V19.1

Open Service
Catalog Manager

QuickStart Guide

February 2021

Trademarks

LINUX is a registered trademark of Linus Torvalds.

Microsoft, Active Directory, Azure, and Excel are either registered trademarks or trademarks of Microsoft Corporation in the United States and/or other countries.

Open Service Catalog Manager is a registered trademark of FUJITSU LIMITED.

The OpenStack Word Mark and OpenStack logo are registered trademarks/service marks or trademarks/ service marks of the OpenStack Foundation in the United States and other countries.

Apache Tomcat, Tomcat, and Apache are trademarks of The Apache Software Foundation.

Java is a registered trademark of Oracle and/or its affiliates.

Other company and product names are trademarks or registered trademarks of their respective owners.

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 how to get started with .

Readers of this Manual

This manual is directed to operators who want to quickly set up a basic installation of OSCM with Docker and Docker Compose. For more detailed information on configuration and usage, refer to the official *OSCM documentation* . It assumes that you are familiar with the following:

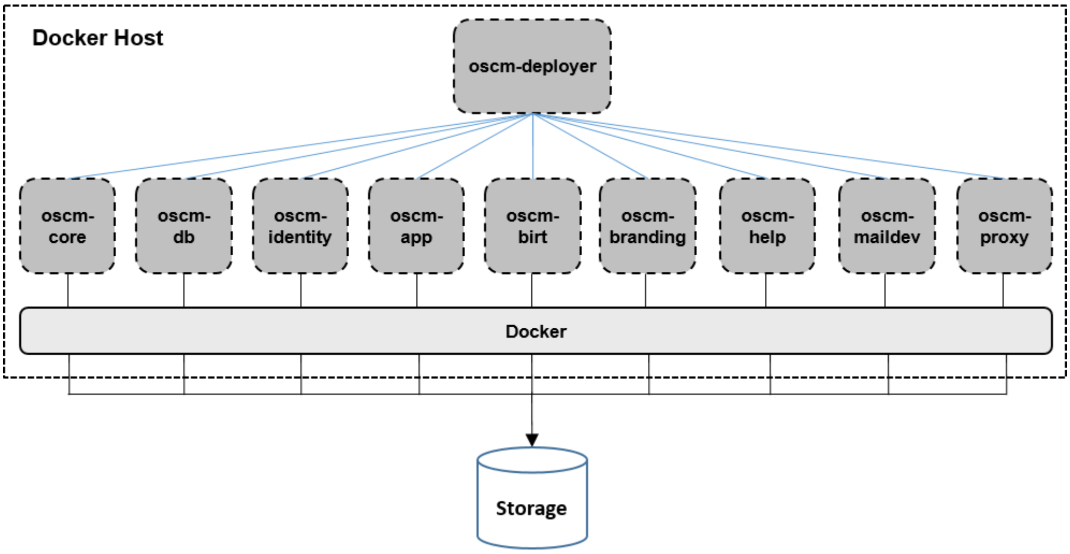
- Administration of the operating systems in use, including the adaption and execution of batch files or shell scripts.
- Java EE technology, particularly as to the deployment on application servers.
- Container technology, particularly Docker and Docker Compose.
- OSCM concepts as explained in the Overview manual.

1 OSCM Containers

OSCM is provided in Docker containers and deployed in a container environment. The applications integrated with OSCM and their data may be hosted on the same system (Docker host) as OSCM or in different locations.

The `oscm-deployer` container is used for configuring and deploying the following OSCM containers:

- `oscm-core`: The OSCM core application, including the platform services and the REST API.
- `oscm-db`: Database SQL server providing the database schema for OSCM and APP.
- `oscm-identity`: Services for authenticating users against an external authorization server based on OpenID Connect (OIDC) and providing for Web browser single sign-on.
- `oscm-app`: The Asynchronous Provisioning Platform (APP) together with an OpenStack, Amazon Web Services (AWS), Microsoft Azure, VMware, and Shell service controller.
- `oscm-birt`: The report engine that OSCM uses for generating reports.
- `oscm-branding`: A static Web server providing an empty directory structure for customizing the layout and branding of OSCM marketplaces.
- `oscm-help`: A static Web server providing the online help for the OSCM administration portal and marketplaces.
- `oscm-maildev`: A mail mock service for quick start-up and testing purposes when no real mail server is available.
- `oscm-proxy`: A proxy enabling access to all OSCM services and applications by the default HTTPS port (443).



OSCM and APP store their data in PostgreSQL databases. For the databases, a directory on the Docker host where OSCM is deployed is mounted as a volume for persistent storage during the deployment process. In this way, the data is preserved in case of container and database updates.

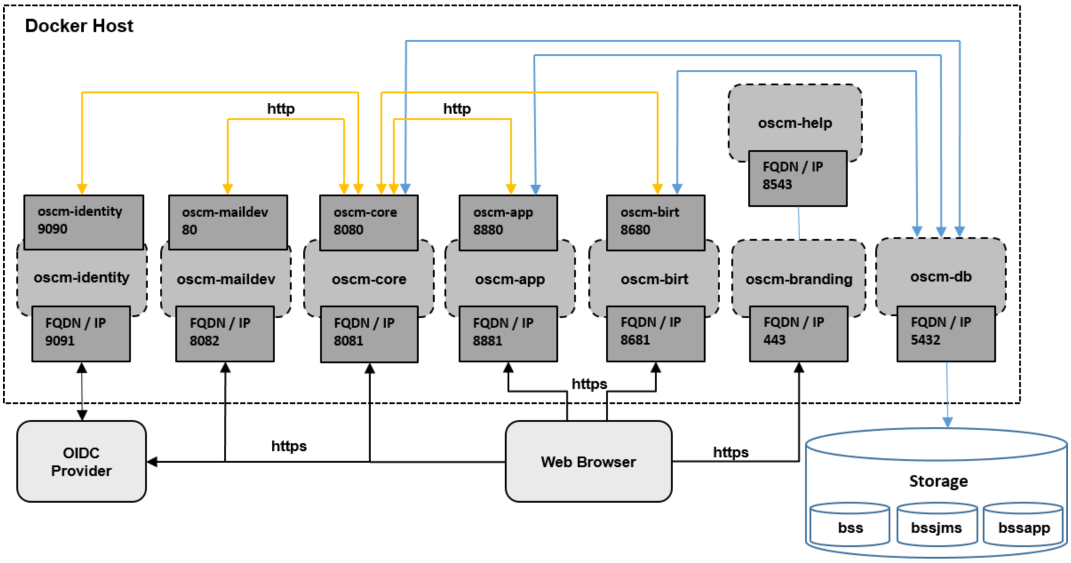
The directory on the Docker host and the path to which it is mounted as a volume in the `oscm-db` container are the following:

`<docker>/data/oscm-db/data:/var/lib/postgresql/data`

`<docker>` is the OSCM data directory on the Docker host specified when OSCM is installed.

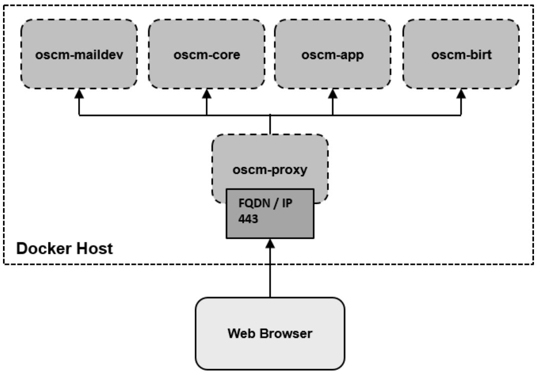
Container Communication

The following figure provides an overview of the container communication on a Docker host:



The internal communication between the Docker containers relies on the HTTP protocol, whereas calls from the outside are secured by HTTPS. The platform operator is responsible for opening the indicated ports. The containers can be addressed by their FQDN or their IP address.

The optional `oscm-proxy` container can be activated as a proxy to enable access to the other containers by the default HTTPS port (443) instead of the ports indicated above.



Logging in to Containers

To look inside a container:

1. List all containers (even stopped ones) to show the container names:

   ```
   docker ps -a
   ```

2. Log in to a container using the following command:

   ```
   docker exec -it <container name> /bin/bash
   ```

   For example:

   ```
   docker exec -it oscm-core /bin/bash
   ```

2 Prequisites and Preparation

For getting started with the setup of OSCM, a Linux system is required with the following software installed:

- Docker
- Docker Compose

This system is referred to as the **Docker host**.

The following system resources are required for the Docker host:

- 2 CPU cores
- 8 GB of RAM
- 20 GB of disk space

Note: This is a minimum configuration for testing purposes only. It is not suitable for production use.

2.1 Preparations for OIDC Authentication Mode

OSCM can be operated in one of the following authentication modes:

- INTERNAL: Users are managed and authenticated in OSCM itself.
- OIDC: Users are managed and authenticated by means of OpenID Connect (OIDC) in an external system such as Microsoft Azure Active Directory, the so-called OIDC provider.

Note: This section is relevant only for OIDC authentication mode.

Before you can install OSCM in OIDC authentication mode, a number of prerequisites and preparations are required in the relevant OIDC provider system.

The prerequisites are specific to the OIDC provider. The following descriptions use Microsoft Azure Active Directory (Azure AD) as an example. If you intend to work with a different OIDC provider, please contact your OSCM support organization for details.

The following prerequisites and preparations are required at the OIDC provider:

1. An account and domain must exist to which you have access as an administrator.

   Azure AD example: domain `mydomain.onmicrosoft.com`

2. A directory must exist in the domain, which contains at least all the users who are to work with OSCM.

   Azure AD example: directory `OSCMOrg`

3. Relevant only for the first deployment of OSCM: Create an access group named `OSCM_PLATFORM_OPERATOR` in the directory, and add the user who is to become the initial OSCM administrator as a member to the group.

   Azure AD example: Create a security group, `OSCM_PLATFORM_OPERATOR`, in the directory `OSCMOrg`. Add the initial administrator, e.g. `oscmadmin@mydomain.onmicrosoft.com`, as a member.

Note: Organizations in OSCM are mapped to groups at the OIDC provider. Each user can be a member of exactly one group only.

4. Register your OSCM installation as an application that connects to the OIDC provider directory.

Azure AD example: In the directory under **App registrations**, create a new registration with the following properties:

- A name and supported account types of your choice.
- Redirect URI: `https://<host_fqdn>:9091/oscm-identity/callback`

  `<host_fqdn>` is the fully qualified name or IP address of the host to access your OSCM installation, `9091` is the port. Omit the port if OSCM is operated with its proxy.

5. Copy the ID of the new application for later use in OSCM.

   Azure AD example: `ef29bb22-369c-424d-9e72-6800ad24239e`

6. Grant the application read and write permissions for the directory, users, and groups.

   Azure AD example: In the properties of the new app, under **API permissions**:

   a. Add the following permissions for the Microsoft Graph API as both, **Delegated permissions** and **Application permissions**: `User.Read.All`, `Group.ReadWrite.All`, `Directory.ReadWrite.All`.

   b. **Grant admin consent** for the directory, `OSCMOrg`, for the new permissions.

7. Obtain a client certificate or secret string for OSCM to access the directory.

   Azure AD example: In the properties of the new app, under **Certificates and secrets**, add a new client secret. Copy the new secret for later use in OSCM.

8. Allow OSCM to obtain tokens for authentication from the OIDC provider.

   Azure AD example: In the properties of the new app, under **Authentication**, enable **Access tokens** and **ID tokens** to be issued by the authorization endpoint.

Based on the items and settings in the OIDC provider system, you can now configure the so-called default tenant for your OSCM installation. Refer to *Preparing Configuration Files* on page 11 for details.

3 Installation

The installation of OSCM consists of the following main steps:

1. Importing the OSCM Docker images into the local Docker registry, if you have received them in archive files (`tar.gz` files).
2. Preparing the data directory on the Docker host.
3. Preparing configuration files.
4. Preparing Docker Compose files and starting OSCM.

3.1 Importing OSCM Docker Images into a Local Registry

Note: This section is relevant only if you have received the OSCM Docker images as `tar.gz` files, and do not install them directly from DockerHub.

In order to install Docker images you received as archive files, you first need to import them into a local Docker registry.

Make sure you have command-line access with write permissions to the Docker registry server in your network. If this is not available, install a Docker registry server in your environment. Refer to the Docker documentation at *https://docs.docker.com/registry/deploying* for details.

In the subsequent sections, the Docker registry is referred to as `${REGISTRY}`. The name of the OSCM registry on DockerHub is `servicecatalog`.

Proceed as follows to import the OSCM Docker images into the local Docker registry:

1. On the Docker registry server, copy the OSCM Docker image archive files into the `/opt` folder.
2. Import the images into the Docker registry using the following Docker command bash script:

```
#!/bin/bash

export REGISTRY=<docker registry service>

docker login ${REGISTRY}

for docker_image in \
        oscm-deployer \
        oscm-core \
        oscm-db \
        oscm-identity \
        oscm-app \
        oscm-birt \
        oscm-branding \
        oscm-help \
        oscm-initdb \
        oscm-maildev \
        oscm-proxy
; do

       docker load –i /opt/${docker_image}.tar.gz

       docker tag ${REGISTRY}/${docker_image}:v19

       docker push ${REGISTRY}/${docker_image}:v19

done
```

3.2 Preparing the Data Directory

On the Docker host, you need to create a directory where various OSCM data can be stored, such as persistent database data or configuration files.

The name of the directory must be **`docker`**. It may be located in the root directory or in any subdirectory. The following descriptions use `/docker` in the root directory.

On the Docker host, execute the following command:

```
mkdir /docker
```

3.3 Preparing Configuration Files

A deployment container and process is available which prepares templates and configuration files in the data directory on the Docker host, where you can then specify settings such as authentication and login information, mail server, etc.

Run the following process, using `-v` to mount the `/docker` data directory on the Docker host to the `/target` directory in the container:

```
docker run --name deployer1 --rm -v /docker:/target
    -e HOST_FQDN=<host_fqdn> ${REGISTRY}/oscm-deployer
```

`<host_fqdn>` is the fully qualified name or IP address of the host used to access OSCM, for example, `host.mydomain.org`. The host specified here is written to the `.env` configuration file (see below) and to the default certificates (see *Integrating Custom SSL Certificates and Key Files* on page 15).

`${REGISTRY}` is the name of your local Docker registry or the OSCM registry on DockerHub.

This command creates the following configuration files in the `/docker` directory:

1. `.env`: Configuration settings for Docker, such as images and the data directory on the Docker host, as well as the host used to access OSCM (`HOST_FQDN`).
2. `var.env`: Configuration settings for OSCM and APP, such as authentication, mail server, database, and other settings. They will be stored in the `bss` and `bssapp` databases.
3. `config/oscm-identity/tenants/tenant-default.properties`: Only relevant, if you set `OIDC` as the authentication mode: the configuration of the default tenant.

Edit these files and adjust the configuration settings to your environment. The settings are described in detail in the Operator's Guide .

3.4 Preparing Docker Compose Files and Starting OSCM

A second process in the deployment container is available which you need to run to do the following:

1. Create the necessary Docker Compose files for running OSCM.
2. Create the necessary subdirectories in the OSCM data directory.
3. Initialize the application databases.
4. Start the application containers.
5. Activate the OSCM proxy.

   Use of the OSCM proxy is optional. If you prefer to work without it, omit `-e PROXY=true` in the command below.

Run the following process on your Docker host:

```
docker run --name deployer2 --rm -v /docker:/target
    -v /var/run/docker.sock:/var/run/docker.sock -e INITDB=true
    -e PROXY=true -e STARTUP=true ${REGISTRY}/oscm-deployer
```

`${REGISTRY}` is the name of your local Docker registry or the OSCM registry on DockerHub.

After OSCM has been deployed, it will take a few minutes to start up. The less power the Docker host has, the longer it will take.

Note: If you activate the OSCM proxy, it is configured to handle the ports of OSCM. You can also use it for additional services and applications by extending its configuration accordingly using NGINX syntax in the following file:

`/docker/config/oscm-proxy/data/proxy.conf`

Do not change any of the existing entries for OSCM!

4 Usage

The following sections provide some basic hints on how to start working with OSCM after it has been deployed and started successfully.

4.1 Accessing the OSCM Administration Portal

The administration portal is the Web interface you use to perform all the configuration and administration tasks in OSCM, like creating and managing organizations, roles, and marketplaces.

You can access the administration portal in your Web browser using a URL in the following format:

```
https://<host_fqdn>:8081/oscm-portal
```

`<host_fqdn>` is the FQDN or IP address to access OSCM as specified in the `HOST_FQDN` setting in the `.env` configuration file, `8081` is the port. Omit the port if OSCM is operated with its proxy. `oscm-portal` is the default context root of OSCM and cannot be changed.

You are prompted for the user ID and password. The login page and the initial credentials depend on the selected authentication mode (`AUTH_MODE` setting in the `var.env` configuration file):

- `INTERNAL` authentication mode:

  Login page of OSCM

  User ID: `administrator`

  Password: `admin123`

  It is recommended that you change the initial password in the OSCM administration portal (**Change Password** page in the **Account** menu).

- `OIDC` authentication mode:

  Login page of the OIDC provider. The page may be skipped if single sign-on is supported and you are already logged in.

  User ID: The ID you specified in the `ADMIN_USER_ID` setting in the `var.env` configuration file

  Password: The password of the user as set in the external authorization system used for authentication. The password can be changed in this system only.

After login, the operator functionality is available in the **Operation** menu.

4.2 Enable Login to APP and Service Controllers

In order to be able to log in to the Asynchronous Provisioning Platform (APP) and its service controllers, some settings have to be made in the administration portal:

1. Choose **Manage organization** in the **Operation** menu.
2. Enter `PLATFORM_OPERATOR` in the **Organization ID** field.
3. Enable the following organization roles: **Supplier** and **Technology provider**.
4. Fill in the mandatory fields (red asterisks).
5. Click **Save**.
6. Go to the **Account** menu and choose **Manage users**.
7. Click the initial administrator account (`administrator` for `INTERNAL` authentication mode, or the administrator specified in the `ADMIN_USER_ID` setting in the `var.env` configuration file for `OIDC` authentication mode).
8. Enter your email address.

9. Enable all user roles.
10. Click **Save**.
11. Log out of the administration portal and log in again to enable the changes.
12. Now you can log in to APP and change APP-specific settings:

    `https://<host_fqdn>:8881/oscm-app/`

    Omit the port if OSCM is operated with its proxy.

    User name: the initial administrator account

    Password: the password of the initial administrator
13. You can also log in to the instance status interface of each service controller:

    `https://<host_fqdn>:8881/oscm-app-<controller-name>/`

    `<controller-name>` is `openstack`, `aws`, `vmware`, `azureARM`, or `shell`. Omit the port if OSCM is operated with its proxy.

    User name: the initial administrator account

    Password: the password of the initial administrator

4.3 Start Using OSCM

For the initial steps for starting to use OSCM, refer to the *Getting Started* document.

5 Integrating Custom SSL Certificates and Key Files

Certificates are required for OSCM to allow for trusted communication between OSCM and the Asynchronous Provisioning Platform (APP), an application underlying a technical service, or an external authorization server. The OSCM deployment creates an appropriate directory structure and Docker Compose configuration, and inserts default certificates for the individual containers, thus allowing for secure communication between OSCM and APP or standard authorization systems such as Microsoft Azure.

In addition, you can import individual certificates to the OSCM containers or make use of custom SSL key pairs for the application listeners. All you need to do is place the certificates and/or key files into the appropriate directories on the Docker host as described in more detail below. The certificates may be self-signed or official. Privacy Enhanced Mail (PEM) format is mandatory. This is a container format that may include just a public certificate or an entire certificate chain with public key, private key, and root certificates.

5.1 Importing Trusted SSL Certificates

If you want OSCM or your applications to trust certain, possibly self-signed SSL certificates, put them in PEM format into the following directory on your Docker host:

`<docker>/config/certs`

`<docker>` is the OSCM data directory on the Docker host specified at installation time.

The `<docker>/config/certs` directory is shared by all OSCM containers. However, you need to restart each container that is to use a new certificate you copy to the directory.

For example, if you want to use the VMware service controller, you need to export the vSphere certificate in PEM format and copy it to the `<docker>/config/certs` directory. Since the VMware service controller is running in the `oscm-app` container, a restart of this container is required.

5.2 Importing SSL Key Pairs for Application Listeners

If you want to use your own SSL key pairs for OSCM and your applications, replace the default key pairs by your PEM files in the following directories on your Docker host:

- Private key: `<docker>/config/<container-name>/ssl/privkey`
- Public certificate: `<docker>/config/<container-name>/ssl/cert`
- Intermediates / chain (optional): `<docker>/config/<container-name>/ssl/chain`

`<docker>` is the OSCM data directory on the Docker host specified at installation time. `<container-name>` is the name of the relevant OSCM container, for example, `oscm-core` or `oscm-app`.

The certificate must also be placed into the following directory shared by all containers so that a trusted relationship between the containers is established:

`<docker>/config/certs`

For example, if you have a custom SSL keypair for the `oscm-core` container, you need to place the private key into the `<docker>/config/oscm-core/ssl/privkey` directory, and the public certificate into the `<docker>/config/oscm-core/ssl/cert` directory. Additionally, you need to place the public certificate into the `<docker>/config/certs` directory. In this case, a restart of the `oscm-core`, `oscm-app`, and `oscm-identity` containers is required.