FUJITSU

FUJITSU Software Enterprise Service Catalog Manager V19.1

Shell Integration

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 4

1 Introduction 6

1.1 Components Involved in the Shell Integration 6

1.2 Usage Scenarios 7

2 Creating and Publishing Services 8

2.1 Prerequisites and Preparation 8

2.1.1 Preparing Shell Scripts 8

2.1.2 Preparing for Azure Automation 10

2.2 Creating Technical Services 11

2.3 Creating and Publishing Marketable Services 12

2.4 Configuring the Status Information Page 12

3 Administrating the Shell Integration 13

3.1 Handling Problems in the Provisioning Process 13

3.2 Updating Service Controller Settings in the Database 14

3.3 Changing the Responsible Organization 14

3.4 Logging 15

Appendix A Service Parameters and Operations 16

Glossary 19

About this Manual

This manual describes the integration of the Shell service with) FUJITSU Software Enterprise Service Catalog Manager, hereafter referred to as ESCM.

This manual is structured as follows:

Chapter	Description
Introduction on page 6	Provides an overview of the ESCM Shell integration, the components involved, and the supported usage scenarios.
Creating and Publishing Services on page 8	Describes how to prepare and provide Shell scripts, how to create and publish services for the Shell integration, as well as how to configure a status information page for viewing details regarding script execution.
Administrating the Shell Integration on page 13	Describes administration tasks related to the ESCM Shell integration.
Service Parameters and Operations on page 16	Describes the technical service parameters and operations which are supported by the Shell service controller.

Readers of this Manual

This manual is intended for operators who want to execute Shell scripts from services on a marketplace provided by ESCM. It assumes that you have access to an existing ESCM installation that supports Shell execution. In addition, you should have basic knowledge of Shell scripts and you should be familiar with the concepts and administration of ESCM.

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:

APP	Asynchronous Provisioning Platform
ESCM	Enterprise Service Catalog Manager
DBMS	Database Management System
IaaS	Infrastructure as a Service
WSDL	Web Services Description Language
WSIT	Web Services Interoperability Technologies

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

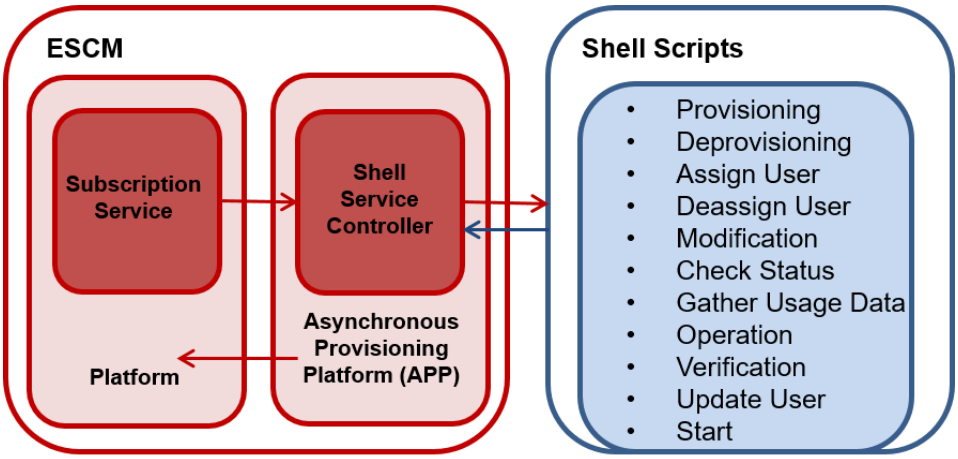
Enterprise Service Catalog Manager (ESCM) is a set of services which provide all business-related functions and features required for turning on-premise applications and tools into 'as a Service' (aaS) offerings and using them in the Cloud. This includes ready-to-use account and subscription management,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

The Shell integration with ESCM allows users to install any software and execute any task by running Shell scripts from services which are published on a marketplace. For example, scripts may be used to provision virtual systems on a cloud platform, or connect ESCM with Microsoft Azure accounts to schedule and execute Azure Automation runbooks built on PowerShell Workflow. The cloud usage costs are calculated and charged by means of the ESCM billing and payment services.

This manual describes the environment and steps involved to execute Shell scripts from services on an ESCM marketplace.

1.1 Components Involved in the Shell Integration

The following picture provides an overview of the main components involved in the integration of ESCM and Shell scripts:



In ESCM, customer subscriptions are managed by means of the **Subscription service**. When a customer creates a subscription or executes specific activities for it, the Subscription service asynchronously triggers the corresponding actions in a Shell through the **Asynchronous Provisioning Platform (APP)** and the **Shell service controller**: Scripts are executed in a Shell.

APP is a framework which provides a provisioning service, an operation service, as well as functions, data persistence, and notification features which are required for integrating applications with ESCM in asynchronous mode. The actual communication with the applications is carried out by service controllers. For each application, a specific and dedicated service controller is required. When APP is deployed (`oscm-app` container), a Shell service controller is also deployed and initialized. It is preconfigured and registered with APP. This service controller can be used for integrating Shell scripts with ESCM. Refer to the *Technology Provider's Guide* for details on APP.

1.2 Usage Scenarios

The ESCM Shell integration supports the following usage scenarios:

- **Provisioning script**: When a customer subscribes to a corresponding service on an ESCM marketplace, the Shell service controller triggers the execution of a provisioning script.
- **Deprovisioning script**: When a customer terminates a subscription, the Shell service controller triggers the execution of a deprovisioning script. The subscription is terminated in ESCM independent of whether the Script execution is successful.
- **Assign user script**: When a customer assigns users to a subscription, the Shell service controller triggers the execution of an assign user script.
- **Deassign user script**: When a customer removes users from a subscription, the Shell service controller triggers the execution of a deassign user script.
- **Update user script**: When a customer updates the data of a user, the Shell service controller triggers the execution of an update user script.
- **Modification script**: When a customer reconfigures an existing subscription to a corresponding service, the Shell service controller triggers the execution of a modification script.
- **Status script**: When a customer opens the **Details** view of a subscription and then clicks the **Custom** tab (e.g. **Status** tab) that has been defined for the underlying marketable service, the Shell service controller triggers the execution of a status script.
- **Operation script**: When a customer selects an operation for his subscription, the Shell service controller triggers the execution of an operation script.
- **Verification script**: When a customer subscribes to a corresponding service, the Shell service controller triggers the execution of a verification script, if it is defined and not empty. In case the verification script returns an error, the subscription process is aborted and the user is informed by a corresponding message. Note that the execution of a verification script is executed in synchronous mode; all other scripts are executed asynchronously.
- **Start script**: When the `oscm-app` container is started, the Shell service controller triggers the execution of a script named `start.sh`, if this exists in the `<docker>/config/oscm-app/scripts` directory. You can use this script, for example, to install additional software or execute remote commands at start-up.
- **Usage data script**: The Shell service controller triggers the execution of a script for collecting events that occurred due to the usage of cloud resources.

2 Creating and Publishing Services

The following sections describe how to create and publish services in ESCM by means of which customers can execute Shell scripts.

2.1 Prerequisites and Preparation

The following prerequisites must be fulfilled before you can create and publish services in ESCM.

- To create technical services for the Shell integration in ESCM, you must have access to ESCM as a technology manager. You must be a member of the technology provider organization responsible for the Shell service controller as specified in the configuration settings for the installation.
- To create marketable services for the Shell integration in ESCM, you must have access to ESCM as a service manager of an organization with the supplier role. This may be the same organization as the technology provider organization or a different one.
- To publish your marketable services, you must have access to an appropriate marketplace in your service manager role.

2.1.1 Preparing Shell Scripts

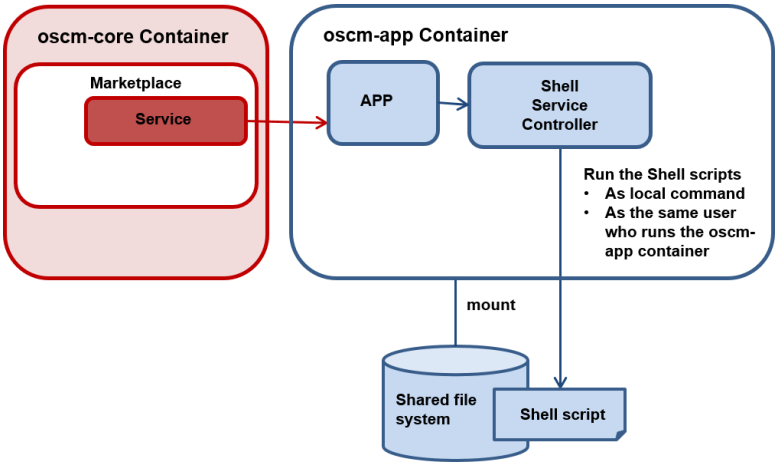
- The Shell scripts to be executed by the Shell service controller must exist. They need to be prepared by the technology provider organization responsible for the Shell service controller and are referenced from the marketable services defined by the supplier organization.
- The scripts may be located on the Docker host where ESCM is deployed, or on an external system (external URL).

  Scripts located on the ESCM Docker host need to be provided in the following directory:

  `<docker>/config/oscm-app/scripts`

  `<docker>` is the ESCM data directory (shared file system) specified at installation time.

  Whenever you provide new scripts or change existing ones, they are automatically made available within the `oscm-app` container in the `/opt/scripts` directory, which is mounted to the `<docker>/config/oscm-app/scripts` directory.



- The Shell service controller will execute the scripts stored on the shared file system as the same UNIX user who runs the `oscm-app` container.

Shell Script Syntax and Rules

In order to execute properly, the Shell scripts must follow some rules.

Script Types

The Shell controller supports standard UNIX shell (`.sh`) scripts as well as PowerShell scripts (`.ps1`) scripts.

Return JSON Object

Each script must return **exactly one JSON object** followed by an `"END_OF_SCRIPT"` string value. In this way, the Shell service controller can determine that a script has finished successfully. Otherwise the script execution will run into a timeout.

The JSON result may contain the following **keywords and elements**:

- `status` (mandatory) - allowed values: `"ok"` or `"error"`
- `message` (mandatory) - information for users about the script execution.
- `usageData` (optional) - expected as the result of the execution of a usage data script. It contains the following JSON keywords and elements:
  - `eventId` - the ID of a billable event as defined in the technical service definition. Sample service definitions with events are available on GitHub under *https://github.com/servicecatalog/oscm-app-shell/tree/master/src/main/resources*.
  - `multiplier` - the number of occurrences of an event. Instead of recording the same event each time it occurs, you can record it only once and set the multiplier to the number of occurrences. The billing services consider the multiplier. For example, one event with a multiplier of 2 is handled in the same way as 2 events of the same type, if both events occur in the same billing period.
- `data` (optional) - any data which can be output in the status information of a subscription after the script has been executed. See *Configuring the Status Information Page* on page 12 for details on how to configure the status information for subscriptions. You can include the following JSON keywords and elements in a `data` element:
  - `output` - any data to be presented, for example, HTML data for the status script.
  - `accessInfo` - access information to be passed to the subscription details for display.
  - `name` - any temporary data.
  - `id` - any ID.
- `parameters` (optional) - any parameters with their values that are to be persisted for later use by other scripts. A `parameters` element may contain any number of parameters defined as key/value pairs:
  - `key` - the key of the parameter to be persisted.
  - `value` - the value of the parameter to be persisted.

  Other scripts can reference a parameter by `$SCRIPT_<key>`, where `<key>` is the parameter's key. For example, a provisioning script may return the ID of a new virtual machine in the `VMID` parameter. This virtual machine can later be addressed in other scripts (update, operation, etc.) using `$SCRIPT_VMID`.

Special Characters

Special characters used in scripts must be properly escaped.

Sample Scripts

Sample scripts are available on GitHub under *https://github.com/servicecatalog/oscm-app-shell/tree/master/src/main/resources/sample_scripts*.

2.1.2 Preparing for Azure Automation

The Shell controller allows customers to connect to Microsoft Azure from ESCM and manage their resources by scheduling and executing Azure Automation runbooks. For this to be possible, preparations need to be taken in Microsoft Azure and ESCM.

Preparations in Microsoft Azure

In order to connect to Microsoft Azure, a valid service principal (application) is required in the context of a customer's Microsoft Azure subscription whose resources are to be managed from ESCM.

The customer needs to take the following steps:

1. Log in to Microsoft Azure as a user with administrative permissions for the subscription.
2. In the **Azure Active Directory** under **App registrations**, create a new registration with a name of your choice.

   Example: `ESCMServicePrincipal`

   Select supported account types as desired. A redirect URI is not required.
3. Select the new application and copy its **(client) ID** and the **directory (tenant) ID** for later use in ESCM.

   Example:

   - Application (client) ID: `ef29bb22-369c-424d-9e72-6800ad24239e`
   - Directory (tenant) ID: `02d72930-3874-4119-80d4-54a4d7f31dfc`
4. Under **Certificates and secrets**, add a **new client secret** for the application. Copy the new secret for later use in ESCM. Be aware that the client secret is no longer accessible in Microsoft Azure once you close the page.
5. Select the Microsoft Azure subscription for which you created the application. From its properties, copy the **subscription ID** for later use in ESCM.

   Example:

   Subscription ID: `98e0f7ef-bb83-408d-9675-0e45f65a54f7`
6. Under **Access control (IAM)** for the subscription, choose **Add role assignment** and add the **Contributor** role to the application (service principal) you created.

Preparations in ESCM

In ESCM, custom attributes need to be created for specifying the authentication information required to connect to Microsoft Azure. The customer administrators later need to provide their values for these custom attributes in their organization's profile on the marketplace.

Proceed as follows:

1. Log in to the ESCM administration portal with your service manager account.
2. Use **Manage custom attributes** in the **Account** menu to create custom attributes for customers with the following **attribute IDs**:
   - `AzureSubscriptionId` - the ID of the customer's Microsoft Azure subscription

- `AzureTenantId` - the customer's directory (tenant) ID in Microsoft Azure
- `AzureApplicationId` - the ID of the application (service principal) created for ESCM in Microsoft Azure
- `AzureSecret` - the client secret created for the application in Microsoft Azure

For each of the attributes, specify a name of your choice as well as `ess.shell` as the **controller ID**.

3. Make sure that your customers specify the appropriate values for the custom attributes in their organization's profile on the marketplace.

The custom attributes are referenced by their ID in the PowerShell scripts to be executed in Microsoft Azure through the Shell controller. Examples are available on GitHub under *https://github.com/servicecatalog/oscm-app-shell/tree/master/src/main/resources/sample_scripts/powershell*.

In addition to the custom attributes, the following parameters should be defined in the technical services created for the Shell integration to work with Microsoft Azure:

- `AzureAutomationAccountName`
- `AzureResourceGroupName`
- `AzureRunbookName`

These parameters can be used in the scripts for working with Azure Automation resources. For details, refer to *Creating Technical Services* on page 11 and *Service Parameters and Operations* on page 16.

2.2 Creating Technical Services

The first step in providing ESCM services for Shell script execution is to create one or more technical services.

Proceed as follows:

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

   As a basis, you can use the technical service sample provided on *https://github.com/servicecatalog/oscm-app-shell/tree/master/src/main/resources*.

   A technical service specifies Shell scripts as parameter options. You need to make them available as described in *Preparing Shell Scripts* on page 8.

   In the technical service definition, be sure to specify:

   - The asynchronous provisioning type
   - The USER access type
   - Service parameters which correspond to the parameters specified in the Shell scripts. For details on the supported service parameters, refer to *Service Parameters and Operations* on page 16.

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

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

2.3 Creating and Publishing Marketable Services

As soon as the technical services for the Shell integration exist in ESCM, you can define and publish marketable services based on them. Your cost calculation for the services should include any external costs for operating any provisioned system, etc..

Proceed as follows:

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

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

2.4 Configuring the Status Information Page

For checking the status of a script execution, you can add a custom tab to the **Details** view for subscriptions on a marketplace when defining a marketable service:

1. Make sure that a script for checking the status of provisioned instances is available. Refer to the sample script *`status.sh`* for details.
2. When creating or editing a marketable service for Shell script execution, specify the **URL of a custom tab**. This URL points to a Web page or Web application the content of which will be retrieved and shown on the custom tab. Using a custom tab, you can provide the information needed for accessing, for example, provisioned instances, such as IP addresses, network information, or user credentials.

   A predefined custom tab, `serverInformation.xhtml`, is available for viewing the output of a status script, e.g. `status.sh`.

   Set the URL for the custom tab in the following format:

   `https://<host_fqdn>:8881/<service controller name>/serverInformation.jsf`

   `<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. `<service controller name>` is the default context root of the relevant service controller and cannot be changed.

   For example:

   `https://myhost:8881/oscm-app-shell/serverInformation.jsf`
3. Enter a **Name of the custom tab**, for example, `Status`. Do not enter `Details` because the first tab is named that way.
4. Enter the name of the status script in the **Retrieve status of provisioned instance. Absolute filesystem path or URL to script file** field of the service parameters. It is assumed that the script is located in the `/opt/scripts/` folder of your Docker host where the `oscm-app` container has been deployed.
5. Save the marketable service definition.
6. Activate the marketable service, log in to the marketplace, and create a subscription to the service.
7. Under **My Subscriptions**, you can now see the **Status** tab next to the **Details** tab.

For details on defining marketable services, refer to the *Supplier's Guide* and to the online help of ESCM.

3 Administrating the Shell Integration

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

3.1 Handling Problems in the Provisioning Process

If there are problems in the communication between the participating systems, the corresponding subscription in ESCM remains pending. The Shell service controller informs the technology managers of its responsible technology provider organization by email of any incomplete operation in the Shell script execution.

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

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

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

   The access URL has the following format:

   `https://<host_fqdn>:8881/oscm-app/controller/?controllerid=ess.shell`

   `<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/controller/?controllerid=ess.shell` is the default context root of the service controller and cannot be changed.

   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 operation.
4. In the **Action** column, select the action for the Shell integration components to execute next. Possible actions are the following:
   - `RESUME` - to resume the processing of a provisioning operation in APP which was suspended.
   - `SUSPEND` - to suspend the processing of a provisioning operation in APP, for example, when a Shell script does not respond.
   - `UNLOCK` - to remove the lock for an instance in APP.
   - `DELETE` - to terminate the subscription in ESCM and remove the instance in APP, but keep a virtual system for later use. The service manager role is required for this action.
   - `DEPROVISION` - to terminate the subscription in ESCM, remove the instance in APP, and delete any virtual system. The service manager role is required for this action.
   - `ABORT_PENDING` - to abort a pending operation in ESCM. ESCM is notified to roll back the changes made for the subscription and return it to its previous state.
   - `COMPLETE_PENDING` - to complete a pending 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.

3.2 Updating Service Controller Settings in the Database

During the deployment of ESCM, several configuration settings are written to the `bssapp` database. This configuration is used for the initial setup of the Shell service controller and its registration with APP. The platform operator is responsible for taking care that the initial settings are correct. Refer to the *Operator's Guide* for details on the initial configuration in the `var.env` configuration file.

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

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

The controller configuration settings are evaluated as follows:

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

To change the user responsible for the Shell service controller, the operator needs to set corresponding environment variables. Refer to the *Operator's Guide* for details.

3.3 Changing the Responsible Organization

You can change the technology provider organization responsible for the Shell service controller using the Web interface of APP:

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 specified for `BSS_USER_KEY` in the configuration settings for APP or as another administrator of the same organization.

3. Specify the technology provider organization for the Shell service controller, `ess.shell`.
4. Save the settings.
5. Make sure that the configuration settings for the Shell service controller are updated.

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

3.4 Logging

The Shell integration software records detailed information regarding script execution in the following log file inside the `oscm-app` container in the following folder:

`/opt/apache-tomee/logs/app-shell.log`

The logging is based on `log4j`.

Note: Be aware that this log file is available only if you set the `TOMEE_DEBUG` configuration setting in the `var.env` configuration file to `true`.

To view the log file, log in to the `oscm-app` container as follows:

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

The default log level is `INFO`.

Appendix A: Service Parameters and Operations

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

You find sample services on GitHub under *https://github.com/servicecatalog/oscm-app-shell/tree/master/src/main/resources*.

Service Parameters

The Shell service controller supports the parameters below.

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

APP_CONTROLLER_ID

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

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

PROVISIONING_SCRIPT

Mandatory. The absolute file system path or URL to a Shell script file. The base folder is `/opt/scripts`. This script will be executed when a user subscribes to a service on the marketplace.

Example: `/provisioning.sh`

DEPROVISIONING_SCRIPT

Mandatory. The absolute file system path or URL to a Shell script file. The base folder is `/opt/scripts`. This script will be executed when a user terminates the subscription.

Example: `/deprovisioning.sh`

UPDATE_SCRIPT

Mandatory. The absolute file system path or URL to a Shell script file. The base folder is `/opt/scripts`. This script will be executed when the configuration of an existing subscription is changed.

Example: `/update.sh`

ASSIGN_USER_SCRIPT

Mandatory. The absolute file system path or URL to a Shell script file. The base folder is `/opt/scripts`. This script will be executed when a user is assigned to a subscription.

Example: `/assign_user.sh`

DEASSIGN_USER_SCRIPT

Mandatory. The absolute file system path or URL to a Shell script file. The base folder is `/opt/scripts`. This script will be executed when a user is removed from a subscription.

Example: `/deassign_user.sh`

CHECK_STATUS_SCRIPT

Optional. The absolute file system path or URL to a Shell script file. The base folder is `/opt/scripts`. This script will be executed when a user opens the **Details** view under **My Subscriptions** and selects the **Status** tab. It retrieves the status of a provisioned instance.

Example: `/status.sh`

SCRIPT_TIMEOUT_SECONDS

Optional. The number of seconds until the execution of a running script will be canceled.

Default: `600`

USAGEDATA_SCRIPT

Optional. The absolute file system path or URL to a Shell script file. The base folder is `/opt/scripts`. This script is automatically executed, by default, on a daily basis if the `APP_TIMER_REFRESH_USAGEDATA` parameter is set accordingly for APP. The script gathers the number of billable events defined in the technical service definition for calculating cloud usage costs.

Example: `/usage_data.sh`

TECHNICAL_SERVICE_ID

Optional. The technical service identifier as defined in the current technical service definition in the `tns:TechnicalService` element. This ID is used for generating events for usage data collection.

Default: `Shell`

<freely definable service parameter>

Optional. Any number of parameters that are mapped from the parameters defined in the Shell script files. For each parameter in the script file, a corresponding parameter must be specified in the technical service definition.

All service parameters are patched into the script file at the top of the file. For example, a service parameter called `MY_PARAM` can be used as `$MY_PARAM` in the script file.

Parameters are used for passing data for script execution, for example, number of allowed transactions, number of bookable CPUs, etc.

OPERATIONS_SCRIPT

Optional. The absolute file system path or URL to a Shell script file defining operations that can be executed for a subscription, for example, starting or stopping a provisioned VM. The base folder is `/opt/scripts`. This script will be executed when a user selects an operation for his subscription.

Example: `/operation.sh`

UPDATE_USER_SCRIPT

Optional. The absolute file system path or URL to a Shell script file. The base folder is `/opt/scripts`. This script will be executed when the data of a user is changed.

Example: `/update.sh`

VERIFICATION_SCRIPT

Optional. The absolute file system path or URL to a Shell script file. The base folder is `/opt/scripts`. If specified, this script will be executed BEFORE any other script. This is useful, for example, for checking whether a specific instance already exists, whether parameters are passed correctely, etc.

Example: `/verification.sh`

Service Parameters for Microsoft Azure Integration

The following parameters can be defined in technical services used for connecting to Microsoft Azure and executing Azure Automation runbooks. The parameters can be referenced in the corresponding PowerShell scripts. Examples are available on GitHub under *https://github.com/servicecatalog/oscm-app-shell/blob/master/src/main/resources/sample_scripts/powershell/*.

AzureAutomationAccountName

Optional. The name of the Azure Automation account to be used.

Example: `MyAutoAccount`

AzureResourceGroupName

Optional. The name of a resource group in Microsoft Azure to work with.

Example: `MyResourceGroup`

AzureRunbookName

Optional. The name of the Azure Automation runbook to execute.

Example: `MyAzureRunbook`

Service Operations for Instances Provisioned by a Shell Script

The Shell service controller supports the definition and execution of service operations for instances provisioned using a Shell script. The operations are defined in the script referenced by the `OPERATIONS_SCRIPT` service parameter.

The `actionURL` for each operation is:

```
http://oscm-app:8880/oscm-app/webservices/oscm-app/oscm-app
         /org.oscm.app.v2_0.service.AsynchronousOperationProxy?wsdl
```

`<oscm-app>` and `<8880>` are the server and port of the container where the Shell service controller is deployed.

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.