FUJITSU

FUJITSU Software Enterprise Service Catalog Manager V19.1

Technology Provider's 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.
Copyright FUJITSU ENABLING SOFTWARE TECHNOLOGY GMBH 2021	All rights reserved, including those of translation into other languages. No part of this manual may be reproduced in any form whatsoever without the written permission of FUJITSU ENABLING SOFTWARE TECHNOLOGY GMBH.

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 tasks of technology providers in order to prepare software applications for usage in a SaaS model and integrate them with) FUJITSU Software Enterprise Service Catalog Manager, hereafter referred to as ESCM.

The manual is structured as follows:

Chapter	Description
Introduction on page 8	Explains how to prepare applications for SaaS and gives an overview of the integration tasks.
Access Types on page 15	Describes how users can access an application that is integrated with ESCM.
Integrating Applications with ESCM on page 20	Describes how to implement the interfaces between an application and ESCM.
Provisioning Applications as Services in ESCM on page 24	Describes how to register applications as services in ESCM and appoint suppliers for them.
Reporting on page 28	Describes the reports available for technology providers in ESCM.
Technical Service Definition XML File on page 29	Describes the elements of a technical service definition.
Integrating Applications with ESCM Using APP on page 38	Describes the Asynchronous Provisioning Platform (APP).

Readers of this Manual

This manual is directed to technology providers who are responsible for integrating applications with ESCM.

This manual assumes that you are familiar with the following:

- ESCM concepts as explained in the *Overview* manual
- Basic Web service concepts
- XML and the XSD language
- A programming language that can be used to create and invoke Web services, for example, Java
- Java, Java servlets, and Java server pages
- Installation and basic administration of Web servers

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:

API	Application programming interface
APP	Asynchronous Provisioning Platform
ESCM	Enterprise Service Catalog Manager
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
SaaS	Software as a Service
WSDL	Web Services Description Language
XSD	XML Schema Definition

Available Documentation

The following documentation on ESCM is available:

- *Overview:* A PDF manual introducing ESCM. It is written for everybody interested in ESCM and does not require any special knowledge.
- *Operator's Guide:* A PDF manual for operators describing how to administrate and maintain ESCM.
- *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 ESCM.
- *Supplier's Guide:* A PDF manual for suppliers describing how to define and manage service offerings for applications that have been integrated with ESCM.
- *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 ESCM.
- *Microsoft Azure Integration:* A PDF manual for operators describing how to offer and use virtual systems controlled by Microsoft Azure through services in ESCM.
- *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 ESCM.
- *OpenStack Integration:* A PDF manual for operators describing how to offer and use virtual systems controlled by OpenStack through services in ESCM.

- *VMware vSphere Integration:* A PDF manual for operators describing how to offer and use virtual machines provisioned on a VMware vSphere server through services in ESCM.
- *Shell Integration:* A PDF manual for operators describing how to use Shell scripts through services in ESCM.
- *Online Help:* Online help pages describing how to work with the administration portal of ESCM. The online help is intended for and available to everybody working with the administration portal.

1 Introduction

Enterprise Service Catalog Manager (ESCM) 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.

With its components, ESCM supports software vendors as well as their customers in leveraging the advantages of Cloud Computing.

The basic scenario of deploying and using applications as services in the ESCM framework involves the following organizations:

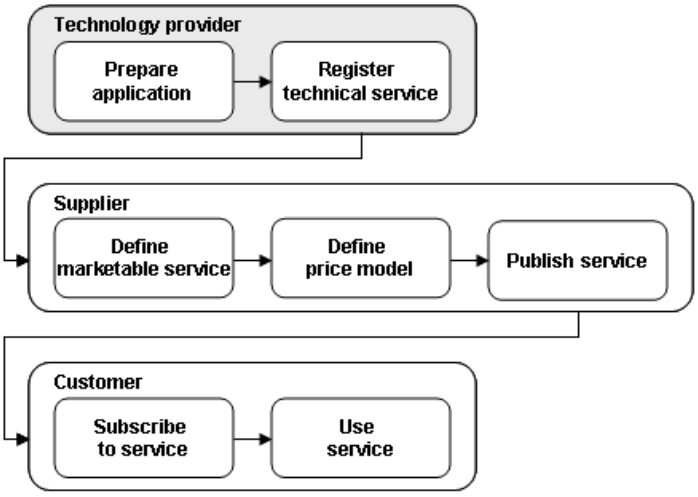
- **Technology providers** (e.g. independent software vendors) technically prepare their applications for usage in the Cloud and integrate them with ESCM. They register the applications as technical services in ESCM.
- **Suppliers** (e.g. independent software vendors or sales organizations) define service offerings, so-called marketable services, for the technical services in ESCM. They publish the services to a marketplace.
- **Customers** register themselves or are registered by an authorized organization in ESCM and subscribe to one or more services. Users appointed by the customers work with the underlying applications under the conditions of the corresponding subscriptions.
- **Marketplace owners** are responsible for administrating and customizing the marketplaces to which services are published.
- **Operators** are responsible for deploying and maintaining ESCM.

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

In extended usage scenarios, the suppliers who define marketable services may involve additional users and organizations in offering and selling these services:

- **Brokers** support suppliers in establishing relationships to customers by offering the suppliers' services on a marketplace. A service subscription is a contract between the customer and the supplier.
- **Resellers** offer services defined by suppliers to customers applying their own terms and conditions. A service subscription establishes a contract between the customer and the reseller.

1.1 The Technology Provider's Tasks in ESCM



As a technology provider, you own or develop applications which you intend to provide as services in the Cloud using ESCM to cover the business-related functionality.

To achieve this goal, you perform the following basic tasks:

- Prepare your applications for usage in a SaaS model, taking into consideration aspects such as a remote interface, multi-tenancy, high scalability and availability, and security.
- Implement the technical interfaces for integrating the applications with ESCM.
- Provision the applications as technical services in ESCM.

As soon as a technical service is available in ESCM, one or more suppliers appointed by you can define actual service offerings, so-called marketable services, for it. The suppliers (or brokers or resellers appointed by them) publish these services to a marketplace, where customers can subscribe to them.

ESCM provides specific reports for technology providers. For example, you can retrieve information on the number of subscriptions for your services and on the usage and load of the underlying applications.

As a technology provider, you can also act as a customer in ESCM. This means that you are automatically privileged to subscribe to services and work with the services you have subscribed to.

If you also need to perform supplier or marketplace owner tasks, your organization must be assigned the corresponding roles by the ESCM operator.

1.2 Preparing Applications for Usage in a SaaS Model

An application that is to be offered in a SaaS model should take the following aspects into consideration:

Remote Interface

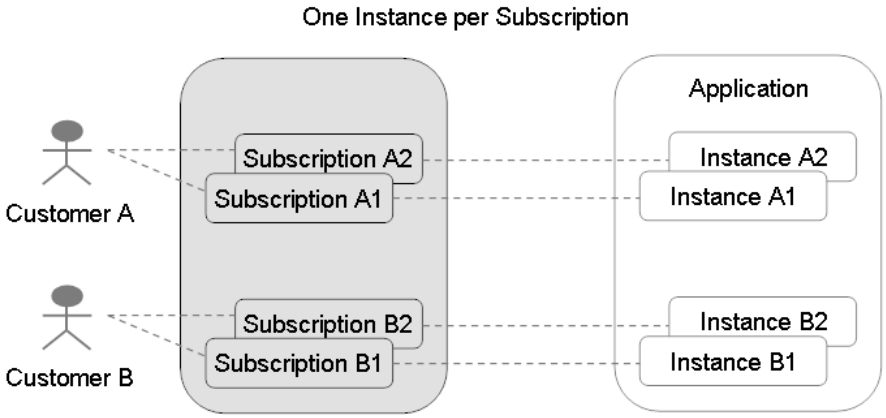
Your application should have a remote interface by which users can access it from anywhere in the Web (Internet/Intranet). This may be, for example, a Web user interface or a Web service.

Multi-Tenancy

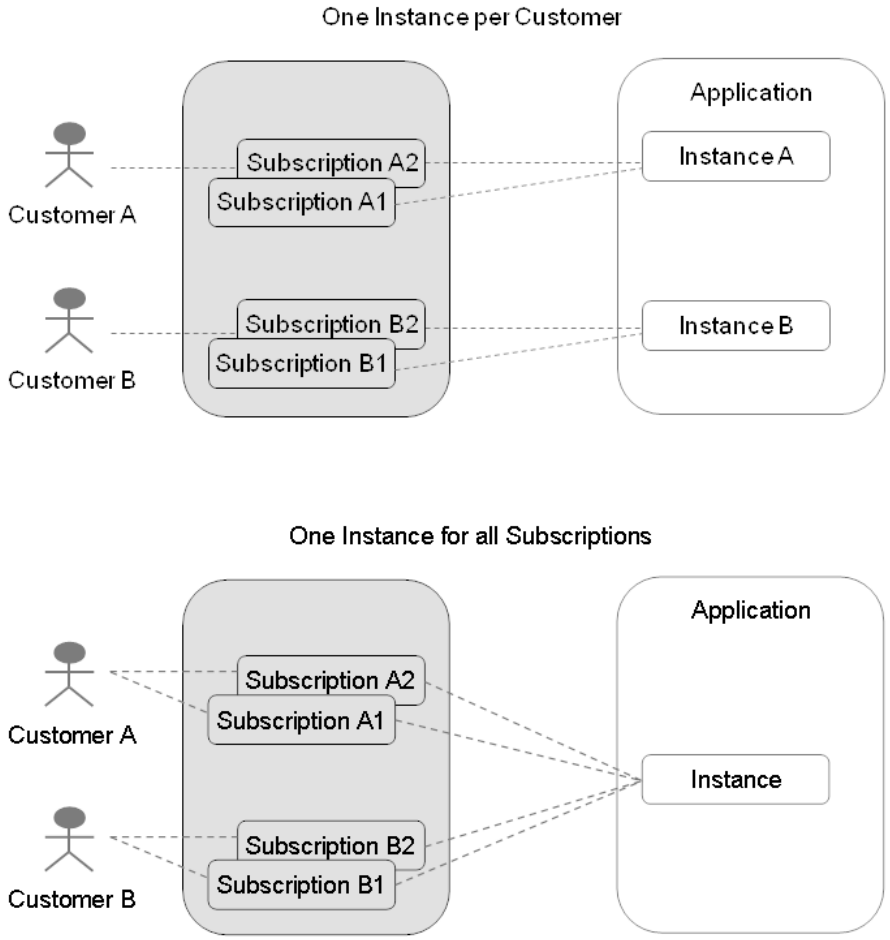
Your application should have multi-tenancy capabilities, at least with respect to the management of data: The data of different customers or of different service subscriptions of a customer should be clearly separated from each other and only be accessible by the respective customer or subscription user.

In ESCM, the term "instance" refers to what is provisioned for a tenant (customer or subscription) on the application side. Before integrating an application with ESCM, you must decide what to consider an instance in the context of the application. Possible solutions range from using different workspaces in one data container over maintaining different databases to instantiating different virtual servers in an Infrastructure as a Service (IaaS) environment.

Also, you need to decide what you want to provision an instance for. The usual and recommended way is to provision one instance for each subscription:



In case your application runs in a stateless mode and there is no need to store any data, or in case you know that a customer will only use one subscription, you could use one instance for a customer or you could even use a single instance for all subscriptions of all customers:



High Scalability

The number of users, performance, and space requirements may differ to a great extent for each customer. Therefore, your application should provide for high scalability.

High Availability

Ideally, your application should be available 24x7. You may consider having your application hosted in a professional data center that takes care of non-stop operation, backup, data security, and regular maintenance.

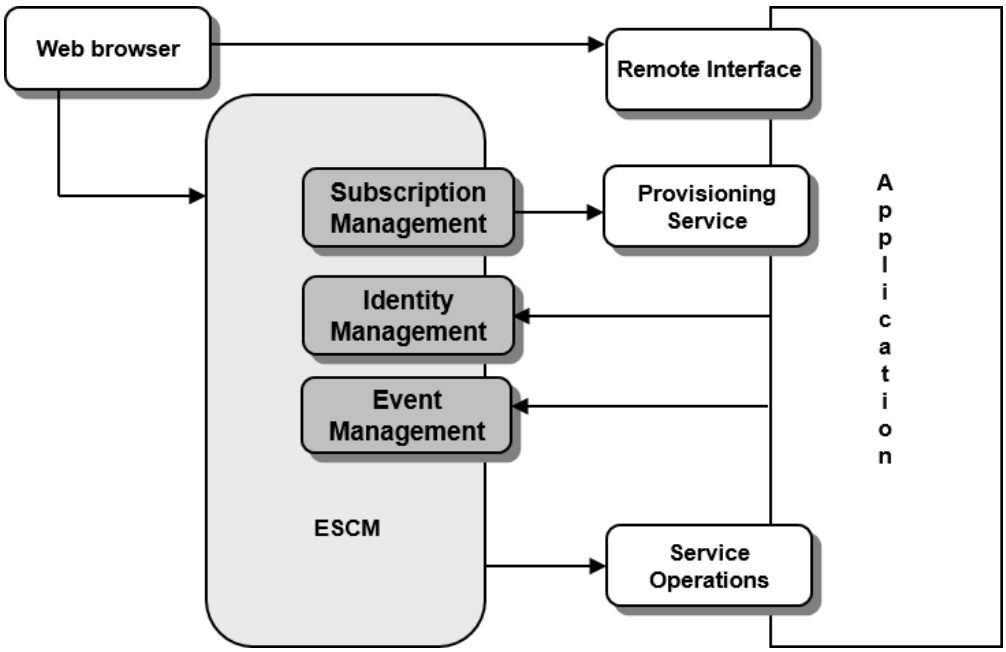
Security

Consider the following security aspects when implementing your application: Web service calls between ESCM and the application can be sent as plain text containing a user ID and password.

For secure communication, the usage of certificates is recommended.

1.3 Overview of Integration Tasks

An application can be integrated with ESCM components as shown in the following illustration. The ESCM components are colored in dark grey:



Depending on your requirements, the integration involves the following tasks:

1. Decide how users access your application. Depending on the type of access, you need to consider different aspects for integration.

   Users may access an application directly or through ESCM. When access occurs through ESCM, user management and authentication are under full control of the ESCM identity management, and price models on a per-user basis can be used.

   For details on access types, refer to *Access Types* on page 15.

2. Implement a provisioning service (optional).

   If an application is to integrate with the subscription management of ESCM, it must provide a corresponding Web service (provisioning service). By the provisioning service, ESCM triggers the application to provision and manage whatever is required for a subscription.

   For details, refer to *Implementing a Provisioning Service* on page 20.

3. Adapt the application's login/logout implementation (optional).

   If an application is to be accessed through ESCM, its login/logout implementation must be adapted. Depending on the access type, specific methods defined by the provisioning service must be implemented.

   For details, refer to *Adapting the Login/Logout Implementation* on page 21.

4. Integrate with the ESCM event management (optional, but recommended for fine-grained pay-per-use billing).

   The application can send events to the event management of ESCM. Events can be used as a basis for price models, billing, and reporting.

For details, refer to *Integrating with ESCM Event Management* on page 22.

5. Implement service operations (optional).

   The application can implement operations that can be executed from the ESCM user interface. Service operations can be used to access the resources of the application and perform administrative tasks.

   For details, refer to *Implementing Technical Service Operations* on page 22.

6. Provision your application as a service in ESCM.

   To do so, you first have to register the application as a technical service with ESCM, either via the ESCM user interface or by preparing an XML service definition and importing it into ESCM. Afterwards, you appoint suppliers for your services, who can then define service offerings.

   For details, refer to *Provisioning Applications as Services in ESCM* on page 24.

1.4 Accessing ESCM

You use the ESCM user interface to perform some of your tasks. The role of your organization as a technology provider basically determines which features are available to you at the ESCM user interface. The actions available to you as an individual user additionally depend on your user role within the organization.

ESCM distinguishes between the following user roles within technology provider organizations:

- **Administrator**: Each technology provider organization must have at least one user with this role. An administrator can manage the organization's account and subscriptions as well as its users and their roles. The first administrator of an organization is defined when the organization is created.
- **Technology manager**: This role allows a user to define technical services in ESCM.

To access the ESCM user interface, you use the login information provided by ESCM in the email confirming the creation of your user account. If your organization uses an external authentication system, passwords are managed in this system. This means that you log in with the password as stored in this system, and the email sent by ESCM does not contain a password.

To log in to the administration portal where you will perform your tasks:

1. Click the link provided in the email, or type the access URL in your Web browser's address bar.

   The access URL has the following format:

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

   `<host_fqdn>` is the fully qualified name or IP address of the host used to access ESCM, `8081` is the port. Omit the port if ESCM is operated with its proxy. `oscm-portal` is the default context root of ESCM and cannot be changed.

   If your organization uses an external system for user management and authentication, the identifier of a corresponding tenant may be appended to the URL:

   `https://<host_fqdn>:8081/oscm-portal/?tenantID=<id>`

2. On the **Login** page, type your user ID and password.

   Depending on the system used for user authentication, the **Login** page may be that of ESCM or of an external provider like Microsoft Azure, or it may be skipped entirely if single sign-on is supported and you are already logged in.

3. Click **Login**, or press **Return**.

You are either logged in directly, or you are prompted to change your initial password when you log in for the first time. It is highly recommended to change the initial password.

If you try to log in with a wrong password, your account may be locked after the third attempt. This depends on whether your organization maintains its user data in an external authentication system. In this case, passwords can only be changed or reset in this system. If user data are maintained in the platform, contact your administrator who can reset your password. You will get a new temporary password for your next login.

If you have forgotten your password, click **Forgot your password?** on the **Login** page. This allows you to define a new password for your user ID. Defining a new password is not possible if your account is locked or if your organization maintains its user data in an external authentication system.

If you have forgotten your user ID, contact your administrator who can look up the IDs of all users registered for your organization.

If your session expires, you have to log in again.

Accessing the ESCM REST API

ESCM provides a REST (Representational State Transfer) API by which you can address and execute the most important functions of the platform services. The REST API and its documentation are accessible and useable in a Web browser by means of the Swagger toolkit:

1. Type the following URL in your Web browser's address bar:

   `https://<host_fqdn>:8081/oscm-rest-api`

   `<host_fqdn>` is the fully qualified name or IP address of the host used to access ESCM, `8081` is the port. Omit the port if ESCM is operated with its proxy.

2. In the login dialog, enter the numeric key for your ESCM user ID and your password.
3. Confirm your entries.

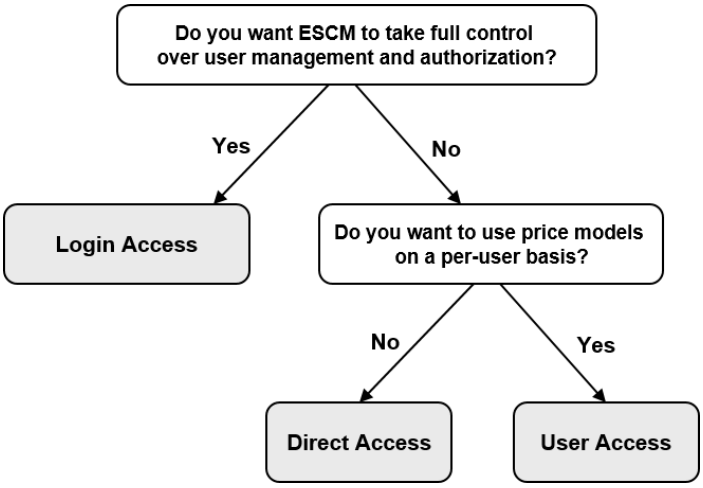
This opens the Swagger UI with the ESCM REST API documentation. From the UI, you can try out and actually execute the individual methods, if you are authorized to do so or specify appropriate user credentials using the **Authorize** option. For more information about Swagger, refer to its documentation in the Internet.

2 Access Types

ESCM offers various ways of integrating and accessing your application. The interaction takes place between the following components:

- The client which in fact is the user who accesses a service using a Web browser or a Web service.
- ESCM.
- The application which is accessed by the user.

There are some criteria to be considered before you decide to use a specific access type. The following decision diagram helps you in deciding which access type to use according to your environment and requirements:



The decision you make is reversible. If the basic requirements of your system are changing, you can, of course, also change the access type.

The access type determines

- whether your application uses ESCM for user authentication and ESCM forwards login information to your application (**application login**).
- whether your application needs to implement a **provisioning service**.

The following table provides an overview of the different access types and the available combinations:

Functionality / Access type	Application Login (to be supported by the application)	Provisioning Service (to be supported by the application)
Login access	yes	yes
Direct access	no	yes without user management
User access	yes *	yes
External access	no	no

* *With user access, the application login can be controlled completely by the application.*

You define the desired access type when preparing a technical service definition for your application. If you opt for access through ESCM (login access), you need to adapt the login/ logout implementation in your application. Depending on the access type you choose, you must implement the methods defined by the provisioning service.

The following sections provide detailed information on the different access types. The sequence diagrams illustrate the interaction between the involved components. Arrows with solid lines represent messages requiring a response, arrows with dotted lines represent the response messages.

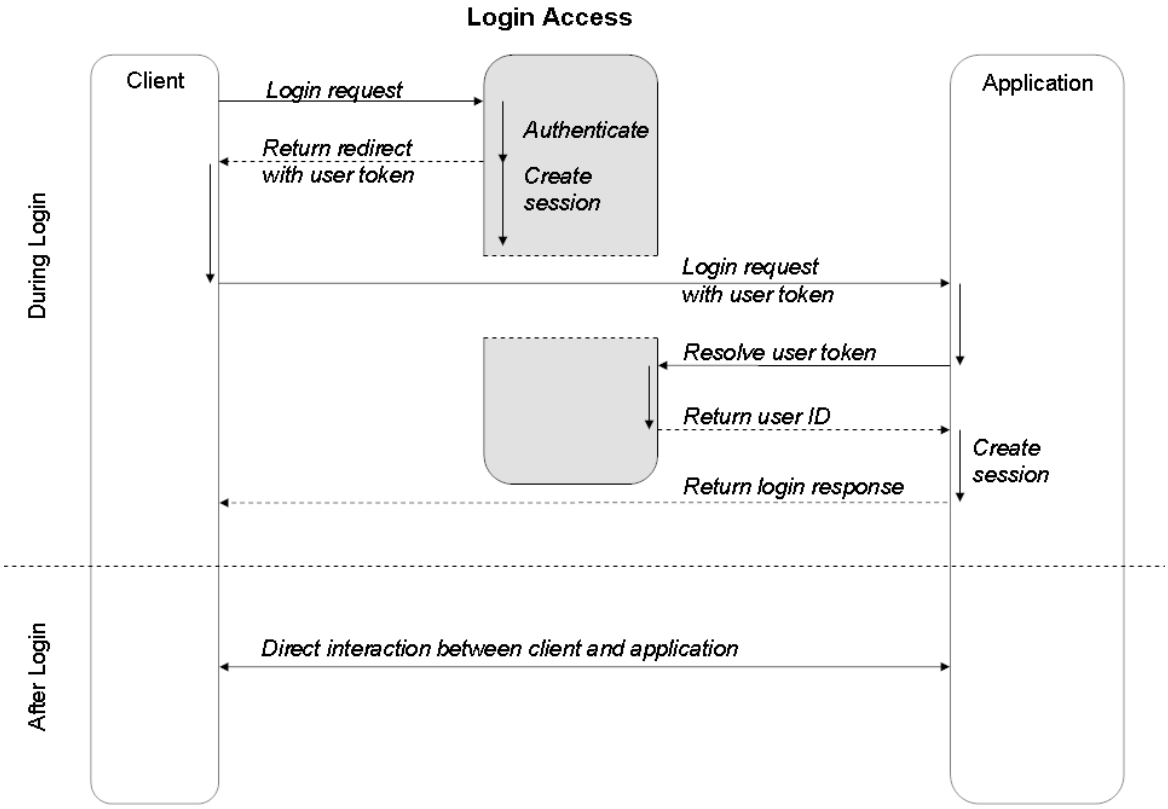
2.1 Login Access

With login access, ESCM is involved during login.

User management and authentication are under full control of the ESCM identity management, and price models on a per-user basis and corresponding billing services can be used.

After login, any interaction between the client and the application takes place directly and without ESCM being involved.

The following figure shows the interaction between the client, ESCM, and the application:



When a user logs in to ESCM in order to use a service, he is authenticated by ESCM and an ESCM session is created. ESCM creates a user token and returns it to the client together with a redirect to the application. The client, in turn, sends the login request with the user token to the application, which must be publicly visible. To be able to log the user in, the application sends a login request to ESCM prompting for the user ID corresponding to the user token. Once the user ID is returned, the application needs to log the user in without requesting any further credentials. Users are trusted because they have been authenticated by ESCM.

After login, any interaction takes place directly between the client and the application.

Note:	Your application should provide some default content for the base URL, since this URL specifies the application's remote interface if a user is already logged in to ESCM. The base URL is specified in the technical service definition (see *Technical Service Definition XML File* on page 29).

Logout

ESCM needs to be notified by the application when a user logs out. The corresponding ESCM user session is then closed and the session data is updated in the ESCM database.

After logout, the application must redirect the user to its own logout page. The full URL of this page is returned by the `deleteServiceSession` method of the ESCM session service.

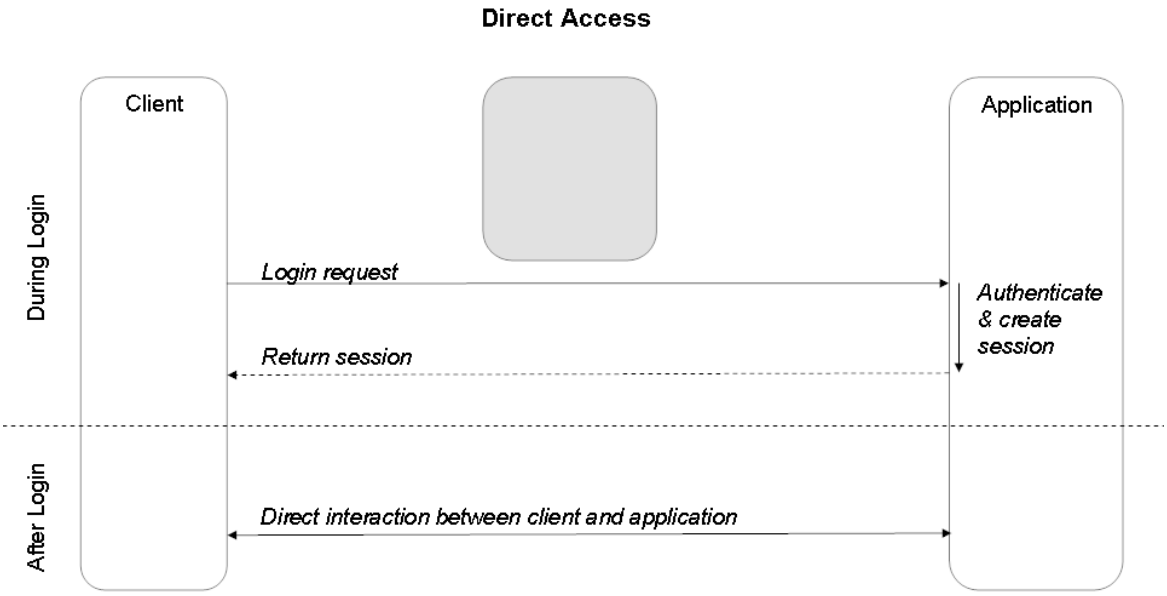
Note:	Users must not be redirected to the session after logout. Access to the application is no longer authorized because the ESCM session has been closed.

2.2 Direct Access

With direct access, users interact directly with the application.

As login is not carried out through ESCM, there are no special requirements from ESCM concerning the application's login/logout functionality. ESCM has no information about the number and duration of user sessions. Therefore, price models on a per-user basis and corresponding billing services in ESCM cannot be used.

The following figure shows the interaction between the client, ESCM, and the application:



The client sends a user's login request directly to the application. The application authenticates the user, creates a session, and returns that session to the client. Any subsequent interaction also takes place directly between the client and the application.

The application needs to ensure that users are directed to the application instance that belongs to the relevant subscription in ESCM. For Web applications, this could be achieved, for example, by configuring separate application URLs for the individual subscriptions. The relevant application URL can be included in the information which is returned to ESCM when the instance for a new subscription is created.

2.3 User Access

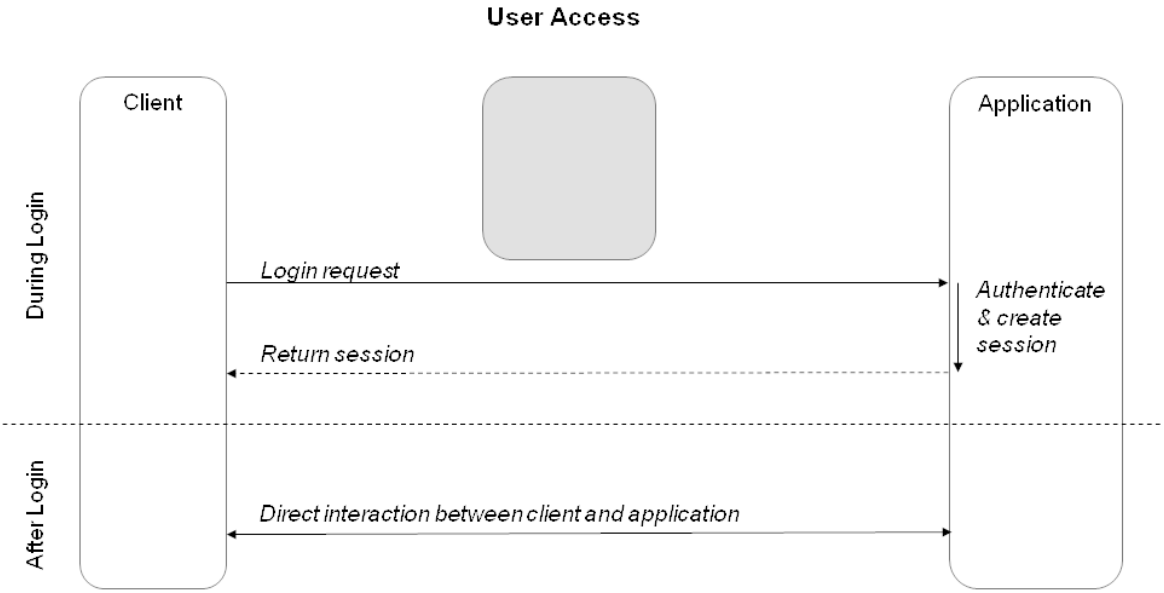
With user access, users interact directly with the application.

However, a basic user management is still carried out in ESCM, which means that the corresponding methods of the provisioning service are called when assigning a user to a

subscription. Thus, price models on a per-user basis and corresponding billing services in ESCM can be used.

Users may interact directly with the application without involving ESCM in any way.

The following figure shows the interaction between the client, ESCM, and the application:



The client sends a user's login request directly to the application. The application authenticates the user, creates a session, and returns that session to the client. Any subsequent interaction also takes place directly between the client and the application.

The application needs to ensure that users are directed to the application instance that belongs to the relevant subscription in ESCM. For Web applications, this could be achieved, for example, by configuring separate application URLs for the individual subscriptions. The relevant application URL can be included in the information which is returned to ESCM when the instance for a new subscription is created.

2.4 External Access

With external access, users can access an application directly after subscribing to a corresponding service. The users are redirected immediately to the application. The URL leading to the application is specified in the technical service definition.

Any further interaction takes place directly between the client and the application without involving ESCM in any way.

3 Integrating Applications with ESCM

Integrating an application with ESCM involves the following implementation tasks:

- Implement a provisioning service
- Adapt the login/logout implementation
- Integrate the application with the ESCM event management
- Implement service operations

These tasks are described in detail in the sections below.

A package with integration samples including source code is available on demand. Contact FUJITSU Enabling Software Technology GmbH at sales@est.fujitsu.com.

Prerequisites

Integrating applications with ESCM involves several interfaces for communication between the ESCM platform and the technical service or application. The communication takes place in two directions: Calls from the application to ESCM (inbound calls) as well as calls from ESCM to the application (outbound calls). For inbound calls, the application must implement a client; for outbound calls, the application must provide a server.

Before you begin, make sure that the following prerequisites are met:

- The application to be integrated is installed and operational.
- An application server is installed and operational.
- You have access to the ESCM integration package.

3.1 Implementing a Provisioning Service

As a first integration step, you implement a so-called provisioning service that exposes its operations as a Web service. A provisioning service is required for integrating an application with the subscription management of ESCM. The provisioning service is called by ESCM when customers subscribe to a service and manage their subscriptions. Additionally, the provisioning service may be called for creating and managing users.

You do not need to implement a provisioning service if you have chosen to use the external access type. For details on access types, refer to *Access Types* on page 15.

For the implementation of a provisioning service, you need to consider the following:

- Instance provisioning
- Provisioning mode
- Application parameters
- User management

These topics are described in detail in the following sections.

3.1.1 Instance Provisioning

When a customer subscribes to a service, the underlying application is supposed to perform specific steps required for the subscription and return an identifier to ESCM for future reference. The term 'instance' denotes all the items that the application has provisioned for a subscription.

The actions to be performed and the items to be created, if any, depend entirely on the concepts and functionality of your application. For example, if a customer creates and stores data when

using your application, your application may create a separate workspace in a data container or a separate database instance.

3.1.2 Provisioning Mode

Instance provisioning can be performed in synchronous or asynchronous mode.

Synchronous Mode

This mode is used if provisioning can be completed right away. The provisioning service triggers the application to perform all the required actions and confirms the operation as complete. ESCM then sets the subscription to active, which means that the service is ready to be used by the customer.

Asynchronous Mode

This mode is used if provisioning operations take a long time because long-running processes or manual steps are involved, or when huge amounts of data or virtual machines need to be set up. In this case, the provisioning service notifies ESCM that the provisioning is pending. Required actions may have started on the application side, but have not been completed yet. The provisioning service needs to notify ESCM using the subscription management service when provisioning is either complete or cannot be completed.

ESCM supports the development of asynchronous provisioning services with the asynchronous provisioning platform (APP). This is a framework which provides a provisioning service as well as functions, data persistence, and notification features which are always required for integrating applications in asynchronous mode. Refer to *Integrating Applications with ESCM Using APP*on page 38 for details on APP.

3.1.3 Application Parameters

An application may have parameters that are of relevance for the service provisioning in ESCM. Parameters can be used to define different feature configurations or service restrictions, for example, the maximum number of folders, files, or objects that can be created. Application parameters are specified in the technical service definition.

ESCM can pass parameters to your application through the instance provisioning call. Any further processing must be carried out by your application. Especially if parameters are used to impose restrictions on service usage, the application needs to ensure that the restrictions are met. For example, if there is a parameter to restrict the maximum number of files created for a subscription, the application needs to track the actual number and ensure that the maximum number is not exceeded.

For details on how to define parameters in the technical service definition, refer to *Technical Service Definition XML File* on page 29.

3.1.4 User Management

If users access your application through ESCM, you need to implement user management operations. These operations are called when users are assigned to or deassigned from a subscription in ESCM, or when user profiles are updated. Your application may take corresponding actions, for example, create corresponding user accounts in its own user management system.

3.2 Adapting the Login/Logout Implementation

If you opt for access through ESCM (login access), you need to adapt the login/logout implementation of your application to pass the control and authentication of users from the application to ESCM, and implement the relevant methods defined by the provisioning service.

The required functionality for login and logout is distributed between a token handler, a custom login module, a custom logout module, and a logout listener:

- The **token handler** is responsible for requesting ESCM to resolve a user token into a user ID. It takes up the task of creating a session object and storing the user ID in that object. Additionally, it forwards requests containing a resolved user token to the custom login module.
- The **custom login module** passes the user ID to the application. It enables users to log in to the application without requesting any further credentials. Users are trusted because they have already been authenticated by ESCM. For example, a custom login module might pass the user ID and a default password to the application.

  To ensure that any login takes place through ESCM, the direct login to your application must be bypassed. For this purpose, you have to implement the required interface methods between the application and ESCM.
- The **custom logout module** closes user sessions on the application side and redirects users to the logout page of ESCM. The URL of the logout page is returned to the application by the `deleteServiceSession` method of the ESCM session service.
- The **logout listener** notifies ESCM when a user logs out or a session timeout occurs.

3.3 Integrating with ESCM Event Management

The event management service in ESCM collects specific events generated during application operation. These events can be used for price models, billing, and reporting. Examples of events are the completion of a specific transaction, or the creation or deletion of specific data.

Your application can send events to ESCM at runtime through the event management service, which is one of the ESCM platform services.

To integrate with ESCM event management:

1. If your application does not generate the required events yet, implement the generation of events.
2. Implement the sending of events to ESCM.
3. When preparing the technical service definition, declare the events that your application will send.

   For details on the technical service definition, refer to *Technical Service Definition XML File* on page 29.

3.4 Implementing Technical Service Operations

You may wish that your technical service offers additional operations or functions that are to be accessible via the ESCM user interface. In a SaaS environment, applications are not installed locally but provisioned as services. Therefore, users cannot access the system resources the applications are using, for example, to perform administrative tasks such as system backup or shutdown. Technical service operations can be used to access the resources of an application and perform administrative tasks without actually opening the application.

To provide technical service operations, a Web service based on the operation service API must be implemented. The operations, their parameters, and the access information of the Web service must be specified in the technical service definition for the application. Refer to *Technical Service Definition XML File* on page 29 for details.

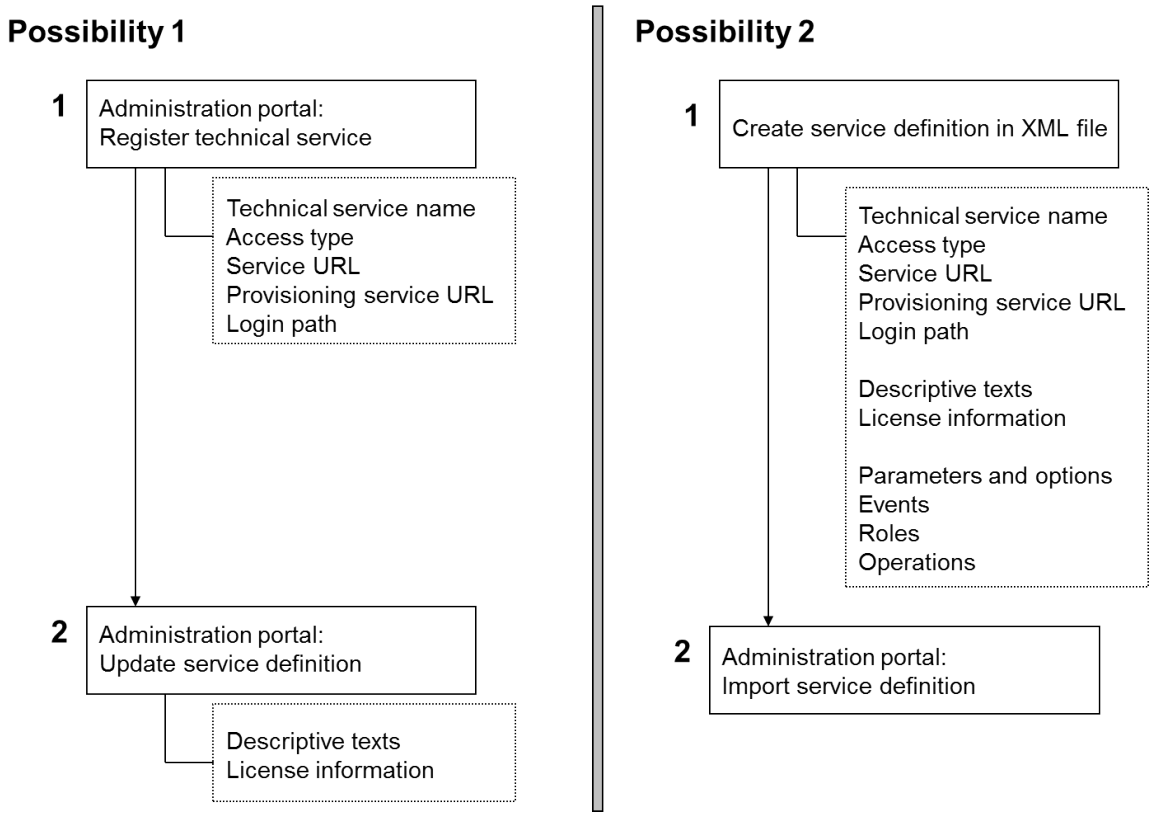
4 Provisioning Applications as Services in ESCM

After the required adaptations of your application are finished, you register your application as a technical service in ESCM.

There are two possibilities for achieving this task:

1. If the application does not provide any parameters, options, roles, events, or operations, register the application as a technical service using the ESCM user interface. Here, you can define the basic features for the technical service you want to provide.
2. If an application is more complex and provides parameters, options, etc., create an XML file containing the technical service definition and import this file into ESCM.

The figure below provides an overview of the two possibilities:



Possibility 1 consists of registering the application and then updating the service definition.
Possibility 2 consists of creating an XML file and importing this file into ESCM.

You can also combine both ways. For example:

1. Register the service using the ESCM user interface.
2. Export the service definition and save it to an XML file.
3. Edit the XML file: Specify descriptions, license information, parameters, options, events, roles, operations, etc.
4. Import the XML file into ESCM again.

After registering your application as a technical service, you need to appoint suppliers for it in ESCM so that marketable services can be defined and published.

The following sections describe the individual steps in detail.

4.1 Registering a Technical Service

By registering an application as a technical service, you make it available in ESCM.

To register a technical service:

1. Log in to the ESCM administration portal.
2. Do one of the following:
   - Select **Technical Service** > **Register service definition**, and fill in the fields as desired.
   - Select **Technical Service** > **Import service definition**, and import the XML file containing your service definition.

     For details on how to create a technical service definition, refer to *Defining a Technical Service in an XML File* on page 25.

4.2 Defining a Technical Service in an XML File

An XML file defining a technical service contains a description of the service's purpose, the license agreement, application parameters, options, events, service roles, and operations.

A service definition must conform to the `TechnicalServices.xsd` XML schema. The schema can be downloaded from *https://github.com/servicecatalog/oscm/releases/*. For detailed information on the elements and attributes, refer to *Technical Service Definition XML File* on page 29.

To prepare a service definition in an XML file:

1. Create an XML file according to the `TechnicalServices.xsd` XML schema, and save it under a name of your choice, for example, `ServiceDefinition.xml`.
2. In the `TechnicalService` section of the file, specify at least the following information:
   - `id`
   - `accessType`
   - `provisioningType` (if not specified, instance provisioning is performed in synchronous mode)
   - `provisioningUrl` (optional for external access, otherwise required)
3. Do one of the following:
   - If users log in directly to your application (user or direct access), describe how to access the application using the `accessInfo` element.
   - Fill in the `baseUrl` attribute (mandatory for login and external access).
   - If login is performed through ESCM (login access), fill in the `loginPath` attribute.
4. If there are any parameters that are to be passed to your application by an instance provisioning call, declare them in the `ParameterDefinition` sections.
5. If your application sends events to ESCM, declare all the events in the `Event` sections.
6. If you want to define your own service roles, declare them in the `Role` sections.
7. If you want to allow users to invoke specific operations on your technical service, declare the operations in the `Operation` sections.

4.3 Appointing Suppliers for a Technical Service

After registering a technical service either by importing the technical service definition XML file or by providing the required information at the user interface, you appoint one or more suppliers for it. Suppliers are the companies, departments, or people who offer your technical service as marketable services. Your organization may act as a supplier of its own, if it is granted the corresponding role by the ESCM operator.

To appoint suppliers:

1. Log in to the ESCM administration portal.
2. Select **Account** > **Manage suppliers** and select the technical service for which you want to appoint a supplier.
3. Specify the organization ID of the supplier to be appointed and click **Add**.

   You can only specify suppliers that have previously been registered by the ESCM operator. You can obtain the organization IDs from your suppliers.

The suppliers can now start defining marketable services for the technical services they are appointed for. These can be different editions of the technical services with individual price models, configurations, upgrade options, and restrictions.

You can also delete suppliers from the list of appointed suppliers. If customer subscriptions exist for the marketable services of a supplier, a corresponding message is displayed. It lists the instance IDs of the underlying application instances. If you want to view the instance IDs for your technical services, you can create an instance report.

4.4 Updating and Maintaining the Service Definition

You can update the descriptive texts and the license agreement text for a technical service either by editing the XML service definition file, or - in a more convenient way - at the ESCM user interface. The changes are effective for all newly defined marketable services based on this technical service.

To update the texts for the technical service in ESCM:

1. Log in to the ESCM administration portal.
2. Select **Technical Service** > **Update service definition**, and edit the texts as desired.

You can also update an existing technical service by editing the technical service definition and importing it again:

1. Log in to the ESCM administration portal.
2. Select **Technical Service** > **Import service definition**.

Observe the following:

- If there is no marketable service and no subscription based on the technical service, you can change all properties of the technical service.
- If there is a marketable service or a subscription based on the technical service, you cannot:
  - Change parameters and their options
  - Change event types
  - Remove parameters, options, or event types
- Roles cannot be removed if prices are defined for them or users having one of the roles are assigned to a related subscription.

You can also delete an existing technical service. If marketable services exist for the technical service, and if there are active subscriptions, it cannot be deleted. The technical service definition XML file is not deleted physically, and you can import it again on demand.

To delete a technical service:

1. Log in to the ESCM administration portal.
2. Select **Technical Service** > **Delete service definition**.

5 Reporting

ESCM offers comprehensive reports for different purposes and at different levels of detail. You can choose from various predefined reports.

The following reports are available for technology providers:

- **Technical service usage report**: Shows all technical services of a technology provider, including the type and number of the events that were collected during the usage of the services.
- **Supplier report**: Shows the suppliers registered by a technology provider for the technical services, and lists the marketable services which the suppliers have defined based on the technical services.
- **Subscription report**: Shows the marketable services suppliers created based on the technical services of the technology provider, and the number of subscriptions to the marketable services.
- **Instance report**: Shows all application instances of the technology provider's technical services and their configuration (parameters and values).

Use the **Create report** menu option in the **Account** menu of the ESCM user interface and choose the desired report. Depending on the report type, you might have to enter additional report parameters.

The generated report is instantly displayed at the ESCM user interface. You can choose to print the report or save it in several formats.

Note:	Contact your platform operator if the reporting functionality is not available. He is responsible for defining the respective configuration parameters.

Appendix A: Technical Service Definition XML File

A service definition contains the information required to register an application as a technical service in ESCM. It is specified in an XML file which needs to conform to the `TechnicalServices.xsd` XML schema. The schema can be downloaded from *https://github.com/servicecatalog/oscm/releases/*.

This section describes the meaning of the elements and attributes that can be defined. For information on the syntax, refer to the `TechnicalServices.xsd` schema.

TechnicalServices

Top-level element of a service definition file.

TechnicalService

Contains all the information required to register an application as a technical service in ESCM.

Attributes:

- **`id`** - ID of the application (required).

  The `id` is the key that uniquely identifies your application as a technical service. The ID must be unique across all your applications registered as technical services.

- **`build`** - Build number of the application (optional).

  You can use this number to specify the build number or patch level of the application underlying the technical service.

- **`provisioningType`** - Specifies whether instance provisioning is performed in synchronous or asynchronous mode (optional). The default is synchronous mode.

  For details on provisioning modes, refer to *Implementing a Provisioning Service* on page 20.

- **`provisioningTimeout`** - Specifies the time in milliseconds after which an asynchronous provisioning operation is considered to have failed (optional). By default, no timeout occurs.
- **`provisioningUrl`** - URL of the WSDL document that defines the provisioning service you implemented for your application (optional for the `EXTERNAL` access type, otherwise required).

  Be aware that for asynchronous provisioning, you need to use the HTTPS protocol.

- **`provisioningVersion`** - Version number of your provisioning service (optional).
- **`accessType`** - Specifies how users access your application (required).

  Use `DIRECT` if users log in directly to your application without involving ESCM.

  Use `LOGIN` if users log in to your application through ESCM. Subsequent interactions take place directly between the client and the application without involving ESCM.

  Use `USER` if the user-specific methods of the provisioning service should be used. With this access type, the application login can be controlled completely by the application.

  Use `EXTERNAL` if users should be able to access an application directly via the URL specified in the `baseUrl` attribute. The application ID (`id` attribute) is appended to the URL when accessing the service.

  For details on access types, refer to *Access Types* on page 15.

- **`baseUrl`** - URL of the remote interface of your application. Users are forwarded to this URL for service access.

Required for the `LOGIN` and `EXTERNAL` access type. For the `EXTERNAL` access type, you specify the external URL here. In the sample below (`baseUrl="https://myserver:7777/myservice"`), your Web application is running on `myserver` at port `7777`.

Optional for the `DIRECT` and `USER` access type. At the ESCM administration portal, a marketplace owner can define featured services for his marketplace home page. If a marketplace owner decides to display services and subscriptions grouped by category on his home page, the specification of the **`baseUrl`** is recommended. It allows the home page visitor to directly use the service if his organization has subscribed to it.

Your application should provide some default content for the base URL, since this URL specifies the application's remote interface.

Be aware that internet domain names must follow the following rules:

- They must start with a letter and end with a letter or number.
- They may contain letters, numbers, or hyphens only. Special characters are not allowed.
- They may consist of a maximum of 63 characters.

- **`loginPath`** - Path to the token handler, a module of your application that handles login requests containing a user token (optional). Required for the `LOGIN` access type.

  The path must be relative to the URL specified as `baseUrl` and start with a `/`.

- **`onlyOneSubscriptionPerUser`** - Defines whether an organization can subscribe only once to a service. Can be set to `true` or `false`. If set to `false` (default), an organization can subscribe to several marketable services based on your technical service. If set to `true`, an organization can subscribe to one marketable service based on your technical service only. In this case, the supplier can still define several marketable services, but as soon as a customer subscribes to one of these services, the other marketable services will no longer be available to him for subscription.

  At the ESCM administration portal, a marketplace owner can define featured services for his marketplace home page. If a marketplace owner decides to display services and subscriptions grouped by category on his home page, it is recommended that you set **`onlyOneSubscriptionPerUser`** to `true`. This allows the home page visitor to start using the service directly from the marketplace home page without the need to select the subscription from the list of existing subscriptions.

- **`allowingOnBehalfActing`** - Defines whether an organization can act on behalf of another organization. Can be set to `true` or `false` (default). If set to `true`, an organization with the technology provider and the supplier role can act in ESCM on behalf of a customer organization. This is achieved via a customer's subscription whose underlying technical service has the `allowingOnBehalfActing` attribute set to `true`.

Example:

```
<TechnicalService
   accessType="LOGIN"
   baseUrl="https://myserver:7777/myservice"
   build="25.01.2010"
   id="SampleService"
   loginPath="/login"
  provisioningType="SYNCHRONOUS"
   provisioningUrl=
       "https://myserver:8090/axis/services/MyProvisioningService?wsdl"
   onlyOneSubscriptionPerUser="false">
```

AccessInfo

Only required for the `DIRECT` or `USER` access type.

Contains a textual description of how users can access your application in the language specified by the `locale` attribute. The access information may consist of up to 4096 characters. The description is required because users do not access the application through ESCM.

For example, if your application has a Web user interface, you could provide some introductory text and the URL of the login page.

The description is displayed at the ESCM user interface and included in the email that is sent to users of services based on your technical service. The description can be specified for multiple languages. Make sure that you always specify the access information at least for the default language, English (`locale=en`). This is also used for any language for which no separate text has been stored.

Attribute:

`locale` - Locale code, for example `en` for English or `en-US` for English - United States.

LocalizedDescription

Contains a description of the item's purpose in the language specified by the `locale` attribute. Can be specified for multiple languages.

`LocalizedDescription` elements are contained in the following elements:

- `TechnicalService`

  At the ESCM user interface, the service description is visible to you as the technology provider and to any supplier, reseller, or broker who sells your service.

- `ParameterDefinition`

  At the ESCM user interface, the parameter description is visible to you as the technology provider, to any supplier, reseller, or broker who sells your service, and to the customers subscribing to the service.

- `Event`

  At the ESCM user interface, the event description is visible to you as the technology provider, to any supplier, reseller, or broker who sells your service, and to the customers subscribing to the service.

- `Role`

  At the ESCM user interface, the role description is visible to you as the technology provider and to any supplier, reseller, or broker who sells your service.

- `Operation`

  At the ESCM user interface, the operation description is visible to you as the technology provider when updating the technical service definition and to customers when they select an operation to be executed for your service.

Attribute:

`locale` - Locale code, for example `en` for English or `en-US` for English - United States.

Example:

```
<LocalizedDescription locale="en">
      Our Service supports you in ...
   </LocalizedDescription>
```

LocalizedLicense

Contains the license agreement for the customer in the language specified by the `locale` attribute. Can be specified in multiple languages.

At the ESCM user interface, the license agreement is visible to you as the technology provider, to any supplier, reseller or broker who sells your service, and to the customers subscribing to your service. It can be changed by suppliers and resellers.

Attribute:

`locale` - Locale code, for example `en` for English or `en-US` for English - United States.

Example:

```
<LocalizedLicense locale="en">
      Please read this software license agreement ... The license terms
      are applied for the concession of the rights ...
   </LocalizedLicense>
```

LocalizedTag

Contains tags (search terms) to be associated with the technical service in the language specified by the `locale` attribute. Can be specified in multiple languages.

At the ESCM administration portal, the tags are visible to you as the technology provider. On a marketplace, customers can use the tags to search for marketable services based on the technical service, provided that the tag display is enabled for the marketplace by the marketplace owner.

You can enter up to five terms, separated by commas. The tags are not case-sensitive. They must not consist of more than 20 characters.

Attribute:

`locale` - Locale code, for example `en` for English or `en-US` for English - United States.

Example:

```
<LocalizedTag locale="en">
      Documentation
   </LocalizedTag>
```

LocalizedName

Contains the name of a service role, operation, or operation parameter in the language specified by the `locale` attribute. Can be specified for multiple languages.

At the ESCM user interface, the role name is visible to you as the technology provider, to any supplier, reseller, or broker who sells your service, and to the customers when they assign users to subscriptions. The operation and operation parameter name is visible to customers when they start to use your service and to you as the technology provider when updating the technical service definition.

Attribute:

`locale` - Locale code, for example `en` for English or `en-US` for English - United States.

ParameterDefinition

Declares a parameter that is passed to your application during instance provisioning.

Suppliers can use parameters to make different values available to customers as different options, for example, different feature configurations or service restrictions. For parameters of type `ENUMERATION`, you can define specific options. For example, for a `MEMORY_STORAGE` parameter that

may take a value of 1 GB, 2 GB, or 4 GB, you could define three options: one for minimum space, one for medium space, and one for maximum space. The options are available to a supplier when defining a price model; a customer can choose between the options when subscribing to a service.

ESCM passes all parameters that you declare in the service definition to your application through the instance provisioning call. Any further processing must be carried out by your application. Especially if parameters are used to impose restrictions on service usage, the application needs to ensure that the restrictions are met. For example, if you declare a `MAX_FILE_NUMBER` parameter that restricts the maximum number of files created by a user, the application needs to track the actual number and ensure that the maximum number is not exceeded.

ESCM offers the following predefined parameters:

Parameter	Description
`NAMED_USER`	This parameter is used to restrict the maximum number of users per subscription. It is relevant for login and user access.
`CONCURRENT_USER`	This parameter is used to restrict the maximum number of concurrent users. It is relevant for login and user access.
`PERIOD`	This parameter is used to define the maximum lifetime of a subscription. As soon as this period has expired, the instances related to the subscription are deactivated. The parameter is relevant for all access types except external access.

Predefined parameters are controlled by ESCM and can be set for any marketable service. No implementation effort is required from your side.

ESCM offers an additional predefined parameter, `VMS_NUMBER`. This parameter is used to store the number of virtual machines booked within the scope of subscriptions to IaaS services. It must be specified in a technical service definition if the number of VMs is to be retrieved.

Attributes:

- **`id`** - ID of the parameter. The ID must be unique across the parameters of a technical service.
- **`valueType`** - Data type of the parameter: `BOOLEAN`, `INTEGER`, `LONG`, `STRING`, `ENUMERATION`, `PWD`. The latter one can be used to encrypt parameter values such as passwords. The values will be encrypted in the database, in reports, and log files, as well as hidden from the user interface.
- **`mandatory`** - Defines whether a value for the parameter must be set for a subscription. Can be set to `true` or `false`.
- **`configurable`** - Defines whether the parameter is visible to suppliers. Can be set to `true` or `false`. If set to `false`, only you can see and modify the parameter. If set to `true`, the supplier can define whether to provide this parameter with different options to customers. Only if the supplier defines the parameter as configurable for his customers, he can define a price for it and the customer can choose an option when subscribing to the service. Irrespective of any setting, the parameter will be sent to the service during the provisioning.
- **`modificationType`** - Defines whether the parameter can be set only at the time a customer subscribes to a service (one-time parameter), or whether it can be set or modified when updating a subscription (standard parameter). Can be set to `ONE_TIME` or `STANDARD`. If set to `ONE_TIME` or left empty, parameter options cannot be changed once a subscription to the marketable service based on this technical service has been created. If a customer upgrades to a service with a parameter with the `modificationType` set to `ONE_TIME`, the parameter options defined by the customer during the initial subscription apply.

If set to `STANDARD`, the parameter options can be changed when a subscription to the marketable service based on this technical service is updated. Irrespective of any setting, the parameter will be sent to the service during the provisioning.

- **`default`** - Defines the default value for the parameter. Values for the data types `INTEGER`, `LONG`, and `BOOLEAN` are evaluated by native Java mechanisms. For boolean values, this means, for example, that all values except `true` or `TRUE` are converted to `false`.
- **`minValue`** - For parameters of type `INTEGER` and `LONG`, defines the minimum value for the parameter.
- **`maxValue`** - For parameters of type `INTEGER` and `LONG`, defines the maximum value for the parameter.
- **`Options`** - For parameters of type `ENUMERATION`, declares the possible options that can be passed to the application for this parameter during instance provisioning. The `Options` element contains various **`Option`** elements that have the following attributes:
  - **`id`** - ID of the option. The ID must be unique across the options of a parameter.
  - **`LocalizedOption`** - Contains a description of the option's purpose in the language specified by the `locale` attribute. Can be specified in multiple languages.

Example:

```
<ParameterDefinition
      configurable="true"
      default="2"
      id="MEMORY_STORAGE"
      mandatory="false"
      valueType="ENUMERATION">

      <Options>
         <Option id="1">
            <LocalizedOption locale="en">Minimum space (1GB)
            </LocalizedOption>
         </Option>
         <Option id="2">
            <LocalizedOption locale="en">Medium space (2GB)
            </LocalizedOption>
         </Option>
         <Option id="3">
            <LocalizedOption locale="en">Maximum space (4GB)
            </LocalizedOption>
         </Option>
      </Options>
   </ParameterDefinition>

   <ParameterDefinition
       configurable="true"
       modificationType="STANDARD"
       default="200"
       id="MAX_FOLDER_NUMBER2"
       mandatory="true"
       maxValue="500"
       minValue="12"
       valueType="INTEGER">
    <LocalizedDescription locale="en">
       Number of folders that can be created.</LocalizedDescription>
   </ParameterDefinition>

 <ParameterDefinition
```

```
      configurable="true"
      id="API_USER_NAME"
      default="User name of HEAT API"
      mandatory="true"
      valueType="STRING"
      modificationType="ONE_TIME">
     <LocalizedDescription locale="en">User name</LocalizedDescription>
   </ParameterDefinition>

<ParameterDefinition
      configurable="true"
      id="API_USER_PWD"
      default="Password for HEAT API"
      mandatory="true"
      valueType="PWD"
      modificationType="ONE_TIME">
     <LocalizedDescription locale="en">Password</LocalizedDescription>
   </ParameterDefinition>

<ParameterDefinition
      configurable="true"
      id="VMS_NUMBER"
      default="0"
      mandatory="true"
      valueType="LONG"
     <LocalizedDescription locale="en">
       Number of VMs of IaaS subscriptions</LocalizedDescription>
   </ParameterDefinition>
```

Event

Declares an event type. The application can notify ESCM about its occurrence. Only required if you implemented the notification of events. For details, refer to *Integrating with ESCM Event Management* on page 22.

Attribute:

id - ID of the event. The ID must be unique across the events of a technical service.

Example:

```
<Event id="FILE_DOWNLOAD">
     <LocalizedDescription locale="en">
        File Download
     </LocalizedDescription>
  </Event>
```

Role

Declares a service role. Service roles provide different types of access to the underlying application. Each service role can be mapped to corresponding access rights or privileges in the application.

Attribute:

id - ID of the service role. The ID must be unique across the service roles of a technical service.

Example:

```
<Role id="ADMIN">
     <LocalizedName locale="en">Administrator</LocalizedName>
     <LocalizedDescription locale="en">
       Administrators have full access to all data entities
```

```
        and can execute administrative tasks such as
        role assignments and user creation.
      </LocalizedDescription>
   </Role>

   <Role id="GUEST">
      <LocalizedName locale="en">Guest</LocalizedName>
      <LocalizedDescription locale="en">
         Guests only have limited read access.
      </LocalizedDescription>
   </Role>
```

Operation

Declares an operation that can be executed on a technical service.

You may wish that your technical service offers additional operations or functions that are to be accessible via the ESCM user interface. In a SaaS environment, applications are not installed locally but provisioned as services. Therefore, users cannot access the system resources the applications are using, for example, to perform administrative tasks such as system backup or shutdown. Technical service operations can be used to access the resources of an application and perform administrative tasks without actually opening the application.

To provide for technical service operations, a Web service based on the operation service API must be implemented.

Attributes:

- **`id`** - ID of the operation. The ID must be unique across the operations of a technical service.
- **`actionURL`** - URL of the WSDL document defining the implemented operation service. The server and port to be used must match the ones where the provisioning service of the application is running. Be aware that for asynchronous provisioning, you need to use the HTTPS protocol. In addition, for ensuring correct communication between ESCM and APP, certificates must have been exchanged, and the server given in the `actionURL` attribute must be specified as the host name whose certificate has been registered. Refer to the APP documentation for details on certificate handling.
- **`OperationParameter`** - For every operation, declares the possible parameters and their values that can be passed to the operation during service provisioning. The `OperationParameter` element has the following attributes:
  - **`id`** - ID of the parameter. The ID must be unique across the operation parameters.
  - **`mandatory`** - Defines whether a value for the operation parameter must be specified. Can be set to `true` or `false`.
  - **`type`** - Defines whether the value for the operation parameter is retrieved from the technical service or from a text string. Can be set to `REQUEST_SELECT` (users can select from a list of options retrieved from the techical service) or `INPUT_STRING` (users can enter a text string for further information).
  - **`LocalizedName`** - Contains the name of the parameter in the language specified by the `locale` attribute. Can be specified in multiple languages.

Example:

```
<Operation id="SNAPSHOT"
      actionURL="https://oscm-app:8881/oscm-app/
                  webservices/oscm-app/oscm-app/
                  org.oscm.app.v2_0.service.AsynchronousOperationProxy?
wsdl">
```

```
      <LocalizedName locale="en">Create snapshot</LocalizedName>
      <LocalizedDescription locale="en">
        Creates a snapshot of the system.
      </LocalizedDescription>
      <OperationParameter
         id="SERVER" mandatory="true" type="REQUEST_SELECT">
         <LocalizedName locale="en">Server</LocalizedName>
      </OperationParameter>
      <OperationParameter
         id="COMMENT" mandatory="false" type="INPUT_STRING">
         <LocalizedName locale="en">Comment</LocalizedName>
      </OperationParameter>
</Operation>
```

Appendix B: Integrating Applications with ESCM Using APP

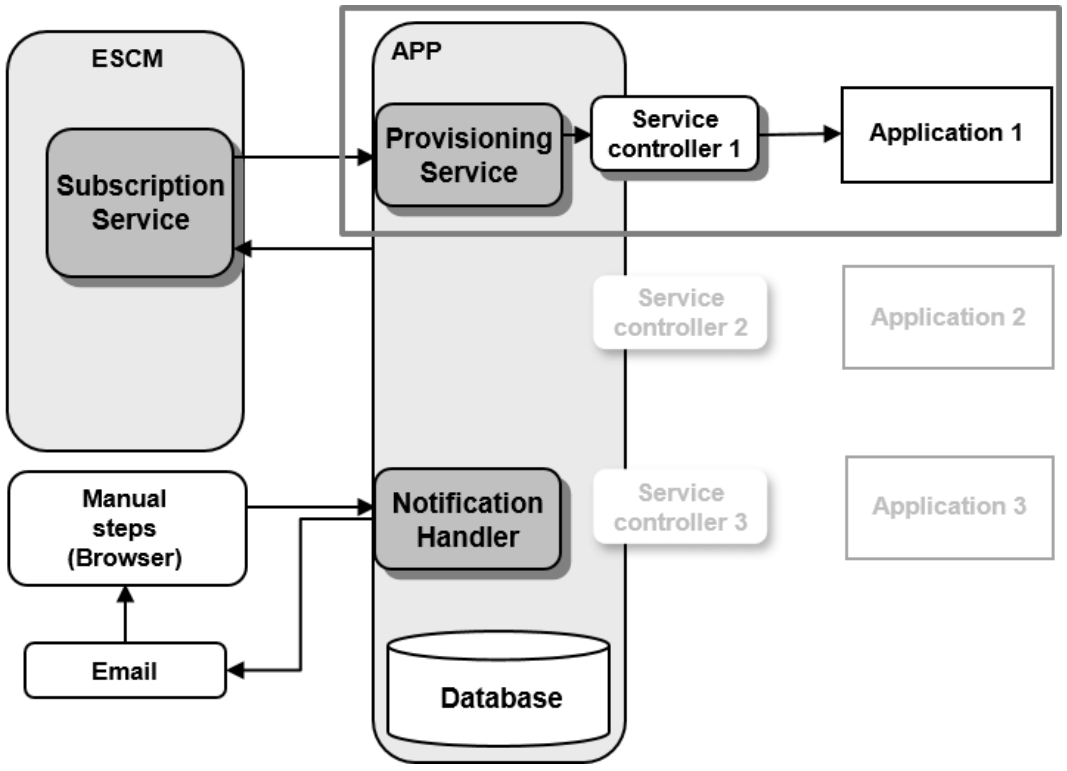
When integrating applications with ESCM, the instance provisioning can be done in two provisioning modes: synchronous or asynchronous mode.

Asynchronous provisioning is required if provisioning operations take a long time because long-running processes or manual steps are involved. This is the case, for example, when provisioning virtual machines on a virtual machine server. ESCM supports the integration of such applications with its asynchronous provisioning platform (APP). This is a framework which provides a provisioning service as well as functions, data persistence, and notification features which are always required for integrating applications in asynchronous mode.

APP also includes the operation service interface for executing technical service operations on the integrated applications from the ESCM user interface.

B.1 Components and Communication Paths

The following figure illustrates the components and communication paths involved when using APP to integrate an application with ESCM.



The following components are involved:

- ESCM with the subscription service handling the provisioning tasks.

- APP including the provisioning service, a database, a notification handler, and an API to enable the communication between the application instances and ESCM.
- A service controller for communication with the application.

  The following ready-to-use service controllers are available, registered, and initialized when deploying the `oscm-app` container:

  - AWS service controller: Can be used for integrating the Amazon Elastic Compute Cloud Web service with ESCM. Refer to the *AWS Integration* guide for details.
  - OpenStack service controller: Can be used for integrating OpenStack services with ESCM. Refer to the *OpenStack Integration* guide for details.
  - VMware service controller: Can be used for integrating VMware vSphere services and using virtual machines provisioned on a VMware vSphere server with ESCM. Refer to the *VMware vSphere Integration* guide for details.
  - Azure service controller: Can be used for integrating Microsoft Azure services with ESCM. Refer to the *Microsoft Azure Integration* guide for details.
- The application with a remote interface and the application instances, if needed.

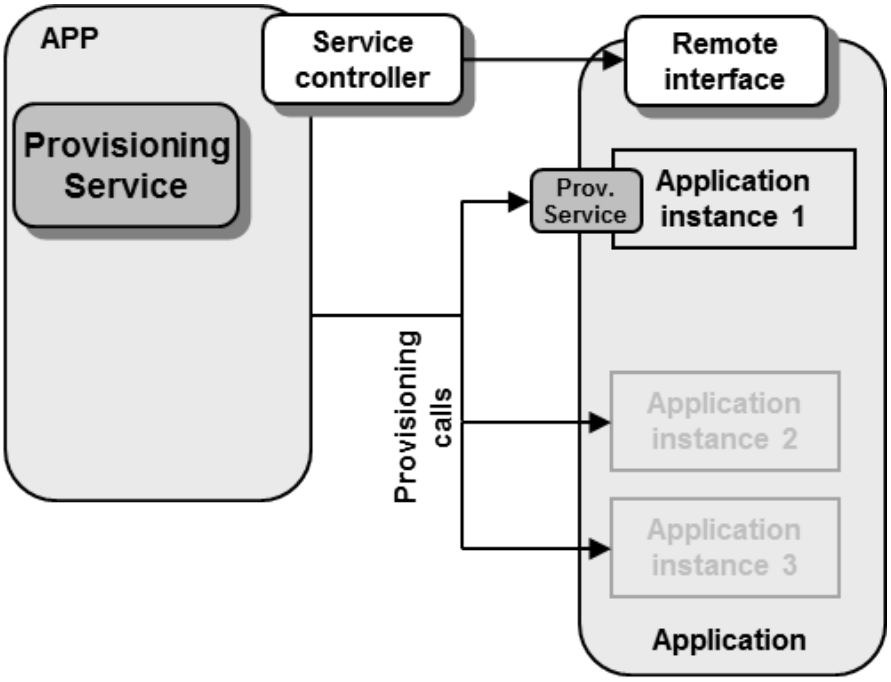
The communication paths are as follows:

- When a subscription is created, deleted, or modified in ESCM, the provisioning service of APP triggers the service controller to execute all the tasks required on the application side.
- It is recommended to divide long-running provisioning operations into several steps using the polling feature of APP. For example, when creating an application instance, the controller could immediately return the instance ID, report the instance status as "not ready", and request APP to poll the status at regular intervals. Only after the instance has actually been set up at the application side, the controller would report its status as "ready" upon the next polling by APP.
- All intermediate artefacts of the tasks are stored in the database of APP.
- When the provisioning operation for a subscription has been successfully completed, ESCM receives an appropriate message, and the status of the subscription is set to `COMPLETE`. When deleting a subscription, ESCM does not expect a response. The task is completed immediately.
- Specific tasks may require manual steps. In this case, the service controller can request the automatic execution to be paused and then send an email notification to the responsible person providing a link to continue the automatic processing. As soon as the link contained in the email is clicked, the task execution is continued.

If the provisioning or modification of an application instance fails or if there are problems in the communication between the participating systems, the corresponding subscription in ESCM remains pending. APP supports you in recovering from such problems with its instance status interface. For details, refer to *Handling Problems in the Provisioning Process* on page 41.

B.2 Provisioning by Instance

The following figure shows details of the communication paths between APP and an application, for example, an OpenStack platform.



Besides communicating with the application's general remote interface via the service controller, APP can forward the provisioning calls directly to the application instance in question.

Each provisioning operation first invokes the service controller, which in turn triggers the relevant actions in the application's general remote interface. When the controller returns with an appropriate status setting, APP calls the provisioning service of the application instance in question. This provisioning service must implement the standard `ProvisioningService` interface and work in synchronous mode.

This feature is useful if not all provisioning operations can be executed via the application's general remote interface. For example, the general remote interface of a virtual machine server allows you to provision virtual machines, but you usually cannot create or manage users within these virtual machines. To do this, you have to address the virtual machine itself.

For proper operation, the application's remote interface needs to support the following network connectivity:

- HTTPS/SOAP requests from APP to the application instances within the application.
- HTTPS/SOAP calls from an application instance to APP (in specific scenarios).

B.3 Service Operations

APP also includes the ESCM operation service interface for executing technical service operations on the integrated applications from the ESCM user interface.

Technical service operations can be used to access the resources of an application and perform administrative tasks without actually opening the application. The operations and the access information of the operation service must be specified in the technical service definition for the application.

APP passes service operation calls from the ESCM user interface directly to the service controller for the underlying application. The service controller is responsible for the execution of the operation in the application.

B.4 Handling Problems in the Provisioning Process

If the provisioning or modification of an application instance fails or if there are problems in the communication between the participating systems, the corresponding subscription in ESCM remains pending.

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

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

1. Work as a technology manager of the technology provider organization which is responsible for the relevant service controller.
2. Invoke the instance status interface of APP for the service controller by opening it's URL in a Web browser.

   The access URL has the following format:

   `https://<host_fqdn>:8881/oscm-app/controller/?controllerid=<controller-ID>`

   `<host_fqdn>` is the fully qualified name or IP address of the host used to access ESCM, `8881` is the port of APP. Omit the port if ESCM is operated with its proxy. The `oscm-app/controller/?controllerid=<controller-ID>` is the default context root of the service controller and cannot be changed. There are the following IDs: `ess.aws`, `ess.openstack`, `ess.vmware`, `ess.azureARM`.

   The Web page shows all subscriptions for the application, including detailed information such as the customer organization, the ID of the related application instance, and the provisioning status.
3. Find the subscription for which you solved the problem in the most recent provisioning, modification, or deprovisioning operation.
4. In the **Action** column, select the action for APP 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 the application does not respond.
   - `UNLOCK` - to remove the lock for an application instance in APP.
   - `DELETE` - to terminate the subscription in ESCM and remove the instance in APP, but keep the actual application instance for later use. For this action, the service manager role is required in addition.
   - `DEPROVISION` - to terminate the subscription in ESCM, remove the instance in APP, and deprovision the application instance. For this action, the service manager role is required in addition.

- `ABORT_PENDING` - to abort a pending provisioning or modification operation in ESCM. ESCM is notified to roll back the changes made for the subscription and return it to its previous state. In the application, no actions are carried out.
- `COMPLETE_PENDING` - to complete a pending provisioning or modification operation in ESCM. ESCM 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..

B.5 Handling Communication Problems Between APP and ESCM

When the communication between APP and ESCM is no longer possible, for example, because ESCM 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 ESCM operator to make sure that ESCM 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 Web interface (base URL) of APP.

   The access URL has the following format:

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

   `<host_fqdn>` is the fully qualified name or IP address of the host used to access ESCM, `8881` is the port of APP. Omit the port if ESCM is operated with its proxy. `oscm-app` is the default context root of APP and cannot be changed.

2. Log in with the ID and password of the user and organization defined by the platform operator in the `BSS_USER_ID` and `BSS_USER_PWD` configuration settings.

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

3. Click **Restart**.

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

B.6 Updating APP and Controller Settings

The APP software requires a number of settings for configuring its container environment. These settings are provided as environment variables in Docker files when deploying the containers. Before the deployment, the platform operator adapted the initial settings to your environment, in particular server names, ports, paths, and user IDs. The platform operator is also responsible for updating the APP settings. Refer to the *Operator's Guide* for details.

You can **view the configuration settings** on the Web interface of APP, where you can also **change the organization that is to be responsible for a service controller**:

1. In a Web browser, access the Web interface (base URL) of APP.

   The access URL has the following format:

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

   `<host_fqdn>` is the fully qualified name or IP address of the host used to access ESCM, `8881` is the port of APP. Omit the port if ESCM is operated with its proxy. `oscm-app` is the default context root of APP and cannot be changed.

2. Log in with the ID and password of the administrator of the technology provider organization specified by the platform operator.

3. Specify another technology provider organization for a service controller, `ess.aws`, `ess.openstack`, `ess.azureARM`, or `ess.vmware`).

   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 ESCM must be changed in the controller configuration settings.

Note: When changing the password for a technology provider responsible for a service controller (`BSS_USER_PWD`), make sure to change it first in the APP Web interface before the technology provider changes it in the ESCM administration portal. Otherwise, he can no longer in to the service controller interface to change it there.

4. Click **Save** to save the settings.

Refer to the *AWS Integration*, *OpenStack Integration*, *Microsoft Azure Integration*, or *VMware vSphere Integration* guides for details on the available service controllers.

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 ESCM 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 ESCM, 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 ESCM to provide their services to customers.

Marketplace Owner

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

Marketplace Manager

A privileged user role within a marketplace owner organization.

OIDC

An authentication mode of ESCM where users are managed and authenticated by means of OpenID Connect in an external system such as Microsoft Azure Active Directory, the so-called OIDC provider.

OIDC Tenant

An entity in ESCM representing a configuration of settings and parameters required to connect to a specific tenant at an OIDC provider, for example, a specific domain and directory in Microsoft Azure Active Directory.

Operator

An organization or person responsible for maintaining and operating ESCM.

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 ESCM; 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. ESCM 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 ESCM. 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 ESCM for offering applications provisioned by technology providers to customers.

Technical Service

The representation of an application in ESCM. 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 ESCM.