V17.6.0

Amazon Web Services Integration

December 2017 - Initial draft

High Risk Activity

The Customer acknowledges and agrees that the Product is designed, developed and manufactured as contemplated for general use, including without limitation, general office use, personal use, household use, and ordinary industrial use, but is not designed, developed and manufactured as contemplated for use accompanying fatal risks or dangers that, unless extremely high safety is secured, could lead directly to death, personal injury, severe physical damage or other loss (hereinafter "High Safety Required Use"), including without limitation, nuclear reaction control in nuclear facility, aircraft flight control, air traffic control, mass transport control, medical life support system, missile launch control in weapon system. The Customer shall not use the Product without securing the sufficient safety required for the High Safety Required Use. In addition, FUJITSU (or other affiliate's name) shall not be liable against the Customer and/or any third party for any claims or damages arising in connection with the High Safety Required Use of the Product.

Export Restrictions

Exportation/release of this document may require necessary procedures in accordance with the regulations of your resident country and/or US export control laws.

Contents

About this Manual

This manual describes the integration of the Amazon Elastic Compute Cloud Web service (Amazon EC2), a major component of Amazon Web Services (AWS), with Open Service Catalog Manager (OSCM) .

This manual is structured as follows:

Chapter	Description
Introduction on page 7	Provides an overview of the OSCM AWS integration, the components involved, and the supported usage scenarios.
Installing the AWS Integration Software on page 9	Describes how to prepare and carry out the installation of the AWS integration software.
Creating and Publishing Services on page 21	Describes how to create and publish services for AWS in OSCM.
Using AWS Services in OSCM on page 23	Describes how to provision and deprovision virtual servers in AWS through services in OSCM.
Administrating the AWS Integration on page 25	Describes administration tasks related to the OSCM AWS integration.
Uninstallation on page 31	Describes how to uninstall the OSCM AWS integration software.

Readers of this Manual

This manual is intended for operators who want to offer virtual servers controlled by AWS through services on a marketplace provided by OSCM. It assumes that you have access to an existing OSCM installation and that you have an AWS account. In addition, you should have basic knowledge of Amazon EC2 and you should be familiar with the concepts and administration of OSCM.

Notational Conventions

This manual uses the following notational conventions:

Add	Names of graphical user interface elements.
`init`	System names, for example command names and text that is entered from the keyboard.
`<variable>`	Variables for which values must be entered.
`[option]`	Optional items, for example optional command parameters.
`one | two`	Alternative entries.
`{one | two}`	Mandatory entries with alternatives.

Abbreviations

This manual uses the following abbreviations:

Amazon EC2	Amazon Elastic Compute Cloud
AMI	Amazon Machine Image
APP	Asynchronous Provisioning Platform
AWS	Amazon Web Services
DBMS	Database Management System
IaaS	Infrastructure as a Service
IdP	SAML Identity Provider
OSCM	Open Service Catalog Manager
SAML	Security Assertion Markup Language
STS	Security Token Service
WSDL	Web Services Description Language
WSIT	Web Services Interoperability Technologies

Available Documentation

The following documentation on OSCM is available:

- *Overview:* A PDF manual introducing OSCM. It is written for everybody interested in OSCM and does not require any special knowledge.
- *Operator's Guide:* A PDF manual for operators describing how to administrate and maintain OSCM.
- *Technology Provider's Guide:* A PDF manual for technology providers describing how to prepare applications for usage in a SaaS model and how to integrate them with OSCM.
- *Supplier's Guide:* A PDF manual for suppliers describing how to define and manage service offerings for applications that have been integrated with OSCM.
- *Reseller's Guide:* A PDF manual for resellers describing how to prepare, offer, and sell services defined by suppliers.
- *Broker's Guide:* A PDF manual for brokers describing how to support suppliers in establishing relationships to customers by offering their services on a marketplace.
- *Marketplace Owner's Guide:* A PDF manual for marketplace owners describing how to administrate and customize marketplaces in OSCM.
- *OpenStack Integration:* A PDF manual for operators describing how to offer and use virtual systems controlled by OpenStack through services in OSCM.
- *Amazon Web Services Integration:* A PDF manual for operators describing how to offer and use virtual servers controlled by the Amazon Elastic Compute Cloud Web service through services in OSCM.
- *Online Help:* Online help pages describing how to work with the administration portal of OSCM. The online help is intended for and available to everybody working with the administration portal.

1 Introduction

Open Service Catalog Manager (OSCM) is a set of services which provide all business-related functions and features required for turning on-premise applications and tools into 'as a Service' (aaS) offerings and using them in the Cloud. This includes ready-to-use account and subscription management, online service provisioning, billing and payment services, and reporting facilities.

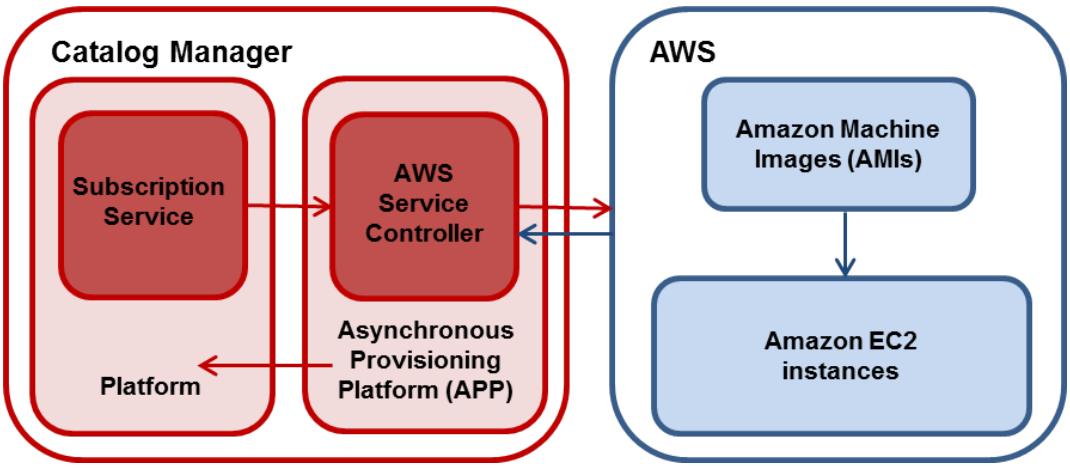
Amazon Web Services (AWS) is a collection of remote computing services that together make up a Cloud computing platform offered by Amazon. Amazon Elastic Compute Cloud (Amazon EC2) is one of the central Web services of AWS. It provides computing capacities in the Cloud and allows you to quickly scale these capacities as your computing requirements change.

The integration of AWS with OSCM provides for an Infrastructure as a Service (IaaS) solution that leverages the features of both products: Through services, which are published on a marketplace in OSCM, users can request and use virtual servers in Amazon EC2. The usage costs can be calculated and charged by means of the OSCM billing and payment services.

The AWS integration package provided with OSCM includes all components required for connecting an existing OSCM installation with AWS. This manual describes how to deploy this package and how to create and use services for Amazon EC2 on an OSCM marketplace.

1.1 Components Involved in the AWS Integration

The following picture provides an overview of the main components involved in the integration of OSCM and AWS:



In OSCM, customer subscriptions are managed by means of the **Subscription service**. When a customer creates or terminates a subscription for an Amazon EC2 instance in AWS, the Subscription service asynchronously triggers the corresponding actions in AWS through the **Asynchronous Provisioning Platform (APP)** and the **AWS service controller**: Virtual servers are created or deleted in AWS.

APP is a framework which provides a provisioning service, an operation service, as well as functions, data persistence, and notification features which are required for integrating applications with OSCM in asynchronous mode. The actual communication with the applications is carried out

by service controllers. APP and the AWS service controller are the main components that make up the AWS integration software.

Amazon EC2 allows customers to provision and use virtual servers on which to run their applications. Each virtual server is based on an Amazon Machine Image (AMI). An AMI serves as the basic unit of deployment for services delivered with Amazon EC2. AWS customers can either request pre-configured AMIs or they can create their own images. They can provision their images with a variety of operating systems and load them with custom application environments.

Each APP installation supports one AWS service controller. This limitation can be overcome by installing APP several times to different application server domains. The need for more than one service controller may arise because multiple AWS accounts or technology provider organizations have to be used.

1.2 Usage Scenarios

The OSCM AWS integration supports the following usage scenarios:

- **Provisioning of a virtual server**: When a customer subscribes to a corresponding service on an OSCM marketplace, the service controller triggers AWS to create an Amazon EC2 instance based on a specific AMI.
- **Automatic execution of user-specific configuration data**: When a customer subscribes to a corresponding service on an OSCM marketplace, he has the option of passing user-specific configuration data to the Amazon EC2 instance to be provisioned. The data can be used to modify the static information defined in the underlying AMI. Thus, the customer can, for example, perform automated configuration tasks or run scripts.
- **Starting and stopping a virtual server**: A customer can explicitly start and stop an Amazon EC2 instance by executing a service operation at the corresponding subscription.
- **Deletion of a virtual server**: When a customer terminates a subscription for an Amazon EC2 instance, the service controller triggers AWS to delete the instance. The subscription is terminated in OSCM independent of whether the deletion is successful in AWS.

In Amazon EC2, the virtual servers created for OSCM subscriptions are managed in the same way as other virtual servers. They can be viewed and monitored with the available AWS tools.

Modifying a subscription and thereby triggering modifications of the virtual server in AWS is not supported. For more details on the supported usage scenarios, refer to *Using AWS Services in OSCM* on page 23.

2 Installing the AWS Integration Software

The following sections describe how to install and configure the AWS integration software as well as the preparations you need to take beforehand.

Installing the AWS integration software consists of installing APP and registering the AWS service controller.

If you already have a working APP installation in your environment, proceed as described in *Installing the AWS Service Controller in an Existing APP Environment* on page 17.

2.1 Prerequisites and Preparation

The following sections describe the prerequisites that must be fulfilled and the preparations you need to take before installing and deploying the AWS integration software.

2.1.1 OSCM and AWS

- You must have access to a fully functional OSCM installation. You can install the AWS integration software in the same environment or on a different server.
- You must have access to OSCM as an administrator and as a technology manager of an organization that has at least the technology provider role.
- You must have an AWS account.

2.1.2 Hardware and Operating Systems

The AWS integration software as a Java application does not rely on specific hardware or operating systems. It can be deployed on any platform supported by the application server and the database management system.

2.1.3 Java and Ant

The AWS integration software requires a Java Development Kit (JDK), version 8, 64 bit. Deployment with JDK 8, Update 121 has been tested and is recommended.

In order to be able to execute the installation scripts, you need to install the Apache Ant 1.10 (or higher) open source software. In the subsequent sections, `<ANT_HOME>` is the installation directory of Apache Ant.

2.1.4 Application Server

The AWS integration software must be deployed on an application server compatible with Java EE version 7. The following application server is supported:

Oracle GlassFish Server, version 4.1.2.

You can deploy the AWS integration software on the application server you use for OSCM. Alternatively, you can use a separate application server installation.

Be aware that when operating OSCM in SAML_SP mode, every Web service client must run in a separate domain of the application server.

Note: Before installing GlassFish, make sure that the `JAVA_HOME` environment variable points to a Java Development Kit (JDK), version 8, 64 bit.

Proceed as follows:

1. Install the application server as described in its documentation, and configure it as required by your environment.

Note: Make sure that the path of the GlassFish installation directory does not contain blanks.

2. After you have configured GlassFish, make a backup copy of the GlassFish installation.
3. Make sure that GlassFish is running in a JDK 8 environment. Also, make sure that no other applications (e.g. Tomcat) are running on your GlassFish ports.

The installation of the AWS integration software creates the `app-domain` domain in your application server. If required, you can change the domain name in the `glassfish.properties` file before starting the installation.

In the subsequent sections, `<GLASSFISH_HOME>` is the installation directory of GlassFish.

2.1.5 Relational Database

The AWS integration software stores its data in a relational database. The following database management system (DBMS) is supported:

PostgreSQL, version 9.1.12.

Install the DBMS as described in its documentation.

If required, you can use a separate machine for the AWS integration database.

Setup and Configuration

Edit the file

`<postgres_dir>/data/postgresql.conf`

as follows (`<postgres_dir>` is the PostgreSQL installation directory):

1. Set the `max_prepared_transactions` property value to `50`.
2. Set the `max_connections` property value to `210`.

   This property determines the maximum number of concurrent connections to the database server.

   Note the following: This setting is used in combination with the JDBC pool size settings for the domains on your application server. If you change the JDBC pool size, you might need to adapt the `max_connections` setting. Refer to the OSCM *Operator's Guide*, section *Tuning Performance*, for details.
3. Set the `listen_addresses` property value:

   Specify the IP addresses of all application servers on which the database server is to listen for connections from client applications. If you use the entry `'*'`, which corresponds to all available IP addresses, you must be aware of possible security holes.
4. Save the file.

If you use a server name in all configuration files instead of `localhost` during installation, edit the file

`<postgres_dir>/data/pg_hba.conf`

as follows (<postgres_dir> is the PostgreSQL installation directory):

1. Add the IP address of the application server that is to host the AWS integration software.
   For example:

```
host all all 123.123.12.1/32 md5
```

   Also add the application server's IPv6 address.
   For example:

```
host all all fe80::cdfb:b6ed:9b38:cf17/128 md5
```

   There are authentication methods other than md5. For details, refer to the PostgreSQL documentation.
2. Save the file.

Restart your PostgreSQL server for the changes to take effect.

2.1.6 Mail Server

To inform users about relevant issues, the AWS integration software requires a mail server in its environment. You can use any mail server that supports SMTP.

The settings for addressing the mail server are defined in the glassfish.properties file of the AWS integration package.

2.2 Installation

The installation of the AWS integration software consists of the following main steps:

1. *Preparing the Software and Setup Utilities* on page 11
2. *Configuring the AWS Integration* on page 12
3. *Setting up the Database* on page 14
4. *Setting up the Application Server Resources* on page 14

2.2.1 Preparing the Software and Setup Utilities

The AWS integration software and setup utilities are provided in the AWS integration package, oscm-aws-install-pack.zip. The contents of the package need to be made available in your environment as follows:

Extract the contents of the oscm-aws-install-pack.zip package to a separate temporary directory on the system from where you want to install and deploy the AWS integration software.

In the following sections, this directory is referred to as <install_pack_dir>.

After extraction, the following directories are available:

- **databases/app_db**
  Configuration files for setting up the database used by the AWS integration software.
- **doc**
  The *Amazon Web Services Integration* guide (this manual).
- **domains/app_domain**
  - Configuration file (glassfish.properties) for setting up the application server resources for the domain to which the AWS integration software is to be deployed.

  - `applications`: The APP application (`oscm-app.ear`) and the AWS service controller (`oscm-app-aws.ear`).

- **`install`**

  XML files that support you in setting up the database and the application server resources for APP.

- **`samples`**:

  Technical service sample.

2.2.2 Configuring the AWS Integration

The AWS integration software and setup utilities require a number of settings. These settings are provided in the following subdirectories and files of `<install_pack_dir>`:

- **`databases/app_db`**
  - `db.properties`: Settings for the database setup and access.
  - `configsettings.properties`: Configuration settings for APP.

    The initial installation stores these settings in the `bssapp` database, where you can change them later, if required. An update installation only adds new settings to the database, but does not overwrite existing ones. In the case that mandatory settings are missing, an error is thrown, and you need to add these settings manually before executing the installation scripts again.

  - `configsettings_controller.properties`: Configuration settings for the AWS service controller.

    The initial installation stores these settings in the `bssapp` database. You can change them later using a graphical user interface.

    The `configsettings_controller.properties` file specifies the organization ID and user credentials for accessing OSCM as well as the AWS access keys. For security reasons, it is recommended that you delete the file as soon as you have successfully installed and configured the AWS integration software.

- **`domains/app_domain`**

  The configuration settings for setting up the application server domain to which APP is deployed are provided in the following file:

  `glassfish.properties`

Additional configuration files contained in other subdirectories are used internally and must not be changed.

For details on the configuration settings, refer to *Configuration Settings* on page 32. For details on updating the configuration settings, refer to *Updating Configuration Settings* on page 27.

You need to adapt the settings in the files above to your environment. In particular, server names, ports, paths, user IDs, and passwords require adaptation.

Proceed as follows to view and adjust the configuration settings:

1. Open each of the configuration files listed above with an editor.
2. Check the settings in each file and adapt them to your environment.
3. Save the files to their original location in `<install_pack_dir>/<subdirectory>`. For future reference, it is a good idea to create a backup of the files.

Observe the following configuration issues:

- The specified ports are suggestions and work with the default settings used in the files.
- If you install everything on the local system, use either the server name or `localhost` in all configuration files for all URLs that need to be resolved by APP.

  Do not mix the specification of server names and `localhost`.

  The `APP_BASE_URL` setting in the `configsettings.properties` file for the `app-domain` domain must be resolved by clients. They always require that the server name be specified.

  Specify the `APP_BASE_URL` setting as follows:

  ```
  APP_BASE_URL=http://<host>:<port>/oscm-app
  ```

  If you have changed the `glassfish.domain.portbase` setting in the `glassfish.properties` file, you must change the port here accordingly.

Configuration for SAML_SP authentication mode:

If the OSCM installation you want to work with is configured for SAML_SP authentication mode, Web service calls to it are secured and authenticated by a Security Token Service (STS). This is a Web service that issues security tokens as defined in the WS-Security/WS-Trust specification. The STS is usually provided by the Identity Provider (IdP) system in use (for example Active Directory Federation Service, Cloudminder, or OpenAM).

To use an STS for Web service calls, you must perform the following steps before installing the AWS integration software:

1. From the IdP or OSCM operator, obtain a metadata exchange file in WSDL format generated with and for the IdP system in use. The metadata includes namespace information required for connecting to the STS.
2. Save the metadata exchange file to the following file, overwriting the existing empty file:

   `<install_pack_dir>/domains/app_domain/wsit/STSService.xml`
3. Open the `STSService.xml` file and retrieve the value of `targetNamespace`, for example:

   `http://schemas.microsoft.com/ws/2008/06/identity/securitytokenservice`
4. Open the following file:

   `<install_pack_dir>/domains/app_domain/wsit/wsit-client.xml`
5. Replace the placeholder in the `namespace` tag of the `wsit-client.xml` file with the `targetNamespace` value copied from the `STSService.xml` file.
6. Close and save the `wsit-client.xml` file to its original location.

   During the installation process, an `OSCM-wsit.jar` file is created containing the `STSService.xml` file as well as the `wsit-client.xml` file; the `.jar` file is then copied to `<install_pack_dir>/domains/app_domain/lib`.
7. Make sure that you enter correct values for the SAML_SP authentication mode in the `configsettings.properties` file in `<install_pack_dir>/databases/app_db`:
   - `BSS_AUTH_MODE=SAML_SP`
   - `BSS_STS_WEBSERVICE_URL=https://<server>:<port>/{SERVICE}/STS`
   - `BSS_STS_WEBSERVICE_WSDL_URL= https://<server>:<port>/oscm/v1.9/{SERVICE}/STS?wsdl`
   - `APP_KEYSTORE_PASSWORD=changeit`
   - `APP_TRUSTSTORE_PASSWORD=changeit`

2.2.3 Setting up the Database

The AWS integration software requires and stores its data in the `bssapp` PostgreSQL database.

The database is created by executing installation scripts. It is initialized with the appropriate schema and settings.

Proceed as follows:

1. Make sure that the database server is running.
2. Open the command prompt (Windows) or a terminal session (UNIX/Linux).
3. Set the following environment variable for your current session:

   `DB_INTERPRETER`: The absolute path and name of the `psql` executable of PostgreSQL. The executable is usually located in the `bin` subdirectory of the PostgreSQL installation directory.

   Example (Unix/Linux):

   ```
   export DB_INTERPRETER="/opt/PostgreSQL/9.1/bin/psql"
   ```

   Example (Windows):

   ```
   set DB_INTERPRETER="C:\Program Files\PostgreSQL\9.1\bin\psql"
   ```

4. Create the database by executing the `build-db.xml` file in `<install_pack_dir>/install` as follows:

   ```
   <ANT_HOME>/bin/ant -f build-db.xml initDB
   ```

   If you set an ID or password other than `postgres` for the PostgreSQL user account (`postgres`) when installing the database management system, you have to specify the ID or password with the call to the `build-db.xml` file as follows:

   ```
   <ANT_HOME>/bin/ant -f build-db.xml initDB
      -Ddb.admin.user=<user ID> -Ddb.admin.pwd=<password>
   ```

Note: It may be required to enclose the `-Ddb.admin.user=<user ID>` and `-Ddb.admin.pwd=<password>` in double or single quotes depending on the operating system.

If the setup of the database fails with errors, proceed as follows:

1. Check and correct the configuration files.
2. Execute the `build-db.xml` file in `<install_pack_dir>/install` as follows:

   ```
   <ANT_HOME>/bin/ant -f build-db.xml DROP.dbsAndUsers
   ```

3. Repeat the setup.

2.2.4 Setting up the Application Server Resources

The AWS integration software requires specific settings and resources in the application server, such as mail settings or a data source.

Proceed as follows to create the resources and make the required settings in the application server:

1. Open the command prompt (Windows) or a terminal session (UNIX/Linux).

2. Execute the `build-glassfish.xml` file in `<install_pack_dir>/install` as follows:

```
<ANT_HOME>/bin/ant -f build-glassfish.xml SETUP
```

This has the following results:

1. **Domains and resources**:
   - The `app-domain` domain is created and started.
   - The settings and resources for APP are created in the application server.
   - APP (`oscm-app.ear`) is deployed to the `app-domain` domain.
   - A key file for encryption and decryption is generated in the location you specified in the `APP_KEY_PATH` setting in the `configsettings.properties` file. By default, this file is named `key` and located in

     `<GLASSFISH_HOME>/glassfish/domains/app-domain/config`
   - The AWS service controller (`oscm-app-aws.ear`) is deployed to the `app-domain` domain.
2. **JVM options set for the `app-domain` domain** during the installation:

```
-Dorg.apache.catalina.loader.WebappClassLoader.
   ENABLE_CLEAR_REFERENCES=false
-Dfile.encoding=UTF8
-Dsun.java2d.print.polling=false
-Dsun.net.inetaddr.ttl=3600
-XX:MaxMetaspaceSize=512m
-Dproduct.name=
```

3. Depending on your environment, it may be required to define a proxy server for the `app-domain` domain in the **JVM Options** of the application server.

   To define a proxy server, specify the following JVM options:

   - `-Dhttps.proxyHost`
   - `-Dhttps.proxyPort`

   If authentication is required, specify the following additional settings:

   - `-Dhttps.proxyUser`
   - `-Dhttps.proxyPassword`

   For all direct communication you need to bypass the proxy server. Specify the hosts which are to be addressed directly and not through the proxy server in the following setting:

   - `-Dhttp.nonProxyHosts`

   For example, APP must not use the configured proxy for Web service calls to OSCM:

   `-Dhttp.nonProxyHosts=localhost|127.0.0.1|myServer*`

   where `myServer` is the host on which OSCM is running.

   In case several service controllers are to run in the same APP domain, and only one of them is to communicate via a proxy server, you need to exclude the target systems of the other service controllers, for example, as follows:

   `-Dhttps.proxyHost=proxy.intern.myserver.com`

   `-Dhttps.proxyPort=8081`

   `-Dhttp.nonProxyHosts=myServer.com|localhost|127.0.0.1|`

   `http://10.140.18.112*|http://myServer.com:8880/templates/*`

   After having configured the proxy server, restart the `app-domain`.

If the setup of the application server domain fails with errors, proceed as follows:

1. Stop the `app-domain` domain.
2. Delete the `app-domain` domain.
3. Repeat the setup.

2.2.5 Exchanging Certificates

For secure communication of the AWS integration software with OSCM, you need to exchange the corresponding certificates.

You need to:

- Import the certificate of OSCM into the truststore of the `app-domain` application server domain of the AWS integration software.
- Export the certificate of the `app-domain` domain and import it into the `bes-domain` application server domain of OSCM.

Proceed as follows:

1. Obtain a `.crt` file with the certificate of OSCM from the OSCM operator.

   The `.crt` file can be created, for example, by executing the following command at the command prompt (Windows) or in a terminal session (UNIX/Linux) on the application server where OSCM is deployed:

   ```
   <AppServerJRE>/bin/keytool -export -rfc -alias s1as
      -file ctmgbss.crt -storepass <password> -keystore
      <GLASSFISH_HOME>/glassfish/domains/bes-domain/config/keystore.jks
   ```

2. Import the certificate of OSCM into the truststore of the `app-domain` application server domain.

   To import the OSCM certificate from the `.crt` file you created, you can use, for example, the following command at the command prompt (Windows) or in a terminal session (UNIX/Linux) on the application server:

   ```
   <AppServerJRE>/bin/keytool -import -trustcacerts -alias <alias>
      -file <filename>.crt -storepass <password> -keystore
      <GLASSFISH_HOME>/glassfish/domains/app-domain/config/cacerts.jks
   ```

3. Create a `.crt` file with the certificate of the `app-domain` domain in which you have deployed the AWS integration software.

   The `.crt` file can be created, for example, by executing the following command at the command prompt (Windows) or in a terminal session (UNIX/Linux) on the application server:

   ```
   <AppServerJRE>/bin/keytool -export -rfc -alias s1as
      -file ctmgapp.crt -storepass <password> -keystore
      <GLASSFISH_HOME>/glassfish/domains/app-domain/config/keystore.jks
   ```

4. Import the certificate of the `app-domain` domain into the `bes-domain` application server domain of OSCM.

   To do this, you can use, for example, the following command at the command prompt (Windows) or in a terminal session (UNIX/Linux) on the application server:

   ```
   <AppServerJRE>/bin/keytool -import -trustcacerts -alias ctmgapp
      -file ctmgapp.crt -storepass <password> -keystore
   ```

```
<GLASSFISH_HOME>/glassfish/domains/bes-domain/config/cacerts.jks
```

5. If the AWS integration software and OSCM are configured for SAML_SP authentication mode, obtain the relevant certificates from the IdP system and import them into the truststore of the `app-domain` domain.

   For example, when using Microsoft Active Directory as the IdP, you need to obtain and import the service communications and token-signing certificates.

6. Stop and restart the `app-domain` and the `bes-domain` domains for the certificates to become effective.

2.3 Installing the AWS Service Controller in an Existing APP Environment

If you already have a working APP installation in your environment, you can use it for the AWS integration and simply register the AWS service controller in it. Proceed as follows:

1. Check the prerequisites described in *OSCM and AWS* on page 9.
2. Deploy the AWS service controller to the `app-domain` domain. To do this, you use the GlassFish administration console, for example:

   `http://127.0.0.1:8848/`

   The `oscm-app-aws.ear` file of the service controller is located in `<install_pack_dir>/domains/app_domain/applications`

3. Register the AWS service controller as follows in APP:
   1. In a Web browser, access the base URL of APP, for example: `http://127.0.0.1:8880/oscm-app`
   2. Log in with the ID and password of the user and organization defined in the `configsettings.properties` file of APP (`BSS_USER_ID` and `BSS_USER_PWD`).
   3. Specify the controller ID (`ess.aws`) and the technology provider organization responsible for the AWS service controller.
   4. Click **Save Configuration** to save the settings.
4. Update the following configuration files so that the settings match your current installation:
   - `db.properties`
   - `configsettings.properties`
   - `configsettings_controller.properties`

   Particularly take care of settings which have been introduced or changed with the new release to which you are upgrading.

5. Update the schema and configuration settings of the `bssapp` database by executing the `build-db.xml` file in `<install_pack_dir>/install` as follows:

```
<ANT_HOME>/bin/ant -f build-db.xml updateDatabase
```

Note: Make sure that Ant runs in a Java 8 runtime environment when calling the `build-db.xml` file.

2.4 Update Installation

Before updating your installation of the AWS integration software, read the *Release Notes* of the new release. They contain information on compatibility issues, changes and enhancements, and known restrictions.

The platform operator and technology managers must make sure that the following rules are observed: The OSCM version must be higher or equal to the APP version. The APP version must be equal to the controller version.

Example: If you want to use the AWS service controller included in the V17.3 release, you must upgrade OSCM and APP to V17.3 first.

Preparing the Update

Before you start with the update installation, carry out the following steps:

1. Make sure that all provisioning operations are complete. Follow the steps as described in *Handling Problems in the Provisioning Process* on page 25.
2. If you are upgrading from V17.0:

   In the `app-domain` application server domain, disable or undeploy the following applications:

   `oscm-app`

   `oscm-app-aws`
3. If you are upgrading from V17.0:

   Check for `.glassfishStaleFiles` files in the `app-domain` domain. If there are any, delete them. The files are located in

   `app-domain/applications/<application name>/.glassfishStaleFiles`

   For example:

   `app-domain/applications/oscm-app/.glassfishStaleFiles`
4. Set the following environment variable for your current session:

   `DB_INTERPRETER`: The absolute path and name of the `psql` executable of PostgreSQL. The executable is usually located in the `bin` subdirectory of the PostgreSQL installation directory.

   Example:

   ```
   export DB_INTERPRETER="/opt/PostgreSQL/9.1/bin/psql"
   ```
5. Create a backup of the key file required for the encryption and decryption of service parameters with data type `PWD` and custom attributes marked for encryption. By default, the file is named `key` and located in the following directory:

   `<GLASSFISH_HOME>/glassfish/domains/app-domain/config`

Updating the Database

Proceed with updating the database as follows:

1. Check whether the file

   `postgresql-9.4-1206-jdbc42.jar`

   is contained in the following directories of the application server:

   - `lib` directory of the `app-domain` domain
   - `<GLASSFISH_HOME>/mq/lib/ext`

If it is not, copy the file from the `<install_pack_dir>/install/lib` directory to the location where it is missing.

2. Create a backup of the database using the standard PostgreSQL commands. The database backup must be compatible with PostgreSQL 9.1.12.
3. Update the following configuration files so that the settings match your current installation:
   - `db.properties`
   - `configsettings.properties`
   - `configsettings_controller.properties`

   Particularly take care of settings which have been introduced or changed with the new release to which you are upgrading.
4. Update the schema and configuration settings of the `bssapp` database by executing the `build-db.xml` file in `<install_pack_dir>/install` as follows:

```
<ANT_HOME>/bin/ant -f build-db.xml updateDatabase
```

Note: Make sure that Ant runs in a Java 8 runtime environment when calling the `build-db.xml` file.

Updating the Application Server

After you have executed the preparation steps:

1. If you are upgrading from a release prior to V17.0:

   Copy the key file of which you have created a backup copy into the following directory of the extracted installation package:

   `oscm-app-install-pack\domains\app_domain\installer`

   Make sure that the key file is named `key`. The installation script will then copy the key file to its default location in the updated version of the application server:

   `<GLASSFISH_HOME>/glassfish/domains/app-domain/config`

Note: For the update installation, make sure that the default path and name are specified in the `configsettings.properties` file (`APP_KEY_PATH="./key"`). The installation script expects this default and does not look up the database.

2. In the **JVM Options** of the application server:
   a. Set the following setting to `false`:

      `-Dorg.apache.catalina.loader.WebappClassLoader.ENABLE_CLEAR_REFERENCES=false`

   b. Delete the following setting:

      `-XX:MaxPermSize`

   c. Add the following setting:

      `XX:MaxMetaspaceSize=512m`
3. Redeploy or deploy the applications that make up the AWS integration software in the `app-domain` domain:
   a. `oscm-app`
   b. `oscm-app-aws`

4. If you are upgrading from a release prior to V17.0:
   - Import the certificates of OSCM and AWS into the truststore of the `app-domain` application server domain of the AWS integration software.
   - Export the certificate of the `app-domain` and import it into the `bes-domain` application server domain of OSCM.
5. Restart the `app-domain` domain.

Updating the Configuration for SAML_SP Mode

If you are running OSCM and the AWS integration software in SAML_SP mode, and if the IdP metadata of your SSO environment have changed, you need to update your WSIT files accordingly:

1. Extract the `OSCM-wsit.jar` file into a separate directory.

   The `.jar` file is located in

   `<GLASSFISH_HOME>/domains/app_domain/lib`

   The `OSCM-wsit.jar` file contains the following files:

   `wsit-client.xml`

   `STSService.xml`
2. Adapt the `.xml` files as required by your environment. For details, refer to *Configuring the AWS Integration* on page 12.
3. Recreate the `OSCM-wsit.jar` file with the modified contents.
4. Stop the `app-domain` domain.
5. Copy the adapted `OSCM-wsit.jar` to the following directory:

   `<GLASSFISH_HOME>/domains/app_domain/lib`
6. Restart the `app-domain` domain.

Note: If, for some reason, you recreate the `app-domain` domain, you also need to recreate the `OSCM-wsit.jar` in the `<GLASSFISH_HOME>/domains/app_domain/lib` directory.

3 Creating and Publishing Services

The following sections describe how to create and publish services in OSCM by means of which customers can request and use virtual servers in AWS.

3.1 Prerequisites and Preparation

The following prerequisites must be fulfilled before you can create and publish services in OSCM:

- To create technical services for the AWS integration in OSCM, you must have access to OSCM as a technology manager. You must be a member of the technology provider organization responsible for the AWS service controller as specified in the configuration settings for the installation.
- In AWS, appropriate AMIs for virtual servers must exist, to which the technical services in OSCM can be mapped. The AWS user specified in the configuration settings for the installation must have the necessary credentials to create and configure virtual servers based on these AMIs.
- To create marketable services for the AWS integration in OSCM, you must have access to OSCM as a service manager of an organization with the supplier role. This may be the same organization as the technology provider organization or a different one.
- To publish your marketable services, you must have access to an appropriate marketplace in OSCM in your service manager role.

3.2 Creating Technical Services

The first step in providing OSCM services for AWS is to create one or more technical services.

Proceed as follows:

1. Define one or more technical services in an XML file.

   The AWS integration package, `oscm-aws-install-pack.zip`, includes a technical service as a sample. Use the sample as a basis for defining your own technical services as required:

   `samples/TechnicalService_AWS.xml`

   In the technical service definition, be sure to specify:

   - The asynchronous provisioning type
   - The direct access type
   - Service parameters which represent the AMIs defined in Amazon EC2. For details on the supported service parameters, refer to *Service Parameters and Operations* on page 37.

Note: Make sure that you do not specify the `baseUrl` attribute in the technical service definition XML file. It specifies an application's remote interface and is not needed for providing OSCM services for AWS.

2. Log in to the OSCM administration portal with your technology manager account.
3. Import the technical services you created and appoint one or more supplier organizations for them.

For details on these steps, refer to the *Technology Provider's Guide* and to the online help of OSCM.

3.3 Creating and Publishing Marketable Services

As soon as the technical services for the AWS integration exist in OSCM, you can define and publish marketable services based on them. Your cost calculation for the services should include any external costs for operating the virtual servers in Amazon EC2.

Proceed as follows:

1. Log in to the OSCM administration portal with your service manager account.
2. Define one or more marketable services based on the technical services you created for AWS.
3. Define price models for your marketable services.
4. Publish the services to a marketplace.

For details on these steps, refer to the *Supplier's Guide* and to the online help of OSCM.

4 Using AWS Services in OSCM

The following sections describe how users can subscribe to and work with the services you have created for AWS in OSCM. You will find details of the supported usage scenarios outlined in *Usage Scenarios* on page 8.

4.1 Subscribing to Services

Users of customer organizations can subscribe to the services you have created for AWS on the marketplace where you have published them. This results in the provisioning of a virtual server in Amazon EC2, as defined in the underlying technical service.

To enable the provisioning of a virtual server, the customer has to enter the name of the key pair of the virtual server when subscribing to the corresponding service in OSCM. The key pair name and the associated private key are used to securely access the Amazon EC2 instance. For details on creating key pairs, refer to the user documentation of Amazon Web Services.

In addition, the customer has to enter a name for the virtual server when subscribing to the corresponding service. The technical service may specify a prefix which is preprended to this name, as well as a pattern against which the name is checked before the provisioning operation is started.

Depending on the parameters defined for the technical service, the customer can choose from different options to configure the virtual server to be provisioned.

The provisioning operations are carried out in asynchronous mode. As long as the provisioning is not complete, the status of the subscription is **pending**. The status changes to **ready** as soon as the provisioning has been finished successfully.

As soon as the provisioning is complete, the users assigned to the subscription can access the virtual server provided by AWS using the IP address indicated in the subscription details on the marketplace in OSCM. The users can access the virtual server according to the connection processes specified by Amazon. For details, refer to the user documentation of Amazon Web Services.

4.2 Executing User-Specific Configuration Data

When an instance is provisioned in Amazon EC2, a customer has the option of passing user-specific configuration data to the instance. The data can be used to perform automated configuration tasks or run scripts. Customers can pass two types of user data to Amazon EC2 instances: shell scripts and `cloud-init` directives. They can also pass this data as plain text, as a file, or as base64-encoded text for API calls.

To access user data scripts or `cloud-init` directives, the technology provider must enter the URL pointing to the scripts or directives into the definition of the technical service. The URL must be accessible for APP.

For details on executing user data, refer to the user documentation of Amazon Web Services.

4.3 Executing Service Operations

Customers can explicitly start and stop a virtual server in AWS from OSCM. To do this, they execute the appropriate service operation from the subscription for the virtual server:

- **Start**: Starts the virtual server if it was stopped.
- **Stop**: Stops the virtual server if it was started.

As a prerequisite, the service operations must be defined in the technical service underlying the subscribed service.

4.4 Terminating Subscriptions

A customer can at any time terminate a subscription for a virtual machine in AWS.

AWS is triggered to delete the virtual server. The subscription is terminated in OSCM independent of whether the deletion is successful in AWS. Note, however, that the subscription name cannot be re-used before the deletion has been completed in AWS.

5 Administrating the AWS Integration

The following sections describe administration tasks you may need to perform in your role as an operator of the AWS integration software:

5.1 Controlling the Provisioning Process

The AWS integration provides you with the following feature for controlling the provisioning and deprovisioning of virtual servers:

In the definition of the technical services for AWS, you can specify the `MAIL_FOR_COMPLETION` parameter. This is an address to which emails are to be sent describing manual steps required to complete an operation.

If you specify this parameter, the AWS service controller interrupts the processing of each operation before its completion and waits for a notification about the execution of a manual action. This notification consists in opening the link given in the email.

Omit the `MAIL_FOR_COMPLETION` parameter if you do not want to interrupt the processing.

5.2 Handling Problems in the Provisioning Process

If the provisioning of a virtual server fails on the AWS side or if there are problems in the communication between the participating systems, the corresponding subscription in OSCM remains pending. The AWS service controller informs the technology managers of its responsible technology provider organization by email of any incomplete provisioning or delete operation in AWS.

You can then take the appropriate actions to solve the problem in AWS or in the communication. For example, you could remove an incomplete virtual server, or you could restore a missing connection.

After solving the problem, the AWS integration components and OSCM need to be synchronized accordingly. You do this by triggering a corresponding action in the APP component. Proceed as follows:

1. Work as a technology manager of the technology provider organization responsible for the AWS service controller.
2. Invoke the instance status interface of APP for the AWS service controller by opening the following URL in a Web browser:

   `https://<server>:<port>/oscm-app/controller/?controllerid=ess.aws`

   For example:

   `https://127.0.0.1:8881/oscm-app/controller/?controllerid=ess.aws`

   The Web page shows all subscriptions for AWS, including detailed information such as the customer organization, the ID of the related AWS instance, and the provisioning status.

3. Find the subscription for which you solved the problem in the most recent provisioning or delete operation.
4. In the **Action** column, select the action for the AWS integration components to execute next. Possible actions are the following:
   - `RESUME` - to resume the processing of a provisioning operation in APP which was suspended.
   - `SUSPEND` - to suspend the processing of a provisioning operation in APP, for example when AWS does not respond.
   - `UNLOCK` - to remove the lock for an AWS instance in APP.
   - `DELETE` - to terminate the subscription in OSCM and remove the instance in APP, but keep the virtual server in AWS for later use. The service manager role is required for this action.
   - `DEPROVISION` - to terminate the subscription in OSCM, remove the instance in APP, and delete the virtual server in AWS. The service manager role is required for this action.
   - `ABORT_PENDING` - to abort a pending provisioning operation in OSCM. OSCM is notified to roll back the changes made for the subscription and return it to its previous state. In AWS, no actions are carried out.
   - `COMPLETE_PENDING` - to complete a pending provisioning operation in OSCM. OSCM is notified to complete the changes for the subscription and set the subscription status to **ready** (or **suspended** if it was suspended before). This is possible only if the operations of the service controller are already completed.
5. Click **Execute** to invoke the selected action.

The instance status interface provides the following additional functionality that is useful for problem-solving purposes:

- You can display service instance details for each subscription by clicking the corresponding entry in the table. This displays all subscription-related information that is stored in the `bssapp` database.
- The **Run with timer** column indicates whether the timer for the interval at which APP polls the status of instances is running. You can reset the timer, if required. For details on the timer setting, refer to *Configuration Settings* on page 32.

5.3 Handling Communication Problems Between APP and OSCM

When the communication between APP and OSCM is no longer possible, for example, because OSCM is stopped, APP suspends the processing of requests. An internal flag is set in the APP database: `APP_SUSPEND=true`, and an email is sent to the address specified in the `APP_ADMIN_MAIL_ADDRESS` configuration setting.

Contact the OSCM operator to make sure that OSCM is up and running again correctly.

You then have the following possibilities to resume the processing of requests by APP:

1. Click the link provided in the email.
2. Log in to APP.

   APP is restarted instantly. In the APP database, the `APP_SUSPEND` key is set to `false`.

As an alternative, you can proceed as follows:

1. In a Web browser, access the base URL of APP, for example:

   `http://127.0.0.1:8880/oscm-app`

2. Log in with the ID and password of the user and organization defined in the `configsettings.properties` file of APP (`BSS_USER_ID` and `BSS_USER_PWD`).

   A message is shown that APP has been suspended due to a communication problem with OSCM.

3. Click **Restart**.

   APP is restarted instantly. In the APP database, the `APP_SUSPEND` key is set to `false`.

5.4 Backup and Recovery

The AWS integration software does not offer integrated backup and recovery mechanisms. Use the standard file system, application server, and database mechanisms instead.

Backup

It is recommended that you create a regular backup of the following data according to the general guidelines of your organization:

- Database (`bssapp`). The frequency of database backups depends on the amount of changes and the availability of time slots with low load. PostgreSQL supports database backups without previous shutdown. For details, refer to the PostgreSQL documentation.

  Make sure to also make a backup of the file containing the key required for encryption and decryption of service parameters with data type `PWD` and custom attributes marked for encryption. By default, this file is named `key` and located in:

  `<GLASSFISH_HOME>/glassfish/domains/app-domain/config`

  The location of this file can be changed using the `APP_KEY_PATH` configuration setting.

- Certificates contained in the truststore of the `app-domain` domain (`cacerts.jks` file).
- Configuration files.

Note:	When preparing for an update installation of the current AWS integration software, always create a backup of the data mentioned above.

Recovery

If you need to recover your AWS integration installation, the recommended procedure is as follows:

1. Restore the `bssapp` database from the backup using the relevant PostgreSQL commands.
2. Make sure that the file containing the key required for encryption and decryption of service parameters with data type `PWD` and custom attributes marked for encryption exists in the location specified in the `APP_KEY_PATH` configuration setting. By default, this file is to be named `key` and located in:

   `<GLASSFISH_HOME>/glassfish/domains/app-domain/config`

   If the file is missing, copy it from your backup to the correct location.

3. Stop the `app-domain` domain of the application server.
4. Restore the certificate truststore of the `app-domain` domain (`cacerts.jks` file) from the backup.
5. Start the `app-domain` domain.

5.5 Updating Configuration Settings

The AWS integration software and setup utilities require a number of settings. In the installation, you adapted the settings, in particular server names, ports, paths, and user IDs, to your environment.

The configuration settings are provided in the following subdirectories and files of `<install_pack_dir>`:

- **`databases/app_db`**
  - `db.properties`: Settings for the database setup and access.
  - `configsettings.properties`: Configuration settings for APP.

    The initial installation stores these settings in the `bssapp` database, where you can change them later, if required. An update installation overwrites the settings. If you don't want existing settings to be overwritten, delete them from the properties file. In case that mandatory settings are missing in the properties file and not yet stored in the database, an exception will occur.
  - `configsettings_controller.properties`: Configuration settings for the AWS service controller.

    The initial installation stores these settings in the `bssapp` database. You can change them later using a graphical user interface.
- **`domains/app_domain`**

  The configuration settings for setting up the application server domain to which APP is deployed are provided in the following file:

  `glassfish.properties`

For details on the configuration settings, refer to *Configuration Settings* on page 32.

If you need to change the settings, proceed as described in the following sections.

To update the configuration settings for database access:

1. Log in to the administration console of the application server.
2. Adapt the settings as required.

To update the configuration settings for the application server:

1. Open the `glassfish.properties` file located in `<install_pack_dir>/domains/app_domain` with an editor.
2. Check the settings in the file and adapt them to your environment if required.
3. Save the file to its original location in `<install_pack_dir>/domains/app_domain`.
4. Update the settings and resources in the application server by executing the `build-glassfish.xml` file in `<install_pack_dir>/install` as follows:

```
<ANT_HOME>/bin/ant -f build-glassfish.xml
   SETUP.configureDomains
```

To update the configuration settings for APP, you have the following options:

- Run the installation script:
  1. Edit the content of the `configsettings.properties` file as required.
  2. Execute the `build-db.xml` file in `<install_pack_dir>/install` as follows:

```
<ANT_HOME>/bin/ant -f build-db.xml
```

```
UPDATE.configSettings
```

- Use the Web interface of APP:
  1. In a Web browser, access the base URL of APP, for example:
     `http://127.0.0.1:8880/oscm-app`
  2. Log in with the ID and password of the user you specified for `BSS_USER_KEY` in the APP *configuration settings* or as another administrator of the same organization.
  3. Edit the settings as required.
  4. Click **Save Configuration** to save the settings.

To update the configuration settings for the AWS service controller:

1. In a Web browser, access the URL of the AWS service controller, for example: `http://127.0.0.1:8880/oscm-app-aws`.
2. Log in with the ID and password of the user specified in the configuration settings for the AWS service controller (`BSS_USER_ID` and `BSS_USER_PWD`) or as another technology manager registered for the same organization.
3. Enter the required settings.
4. Save the settings.

If you want to **change the technology provider** organization responsible for the AWS service controller, you can use the Web interface of APP:

1. In a Web browser, access the base URL of APP, for example: `http://127.0.0.1:8880/oscm-app`
2. Log in with the ID and password of the user specified for `BSS_USER_KEY` in the configuration settings for APP or as another administrator of the same organization.
3. Specify the technology provider organization for the AWS service controller, `ess.aws`.
4. Save the settings.
5. Make sure that the configuration settings for the AWS service controller are updated.

   Any technology manager registered for the technology provider organization you specified can log in to the graphical user interface for updating the controller configuration settings (see above). At least the ID and password of the user to be used for accessing OSCM must be changed in the controller configuration settings.

5.6 Adapting the Log Configuration

The AWS integration software records information and problems such as connection issues in the following log files on the application server:

- `<GLASSFISH_HOME>/domains/<DOMAIN_NAME>/logs/app-aws.log`: Log of the AWS service controller
- `<GLASSFISH_HOME>/domains/<DOMAIN_NAME>/logs/app-core.log`: Log of the APP component

The logging is based on `log4j`. The default log level is `INFO`, which may not be sufficient depending on the circumstances. In such a case, you will need to adapt the log level in the configuration files. The following configuration files are of relevance:

- `<GLASSFISH_HOME>/domains/<DOMAIN_NAME>/config/log4j.ess.aws.properties`: Log configuration of the AWS service controller

- `<GLASSFISH_HOME>/domains/<DOMAIN_NAME>/config/log4j.app.core.properties`: Log configuration of the APP component

Proceed as follows to adapt the log level:

1. Open the relevant configuration file.
2. Find the string `log4j.logger.org.oscm.app` in the configuration file.
3. Change the log level as desired to one of the following:
   - `ERROR` - designates error events that might still allow the application to continue running.
   - `WARN` - designates potentially harmful situations.
   - `INFO` - designates informational messages that highlight the progress of the application at coarse-grained level.
   - `DEBUG` - designates fine-grained informational events that are most useful to debug an application.

   Example:

```
log4j.logger.org.oscm.app=INFO
```

Every 60 seconds, the AWS integration software checks for changes in the log configuration. There is no need to restart the application.

6 Uninstallation

If you want to uninstall the AWS integration software, take the following preparations:

- Back up resources and data you would like to keep. For details, refer to *Backup and Recovery* on page 27.
- In OSCM, delete the marketable services and technical services related to AWS.

To uninstall the AWS integration software:

1. Stop the `app-domain` domain in the application server.
2. Delete the `app-domain` domain.
3. Delete the `bssapp` database in the database management system.
4. Uninstall the database management system and the application server if you no longer need them for other purposes.

For details on how to proceed, refer to the documentation of the database management system and the application server.

Appendix A: Configuration Settings

The configuration settings for the AWS integration software are provided in the following files in subfolders of the directory to which you extracted the `oscm-aws-install-pack.zip` file (`<install_pack_dir>`):

- `domains/app_domain/glassfish.properties`
- `databases/app_db/db.properties`
- `databases/app_db/configsettings.properties`
- `databases/app_db/configsettings_controller.properties`

This appendix describes the settings in detail.

A.1 Database Configuration Settings

The `db.properties` file located in `<install_pack_dir>/databases/app_db` contains the configuration settings for database access. This configuration is used for the initial setup and schema updates.

db.driver.class

The Java class of the JDBC driver.

Default: `org.postgresql.Driver`

db.host

The database host.

Default: `localhost`

db.port

The database port.

Default: `5432`

db.name

The name of the database.

Default: `bssapp`

db.user

The name of the user to connect to the database.

Default: `bssuser`

db.pwd

The password of the user to connect to the database.

Default: `bssuser`

db.type

The type of the database.

Default: `postgresql`

A.2 APP Configuration Settings

The `configsettings.properties` file located in `<install_pack_dir>/databases/app_db` contains the configuration settings for APP.

APP_BASE_URL

`APP_BASE_URL=http://<server>:<port>/oscm-app`

The URL used to access APP.

APP_TIMER_INTERVAL

`APP_TIMER_INTERVAL=15000`

The interval (in milliseconds) at which APP polls the status of instances. If you increase the value, provisioning takes longer. If you decrease it, more load is put on the system. We strongly recommend that you do not set a value of more than 180000 milliseconds (3 minutes), although the maximum value is much higher (922337203685477580).

If you do not specify this setting at all, the default value used is 15000.

APP_TIMER_REFRESH_SUBSCRIPTIONS

`APP_TIMER_REFRESH_SUBSCRIPTIONS=86400000`

The interval (in milliseconds) at which APP polls the status of instances and updates the number of virtual machines (VMs) provisioned for subscriptions to IaaS services, for example in OpenStack. The number is updated only in case the technical service definition specifies a `VMS_NUMBER` parameter.

If you do not specify this setting, the default value used is 86400000 (once a day).

APP_MAIL_RESOURCE

`APP_MAIL_RESOURCE=mail/APPMail`

The JNDI name of the GlassFish mail resource used to send emails.

The resource `mail/APPMail` is created during setup with the parameters defined in the `glassfish.properties` file. This setting needs to be changed only if you want to use a different mail resource.

APP_ADMIN_MAIL_ADDRESS

`APP_ADMIN_MAIL_ADDRESS=admin@example.com`

The email address to which email notifications are sent.

APP_KEY_PATH

The path to the file containing the key required for encryption and decryption of service parameters with data type `PWD` and custom attributes marked for encryption.

Upon the start of APP, it is checked whether a key file exists in the location specified in this setting. If this not the case, the key file is automatically generated and stored in the location specified here. If nothing is specified, the file will be named `key` and generated into the following location:

`<GLASSFISH_HOME>/glassfish/domains/app-domain/config`

Default:

`./key`

Note:	Be aware that the key file must not be deleted. Otherwise, encryption and decryption is no longer possible. It is recommended to create a backup of this file once generated.

APP_KEYSTORE_PASSWORD

`APP_KEYSTORE_PASSWORD=changeit`

The password required to access the keystore of the domain used for APP in the application server.

APP_TRUSTSTORE

`APP_TRUSTSTORE=./cacerts.jks`

The path and file name of the truststore of the application server domain used for APP. The certificate of OSCM is stored in this truststore.

APP_TRUSTSTORE_BSS_ALIAS

`APP_TRUSTSTORE_PASSWORD=bes-s1as`

The alias of the certificate of OSCM as stored in the truststore of the application server domain used for APP.

APP_TRUSTSTORE_PASSWORD

`APP_TRUSTSTORE_PASSWORD=changeit`

The password required to access the truststore of the application server domain used for APP.

BSS_AUTH_MODE

`BSS_AUTH_MODE=INTERNAL`

Specifies that OSCM is used for user authentication.

Possible value: `INTERNAL`

BSS_USER_KEY

`BSS_USER_KEY=<userKey>`

The user key for accessing OSCM.

Replace `<userKey>` with the user key which you receive with the confirmation email for your user account.

The user specified here must have the administrator role for your organization in OSCM. The user account is used for carrying out actions on behalf of APP in OSCM. In addition, the user is allowed to register service controllers in APP.

BSS_USER_ID

`BSS_USER_ID=<userId>`

The identifier of the user specified in `BSS_USER_KEY` for accessing OSCM.

Replace `<userId>` with the user ID.

BSS_USER_PWD

`BSS_USER_PWD=_crypt:<password>`

The password of the user specified in `BSS_USER_KEY` for accessing OSCM.

Replace `<password>` with the plain text password. The password is encrypted when it is stored in the database.

BSS_WEBSERVICE_URL

```
BSS_WEBSERVICE_URL=https://<server>:<port>/{SERVICE}/BASIC
```

Mandatory when `BSS_AUTH_MODE` is set to `INTERNAL` and basic authentication is used. The endpoint of the OSCM Web services to be used. The `{SERVICE}` placeholder must not be replaced.

BSS_WEBSERVICE_WSDL_URL

```
BSS_WEBSERVICE_WSDL_URL=https://<server>:<port>/oscm/v1.9/{SERVICE}/BASIC?wsdl
```

Mandatory when `BSS_AUTH_MODE` is set to `INTERNAL` and basic authentication is used. The URL specifying the version of the OSCM Web services to be used. The `{SERVICE}` placeholder must not be replaced.

A.3 Controller Configuration Settings

The `configsettings_controller.properties` file located in `<install_pack_dir>/databases/app_db` contains the configuration settings for the AWS service controller. This configuration is used for the initial setup and stored in the APP database.

A technology provider can define service parameters in the technical service definition. If such a parameter has the same ID as a controller configuration setting stored in the APP database, it overrules the configuration setting in the database when the marketable service based on such a technical service is subscribed to. By default the values in the controller configuration settings are used. Refer to the *Technology Provider's Guide* for details on defining technical services.

In addition, a supplier can define custom attributes for subscriptions and for customers. If such an attribute has the same ID as a controller configuration setting stored in the APP database as well as a corresponding technical service parameter, it overrules the technical service parameter as well as the configuration setting in the database when the marketable service based on such a technical service is subscribed to.

The controller configuration settings are evaluated as follows:

1. Configuration setting as stored in the APP database.
2. Technical service parameter. If defined, it overrules 1.
3. Custom attribute for customer. If defined, it overrules 1. and 2.
4. Custom attribute for subscription. If defined, it overrules 1. and 2. and 3.

CONTROLLER_ID

```
CONTROLLER_ID=ess.aws
```

The identifier of the service controller.

BSS_ORGANIZATION_ID

```
BSS_ORGANIZATION_ID=<organizationID>
```

The ID of the organization in OSCM which is responsible for the service controller. The organization must have the technology provider role.

BSS_USER_ID

`BSS_USER_ID=<userId>`

The identifier of the user specified in `BSS_USER_KEY` for accessing OSCM.

Replace `<userId>` with the user ID.

BSS_USER_KEY

`BSS_USER_KEY=<userKey>`

The user key for accessing OSCM.

Replace `<userKey>` with the user key which you received with the confirmation email for your user account.

The user specified here must have the technology manager role in OSCM and belong to the organization specified in the `BSS_ORGANIZATION_ID` setting.

It is recommended that the user account is used only for carrying out actions on behalf of the service controller in OSCM.

BSS_USER_PWD

`BSS_USER_PWD=_crypt:<password>`

The password of the user specified in `BSS_USER_KEY` for accessing OSCM.

Replace `<password>` with the plain text password. The password is encrypted when it is stored in the database.

ACCESS_KEY_ID_PWD

`ACCESS_KEY_ID_PWD=_crypt:<accessKeyID>`

The identifier of the access key for the AWS account.

A technology provider who is responsible for creating technical services for appropriate AMIs needs to have an AWS account to create Amazon EC2 instances. For details about creating an AWS account, refer to the user documentation of Amazon Web Services.

Together with the secret access key, the access key ID is used to authenticate API calls to Amazon EC2.

SECRET_KEY_PWD

`SECRET_KEY_PWD=_crypt:<secretAccessKey>`

The secret access key for the AWS account.

Together with the access key ID, the secret access key is used to authenticate API calls to Amazon EC2.

Appendix B: Service Parameters and Operations

The following sections describe the technical service parameters and service operations which are supported by the AWS service controller.

Service Parameters

The AWS service controller supports the parameters below.

Note: All parameters defined in the technical service definition must be one-time parameters, since the modification of parameters is not supported. Be sure to set their `modificationType` to `ONE_TIME`.

APP_CONTROLLER_ID

Mandatory. The ID of the service controller as defined in its implementation. The ID is set during the installation of the AWS integration software.

Default (must not be changed): `ess.aws`

DISK_SIZE

Optional. The maximum disk size of the virtual server to be instantiated. The value is a number specifying the size in GB. If not specified, the default size for the selected type of server is used.

Users should not be able to enter a value for this parameter. This means the parameter should not be configurable for customers or, in case you specify more than one disk size, have fixed parameter options for selection.

Example: `3`

IMAGE_NAME

Mandatory. Name of the AMI which is the basis for virtual servers.

Users should not be able to enter a value for this parameter. This means the parameter should not be configurable for customers or, in case you specify more than one image, have fixed parameter options for selection.

Example: `ami-0d77397e`

INSTANCENAME

Mandatory. The name of the virtual server to be instantiated. This name must be specified by customers when they subscribe to a corresponding service. The string given in `INSTANCENAME_PREFIX` is prepended to the name. The name including the prefix must match the pattern given in `INSTANCENAME_PATTERN`.

As the instance name is stored as a tag on the Amazon EC2 instance, the maximum length is 255 Unicode characters.

Example: `MyServer`

INSTANCENAME_PATTERN

Mandatory. A regular expression specifying a pattern for the virtual server instance names entered by the users when they subscribe to a corresponding service. If the names do not match the pattern, the subscription is rejected.

The regular expression must be specified in the technical service definition.

Example: `.*{1,255}`

INSTANCENAME_PREFIX

Optional. A string to be prepended to the virtual server instance names entered by the users when they subscribe to a corresponding service.

The string must be specified in the technical service definition.

Example: `aws`

INSTANCE_TYPE

Mandatory. The type of the virtual server to be instantiated. Any valid Amazon EC2 instance type can be specified. In the sample technical service, the following types are defined:

- **t1.micro**
- **t1.small**
- **t1.medium**

Users should not be able to enter a value for this parameter. This means the parameter should not be configurable for customers or, in case you specify more than one type, have fixed parameter options for selection.

Example: `t1.micro`

KEY_PAIR_NAME

Mandatory. The key pair name of the virtual server to be instantiated.

The key pair name must be specified by the customer when subscribing to an AWS service. To log in to the instance, the customer must enter the key pair name and the associated private key.

For details on creating key pairs, refer to the user documentation of Amazon Web Services.

Example: `my-key-pair`

MAIL_FOR_COMPLETION

Optional. The address to which emails are to be sent that describe manual steps required to complete an operation. If you specify this parameter, the service controller interrupts the processing of each operation before its completion and waits for a notification about the execution of a manual action. Omit this parameter if you do not want to interrupt the processing.

Example: `info@company.com`

PUBLIC_IP

Optional. Specifies whether the virtual server to be instantiated is to be assigned a public IP address. The value can be `true` or `false`. If not specified, the default is `false`.

Example: `true`

REGION

Mandatory. The region where the data center hosting the virtual servers is located. Any valid region can be specified. In the sample technical service, the following regions are defined:

- **us-west-1** (US West (Northern California) Region)
- **us-west-2** (US West (Oregon) Region)
- **us-east-1** (US East (Northern Virginia) Region)

Users should not be able to enter a value for this parameter. This means the parameter should not be configurable for customers or, in case you specify more than one region, have fixed parameter options for selection.

Example: `us-east-1`

SECURITY_GROUP_NAMES

Optional. Comma-separated list of security group names for the virtual server to be instantiated.

A security group acts as a firewall that controls the traffic to an Amazon EC2 instance. An Amazon EC2 instance can be assigned to one or more security groups. For details on security groups, refer to the user documentation of Amazon Web Services.

Users should not be able to enter a value for this parameter. This means the parameter should not be configurable for customers or, in case you specify more than one security group, have fixed parameter options for selection.

Be aware that for the specification of security groups, the specification of the `SUBNET` parameter is mandatory. Otherwise the groups will be ignored.

Example: `MySecurityGroup`

SUBNET

Optional. The subnet to which the virtual server is to be assigned.

Users should not be able to enter a value for this parameter. This means the parameter should not be configurable for customers or, in case you specify more than one subnet, have fixed parameter options for selection.

If you specify security groups, you also need to specify the corresponding subnet. If no subnet parameter is specified, the AWS service controller ignores any specified security groups, and the service instance is created in a default subnet and a default security group is assigned.

Example: `subnet-a77430d0`

USERDATA_URL

Optional. URL to access the user data scripts or `cloud-init` directives for the automatic execution of user-specific configuration data. This URL must be accessible for APP.

Users should not be able to enter a value for this parameter. This means the parameter should not be configurable for customers or, in case you specify more than one URL, have fixed parameter options for selection.

Example: `http://127.0.0.1:8880/cloud-init/LAMP.script` (if the file was created under `<appdomain>/docroot/cloud-init/LAMP.script`)

Service Operations for Virtual Servers

The AWS service controller supports the service operations below for virtual servers.

The `actionURL` for each operation is:

`http://oscm-app:8880/oscm-app/webservices/oscm-app/oscm-app/org.oscm.app.v2_0.service.Asynch`

`<oscm-app>` and `<8880>` are the server and port of the container where the AWS integration software is deployed.

Note: If you provision a virtual server that does not support start and stop operations, make sure that you remove the service operations from the technical service definition.

START_VIRTUAL_SYSTEM

Starts a virtual server in AWS if it was stopped.

STOP_VIRTUAL_SYSTEM

Stops a virtual server in AWS if it was started.

Glossary

Administrator

A privileged user role within an organization with the permission to manage the organization's account and subscriptions as well as its users and their roles. Each organization has at least one administrator.

Application

A software, including procedures and documentation, which performs productive tasks for users.

Billing System

A system responsible for calculating the charges for using a service.

Broker

An organization which supports suppliers in establishing relationships to customers by offering the suppliers' services on a marketplace, as well as a privileged user role within such an organization.

Cloud

A metaphor for the Internet and an abstraction of the underlying infrastructure it conceals.

Cloud Computing

The provisioning of dynamically scalable and often virtualized resources as a service over the Internet on a utility basis.

Customer

An organization which subscribes to one or more marketable services in OSCM in order to use the underlying applications in the Cloud.

Infrastructure as a Service (IaaS)

The delivery of computer infrastructure (typically a platform virtualization environment) as a service.

Marketable Service

A service offering to customers in OSCM, based on a technical service. A marketable service defines prices, conditions, and restrictions for using the underlying application.

Marketplace

A virtual platform for suppliers, brokers, and resellers in OSCM to provide their services to customers.

Marketplace Owner

An organization which holds a marketplace in OSCM, where one or more suppliers, brokers, or resellers can offer their marketable services.

Marketplace Manager

A privileged user role within a marketplace owner organization.

Operator

An organization or person responsible for maintaining and operating OSCM.

Organization

An organization typically represents a company, but it may also stand for a department of a company or a single person. An organization has a unique account and ID, and is assigned one or more of the following roles: technology provider, supplier, customer, broker, reseller, marketplace owner, operator.

Organizational Unit

A set of one or more users within an organization representing, for example, a department in a company, an individual project, a cost center, or a single person. A user may be assigned to one or more organizational units.

OU Administrator

A privileged user role within an organization allowing a user to manage the organizational units for which he has been appointed as an administrator, and to create, modify, and terminate subscriptions for these units.

Payment Type

A specification of how a customer may pay for the usage of his subscriptions. The operator defines the payment types available in OSCM; the supplier or reseller determines which payment types are offered to his customers, for example payment on receipt of invoice, direct debit, or credit card.

Platform as a Service (PaaS)

The delivery of a computing platform and solution stack as a service.

Price Model

A specification for a marketable service defining whether and how much customers subscribing to the service will be charged for the subscription as such, each user assigned to the subscription, specific events, or parameters and their options.

Reseller

An organization which offers services defined by suppliers to customers applying its own terms and conditions, as well as a privileged user role within such an organization.

Role

A collection of authorities that control which actions can be carried out by an organization or user to whom the role is assigned.

Seller

Collective term for supplier, broker, and reseller organizations.

Service

Generally, a discretely defined set of contiguous or autonomous business or technical functionality, for example an infrastructure or Web service. OSCM distinguishes between technical services and marketable services, and uses the term "service" as a synonym for "marketable service".

Service Manager

A privileged user role within a supplier organization.

Standard User

A non-privileged user role within an organization.

Software as a Service (SaaS)

A model of software deployment where a provider licenses an application to customers for use as a service on demand.

Subscription

An agreement registered by a customer for a marketable service in OSCM. By subscribing to a service, the customer is given access to the underlying application under the conditions defined in the marketable service.

Subscription Manager

A privileged user role within an organization with the permission to create and manage his own subscriptions.

Supplier

An organization which defines marketable services in OSCM for offering applications provisioned by technology providers to customers.

Technical Service

The representation of an application in OSCM. A technical service describes parameters and interfaces of the underlying application and is the basis for one or more marketable services.

Technology Manager

A privileged user role within a technology provider organization.

Technology Provider

An organization which provisions applications as technical services in OSCM.