception e) {
    e.printStackTrace();
} catch (Exception eg) {
    eg.printStackTrace();
}
msg.setSubject(subject);
msg.setSentDate(new Date());
msg.setText(msgText);
Transport transport = session.getTransport("smtp");
// Provide the user and password if you do not specify an authenticator when
creating the session
transport.connect(uname, pwd);
// If you provide an authenticator when creating the session, you do not
need user and password in next line
//transport.connect();
transport.sendMessage(msg, recipientAddress);
} catch (MessagingException e) {
    e.printStackTrace();
}

Task 4: Update weblogic.xml to Reference the Jersey JARs

The email feature internally uses REST API's to make a call to the Oracle managed
email gateway. This requires that your application references the JAX-RS 1.1 API
library (Jersey). This can be achieved by updating your application's weblogic.xml file.

To update weblogic.xml, add the following code snippet to the file

<library-ref>
  <library-name>jax-rs</library-name>
  <specification-version>1.1</specification-version>
  <implementation-version>1.9</implementation-version>
  <exact-match>false</exact-match>
</library-ref>

Managing Shared Libraries

The CLI enables you to deploy and manage any shared Java EE library and optional
package supported Oracle WebLogic Server.

The libraries you can manage include both out of the box libraries and optional
packages provided by Oracle Cloud, as well as any user-defined custom shared
libraries or optional packages that are packaged using the standard process as
supported by WebLogic Server.

This table describes the commands for managing the deployment of shared libraries
and optional packages.

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>list-libraries</td>
<td>Lists all the shared libraries that are installed and available in the service instance.</td>
<td>user, password, identitydomain, serviceinstance</td>
</tr>
<tr>
<td>describe-library</td>
<td>Describes a shared library identified by its name, spec version and impl version. The description includes the status, deploy type, type and the name of the applications that references this library.</td>
<td>user, password, identitydomain, serviceinstance, library, specversion, implversion</td>
</tr>
<tr>
<td>install-library</td>
<td>Installs a custom shared library that is available in the local disk.</td>
<td>user, password, identitydomain, serviceinstance, library, specversion, implversion</td>
</tr>
</tbody>
</table>
### Managing Configurations

You can use the `list-config` and `set-config` CLI commands to manage your JCS-SaaS Extension configuration.

**Note:**

You will need to download the latest JCS-SaaS Extension SDK to use this feature. See [Downloading the Oracle Java Cloud Service - SaaS Extension SDK](#).

JCS-SaaS Extension hides its internal implementation by simplifying the configuration into a simple key and a value, allowing you to view configurations and modify them by using the CLI to specify this name/value pair. Be aware that you can only modify a limited set of configurations; you can see the modifiable configurations by using the `list-config` command.

To manage JCS-SaaS Extension configurations, use these commands:

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
</table>
| list-config | Creates a list of all available configurations. To see a full list of configurations, you must include the `-showvalues (-s)` command. | user, password, identitydomain, serviceinstance. Three other arguments you need to consider are:  
  - showvalues (alias: sv)  
    When set, the value for each configuration is fetched; **Default**: false.  
  - search (alias: s)  
    Limits the configurations listed to just those containing the string specified with this option.  
  - verbose (alias: v)  
    The true/false flag that indicates if the listing should be “verbose” (full-format); **Default**: false. |
### Command Description Mandatory Arguments

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>set-config</td>
<td>Allows you to set any of the available configurations by specifying the configuration name—obtained from the list-config command—and appropriate configuration value.</td>
<td>user, password, identitydomain, serviceinstance, name, value</td>
</tr>
</tbody>
</table>

### To manage configurations:

**Note:**

To simplify command entry, you can store values for all parameters except `-password (-p)` in the `javacloud.properties` file. See Using `javacloud.properties`.

1. **Use** `list-config` to display the properties you can configure:

   ```bash
   ./javacloud -list-config -showvalues -identityDomain myIdentityDomain -serviceInstance myServiceInstance -userName myUserName -verbose
   ```

   The system responds with a list similar to this:

   ```bash
   #=======================================================================
   #========================================================================
   #===================================#===================================
   #| Listing 20 Simple Configs |
   | [Identity Domain=jcsdc, Service Instance=javas2] |
   #=======================================================================
   #========================================================================
   #=#===================#==========#======#=======#========#==============
   #| Name        |Value Type|Readabl|Writabl|Required|Description |
   #| Value |Value | Restart | |
   #=#===================|==========|======|=======|========|
   #========================================================================
   #========================================================================
   ```
<table>
<thead>
<tr>
<th>Field</th>
<th>Type</th>
<th>Required</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. jta.transaction.timeout</td>
<td>INTEGER</td>
<td>Y</td>
<td>Y</td>
<td>JTA Transaction</td>
</tr>
<tr>
<td>2. alias</td>
<td>STRING</td>
<td>Y</td>
<td>Y</td>
<td>enabled endpoint</td>
</tr>
<tr>
<td>3. servers.status</td>
<td>STRING</td>
<td>Y</td>
<td>Y</td>
<td>Managed Server Status(es)</td>
</tr>
<tr>
<td>4. size</td>
<td>INTEGER</td>
<td>Y</td>
<td>Y</td>
<td>Maximum value</td>
</tr>
</tbody>
</table>

Note: The table provides a summary of the command-line interface options for managing Oracle Java Cloud Service - SaaS Extension. The options include settings for transaction timeouts, alias management, server status, and JVM arguments.

Chapter 8
Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension

---

ORA7CE

---

Page 8-14
<table>
<thead>
<tr>
<th></th>
<th>jvm.arg.stack.size</th>
<th>INTEGER</th>
<th>Y</th>
<th>Y</th>
<th>Y</th>
<th>Minimum value : 228 (KB)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>jvm.standard.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Standard Java Security manager enabled?</td>
</tr>
<tr>
<td></td>
<td>security.manager.</td>
<td>BOOLEAN</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Sun HTTP handler enabled?</td>
</tr>
</tbody>
</table>
|   | minimum.protocol.  | STRING  |   Y   |   Y   |   Y    | x is an integer between 1 and 9,
inclusive

<table>
<thead>
<tr>
<th>version</th>
<th>inclusive</th>
</tr>
</thead>
<tbody>
<tr>
<td>y</td>
<td></td>
</tr>
</tbody>
</table>

is an integer between 0 and 9, inclusive

e.g.

TLSv1.2

"" : The value of "" indicates the minimum protocol version is not set and default value will be used

This table shows the name of the configurations along with other information critical to making configuration decisions. The **Value** column shows the information you can change with the `set-config` command. Those you can change are denoted by a **Y** in the **Value Writable** column. Note that some values can’t be changed and others will require system restart for the change to occur.

The configurations you can modify are:

<table>
<thead>
<tr>
<th>Name</th>
<th>Value Writable</th>
<th>Description</th>
<th>Specif y As</th>
</tr>
</thead>
<tbody>
<tr>
<td>credential.custom.map.enabled</td>
<td>Boolean</td>
<td>Specifies whether the Credential Store custom map is enabled. When value is set to true, all authenticated users with the role UserMapAccessRole will be able to read credentials from this map. Ensure UserMapAccessRole role is available before using the set-config command to enable or disable this map.</td>
<td>true</td>
</tr>
<tr>
<td>credential.public.map.enabled</td>
<td>Boolean</td>
<td>Specifies whether or not the Credential Store public map is enabled. <strong>Note:</strong> Setting this value to true permits unauthenticated users to manage the user.public.map credential map.</td>
<td>true</td>
</tr>
<tr>
<td>Name</td>
<td>Value Type</td>
<td>Readable</td>
<td>Writable</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------</td>
<td>----------</td>
<td>----------</td>
</tr>
<tr>
<td><code>internal.datasource.opss.global.txn</code></td>
<td>String</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><code>java.mail.enable</code></td>
<td>Boolean</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td><code>jdbc.datasource.idle.trust.seconds</code></td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>Name</td>
<td>Value Type</td>
<td>Readable</td>
<td>Writeable</td>
</tr>
<tr>
<td>------------------------------</td>
<td>------------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>jdbc.datasource.inactive.timeout</td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>jdbc.datasource.initial.capacity</td>
<td>Integer</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>jdbc.datasource.max.capacity</td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>jdbc.datasource.min.capacity</td>
<td>Integer</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>Name</td>
<td>Value Type</td>
<td>Readable</td>
<td>Writeable</td>
</tr>
<tr>
<td>-------------------------------------------</td>
<td>------------</td>
<td>----------</td>
<td>-----------</td>
</tr>
<tr>
<td>jdbc.datasource.shrink.frequency</td>
<td>Integer</td>
<td>Y</td>
<td>N</td>
</tr>
<tr>
<td>jdbc.datasource.Statement.setTimeout</td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>jdbc.datasource.Timeout</td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>jta.transaction.Timeout</td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>jvm.arg.max.perm.size</td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
</tr>
</tbody>
</table>
### Chapter 8
Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension

<table>
<thead>
<tr>
<th>Name</th>
<th>Value Type</th>
<th>Readable</th>
<th>Writeable</th>
<th>Restart</th>
<th>Description</th>
<th>Specify As</th>
</tr>
</thead>
<tbody>
<tr>
<td>jvm.arg.reserved.code.cache.size</td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Determines the maximum size of the code cache. • <strong>Minimum value</strong>: 48 (MB) • <strong>Maximum value</strong>: 240 (MB) A negative value indicates that maximum size is not set; that is, it's the default. The default depends on the JRE version on which your application is running. Refer to the Welcome app on your instance to learn your server's JRE version.</td>
<td>MB</td>
</tr>
<tr>
<td>jvm.arg.stack.size</td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>JVM's Stack Size argument value. Negative value indicates that the value is not set. ie) the default.</td>
<td>KB</td>
</tr>
<tr>
<td>jvm.arg.use.code.cache.flushing</td>
<td>String</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Turns code cache flushing on and off. When enabled, the code cache flushing is triggered when the memory available in the code cache is low. You must enable code cache flushing if you constrain the code cache. If flushing is disabled and the code cache is full, the JIT compiler will not compile methods. • Accepted values are on and off. • The value of &quot;&quot; indicates that code cache flushing is not set and the default value will be used. The default depends on the JRE version on which your application is running. Refer to the Welcome app on your instance to learn your server's JRE version</td>
<td>on</td>
</tr>
<tr>
<td>jvm.standard.security.manager.enabled</td>
<td>Boolean</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Specifies whether or not the standard JVM security manager is enabled.</td>
<td>true</td>
</tr>
</tbody>
</table>
### Table 8-1: Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension

<table>
<thead>
<tr>
<th>Name</th>
<th>Value Type</th>
<th>Readable</th>
<th>Writeable</th>
<th>Restart</th>
<th>Description</th>
<th>Specific As</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>logging.system.loggers.enabled</code></td>
<td>Boolean</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Allows you to enable JDK system loggers (such as oracle.wsm) in runtime. When this value is set to true, system loggers will be enabled and you will be allowed to set logging levels in runtime. Setting this value to false disables the loggers.</td>
<td>true</td>
</tr>
<tr>
<td><code>security.SSL.minimum.protocol.version</code></td>
<td>String</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Minimum version of SSL or TLS protocols enabled for SSL connections. • SSLv3: Specifies SSL V3.0 as the minimum protocol version enabled in SSL connections. • TLSv1: Specifies TLS V1.0 as the minimum protocol version enabled in SSL connections. • TLSvx.y: Specifies TLS Vx.y as the minimum protocol version enabled in SSL connections, where: – x is an integer between 1 and 9, inclusive. – y is an integer between 0 and 9, inclusive; for example, TLSv1.2. • “”: The value of “” indicates the minimum protocol version is not set and default value will be used.</td>
<td>Alphanumeric</td>
</tr>
<tr>
<td><code>server.max.thread.stuck.time</code></td>
<td>Integer</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>The number of seconds that a thread must continually be working before the server determines that the thread is stuck.</td>
<td>Seconds</td>
</tr>
<tr>
<td><code>servers.status.running</code></td>
<td>String</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Managed Server Status(es). If any of the servers are in the admin state, set the value to be true to resume it.</td>
<td>true</td>
</tr>
<tr>
<td>Name</td>
<td>Value Type</td>
<td>Readable</td>
<td>Writeable</td>
<td>Restart</td>
<td>Description</td>
<td>Specific As</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------</td>
<td>----------</td>
<td>-----------</td>
<td>---------</td>
<td>-----------------------------------------------------------------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>ssl.twoway.client.enabled</td>
<td>String</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Set the alias of the private key that the client should use while authenticating with the two-way ssl</td>
<td>Alphanumeric</td>
</tr>
<tr>
<td>sun.http.handler.enabled</td>
<td>Boolean</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>Indicates whether or not the Sun HTTP handler is enabled.</td>
<td>true</td>
</tr>
</tbody>
</table>
Note:
Whenever you change credentials for a domain you need to restart the service for the changes to take effect.

User credentials are stored in any of three maps:

- **user.custom.map;** for storing credentials to which user applications have access. All the authenticated users assigned the role UserMapAccessRole will be able to read credentials from this map. See *Allowing Users Access to User Map in the Credential Store.*
- **oracle.wsm.security;** for storing credentials that web service clients can use.
- **user.public.map;** for storing credentials to which user applications can have anonymous access. Users will be able to read credentials from this map without authentication.

This table describes the commands for managing credentials.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>list-credentials</td>
<td>Lists all the credentials.</td>
<td>user, password, identitydomain, serviceinstance</td>
</tr>
<tr>
<td>describe-credential</td>
<td>Describes a credential identified by a credential map and a key.</td>
<td>user, password, identitydomain, serviceinstance, key</td>
</tr>
<tr>
<td>set-credential</td>
<td>Adds or updates a credential. The map is created if that is not available.</td>
<td>user, password, identitydomain, serviceinstance, key, keyuser, keypassword</td>
</tr>
<tr>
<td>delete-credential</td>
<td>Deletes an existing credential.</td>
<td>user, password, identitydomain, serviceinstance, key</td>
</tr>
</tbody>
</table>

**Enabling Access to Credentials in user.public.map**

For newly provisioned instances, you can access credentials stored in user.public.map even if you haven't been authenticated so long as the property `credential.public.map.enabled` is true.

By default, `credential.public.map.enabled` is set to true, thus enabling you to access credentials in user.public.map without being authenticated. Occasionally, this property will not be true. In that case, you need to enable it by using the `set-config` command; for example:

Note:
This procedure assumes you have downloaded the latest version of the JCS-SaaS Extension SDK. See *Downloading the Oracle Java Cloud Service - SaaS Extension SDK.*
1. First, use `list-config` to determine the property status:

   ```
   ./javacloud -list-config -identityDomain myIdentityDomain -serviceInstance myServiceInstance -userName myUserName -password myUserPassword -verbose -showvalues
   ```

2. On the configuration listing, check the value for `credential.public.map.enabled`. If it is true, the property is enabled.

   +---------------------------------------------------------------------------------------+
   | credential. public map enabled? | public. boolean | Y | Y | Y |
   | user.public.map will be manageable to unauthenticated users | true | |
   +---------------------------------------------------------------------------------------+

   If it is false, use `set-config` to enable the property:

   ```
   ./javacloud -set-config -name credential.public.map.enabled -value true -identityDomain myIdentityDomain -serviceInstance myServiceInstance -userName myUserName -password myUserPassword
   ```

3. Verify the status change by rerunning the `list-config` command.

   For more information on `list-config` and `set-config`, see Managing Configurations.

Enabling Access to Credentials in `user.custom.map`

If you are assigned the role UserMapAccessRole, you can create, read, and update credentials stored in the `user.custom.map`.

To allow users to access credential maps, use the following procedure:

1. Ensure the following API is in the application from which you want to fetch the credential:

   ```java
   CredentialStore credentialStore =
   credentialStore =
   oracle.security.jps.service.JpsServiceLocator.getServiceLocator().lookup(oracle.security.jps.service.credstore.CredentialStore.class);
   
   String map = "user.custom.map";
   String mykey = "mykey";
   
   Credential cred = credentialStore.getCredential(map, mykey);
   out.println("Password got:" + cred);
   if (cred != null) {
   ```
2. Add the credential to the map by using the set-credential command:

```
Note:

./javacloud is a script you can use to execute java -jar
javacloud.jar. In order to use it, you must add execute permission to
this script.
```

$ ./javacloud -dc us1 -id jcscdc -u jcsteam -si javas2 -set-credential -map
user.custom.map -key mykey -keyuser user-name

The system response will be:

[The password for the user specified with the argument user.]
[password] ****
[The password that is bound with the key.]
[keypassword] ****
[INFO] - Update - OK

3. Create the new role in the ID Management Console:
   a. Go to the MyServices application supplied with your Oracle Java Cloud
      Service - SaaS Extension account and click Security.
   b. Click the Customer Roles tab to open the page and then click Add.
      The Add Custom Role dialog appears.
   c. In the Add Custom Role dialog, enter the Role Name, UserMapAccessRole,
      along with a Display Name and, optionally, a short Description of the role.
      Then click Add.
      The new role, UserMapAccessRole appears on the Custom Role list.

4. Assign the new role to a user or users:
   a. Click the Users tab to go back to the Users list.
   b. Click associated with the user to whom you want to assign the custom role.
   c. From the drop-down menu, select Manage Roles.
      The Manage Roles dialog for that user appears.
   d. From Available Roles, move the custom role created in step 3 (it will be listed
      by its Display Name) to the Assigned Roles list and click Save.

5. Have the user who has been assigned with the new role log on to the custom
   application that executes the code specified at step 1.
   The user assigned the new role can now create, read, or delete credentials in the
   credential map.
Managing Web Services Security Truststore

The CLI enables you to manage Oracle Web Services Management (OWSM) security policies (WSS) for an OWSM truststore used for web services with OWSM policies.

**Note:**
- Adding a WSS certificate to the `wss trust-store` requires a service restart for the changes to take effect.
- Oracle does not support the use of special characters in the alias name of OWSM truststore certificates.

This table describes the commands for managing OWSM security policies.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>list-wss-certificates</td>
<td>Lists all the trusted certificates from the OWSM truststore.</td>
<td>user, password, identitydomain, serviceinstance</td>
</tr>
<tr>
<td>add-wss-certificates</td>
<td>Imports a new certificate into the OWSM truststore.</td>
<td>user, password, identitydomain, serviceinstance, path</td>
</tr>
<tr>
<td>delete-wss-certificates</td>
<td>Deletes an existing certificate from the outbound OWSM truststore.</td>
<td>user, password, identitydomain, serviceinstance, alias</td>
</tr>
<tr>
<td>download-wss-certificates</td>
<td>Downloads a certificate from the OWSM truststore.</td>
<td>user, password, identitydomain, serviceinstance</td>
</tr>
</tbody>
</table>

**Note:**
You can automate the process of setting up WSS trust from a local WebLogic Server domain to a JCS-SaaS Extension instance in the cloud by using the `setup-wss-trust` command. See Setting Up Trust Between WebLogic Domains and JCS-SaaS Extension.

Setting Up WSS Trust Between Two Instances or Between an On-premises WLS Domain and One Instance

The `setup-wss-trust` command-line tool that automates the process of setting up Web Service Security (WSS) trust from a local WebLogic Server domain to a JCS-SaaS Extension instance in the cloud or between two JCS-SaaS Extension instances running in different identity domains.

This table describes the commands for setting up these particular patterns of WSS trust.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>setup-wss-trust</td>
<td>Sets up WSS trust from a local WebLogic domain to the JCS-SaaS Extension instance on the cloud. Once the trust is set up, the on-premises local WebLogic domain will be able to propagate identity and protect message.</td>
<td>user, password, identitydomain, serviceinstance, alias, issuer</td>
</tr>
</tbody>
</table>

See Setting Up Trust Between WebLogic Domains and JCS-SaaS Extension.

Managing SSL Truststores

The CLI enables you to manage WebLogic Server security policies (SSL) for a WebLogic truststore used for web services with WebLogic policies.

![Note:](image)

- Adding an SSL certificate to the `ssl trust-store` requires a service restart for the changes to take effect.
- Oracle does not support the use of special characters in the alias name of WebLogic SSL truststore certificates.

This table describes the commands for managing SSL truststores.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>list-ssl-certificates</td>
<td>Lists all the trusted certificates from the SSL truststore.</td>
<td>user, password, identitydomain, serviceinstance</td>
</tr>
<tr>
<td>add-ssl-certificates</td>
<td>Imports a new certificate into the outbound SSL truststore.</td>
<td>user, password, identitydomain, serviceinstance, path</td>
</tr>
<tr>
<td>delete-ssl-certificates</td>
<td>Deletes an existing certificate from the outbound SSL truststore.</td>
<td>user, password, identitydomain, serviceinstance, alias</td>
</tr>
<tr>
<td>download-ssl-certificates</td>
<td>Downloads a certificate from the outbound SSL truststore.</td>
<td>user, password, identitydomain, serviceinstance</td>
</tr>
</tbody>
</table>

**Supported Certificate File Formats**

Java Cloud Service - SaaS Extension supports three certificate file formats:

- Single certificates appear in DER binary format.
• One or more certificates can appear in PEM format. The base64-encoded content of certificates is printed in base64-encoded format with the alias for each certificate shown in custom headers in the .pem file itself; for example:

```plaintext
alias: orakey
-----BEGIN CERTIFICATE-----
MIICXjCCAcegAwIBAgIIHiLb185PqPEwDQYJKoZIhvcNAQEFBQAwVzETMBSGcmS
JomT8ixkARKW2NaVbtevNBQGCmSj0mT8ixkARKWb9yYWNsZTEvMBMGcSmJomT
8ixkARKWBhSb3kMREwDVQDEwhDbG91ZDQTATeFw0xNDExMjExNzI5NDBa
Fw0yNDEwMTgxNzI5NDBaMGkzEzARBoGjikiaJk/Is2AZEZFgNj20xFjUEBoGjikiaJ
k/ls2AZEZFgVcmBfJbOkFzATBgoGjikiaJk/Is2AZEZFgVjgB91ZDEjMCEGALw
axNvcsmfY2V0cm1hDEQxNjA4X2phdmFzdmMwG28wDQYJoZIhvcNAQEEBBQAQAgY0A
MIGJAgGBAlIndinsAS5WTeC30fipsLT04XPHIcT99y6wnIyWdX8hP3K+epN87s7C
IvH2lm6WOKSflyOOCrynqRf4p6F0uF24F5Gy+aAisQkhdpwzRQsRy4VDuFaca
8FCyuoGQisuq8PoNdXlnUBvYGBhiOU2ZfLF2fsZuUg/pkiNTyvAgMBAAgjITAf
MBQGA1UdDgQWBBRBcruDY193MS3t1LBD4yyrW5v0sJTNBqkq0k9Gw0BAQUFA
agQA3RexPCqjF120vSo93UD33yWlSg7J0VQa3FRTYs3WC8481XrXq50WCJ0/yY/NV
PYDAXzms8HHvW+j1/Cp8AIVSesOD0zq8fWu6ub76NyA5dULcKXHJNJKW/K0
P15cSS93Kr7X2UfMlj9WOUR1/oY/tLiQy/R7Tsz5ft17w==
-----END CERTIFICATE-----
alias: entrustpremium
-----BEGIN CERTIFICATE-----
MIIEvDCCA0gAwIBAgIEOOG5ZfjANBqkq0k9Gw0BAQUFADBvAEMBIA1UECgkML
RW50cnVzdCuZXRQxQDABgNVBAsUN3d3dy51bnRydXN0Lm5ldC9DUFNf0AI0BCp
bnNvcnAuIG51H2l14gKXpwbU10cySaWFliktxJTAJBGVBasTHchjKSAxGTX5
IEVudH1jCQbmnVexb10ZQxMzAzBqNBAMwTKVxHzJc3QubmVOIEN1cnR
ZmljYXEpB24gQX0gV9yaX515QyMDQ4QTAMFWt050TcEz5JMQxNzUwNTAFAw0bO0TEy
MDM1ODlwNTAFAw0QMDc0Q1MFAw0bO0TEyMTM1ODlwNTAFAw0bO0TEyMTQ0MDAw
Iy1CMQc0Q1MFAw0bO0TEyMTU0MDAwIy1CMQc0Q1MFAw0bO0TEyMTU0MDAwIy1CMQc
-----END CERTIFICATE-----
```

• One or more certificates can appear in JKS format. Multiple certificates appear in the JKS file, listed by their respective aliases, but are otherwise in binary format.

### Managing System Properties

From the CLI, you can list, add, or delete system properties by using system property commands in the server startup argument.

The following table describes the commands you can use to manage system properties.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>list-system-properties</td>
<td>Lists all persisted system properties.</td>
<td>user, password, identitydomain, serviceinstance</td>
</tr>
<tr>
<td>set-system-property</td>
<td>Adds or updates an existing system property. Requires service instance restart to be effective.</td>
<td>user, password, identitydomain, serviceinstance, name, value</td>
</tr>
<tr>
<td>delete-system-property</td>
<td>Deletes a persisted system property. Requires service instance restart to be effective.</td>
<td>user, password, identitydomain, serviceinstance, name, value</td>
</tr>
</tbody>
</table>
You must restart the service instance for the properties to be implemented.

**Special Note on the UseSunHttpHandler Property**

Oracle Java Cloud Service - SaaS Extension doesn't allow you to set the UseSunHttpHandler property by using the System Properties section of the Administration console, using the set-system-property CLI command or directly by calling System.setProperty("UseSunHttpHandler", "true") in the code. Instead, use the sun.http.handler.enabled configuration, where you can check whether the UseSunHttpHandler system property is set. You must do this because additional behavior was added to improve instance functionality within the UseSunHttpHandler system property. This property is widely used and it solves several outbound connectivity problems.

*Note:*
To enable the sun.http.handler.enabled configuration, you can change the value in the Settings section of the Administration Console or via SDK as follows:

```bash
./javacloud -u UserName -si Service Instance Name -id IdentityDomain -dc DataCenter Code -set-config -name sun.http.handler.enabled -value true
```

See [Managing Configurations](#) and [Viewing System Configurations on the Settings Page](#).

**Special Note on Disabling the Security Manager**

Oracle Java Cloud Service - SaaS Extension no longer uses the Java Standard security manager. Instead, it employs the Java Cloud Service - SaaS Extension specific byte-code translation-based security manager. You should use the list-system-properties command to see if the Java Standard security manager is still enabled and, if it is, use the delete-system-property command to disable it.

*Note:*
./javacloud is a script you can use to execute java -jar javacloud.jar. In order to use it, you must add execute permission to this script.

To see if the Java Standard security manager is enabled:

```bash
./javacloud -u UserName -si Service Instance Name -id IdentityDomain -dc DataCenter Code -list-system-properties
```

If it is, to disable it:

```bash
./javacloud -u UserName -si Service Instance Name -id IdentityDomain -dc DataCenter Code -delete-system-property -name java.security.manager
```

Now, restart it by running this command:
You can also disable the security manager by using the `-list-config` command to set the `jvm.standard.security.manager.enabled` configuration to false. See Managing Configurations.

Viewing Access Logs

Use the `query-access-logs` command to view information in the `access.log` file.

WebLogic Server keeps a log of all HTTP transactions in a text file called `access.log`. You can access these logs from JCS-SaaS Extension by using the `query-access-logs` command provided by the SDK; for example:

```
./javacloud -user joe.user@myCo.com -password imjoespw -identitydomain mycopaid17110 -serviceinstance paidstandard1 -query-access-logs
```

Weblogic Server has a dedicated buffer that stores the HTTP requests to the server before they appear in the `access.log` file. The server will flush the buffer with the HTTP access information to the `access.log` once the buffer is full. By default this buffer size is set to 8Kb, so, in order to be reflected more recent logs via this `query-access-logs`, it is required reach the max of this buffer size.

Using the Command

When used without optional, filtering parameters (as in the preceding example), `-query-access-logs`, returns a listing similar to this:

```
#==========================================================================
#                         Listing 32 log records                         #
#==========================================================================
```

8-30
<table>
<thead>
<tr>
<th>No</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>==========</td>
</tr>
</tbody>
</table>
|===========================================================================
|===========================================================================
|=|Sat Nov 19
|
|1 |-          |10:03:00   |GET:/Diagnostic/DiagnosticService? WSDL |PST 2016
|
|2 |-          |10:05:07   |GET:/Diagnostic/DiagnosticService? WSDL |PST 2016
|
|3 |-          |11:40:53   |GET:/basic/faces/view1.jspx |PST 2016
By using the optional parameters, you can tailor reports to show just information germane to your requirements. For example, you can filter query access logs by:

- The maximum number of search records to be returned.
- Whether the listing should be in a verbose mode.
- Time and date range.
- The HTTP method used for request search for in the access logs. You can provide one or multiple values for this parameter such as: GET, PUT, POST, DELETE.
- An authorized username in the access log.
- The source IP from which the access log search was requested.
- The URL path from which the access log search was requested.

You can also combine these parameters. For example, if you wanted to limit the number of results returned and show just those for a specific date and time range and return the results in a specific format, you would enter something like this:

```
./javacloud -user joe.user@myCo.com -password imjoespw -identitydomain mycopaid17110 -serviceinstance paidstandard1 -query-access-logs -limit 10 -starttime 19/11/2016:10:00 -endtime 19/11/2016:11:50 -datetimeformat dd/MM/yyyy:HH:mm
```

You would receive a listing like this:

```
-----------[Fetching at Mon Nov 21 18:27:55 PST 2016]-----------
```

```
Listing 6 log records
```
<table>
<thead>
<tr>
<th>S.No</th>
<th>Auth User</th>
<th>DateTime</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Sat Nov 19 10:03:00 PST 2016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET:/Diagnostic/ DiagnosticService? WSDL</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>Sat Nov 19 10:05:07 PST 2016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET:/Diagnostic/ DiagnosticService? WSDL</td>
</tr>
<tr>
<td>3</td>
<td>-</td>
<td>Sat Nov 19 11:40:53 PST 2016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET:/basic/faces/view1.jspx</td>
</tr>
<tr>
<td>4</td>
<td>-</td>
<td>Sat Nov 19 11:40:53 PST 2016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET:/basic/faces/view1.jspx</td>
</tr>
<tr>
<td>5</td>
<td>-</td>
<td>Sat Nov 19 11:42:28 PST 2016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET:/basic/index.jsp</td>
</tr>
<tr>
<td>6</td>
<td>-</td>
<td>Sat Nov 19 11:42:28 PST 2016</td>
</tr>
<tr>
<td></td>
<td></td>
<td>GET:/basic/index.jsp</td>
</tr>
</tbody>
</table>

----------[Fetched at Mon Nov 21 18:27:56 PST 2016]----------
For another example, if you wanted to filter the results to just the most recent five logs based on a source IP and return those results in a verbose status, you might enter:

```
./javacloud -user joe.user@myCo.com -password imjoespw -identitydomain
mycopaid17110 -serviceinstance paidstandard1 -query-access-logs -sourceip
10.242.200.4 -verbose -limit 5
```

which would return:

```
--------[Fetching at Mon Nov 21 20:57:40 CST 2016]--------

Listing 10 log records
<table>
<thead>
<tr>
<th>S.</th>
<th>Auth User</th>
<th>Source</th>
<th>DateTime</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>jose.hijar@ora</td>
<td>10.242.19</td>
<td>Sat Nov 19</td>
</tr>
<tr>
<td></td>
<td>cle.com</td>
<td>200.4</td>
<td>14:16:08</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>jose.hijar@ora</td>
<td>10.242.19</td>
<td>Sat Nov 19</td>
</tr>
<tr>
<td></td>
<td>cle.com</td>
<td>200.4</td>
<td>14:16:10</td>
</tr>
</tbody>
</table>
```
Chapter 8
Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension

---

-------------[Fetched at Mon Nov 21 20:57:42 CST 2016]-------------

---

---

---

---

---

---

---
For a complete list of all optional parameters for -query-access-logs, navigate to the $SDK_HOME/doc/index.html file (where $SDK_HOME is the directory containing your Oracle Java Cloud Service - SaaS Extension installation) and search on the command name.

**Log Format**

The common log format is the default:

```
host RFC931 auth_user [day/month/year:hour:minute:second UTC_offset]
"request" status bytes
```

where:

- **host** — Either the DNS name or the IP number of the remote client.
- **RFC931** — Any information returned by IDENTD for the remote client; WebLogic Server does not support user identification.
- **auth_user** — If the remote client user sent a userid for authentication, the user name; otherwise "-".
- **day/month/year:hour:minute:second UTC_offset** — Day, calendar month, year and time of day (24-hour format) with the hours difference between local time and GMT, enclosed in square brackets.
- **"request"** — First line of the HTTP request submitted by the remote client enclosed in double quotes.
- **status** — HTTP status code returned by the server, if available; otherwise "-".
- **bytes** Number of bytes listed as the content-length in the HTTP header, not including the HTTP header, if known; otherwise "-".

You can extend log formats to customize the information that is recorded. You can set the attributes that define the behavior of HTTP access logs for each server instance or for each virtual host that you define.

**Viewing Service Logs**

You can access and view service log files by using the query-service-log command. These logs are useful when troubleshooting issues that might arise with your JCS-SaaS Extension instance.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>query-service-log</td>
<td>Gets application log records that match the given search criteria.</td>
<td>identityDomain, userName, serviceInstance</td>
</tr>
</tbody>
</table>

This command gets application log records based on specified search criteria. You can further restrict the search scope by using one of the combinations of last,unit or starttime,endtime. Also, you can limit the number of log records returned in the search results by using -limit.
For example:

```
./javacloud -query-service-log -identityDomain myIdentityDomain -userName myUserName -serviceInstance myServiceInstance -starttime 18:00:00 -endtime 17:59:59 -limit 10 -last HOUR
```

In this example, service logs compiled up through the last hour of a 24–hour period beginning at 18:00:00 would be produced for viewing.

Viewing Service Metrics

Use the `list-service-metrics` to show the metrics for a running/active service.

This command provides service performance statistics that help you measure an application's performance, identify performance bottlenecks, and monitor the health of the service as a whole. `list-service-metrics` is a super-set of the `query-service-metrics` command (now deprecated; see Note, below) that lets you access the metrics based on this hierarchy:

```
instance
  applications
    OpenSessionsCurrentCount
    RequestProcessingTime
    RequestCountPerMinute
  databases/datasources
    ConnectionsTotalCount
    ConnectionsCreateRate
    ActiveConnectionsCurrentCount
  infra/storage
    free
  infra
    storage
      free
      max
  applications
    <app_name>
      OpenSessionsCurrentCount
      RequestProcessingTime
      RequestCountPerMinute
    servlets
      <servlet_name>
        OpenSessionsCurrentCount
        RequestProcessingTime
        RequestCountPerMinute
  databases
    datasources
      <datasource_name>
        ConnectionsTotalCount
        ConnectionsCreateRate
        ActiveConnectionsCurrentCount
  servers
    <server_name>
      memory
        HeapFreeCurrent
        HeapSizeMax
      workmanager
        <workmanager_name>
          CompletedRequests
```

Chapter 8
Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension
8-38
Note:

query-service-metrics is deprecated and a warning message appears if you use this command; however, for the near future, the command will remain in the SDK to support legacy applications.

list-service-metrics Arguments

list-service-metrics takes these arguments:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>metric</td>
<td>The name of the metric. It should be of the format parent/child**; for example: -metric infra/storage. The values for parent are: • instance • infra • databases • servers • applications. The value for child can be empty. If this attribute is not provided, the instance level metric is returned. If the value is '/' or empty, all metrics will be listed. <strong>Shortcut:</strong> mn</td>
<td>user (-u), password (-p), identity domain (-id), service instance (-si)</td>
</tr>
<tr>
<td>verbose</td>
<td>The true/false flag that indicates whether or not the listing should be verbose (that is, full-format). <strong>Shortcut:</strong> v</td>
<td>user (-u), password (-p), identity domain (-id), service instance (-si)</td>
</tr>
<tr>
<td>gridwidth</td>
<td>The maximum width of the grid. You can use this if you want to limit the width of the grid display (for example, when you have a smaller display). Be aware that, if you specify a smaller width, the grid might not be formed to fit within the width. This is applicable when the argument grid is true. <strong>Shortcut:</strong> gw</td>
<td>user (-u), password (-p), identity domain (-id), service instance (-si)</td>
</tr>
</tbody>
</table>

Default: false
<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>gridtree</td>
<td>The true/false flag that indicates whether or not to show the grid content in a tree-like format, by grouping columns with the same value.</td>
<td>user (-u), password (-p), identity domain (-id), service instance (-si)</td>
</tr>
<tr>
<td></td>
<td><strong>Shortcut:</strong>    g</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Default:</strong> false</td>
<td></td>
</tr>
<tr>
<td>sorton</td>
<td>The sort order for the metrics. Metrics are sorted in ascending order unless the argument -descending is specified. Acceptable values are:</td>
<td>user (-u), password (-p), identity domain (-id), service instance (-si)</td>
</tr>
<tr>
<td></td>
<td>• METRIC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• VALUE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• TYPE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• COMPONENT</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Shortcut:</strong> so</td>
<td></td>
</tr>
<tr>
<td></td>
<td><strong>Default:</strong> false</td>
<td></td>
</tr>
<tr>
<td>descending</td>
<td>The true/false flag that, when used with -sorton indicates whether or not the metrics should be sorted in descending order rather than ascending, which is the default. Adding this argument to this command without specifying true or false is same as specifying true.</td>
<td>user (-u), password (-p), identity domain (-id), service instance (-si)</td>
</tr>
<tr>
<td></td>
<td><strong>Default:</strong> true</td>
<td></td>
</tr>
</tbody>
</table>

**Note:**

While a password is required by this command, you should not specify the -p argument on a command-line that takes the password in plain text. Instead, execute the command without specifying this argument and the system will prompt you for the password, which you can then enter securely.

**Using the Command**

**Note:**

To simplify command entry, you can store values for all parameters except -password (-p) in the javacloud.properties file. See Using javacloud.properties.

To display metrics, enter the command list-service-metrics, specifying the required user, password, identity domain, and service instance along with the metric.
you want to view. If you don’t specify a metric, the command displays metrics at the instance level and, by default sorted on the COMPONENT attribute. Some examples follow:

**Note:**

`.javacloud` is a script you can use to execute `java -jar javacloud.jar`. In order to use it, you must add execute permission to this script.

To display metrics at the instance level:

`.javacloud list-service-metrics -dc us1 -id usoracletrial09442 -si javatrial6938 -gridtree [-metric instance]

Note that specifying the level (`-metric instance`) is optional.

The output would be (with `-gridtree` specified):

```
#====================================================================================
# |                               Listing one Metric(s)                                |
# | [Identity Domain=usoracletrial09442, Service Instance=javatrial6938],        |
# Instance Level Summary                                                        |
#====================================================================================
# Component| Metrics
#----------|--------
# | Name | Value
#| Active Sessions Count | 0
#| applications|-----------------------|
#| Request Processing Time| 0
#| Requests Count | 0.000 per minute
#| Component | Metrics
#| instance | | Name | Value
```
To view all storage metrics:

```
./javacloud list-service-metrics -dc us1 -u system -id usoracletrial09442 -si javatrial6938 -metric infra/storage
```

The output would be:

```
#====================================#
|       Listing 2 Metric(s)          |
|
| [Identity Domain=usoracletrial09442, |
| Service Instance=javatrial6938],   |
| Component Level                    |
#===========#=============#==========#
<p>| Metric Name|  Component  | Value    |
|===========|=============|==========|</p>
<table>
<thead>
<tr>
<th>Free</th>
<th>infra/storage</th>
<th>5118 MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>infra/storage</td>
<td>5120 MB</td>
</tr>
</tbody>
</table>
#====================================#
```

To view an application-level metric:

```
./javacloud list-service-metrics -dc us1 -u system -id usoracletrial09442 -si javatrial6938 -gridtree -metric applications/welcome-app/RequestcountPerMinute
```

The output would be:

```
#========================================================#
|                 Listing one Metric(s)                   |
|
|   [Identity Domain=usoracletrial09442,                 |
|    Service Instance=javatrial6938], Component Level    |
#============#===========================================#
| Component  |                  Metrics                  |
|============|===========================================|
```

---

<table>
<thead>
<tr>
<th>databases</th>
<th>Connections Total Count</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>datasources</td>
<td></td>
<td></td>
</tr>
<tr>
<td>minute</td>
<td>JDBC Connection Create Rate</td>
<td>0.000 per minute</td>
</tr>
<tr>
<td></td>
<td>Open JDBC Connections Count</td>
<td>0</td>
</tr>
</tbody>
</table>

---

Chapter 8

Using the Command-Line Interface to Manage Oracle Java Cloud Service - SaaS Extension

---

```bash
./javacloud list-service-metrics --dc us1 --u system --id usoracletrial09442 --si javatrial6938 --metric infra/storage
```

The output would be:

```
#====================================#
|       Listing 2 Metric(s)          |
|
| [Identity Domain=usoracletrial09442, |
| Service Instance=javatrial6938],   |
| Component Level                    |
#===========#=============#==========#
<p>| Metric Name|  Component  | Value    |
|===========|=============|==========|</p>
<table>
<thead>
<tr>
<th>Free</th>
<th>infra/storage</th>
<th>5118 MB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maximum</td>
<td>infra/storage</td>
<td>5120 MB</td>
</tr>
</tbody>
</table>
#====================================#
```

```
./javacloud list-service-metrics --dc us1 --u system --id usoracletrial09442 --si javatrial6938 --gridtree --metric applications/welcome-app/RequestcountPerMinute
```

The output would be:

```
#========================================================#
|                 Listing one Metric(s)                   |
|
|   [Identity Domain=usoracletrial09442,                 |
|    Service Instance=javatrial6938], Component Level    |
#============#===========================================#
| Component  |                  Metrics                  |
|============|===========================================|
```
## Refreshing an Application

Use the `-refresh` command whenever you need to download and redeploy applications that were previously deployed to JCS-SaaS Extension instances.

You can use this command with any out-of-date version of JCS-SaaS Extension. It is particularly useful if you are encountering login issues with older applications. For example, if you provisioned an instance with a version of JCS-SaaS Extension prior to 17.1.3 but then patched your service with this later release, you might encounter a blank page when trying to log in to applications deployed on that instance. If this occurs, you need to download and redeploy these applications to load the latest tag libraries. `-refresh` facilitates this task.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>refresh</td>
<td>Downloads and redeploys applications to ensure that the latest tag libraries are included.</td>
<td>user (-u), identitydomain (-id), serviceinstance (-si), application (-app)</td>
</tr>
</tbody>
</table>

**Note:**

This command resolves an issue wherein a blank screen would appear after logging in to an ADF application deployed on a JCS-SaaS Extension instance provisioned before version 17.1.3 but subsequently upgraded to that version (see ADF Application Login Results in Blank Page). If you have any deployments that match this condition, you should run this command.
For example:

$ ./javacloud -u username -id myiddomain123 -si javatrial23 -refresh -app hcmconnect-ear-1.1.0-SNAPSHOT

Note that this command will return a job identifier; for example:

[INFO]    - The application is being refreshed.

1:Job Id   - 5103
------------ - -----------------------

You can use this job number with the job-status command to track the status of the refresh.

Synchronizing UI and SDK Data

You can use the SDK command sync-system to synchronize data between the resource management UI and the SDK.

Occasionally, you might notice that association and utilities data that appear in the resource management UI's Home section might not appear or don't match similar data for the same instance returned when you run the describe-service-instance SDK command. Values such as state, version, and associations should match and when they don't, use the SDK command sync-system to synchronize them.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>sync-system</td>
<td>Synchronizes client and service instance data.</td>
<td>user (u), identitydomain (id), serviceinstance (si)</td>
</tr>
</tbody>
</table>

**Note:** For a complete list of optional, advanced, and diagnostic arguments.
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>tic</td>
<td>help</td>
<td>parameters, use the -d parameter with the Help command</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>Mandatory Arguments</td>
</tr>
<tr>
<td>-------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>----------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td></td>
<td>a\nd</td>
</tr>
</tbody>
</table>

For example:

```bash
./javacloud -a http://myServer.us.MyCorp.com:7003 -si myServiceInstance -id myIdentityDomain -user joe.user@MyCorp.com -p jcssx1234 -sync-system
```

If the synchronization is successful, the system will respond:

```
[INFO]    - System is synchronized.
```

Otherwise, if it fails the system will respond:

```
[ERROR]   - The system could not be synchronized.
```

### Accessing the Local File System

The Oracle Java Cloud Service - SaaS Extension SDK contains two tools that enable you to manage the files in the `/customer/scratch/` directory of your Oracle Java Cloud Service - SaaS Extension instance.

**Topics**

- Using the File Browser
- Using the File System Access Shell

### Using the File Browser

The Oracle Java Cloud Service - SaaS Extension SDK includes a Maven plug-in project that can be used to manage the files in your `/customer/scratch/` directory. The sample File Browser application also shows how `java.io.*` APIs can be used to read and write files.

To build and launch the File Browser sample:
1. Navigate to the $SDK_HOME/samples/maven/filebrowser directory (where $SDK_HOME is the directory containing your Oracle Java Cloud Service - SaaS Extension installation).

2. Run the following command:

   mvn clean package

3. Once the sample is built, enter the following URL in your browser:

   https://<servicename-identitydomain>.java.cloud.oracle.com/filebrowser/

   This opens the File Browser's "welcome" window:

4. Click the Local File System Access Test link.

   This opens the Filer Browser's current directory page:

5. You can use this page to browse the /customer/scratch directory. You can use the options on this page to upload and download files from that volume, navigate to the parent directory, or create a new directory.

Using the File System Access Shell

   You can use the CLI to open a File System Access Shell to manage the files in your Oracle Java Cloud Service - SaaS Extension instance.

   The File System Access Shell accepts basic file management commands, such as ls, cp, mv, put, and get, to manage the files in your /customer/scratch/ directory.

   For detailed information about all the available File Shell commands and their usage, navigate to the $SDK_HOME/doc/javacloud-fs-usage.html file (where $SDK_HOME is the directory containing your Oracle Java Cloud Service - SaaS Extension installation). You can also access all the SDK documentation via the "Welcome App".
Here is an example of using the CLI to open a file shell session:

```
$ ./javacloud -dc us1 -u username@oracle.com -id usoracletrial08411 -si javatrial5334 -fs
Java service file-system access shell.
The root directory "/" points "/customer/scratch/"
```

Here is an example of using the File Shell to list all files in the /customer/scratch directory:

```
/>-fs -grid
#=======================================================================================#
|                               Listing 5 file(s) under /                               |
#=#============#===#========================#===========================================#
|#|    Name    |Dir|       File Type        |         Last Modified Description         |
|=|============|===|========================|===========================================|
|1|cloudappc   |d  |                        |                                           |
|2|a.txt       |   |text/plain              |55 days, 4 hours, 36 minutes and 13 seconds|
|3|myzip       |d  |                        |                                           |
|4|FirstPdf.pdf|   |application/octet-stream|2 days, 4 hours, 21 minutes and 19 seconds |
|5|metrics     |d  |                        |                                           |
#=======================================================================================#
```

Using the Application and Domain Configuration Shell

The Application and Domain Configuration Shell (the "Config Shell") enables you to perform general web service and WebLogic domain configuration tasks.

Use the CLI commands to perform the following tasks against the WebLogic domain of your Oracle Java Cloud Service - SaaS Extension instance:

- Lists all JRF web services and web service clients
- Manages OWSM policies on web service endpoints and web service client ports
- Sets web services configuration and policy overrides
- Sets web services client stub properties
- Sets SAML DN configuration to the WebLogic domain
- Lists SAML DN configuration

In multi-node environments, a single command can translate into multiple commands (one for each managed server) and URL. For example, if you run the attach-webservice-policy command on the S3 node in a four-node environment, you do not need to repeat this action for nodes S1, S2, or S4.

For information about all the available Config Shell commands and how to use them, navigate to the $SDK_HOME/doc/javacloud-app-config.html file (where $SDK_HOME is the directory containing your Oracle Java Cloud Service - SaaS Extension installation). You can also access all the SDK documentation via the "Welcome App".
Using the Basic Config Shell Commands

The "Config Shell" enables you to use the CLI to perform general web service and WebLogic domain configuration tasks.

This section provides some examples for using certain `config-shell` commands.

Starting the Config Shell

Here is an example of entering the Config Shell.

```
$ ./javacloud -dc us1 -u username@oracle.com -id usoracletrial08411 -si javatrial5334 -config-shell

[INFO] - Java service config shell.
    Initializing ...

Config-shell:
```

Using the `set` Command and the `command` Argument

The `config-shell` takes a `-command` argument that can contain a list of commands that will be automatically executed upon entering the shell. The list of commands can be separated with a semicolon. If the shell needs to exit at the end of running all the listed commands, then the `exit` command should also be specified in the command list.

The `config-shell` also supports a special `set` command that allows you to set frequently used arguments across commands. Once an argument is set, the commands requiring that argument can take the `set` value as the default value. This is similar to `javacloud.properties` for the configuration shell.

In the following example, the `set` command can be used to set the default arguments (for example, `application` and `module`), and then perform the commands without the need for passing those arguments in every command within the `config-shell`.

```
$ ./javacloud -dc us1 -u username@oracle.com -id usoracletrial08411 -si javatrial5334 -config-shell -command "set application=myapp;;set module=mymodule"

Now the `config-shell.command` can be defined in the `javacloud.properties` file.
```

Note:

Arguments, such as `module`, that are supported by the `config-shell` command `set-webservice-client-property`, cannot be directly specified in the `javacloud.properties` file. It can be only specified as `config-shell.command=set module=dctest` in the properties file.

Here is an example of using the `set` command to specify the arguments only once in the shell:

```
Config-shell:>set application=Application3_ViewController_webapp1
Added: application
Config-shell:>list-webservice-clients
</domain>/m0/Application3_ViewController_webapp1:
   moduleName=dctest, moduleType=wsconn, serviceRefName= AppModuleService
```
Note that the application name is taken automatically since it was already set in the shell. Just type `set` in the shell to list all the arguments that you have set:

```
Config-shell:>set
#===============================================================================================#
|                           Listing 7 argument(s) and their values.                             |
#===============================================================================================#
|                                          argument                  |                  value            |
|================================================================================================|
| application                                                |Application3_ViewController_webapp1|
|-----------------------------------------------------------+-----------------------------------|
| gridwidth                                                  |140                                |
|-----------------------------------------------------------+-----------------------------------|
| module                                                     |dctest                             |
|-----------------------------------------------------------+-----------------------------------|
| output                                                     |/Users/velsubra/Desktop/ade/twork/|
|-----------------------------------------------------------+-----------------------------------|
| port                                                       |AppModuleServiceSoapHttpPort       |
|-----------------------------------------------------------+-----------------------------------|
| serviceref                                                 |AppModuleService                   |
|----------------------------------------------------------------|                                   |
| [alias, clienttype, configprops, debug, dump, help, issuer, |           --NOT SET--             |
| overrideprops, policyuri, retain, service,stubprops,       |                                   |
| subject, tokentype, trustedDN, verbose]                    |                                   |
|----------------------------------------------------------------|                                   |
```

**Displaying Help for a Config Shell Command**

You can use the `-help` command to display detailed information for each Config Shell command.

```
Config-shell:>list-webservice-clients -help
Command:
--------
list-webservice-clients - Lists all the web service clients.
E.g) list-webservice-clients -application myapp;list-webservice-clients -
application myapp
                                   -verbose
                               Command alias:[listwebserviceclients]
Mandatory argument(s):
----------------------
application - The name of the application.

Shortcut:app

Optional arguments(s):
----------------------
verbose - The flag (true/false) that indicates if the listing should be done in verbose{full-format}.

Shortcut:v

Default Value: false

Advanced argument(s):
----------------------

Diagnostic/Help argument(s):
----------------------------
help - The flag (true/false) to indicate whether the help text should be printed. The default value
is false. When true, only the help is printed and all the other arguments, if specified, are ignored.

Shortcut: h
Default Value: false

debug - The flag (true/false) to indicate whether the debug-level messages should be printed. The debug messages are more detailed than INFO-level messages. The default value is false.

Shortcut: d
Default Value: false

Managing Logging Levels

Use the CLI to list loggers and set their levels. Log levels indicate the amount of detail presented by the logged information.

Listing loggers and setting their log levels is particularly useful when you want to control the debugging resolution of the log statements.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>Mandatory Arguments</th>
</tr>
</thead>
<tbody>
<tr>
<td>list-loggers</td>
<td>Lists the name and log level of all the Loggers or of a given Logger and, optionally, its children. Note that, while this command lists all the loggers and their logging levels, it hides any internal loggers.</td>
<td>No mandatory arguments</td>
</tr>
</tbody>
</table>
| set-log-level | Sets the Log level of the Logger to the given level. The JDK’s available log levels are:  
  • FINEST  
  • FINER  
  • FINE  
  • INFO  
  • WARNING  
  • SEVERE  
  Oracle Java Cloud Service - SaaS Extension also supports Weblogic’s logging convention, so following are also accepted log levels.  
  • TRACE  
  • NOTIFICATION  
  • WARNING  
  • ERROR | logger, level |
Displaying Application Details

You can use the Config Shell to list all the OWSM policies, OWSM client policies, web services, and web service clients in your domain.

Listing OWSM Policies

You can list the service policies in your domain. By default, `list-all-webservice-policies` only lists all service policies but does not include any client policies. The argument `-subject` tells whether the listing should be done for client or service.

```
Config-shell:>list-all-webservice-policies
List of available OWSM policies
security : oracle/http_basic_auth_over_ssl_service_policy
security : oracle/wss_saml_or_username_token_over_ssl_service_policy
security : oracle/wss_saml_token_bearer_over_ssl_service_policy
security : oracle/wss11_message_protection_service_policy
security : oracle/wss11_saml_token_with_message_protection_service_policy
security : oracle/wss_saml120_token_bearer_over_ssl_service_policy
security : oracle/wss11_username_token_with_message_protection_service_policy
security : oracle/wss_http_token_over_ssl_service_policy
security : oracle/wss_username_token_over_ssl_service_policy
security : oracle/wss11_x509_token_with_message_protection_service_policy
security : oracle/wss_saml_token_over_ssl_service_policy
security : oracle/multi_token_rest_service_policy
security : oracle/http_saml20_token_bearer_over_ssl_service_policy
security : oracle/wss11_saml_or_username_token_with_message_protection_service_policy
security : oracle/wss_saml20_token_bearer_over_ssl_service_policy
security : oracle/multi_token_over_ssl_rest_service_policy
Config-shell:>
```

Listing OWSM Client Policies

You can list the client policies in your domain by adding the `-subject client` argument to the `list-all-webservice-policies` command.

```
Config-shell:>list-all-webservice-policies -subject client
List of available OWSM policies
security : oracle/wss_http_token_client_policy
security : oracle/http_basic_auth_over_ssl_client_policy
security : oracle/http_saml20_token_bearer_over_ssl_client_policy
security : oracle/wss_http_token_over_ssl_client_policy
security : oracle/wss11_saml_token_with_message_protection_client_policy
security : oracle/wss11_x509_token_with_message_protection_client_policy
security : oracle/wss11_username_token_with_message_protection_client_policy
security : oracle/wss_saml120_token_bearer_over_ssl_client_policy
security : oracle/wss11_saml_token_bearer_over_ssl_client_policy
security : oracle/wss_username_token_over_ssl_client_policy
security : oracle/wss_saml_token_over_ssl_client_policy
security : oracle/wss11_message_protection_client_policy
security : oracle/http_saml20_token_bearer_client_policy
security : oracle/wss_saml20_token_over_ssl_client_policy
security : oracle/wss_saml120_token_over_ssl_client_policy
Config-shell:>
```

Listing Web Services

You can list the web services in your domain.
Config-shell:>list-webservices -app adfbc_bcProfile1 -v
Server:m0
========
/<domain>/m0/adfbc_bcProfile1 :
moduleName=cloudapps-adfbc-context-root, moduleType=web, serviceName=/{adfbc/
common{/AppModuleService
   enableTestPage: true
   enableWSDL: true
      AppModuleServiceSoapHttpPort http://server:port/cloudapps-adfbc-context-
         root/AppModuleService
         enable: true
         enableREST: false
         enableSOAP: true
         maxRequestSize: -1
         loggingLevel: NULL
         wsat.flowOption: NEVER
         wsat.version: DEFAULT
No policies attached; endpoint is not secure.

Listing Web Service Clients
You can list the web service clients in your domain.

Config-shell:>list-webservice-clients -app Application3_ViewController_webapp1 -v
/<domain>/m0/Application3_ViewController_webapp1 :
moduleName=dctest, moduleType=wsconn, serviceRefName=AppModuleService
   AppModuleServiceSoapHttpPort serviceWSDLURI=http://server:port/cloudapps-
      adfbc-context-root/AppModuleService?wsdl
No policies attached; endpoint is not secure.

Note that in this example there is only a single client. By using the verbose (-v) argument, the output will try to describe the attached policies as well. In this case the client does not have any policies.

Example Use-case: Overriding an Endpoint Address for a Web Service Client

You can use the Config Shell commands discussed in this section to override a web service endpoint address for a web service client.

The following use case shows one way to override an web service endpoint address for a web service client.

1. List the web service clients for the details that would be required when setting the endpoint address.

   Config-shell:>list-webservice-clients -app Application3_ViewController_webapp1 -v
   /<domain>/m0/Application3_ViewController_webapp1 :
      moduleName=dctest, moduleType=wsconn, serviceRefName=AppModuleService
         AppModuleServiceSoapHttpPort serviceWSDLURI=http://server:port/cloudapps-
            adfbc-context-root/AppModuleService?wsdl
No policies attached; endpoint is not secure.

2. Set the endpoint address:
   a. Set the various parameters that identify the web service client. (Note that the following parameters map to the highlighted client details in the list-webservice-clients output) in Step 1:

      Config-shell:>set application=Application3_ViewController_webapp1;set
         module=dctest;set serviceref=AppModuleService;set
         port=AppModuleServiceSoapHttpPort
Note:
Step 2a is optional since these arguments can be directly passed when using `set-webservice-client-property` to change the endpoint address, as shown in Step 2b. Also, any values that are passed on the command-line will override values that are set using the `set` command.

b. Change the endpoint address using the `set-webservice-client-property` command:

```
Config-shell:>set-webservice-client-property -stubprops
javax.xml.ws.service.endpoint.address=http://server:port/cloudapps-adfbc-context-root/AppModuleService
```

Please restart application to uptake any policy or configuration change.

3. Restart the application:

```
Config-shell:>restart-application
INFO] - Stopping the application : Application3_ViewController_webapp1
INFO] - Job:1752 Operation:Stop Application
INFO] - Starting the application : Application3_ViewController_webapp1
INFO] - Job:1753 Operation:Start Application
```

CLI Commands in the SDK

Use the commands described in this topic with the Oracle Java Cloud Service - SaaS Extension command-line interface to monitor applications deployed on your service instance.

Some of the following commands are documented elsewhere in this guide but you can more information on each of them by navigating to the `$SDK_HOME/doc/index.html` file (where `SDK_HOME` is the directory containing your Oracle Java Cloud Service - SaaS Extension installation). You can also access all the SDK documentation via the Welcome Application. See SDK Documentation.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
<th>For More Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>add-datasource-jndiname</td>
<td>Add a new JNDI name for a data source.</td>
<td></td>
</tr>
<tr>
<td>add-ssl-certificates</td>
<td>Uploads one or more new certificates into the outbound SSL truststore from the local disk.</td>
<td>Managing SSL Truststores</td>
</tr>
<tr>
<td>add-wss-certificates</td>
<td>Uploads one or more new certificate into the web service security (WSS) truststore.</td>
<td>Managing Web Services Security Truststore</td>
</tr>
<tr>
<td>config-shell</td>
<td>Executes service/application configuration commands.</td>
<td></td>
</tr>
<tr>
<td>delete</td>
<td>Deletes an installed application permanently. The existing user sessions for the application, if any, will be lost.</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>For More Information</td>
</tr>
<tr>
<td>------------------------</td>
<td>-----------------------------------------------------------------------------------------------</td>
<td>-----------------------------------------------</td>
</tr>
<tr>
<td>delete-credential</td>
<td>Deletes an existing credential.</td>
<td>Managing Credentials</td>
</tr>
<tr>
<td>delete-datasource-jndiname</td>
<td>Delete existing JNDI name for a data source.</td>
<td></td>
</tr>
<tr>
<td>delete-library</td>
<td>Deletes an installed shared library permanently.</td>
<td>Managing Shared Libraries</td>
</tr>
<tr>
<td>delete-ssl-certificates</td>
<td>Deletes one or more existing certificates from the outbound SSL truststore.</td>
<td>Managing SSL Truststores</td>
</tr>
<tr>
<td>delete-system-property</td>
<td>Deletes a persisted system property.</td>
<td>Managing System Properties</td>
</tr>
<tr>
<td>delete-wss-certificates</td>
<td>Deletes one or more existing certificates from the web service security (WSS) truststore.</td>
<td>Managing Web Services Security Truststore</td>
</tr>
<tr>
<td>describe-application</td>
<td>Describes an application identified by its name. The description includes the current status and the application URLs (one for each web module) that can be used to access the application. If you would like to know the run-time metrics of web modules, use the command <code>query-service-metrics</code>.</td>
<td></td>
</tr>
<tr>
<td>describe-credential</td>
<td>Describes a credential identified by a key.</td>
<td>Managing Credentials</td>
</tr>
<tr>
<td>describe-library</td>
<td>Describes a shared library identified by its name, spec version, and impl version. The description includes the status, deploy type, type, and the name of the applications that reference this library.</td>
<td>Managing Shared Libraries</td>
</tr>
<tr>
<td>describe-service-instance</td>
<td>Describes a service instance under an identity domain. The description includes the status of the service instance and the size of the offering.</td>
<td></td>
</tr>
<tr>
<td>download-artifacts</td>
<td>Downloads artifacts such as:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Whitelist configuration file used by the service instance.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Applications deployed by the users.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Shared libraries that are available to user applications.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Multiple artifacts can be downloaded using a single command-line.</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>For More Information</td>
</tr>
<tr>
<td>------------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------</td>
</tr>
<tr>
<td>download-ssl-certificates</td>
<td>Downloads one or more certificates from the outbound SSL truststore to the local disk.</td>
<td>Managing SSL Truststores</td>
</tr>
<tr>
<td>download-wss-certificates</td>
<td>Downloads a certificate from the web services security (WSS) truststore.</td>
<td>Managing Web Services Security Truststore</td>
</tr>
<tr>
<td>fs-shell</td>
<td>Executes file system-specific shell commands. <strong>Note:</strong> The shell is not like an OS shell. This shell supports only simple commands that are useful in managing files in a Java Cloud Service - SaaS Extension instance. The options that are available for a standard OS command are not available in this shell. For example, <code>ls -ltr</code> will not work here. In this shell, the supported options work his shell. • Piping is not allowed. • Redirection is not allowed. • Special characters such as <code>*,:!?[]()&lt;&gt;%@$</code> are not allowed. See <code>validatespecialchars</code>.</td>
<td></td>
</tr>
<tr>
<td>install</td>
<td>Installs a user application that is already bundled and available in the local disk. Use the command <code>install-library</code>, if you want to install a shared library.</td>
<td></td>
</tr>
<tr>
<td>install-library</td>
<td>Installs a custom shared library.</td>
<td>Managing Shared Libraries</td>
</tr>
<tr>
<td>job-log-file</td>
<td>Downloads the job log file and writes to the local disk.</td>
<td></td>
</tr>
<tr>
<td>job-status</td>
<td>Describes a job identified by its job ID. The description includes the current status, start time, and end time of the job.</td>
<td></td>
</tr>
<tr>
<td>list-applications</td>
<td>Lists all the applications that are installed and available in the service instance.</td>
<td></td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>For More Information</td>
</tr>
<tr>
<td>-----------------------</td>
<td>------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------------</td>
</tr>
<tr>
<td>list-commands</td>
<td>Lists all the commands. You can use the argument <code>-search</code> to find specific set of commands. This is the default command</td>
<td></td>
</tr>
<tr>
<td>list-config</td>
<td>Lists all available editable configurations. Use this command to see which configurations you can change by using <code>set-config</code>.</td>
<td>Managing Configurations</td>
</tr>
<tr>
<td>list-credentials</td>
<td>Lists all the credentials.</td>
<td>Managing Credentials</td>
</tr>
<tr>
<td>list-datasource-jndinames</td>
<td>Lists all the JNDI names for a data source.</td>
<td></td>
</tr>
<tr>
<td>list-job-logs</td>
<td>Lists all the logs associated with a job.</td>
<td></td>
</tr>
<tr>
<td>list-jobs</td>
<td>Lists all job details that are visible to the user. You can scope the listing using options &quot;serviceinstance&quot; and/or &quot;application&quot;.</td>
<td></td>
</tr>
<tr>
<td>list-libraries</td>
<td>Lists all the shared libraries that are installed and available in the service instance.</td>
<td>Managing Shared Libraries</td>
</tr>
<tr>
<td>list-loggers</td>
<td>Lists the name and log level of all the Loggers or of a given Logger and, optionally, its children. Note that, while this command lists all the loggers and their logging levels, it hides any internal loggers.</td>
<td>Managing Logging Levels</td>
</tr>
<tr>
<td>list-service-metrics</td>
<td>Provides service performance statistics that help you measure an application's performance, identify performance bottlenecks, and monitor the health of the service as a whole. list-service-metrics is a superset of the query-service-metrics command (now deprecated) that lets you access the metrics based on this hierarchy:</td>
<td>Viewing Service Metrics</td>
</tr>
<tr>
<td>list-ssl-certificates</td>
<td>Lists all the trusted certificates from the SSL truststore. SSL outbound calls from Java Cloud Service - SaaS Extension are authorized based on these certificates.</td>
<td>Managing SSL Truststores</td>
</tr>
<tr>
<td>list-system-properties</td>
<td>Lists all persisted system properties.</td>
<td>Managing System Properties</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>For More Information</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>--------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>list-wss-certificates</td>
<td>Lists all the trusted certificates from the web service security (WSS) truststore.</td>
<td>Managing Web Services Security Truststore</td>
</tr>
<tr>
<td>query-access-logs</td>
<td>Gets access log records that match the given search criteria. The search scope is restricted using one of the combinations &quot;starttime,endtime&quot;. You can limit the number of log records in the search results to be returned by using -limit.</td>
<td>Viewing Access Logs</td>
</tr>
<tr>
<td>query-service-logs</td>
<td>Gets application log records that match the given search criteria. The search scope is restricted by using one of the combinations &quot;last,unit&quot; or &quot;starttime,endtime&quot;. You can limit the number of log records in the search results to be returned by using the -limit parameter.</td>
<td>Viewing Service Logs</td>
</tr>
<tr>
<td>query-service-metrics</td>
<td>Lists service instance application metrics.</td>
<td>Refreshing ADF Applications</td>
</tr>
<tr>
<td>refresh</td>
<td>Downloads the necessary applications and place them in a temporary location. It then redeployes those applications to your instance.</td>
<td>Refreshing ADF Applications</td>
</tr>
<tr>
<td>restart-service-instance</td>
<td>Restarts the service instance.</td>
<td>Managing Configurations</td>
</tr>
<tr>
<td>set-config</td>
<td>Sets the value for a configuration listed by the list-config command.</td>
<td>Managing Configurations</td>
</tr>
<tr>
<td>set-credential</td>
<td>Adds or updates a credential against key.</td>
<td>Managing Credentials</td>
</tr>
<tr>
<td>set-log-level</td>
<td>Sets the Log level of the Logger to the given level. Note that Oracle Java Cloud Service - SaaS Extension also supports Weblogic's logging convention.</td>
<td>Managing Logging Levels</td>
</tr>
<tr>
<td>set-system-property</td>
<td>Adds or updates an existing system property. Requires service instance restart to be effective.</td>
<td>Managing System Properties</td>
</tr>
<tr>
<td>setup-wss-trust</td>
<td>Automates the process of setting up Web Service Security (WSS) trust from a local WebLogic Server domain to a JCS-SaaS Extension instance in the cloud.</td>
<td>Setting Up Trust Between WebLogic Domains and JCS-SaaS Extension</td>
</tr>
<tr>
<td>Command</td>
<td>Description</td>
<td>For More Information</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>start</td>
<td>Starts an already installed application that is in the stopped state.</td>
<td></td>
</tr>
<tr>
<td>stop</td>
<td>Stops an installed and running application.</td>
<td></td>
</tr>
<tr>
<td>update</td>
<td>Updates an existing installed application.</td>
<td></td>
</tr>
<tr>
<td>update-library</td>
<td>Updates an existing installed shared library.</td>
<td>Managing Shared Libraries</td>
</tr>
<tr>
<td>usage</td>
<td>Prints the usage of this tool into the HTML file (defaulted to - usage.html) for off-line reference.</td>
<td></td>
</tr>
<tr>
<td>version</td>
<td>Provides the build (version) number of this tool.</td>
<td></td>
</tr>
</tbody>
</table>
Frequently Asked Questions for Oracle Java Cloud Service - SaaS Extension

This section provides answers to frequently asked questions (FAQ) about configuring and using Oracle Java Cloud Service - SaaS Extension. This technical FAQ supplements the more general Java FAQ on the Oracle Cloud website, at the following address: http://cloud.oracle.com/java.

Topics:

- How is Oracle Java Cloud Service - SaaS Extension different from Oracle Java Cloud Service?
- How do I create an on-premise WebLogic Server environment that is comparable to an Oracle Java Cloud Service - SaaS Extension instance?
- Can I set Log4j or JDK logging levels for my applications?
- If I’m using the JDeveloper IDE can I use Log4j with applications deployed to Oracle Java Cloud Service - SaaS Extension?
- Do I need to put my Log4j properties files in a particular location?
- How do I pass UTF-8 encoded characters in the request URL and how do I get the value into the application?
- Can I change a service name after I’ve already activated the service?

How is Oracle Java Cloud Service - SaaS Extension different from Oracle Java Cloud Service?

Like Oracle Java Cloud Service, Oracle Java Cloud Service - SaaS Extension provides an enterprise-grade platform to develop and deploy business applications in the cloud. The differences lie in how you will use the service: Java Cloud Service supports deployment of custom business application development while you would use Java Cloud Service–SaaS Extension to build extensions to existing Oracle SaaS products, such as CRM, HCM, and so on.

Some of the major difference between these two services are:

- While both services can be managed and monitored through their specific consoles—Java Cloud Service - SaaS Extension Control and the Java Cloud Service Console (both web-based interfaces)—Oracle Java Cloud Service - SaaS Extension can also be controlled via a command-line interface available with the Java Cloud Service - SaaS Extension SDK. Conversely, Oracle Java Cloud Service can be managed through REST APIs.
• With Oracle Java Cloud Service - SaaS Extension, you cannot access the configuration of the underlying application server, JVM, and/or operating system for any services, while this is possible with Oracle Java Cloud Service.

• Java Cloud Service - SaaS Extension supports Single-Sign On (SSO) out of the box. Once federation is enabled in the data center where your Software as a Service (such as Sales Cloud, Service Cloud, Marketing Cloud, and so on) is running, your Java Cloud Service - SaaS Extension applications won't require any changes to enable SSO capabilities.

• With Oracle Java Cloud Service - SaaS Extension, you deploy applications directly from Oracle Java Cloud Service - SaaS Extension Control (a web-based console), whereas you deploy applications for Oracle Java Cloud Service by using Fusion Middleware Control, the WebLogic Server Administration Console, WebLogic Scripting Tool (WLST) commands, or an IDE.

• Currently, Oracle Java Cloud Service - SaaS Extension is available to customers in North America and Europe. Oracle Java Cloud Service is available only from North American data centers.

For more detailed information and comparisons between both services, see the Oracle Cloud Java FAQ.

How do I create an on-premise WebLogic Server environment that is comparable to an Oracle Java Cloud Service - SaaS Extension instance?

An on-premise environment is a local WebLogic Server/Java EE environment that is comparable to an Oracle Java Cloud Service - SaaS Extension instance. An on-premise environment is useful for both developing and troubleshooting applications deployed to Oracle Java Cloud Service - SaaS Extension. See Creating an On-premise WebLogic Server Environment.

Can I set Log4j or JDK logging levels for my applications?

Yes, by using commands available from the Java Cloud Service - SaaS Extension SDK. This feature is particularly useful when you want to adjust the log level of loggers your application is using as it allows you to control the debugging resolution of the log statements. See Managing Logging Levels. See Downloading the Oracle Java Cloud Service - SaaS Extension SDK.

If I’m using the JDeveloper IDE can I use Log4j with applications deployed to Oracle Java Cloud Service - SaaS Extension?

Yes, however, Log4j is not part of the libraries available in JDeveloper, so you would need to explicitly download and include the Log4j library into your application.

Do I need to put my Log4j properties files in a particular location?

Oracle Cloud does not affect Log4j’s mechanism for locating its own configuration files; therefore, the location of Log4j properties files should be by default found on the system CLASSPATH. Note that FileAppenders can only write to /customer/scratch/**. For configuration file information see the Log4j documentation. See Guidelines for Applications When Accessing System Properties.

How do I pass UTF-8 encoded characters in the request URL and how do I get the value into the application?

Pass the encoded character in a format like myparam=%E6%B5%8B%E8%AF%95.
To get the actual value back from a servlet you must do the following:

```java
String paramValue = request.getParameter("myparam");  // returns value
encoded in WLS default encoding that is iso-8859-1
paramValue = new String(paramValue.getBytes("iso-8859-1"),"UTF-8") ;  //
Encodes to UTF-8 now.
```

**Note:**

An application, for example "gbk", can override the default WLS encoding. If that happens, set a `paramValue` variable as follows:

```java
paramValue = new String(paramValue.getBytes("gbk"),"UTF-8") ;
```

Can I change a service name after I've already activated the service?

No. The service name is unique within an identity domain and is used as part of the service URL:

```
Oracle Java Cloud Service - SaaS Extension - S1
Service URL Preview: https://[java1]-financedepthJAVA.us1.oraclecloud.com/...

* Service Name: java1
  Description: Java Service for Finance Dept
```

It cannot be changed
Troubleshooting Java Cloud Service - SaaS Extension

This section describes common issues that you might encounter when using Oracle Java Cloud Service - SaaS Extension and explains how you can resolve them.

Topics

• Use the Whitelist Tool
• ADF Deployment is Failing
• ADF Application Login Results in Blank Page
• A Signed JAR Appears as Unsigned After being Uploaded to the Cloud
• Java User Role Doesn't Allow Access to Console or SDK
• `java_OPSS_wlsaas` Resource Could Not be Prepared Because JDBC Driver Does Not Support XA (Global) Transactions
• Certificate in WSDL Doesn't Match the Certificate Being Validated
• Service Instance Does Not Restart
• Set UseSunHttpHandler property to True When Making Outbound HTTP(S) Calls
• How Do I Expose the WSDL for an Application Deployed in Java Cloud Service - SaaS Extension?
• SAAJ 1.1 Not Always Supported
• Memory Errors Affecting Application Deployment
• Problems with Outbound Connections

Use the Whitelist Tool

The Java Cloud Service - SaaS Extension Whitelist tool makes it easy for you to verify that applications you are trying to deploy are not using disallowed packages.

Quite often, problems with application deployment can be traced to disallowed Java packages in the application. Some Java packages, for example `java.rmi`, cannot be used in applications deployed to Oracle Java Cloud Service - SaaS Extension. You can test your applications for these disallowed packages by using the Whitelist tool included in the Oracle Java Cloud Service - SaaS Extension SDK. As part of the Java API validation, the Whitelist tool performs a type of compatibility test on every application installed or updated in a Java Cloud Service - SaaS Extension instance by validating deployment descriptors and other application configuration files, such as the `log4j.properties` file. If you are encountering additional deployment problems, you can locally validate an application by using `whitelist.jar`, which is available in the Java Cloud Service - SaaS Extension SDK (you can download the SDK from the Oracle Cloud Downloads page). This tool lets you scan one or more class files, JAR files, deployable archives (WAR or EAR), or exploded directories to locates any
disallowed or otherwise impermissible usages. You can also use it to verify whether
the input file (when the input file is not a class, JAR, WAR, or EAR) can be packaged
inside a deployable archive.

It is recommended that, whenever you encounter an application deployment problem,
you perform a Whitelist check on the application before proceeding to more intensive
troubleshooting.

The Whitelist Tool Command

Run the Whitelist by issuing this command:

./whitelist [-argument ...] [-help] [file1 file2 dir1 dir2 ...]

For example:

./whitelist -log /home/log/newlog.log /home/apps/myapp.war

The valid arguments are:

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
<th>Default</th>
<th>Shortcut</th>
</tr>
</thead>
<tbody>
<tr>
<td>log</td>
<td>The path to the log file to which the scan result will be written.</td>
<td>l</td>
<td>l</td>
</tr>
<tr>
<td>grid</td>
<td>The true/false flag that indicates if the error listing should be rendered in a grid. Ensure that your console window is wide enough so that the grid does not wrap.</td>
<td>false</td>
<td>g</td>
</tr>
<tr>
<td>gridwidth</td>
<td>The maximum width of the grid. You can use this value if you want to limit the width of the grid display (for instance, when you have a smaller display). <strong>Note:</strong> If you specify a smaller width, the grid might not fit within the width. This is applicable when grid is true.</td>
<td>100</td>
<td>gw</td>
</tr>
<tr>
<td>includesummary</td>
<td>A true/false flag that, when set to true, causes a summary report to be printed.</td>
<td>false</td>
<td>is</td>
</tr>
<tr>
<td>showall</td>
<td>A true/false flag that, when set to true, displays all the warning and errors from a trusted third party API. If you are trying to deploy an application that uses APIs from trusted third parties, it is recommended that you set this flag to true.</td>
<td>false</td>
<td>sa</td>
</tr>
</tbody>
</table>
Using the Whitelist Tool

In this example, we’ll test the file benefits.war, which is on the local file system in the D:\Applications folder, and send the log, as file called benefits.log, to the folder C:\java_logs.

To use the Whitelist tool and specify a path for the log file, do the following:

1. Locate the JAR files, deployable archives (WAR or EAR), or exploded directories you want to check.
2. Open a command prompt and navigate to the SDK_HOME\lib directory (where SDK_HOME is the Oracle Java Cloud Service - SaaS Extension SDK installation directory; for example, D:\oracle_javacloud_sdk 15.1.2\oracle-javacloud-sdk\lib).
3. Run the Whitelist tool by entering:

   ./whitelist -log C:\java_logs\benefits.log D:\Applications\benefits.war

A check of C:\java_logs\ shows the file benefits.log:

If you wanted to output the log directly to your screen, you could use the grid argument, like this:

   ./whitelist D:\Applications\benefits.war -grid

The system would respond:
ADF Deployment is Failing

ADF application deployments are failing because an `<exact-match>` value is not supplied in the WebLogic deployment descriptor.

When deploying an ADF application, that deployment might fail. The log will contain a message similar to this:

```
2014-09-03 12:45:03 CDT: Starting action "Deploy Application"
2014-09-03 12:45:03 CDT: Deploy Application started
2014-09-03 12:45:12 CDT: weblogic.application.ModuleException: weblogic.application.ModuleException: 
at weblogic.servlet.internal.WebAppModule.startContexts(WebAppModule.java:1531)
   at weblogic.servlet.internal.WebAppModule.start(WebAppModule.java:488)
```

This is most likely because the ADF application’s WebLogic deployment descriptor, `weblogic.xml`, has an incorrect JSF library reference. Whereas Java Cloud Service -
SaaS Extension supports both JSF 1.2 and JSF 2.0 libraries, ADF 11.1.1.9.0 only works with JSF 1.2. If your weblogic.xml file does not use an <exact-match> specification within the <library-ref> element:

```xml
<library-ref>
  <library-name>jsf</library-name>
  <specification-version>1.2</specification-version>
</library-ref>
```

WebLogic Server will select the latest library (that being the unsupported JSF 2.0 library), which causes deployment to fail. You can verify this error by running the Whitelist tool against the archive. You should see the following warning:

Recommended child element "exact-match" missing under element bea-weblogic:weblogic-web-app****-weblogic:library-ref.
Element exact-match should be specified and set to true for JSF 1.2 applications..

See Use the Whitelist Tool.

Workaround

To rectify this situation, add the <exact-match> element with the appropriate <library-ref>:

```xml
<library-ref>
  <library-name>jsf</library-name>
  <specification-version>1.2</specification-version>
  <exact-match>true</exact-match>
</library-ref>
```

ADF Application Login Results in Blank Page

When logging in to an ADF application deployed on JCS-SaaS Extension instances provisioned before 17.1.3, you might get a blank screen. Use the -refresh CLI command to resolve this issue.

ADF users trying to start instances created with a version of JCS-SaaS Extension predating 17.1.3 but then patched with a 17.1.3 (or later) release might encounter a blank page after ADF login. Because the ADF libraries were updated for the later version of JCS-SaaS Extension, you need to recompile the JSP pages of any ADF applications deployed on JCS-SaaS Extension instances that predate 17.1.3.

To redeploy, use the -refresh command available in the 17.2.1 SDK CLI. This command does the following:

- Downloads the necessary applications and place them in a temporary CLI location.
- Redeploys those applications to your instance.
For example:

$ ./javacloud -u user.self@myCo.com -dc dataCenterCode -id myDomain1234 -si myInstance1715 -refresh -app hcmconnect-ear-1.1.0-SNAPSHOT

**Note:**

-dc is the data center code. The accepted values are us1 (United States), us2, em1 (Europe, Middle East, Africa), em2, ap1 (Asia, Pacific), or ap2, depending on where your instance is located.

You should get this response:

```
[INFO]    - The application is being updated.

1:Job Id         - 1912
--------------      - -----------------------
->               - Properties
------------     - -----------------------
Status           - NEW
Identity Domain  - mycotrial1715
Service Instance - prodtria1715
Application      - hcmconnect-ear-1.1.0-SNAPSHOT
Start Time       - Tuesday, February 21, 2017 1:19:30 PM PST
Operation        - Redeploy Application
--------------    - -----------------------

[TIP]            - You can use the command "job-status" to monitor a job.
```

The -refresh command is described in detail in Refreshing ADF Applications.

**A Signed JAR Appears as Unsigned After Being Uploaded to the Cloud**

If you upload an application—for example, an EAR file—to the Cloud and that archive contains a signed JAR, when you retrieve that JAR, it might then be unsigned. For security reasons, Oracle Java Cloud Service - SaaS Extension recompiles user applications and repackagess them before deployment (see Considerations When Developing Applications on Oracle Java Cloud Service - SaaS Extension). This process will cause any applications containing a signed applet JAR to lose the signature.

**Workaround**

To avoid any changes to the JAR files that you don't want executed in Java Cloud Service - SaaS Extension need to be packaged and accessed differently:
Note:

Applet JARs are loaded on the browser and not on the Java Cloud Service - SaaS Extension runtime. If a class needs to be loaded into Java Cloud Service - SaaS Extension runtime, the following suggestions do not apply.

- Package the signed JAR with an extension other than .jar (for example, DME.jar). When the JAR is served to the client, the application servlet should convert the JAR with the correct name; for example, DME.jar. This will prevent recompilation of this file.

Note:

If the code contained in this JAR file is executed on the server side, it will be recompiled before execution.

- Upload the signed JAR to /customer/scratch on the file system share accessible to the application on the Java Cloud Service - SaaS Extension service instance. This location can be accessed inside the user application by using Java standard file I/O API's. Files can be uploaded by using a file system shell feature of the Java Cloud Service - SaaS Extension command-line interface (supplied with the Java Cloud Service - SaaS Extension SDK).

- Upload the signed JAR to Oracle Storage Cloud Service, where it can be accessed by using the Oracle Storage Cloud Service REST APIs or Java APIs from within Oracle Java Cloud Service - SaaS Extension.

Java_User Role Doesn’t Allow Access to Console or SDK

Users assigned the role Java_User role cannot access the SDK or Java Console.

If you have been granted the role Java_User for Oracle Java Cloud Service - SaaS Extension, you have limited options with what you can do with the service. For example, you can't see the Java Cloud Service - SaaS Extension instance in My Services/My Account, nor can you access Java Cloud Service - SaaS Extension Control or the SDK.

Workaround

To access these — and other blocked — features, contact your ID administrator and have your role changed to Java_Administrator. The role of Java_User is an out-of-the-box sample role that allows users to access those samples. Otherwise for general application access, this role is not useful.
java_OPSS_wlsaas Resource Could Not be Prepared Because JDBC Driver Does Not Support XA (Global) Transactions

You might experience issues with an instance because the java_OPSS_wlsaas resource could not be prepared.

Commencing with JCS - SaaS Extension 17.1.3, newly provisioned instances support Fusion Middleware 11.1.1.9. This support requires that JCS - SaaS Extension has an internal datasource for the OPSS policy store. This datasource is created in the JCS - SaaS Extension instance domain to support the policy store in the created schema. By default, the OPSS datasource must support the global transactions protocol as OnePhaseCommit. With this configuration, the customer might receive an error message similar to this:

```
Exception occurred during commit of transaction ...
weblogic.transaction.RollbackException: Could not prepare resource 'java_OPSS_wlsaas' JDBC driver does not support XA, hence cannot be a participant in two-phase commit. To force this participation, set the GlobalTransactionsProtocol attribute to LoggingLastResource (recommended) or EmulateTwoPhaseCommit for the Data Source = java_OPSS
```

This happens because the OPSS datasource doesn’t support XA Commits distributed transactions.

**Workaround**

To resolve this issue for instances created subsequent to 17.1.3 but before 17.4.1, configure the global support transactions value for the OPSS datasource by using the set-config SDK command to set the `internal.datasource.opss.global.txn` configuration value to None; for example:

```
./javacloud.jar -user myUsername -dc myDataCenter -serviceinstance jcssx001b -identitydomain myIdDomain -set-config -name internal.datasource.opss.global.txn -value None
```

For this new configuration to take effect, restart the instance by using the command `- restart-service-instance`.

For additional information on setting this configuration, see Managing Configurations.

Certificate in WSDL Doesn’t Match the Certificate Being Validated

As a best practice, you should not imbed the web service descriptor (WSDL) in the application or you risk a mismatch with the certificate being validated.

If the certificate in the WSDL of an application deployed on Java Cloud Service - SaaS Extension attempting to access web services for a SaaS application doesn’t match the
certificate being validated for that application, you will receive a message similar to this:


Workaround

To avoid this, do not imbed the WSDL in the application. If you have imbedded the WSDL, you will need to override it at run-time by using something like this:

salesPartyService_Service = new salesPartyService_Service(new URL("desiredwsdl"), QName );

This overrides the WSDL associated with the proxy instead of the end point. This will account for the unavailability of the old WSDL and the possibility that certificate information in the WSDL will change

Service Instance Does Not Restart

When you click Restart Service Instance, the application fails to restart.

Often, you will attempt to restart a service instance to relaunch a stalled application but nothing happens. To ensure that the instance and any applications running on it restart successfully, you must perform a forced restart by selecting Force a restart of the service instance, even if there are active jobs? on the Restart Service Instance’s Confirmation dialog box.

You can also force a service instance restart from the command-line interface by using the restart-service-instance command with the force option (-f); for example:

./javacloud restart-service-instance -f -u myUser -id myIdDomain -si myServiceInstance

Using restart-service-instance requires a password (-p) but if you omit it in the command (as in the preceding example), the system will prompt you for it before forcing the restart. This is preferred to using the -p flag, if your command line renders the password in plain text.
Be aware that when you restart a service instance, all servers in that instance will be stopped and restarted. If the service instance is running a single server, you might experience some downtime. If the service instance has multiple servers, they will be restarted sequentially. During this time, other operations will not be permitted.

**Set UseSunHttpHandler property to True When Making Outbound HTTP(S) Calls**

As a best practice, set the `UseSunHttpHandler` property to `true`.

If you are making an outbound http(s) call, you should use your own or some other third party code to set the property `UseSunHttpHandler` to be true. In the past when the standard Java security manager was turned on in JCS-SaaS Extension instances, Oracle discouraged this practice; however, since the standard Java security manager is turned off, we now recommend using Sun's handlers.

**Note:**

Before using Sun's handler, be sure to turn off the security manager. See [Special Note on Disabling the Security Manager](#).

To set the `UseSunHttpHandler` property, use the Settings section of the Administration Console, or use the `set-config` CLI command to set the `sun.http.handler.enabled` configuration to `true`. Do not call `System.setProperty("UseSunHttpHandler", "true")` from code. See [Special Note on the UseSunHttpHandler Property](#).

**How Do I Expose the WSDL for an Application Deployed in Java Cloud Service - SaaS Extension?**

You can expose a web service descriptor (WSDL) for an application by using an empty `<login-config>` element in your `web.xml` deployment descriptor.

To expose a WSDL to an application deployed in Oracle Java Cloud Service - SaaS Extension, you need to treat the application like you would Internet Public Pages and make the WSDL and the WebService endpoint available on the Internet while bypassing SSO perimeter security. You do this by providing an empty security element called `<login-config/>` in the `web.xml` deployment descriptor, as shown in this example:

```xml
  ...
  <login-config/>
  ...
</web-app>
```

**SAAJ 1.1 Not Always Supported**

You might encounter a message telling you that a class in your implementation of JCS-SaaS Extension with Weblogic Server 10.3.6 does not support SAAJ 1.1.
If you are running JCS-SaaS Extension with Weblogic Server 10.3.6 and a JAX-WS custom business service and you call the external Web Service on the server, you might get this exception:

```
Call to Web Service failed, error:
java.lang.UnsupportedOperationException: This class does not support SAAJ 1.1
```

This occurs because of an issue with WebLogic's default SAAJ implementation in package `weblogic.webservice.core.soap`. To correct this, from the command line interface, change the SAAJ implementation by setting the `-name` parameter of the `set-system-property` command to `javax.xml.soap.MessageFactory` and the `-value parameter` to `weblogic.xml.saaj.MessageFactoryImpl`:

```
./javacloud set-system-property -user userName -identitydomain identityDomain -serviceinstance serviceInstance -name javax.xml.soap.MessageFactory -value weblogic.xml.saaj.MessageFactoryImpl
```

For more information on `set-system-properties`, navigate to the `$$SDK_HOME/doc/index.html` file (where `SDK_HOME` is the directory containing your Oracle Java Cloud Service - SaaS Extension installation) and click `CLI-Javacloud.jar`. You can also access all of the SDK documentation via the "Welcome App". Also see Using the Command-Line Interface to Monitor Oracle Java Cloud Service - SaaS Extension.

### Memory Errors Affecting Application Deployment

If you are receiving either PermGen or OutofMemory errors during deployment, you might be able to correct them by using flattened configuration management tools to adjust the Perm or Stack size.

#### PermGen Errors

If you are encountering PermGen errors during deployment, set the configuration `jvm.arg.max.perm.size` to a value between 512M (default) and 1024M and restart the instance:

```
./javacloud set-config -name jvm.arg.max.perm.size -value 1024 -identityDomain myIdentityDomain -serviceInstance myServiceInstance -name userName myUserName -password myUserPassword
```

The minimum and maximum values will be enforced by the tools.

#### OutofMemory Errors

If you are encountering OutofMemory error during deployments, use the argument `jvm.arg.stack.size` to reduce the stack size value, thus increasing the availability of heap:

```
./javacloud set-config -name jvm.arg.stack.size -value 256 -identityDomain myIdentityDomain -serviceInstance myServiceInstance -name userName myUserName -password myUserPassword
```
Problems with Outbound Connections

If you are encountering problems making outbound connections, you should use an outbound proxy.

You can use one of these get-able properties:

- `http.proxyHost`
- `http.proxyPort`
- `https.proxyHost`
- `https.proxyPort`

See Guidelines for Applications When Accessing System Properties.

Can’t Enable Java Mail Service from the JCS-SaaS Extension Administration Console

If you can't set `java.mail.enabler` to `true` from JCS-SaaS Extension Administration Console, use `set-config` to resolve the issue.

When using JCS-SaaS Extension Administration Console for the first time and are changing the value of `java.mail.enabler` setting to `true`, you might receive this error:

Error setting value: Java mail service couldn't be enabled as one of the prerequisites is missing. Please contact ORACLE support.

You can correct this situation by using the `set-config` command in the command-line interface (CLI), for example:

```
./javacloud -id myIDDomain62337 -si javatrial1870 -u johnsmith@example.com -set-config -name java.mail.enabler -value true
```

For details on using `set-config`, see Managing Configurations.
Oracle Java Cloud Service - SaaS Extension Deprecated Features and APIs

Certain features and APIs are either unsupported or are deprecated in this release of Oracle Java Cloud Service - SaaS Extension.

Topics:
• About the Oracle Java Cloud Service - SaaS Extension Deprecation Policy
• Unsupported Features and APIs

About the Oracle Java Cloud Service - SaaS Extension Deprecation Policy

The following describes the deprecation policy for Oracle Java Cloud Service - SaaS Extension:

• All APIs marked as deprecated in Javadoc for WebLogic Server release 10.3.6 and ADF release 11.1.1.9.0 are deprecated for the Oracle Java Cloud Service - SaaS Extension. See the Oracle WebLogic Server API Reference.
• As a general rule, APIs that are marked as deprecated for Oracle Java Cloud Service - SaaS Extension in a specific version of the product, will be fully removed in the next major product update.

Unsupported Features and APIs

There are a number of features and APIs not supported in this release of Oracle Java Cloud Service - SaaS Extension.

Oracle Java Cloud Service - SaaS Extension does not support:

• Any API deprecated in WebLogic Server release 10.3.6 or earlier.
• Any API deprecated in ADF release 11.1.1.9.0 or earlier.
• In addition to the areas detected by the Oracle Java Cloud Service - SaaS Extension whitelist, Oracle Java Cloud Service - SaaS Extension does not support the features and capabilities listed in the following table. This table includes workarounds where applicable.

<table>
<thead>
<tr>
<th>Unsupported Feature</th>
<th>Alternative</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct socket connections.</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>Direct JAR deployment.</td>
<td>Embed JAR in EAR.</td>
</tr>
<tr>
<td>Java EE Connector Architecture (JCA) Container - RAR deployments.</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>Unsupported Feature</td>
<td>Alternative</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>JAX-RPC-based web services.</td>
<td>Convert to JAX-WS web services.</td>
</tr>
<tr>
<td>Applications exposing asynchronous SOAP based web services using WS-Addressing.</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>Use of WS-* specifications other than WS-Security (through OWSM policies).</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>Remote invocations with a transport protocol other than HTTPS (including plain text HTTP).</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>Coherence applications, managed or used through WebLogic Server ActiveCache.</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>Direct usage of any JRF API components other than ADF (for example, the direct use of Oracle Platform Security Services (OPSS) and ODL APIs).</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>Direct use of Oracle JDBC Driver APIs.</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>Use of SQL statements specific to a database instance other than Oracle Database 11g (11.2).</td>
<td>Convert application to use Oracle Database.</td>
</tr>
<tr>
<td>Direct modification of the Java command-line parameters.</td>
<td>Use web.xml context parameters or set system properties programmatically. See context-param in Oracle Fusion Middleware Developing Web Applications, Servlets, and JSPs for Oracle WebLogic Server.</td>
</tr>
<tr>
<td>Application scoped JDBC modules.</td>
<td>Modify application to use the system scoped data source created through the database association.</td>
</tr>
<tr>
<td>Setting of operating system environment variables or JVM/Server command-line parameters.</td>
<td>Dependencies on these variables would need to be brought into the application deployment archive (for example, the packaging of a properties.xml file and reading of the information from it).</td>
</tr>
<tr>
<td>EJB 2.x Entity Beans.</td>
<td>EJB 3.0 and JPA.</td>
</tr>
<tr>
<td>The following ADF features:</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>– ADF Desktop Integration</td>
<td></td>
</tr>
<tr>
<td>– ADF MBeans</td>
<td></td>
</tr>
<tr>
<td>– ADF seeded customizations or cross-session personalization (MDS)</td>
<td></td>
</tr>
<tr>
<td>– ADF Mobile</td>
<td></td>
</tr>
<tr>
<td><strong>Note:</strong> ADF Mobile applications are designed to run on mobile devices, and so cannot run on Oracle Cloud. However, you can integrate ADF Mobile applications with Oracle Java Cloud Service - SaaS Extension instances (for example, using supported RESTful APIs).</td>
<td></td>
</tr>
<tr>
<td>– ADF Active Data Services</td>
<td></td>
</tr>
<tr>
<td>– ADF Data Controls for BI, Essbase, BAM, and JMX</td>
<td></td>
</tr>
<tr>
<td>Application deployment archives that have a size of more than 95MB.</td>
<td>Not applicable.</td>
</tr>
<tr>
<td>Unsupported Feature</td>
<td>Alternative</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>All <code>sun.*</code> packages in the Java SDK, including <code>sun.misc.BASE64Encoder</code>, are not supported. The <code>sun.*</code> packages are a security risk because they are internal-only, and so are not part of the public API.</td>
<td>Alternative options for common encoders that have similar functionality are:</td>
</tr>
<tr>
<td></td>
<td>– Java SE ships with JAXB. The <code>javax.xml.bind.DatatypeConverter</code> has similar static methods, see <code>parseBase64Binary()</code> and <code>printBase64Binary()</code>.</td>
</tr>
<tr>
<td></td>
<td>– Apache Commons Codec, see <code>http://commons.apache.org/proper/commons-codec/a security risk</code>.</td>
</tr>
<tr>
<td>Java Standard security manager</td>
<td>This security manager has been replaced by a Java Cloud Service - SaaS Extension-specific byte-code translation based security manager. For instructions on disabling the Java standard security manager, see the Special Note on Disabling the Security Manager in Managing System Properties.</td>
</tr>
</tbody>
</table>

In addition, Oracle Java Cloud Service - SaaS Extension does not support the use of the public WebLogic Server 10.3.6 APIs and capabilities as described in the following table.

<table>
<thead>
<tr>
<th>Unsupported WebLogic Server Capability</th>
<th>Description/Rationale</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>weblogic.wtc.*</code></td>
<td>Tux integration is not supported.</td>
</tr>
<tr>
<td><code>com.bea.logging</code></td>
<td>Deprecated logging API.</td>
</tr>
<tr>
<td><code>com.bea.httppubsub</code></td>
<td>Not supported.</td>
</tr>
<tr>
<td><code>com.bea.security.*</code></td>
<td>All security in Oracle Cloud is handled at the identity management level. No custom security provider or model is supported.</td>
</tr>
<tr>
<td><code>weblogic.apache.*</code></td>
<td>Deprecated and replaced by <code>org.apache.html.dom</code>.</td>
</tr>
<tr>
<td><code>weblogic.webservice.*</code></td>
<td>Deprecated WebLogic Server 8.1 web services features.</td>
</tr>
<tr>
<td><code>weblogic.cluster.*</code></td>
<td>Do not expose WebLogic Server clustering (including the Singleton service) at the Oracle Java Cloud Service - SaaS Extension level.</td>
</tr>
<tr>
<td><code>weblogic.connector.*</code></td>
<td>Oracle Java Cloud Service - SaaS Extension does not support JCA.</td>
</tr>
<tr>
<td><code>weblogic.deploy.*</code></td>
<td>Deployment must be performed through Oracle Java Cloud Service - SaaS Extension-specific interfaces.</td>
</tr>
<tr>
<td><code>weblogic.management.*</code></td>
<td>Do not expose the WLS JMX tree at Oracle Java Cloud Service - SaaS Extension level.</td>
</tr>
<tr>
<td><code>weblogic.security.*</code></td>
<td>All security is handled at the Oracle Cloud identity management level. No custom security provider or model is supported. However authenticated user's principles can be read.</td>
</tr>
<tr>
<td><code>weblogic.time.*</code></td>
<td>Deprecated and not supported at Oracle Java Cloud Service - SaaS Extension level.</td>
</tr>
<tr>
<td>Unsupported WebLogic Server Capability</td>
<td>Description/Rationale</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>-----------------------</td>
</tr>
<tr>
<td>weblogic.uddi.*</td>
<td>Deprecated in previous version of WebLogic Server.</td>
</tr>
<tr>
<td>weblogic.workarea.*</td>
<td>Deprecated in previous version of WebLogic Server.</td>
</tr>
<tr>
<td>.NET and C APIs for JMS</td>
<td>Oracle Java Cloud Service - SaaS Extension does not support C or .NET clients.</td>
</tr>
</tbody>
</table>