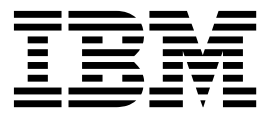


IBM Cloud Orchestrator
Version 2.5.0.4

Content Development Guide



Note

Before using this information and the product it supports, read the information in Notices.

This edition applies to version 2, release 5, fix pack 4 of IBM Cloud Orchestrator (program number 5725-H28) and to all subsequent releases and modifications until otherwise indicated in new editions.

The material in this document is an excerpt from the IBM Cloud Orchestrator knowledge center and is provided for convenience. This document should be used in conjunction with the knowledge center.

© **Copyright IBM Corporation 2013, 2017.**

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Preface	v
Who should read this information	v

Chapter 1. Getting started with IBM Cloud Orchestrator content	1
Content packs	1
Virtual images	1

Chapter 2. IBM Cloud Orchestrator overview	3
Architecture	3

Chapter 3. Installing and configuring the IBM Process Designer	5
---	----------

Chapter 4. Developing toolkit and application content	7
Implementing workflow orchestration	7
Managing business processes	7
Process application	8
Toolkit	8
Snapshot	8
Business Process Definition	9
Service	9
Participant group	10
Coach validation framework	10
Implementing extensions with IBM Cloud Orchestrator	11
Implementing a custom operation for IBM Cloud Orchestrator	11
Start a custom extension operation	12
Using instance data provided by OperationContext	13
Implementation of a human service as a UI extension	13
Implementation a Business Process Manager process as an extension	14
Best practices and guidelines	15
Importing and exporting toolkits and process applications	17
Importing toolkits	17
Exporting toolkits	18
Business Process Manager security	18

Chapter 5. Importing by using the command-line interface	19
---	-----------

Chapter 6. Example scenarios for workflow-based orchestration	21
Self-service offerings scenario	21

Chapter 7. Automating the creation of categories, offerings, and instance actions	25
Exporting by using the catalogTool.sh tool	28

Chapter 8. Publishing IBM Cloud Orchestrator content to IBM Cloud Orchestrator Catalog	29
---	-----------

Appendix A. Base orchestrator toolkit	31
Business objects	32
OperationContext	32
Service Instance	33
VirtualMachine	34
NetworkInterface	34
Services	35
General system services	35
Integration services	37
Cloud management services	44
Samples	45

Appendix B. Scripting utilities toolkit	47
File upload restrictions	47
Public Cloud Gateway restrictions	47
Business objects	49
Implementation services	51
Sample processes	57
Sample UIs	57

Appendix C. Support IaaS toolkit	59
Generic IaaS REST call	60
OpenStack and Cloud API library in Business Process Manager	61
Get IaaS Regions	61
Samples for the IaaS support toolkit	62
Sample to create a tenant in OpenStack identity service	63
Sample to get a list of servers from OpenStack compute service	63
Troubleshooting	64

Appendix D. Email notification toolkit	65
Business objects	65
Implementation services	66
Sample processes	67
Sample UI	67

Appendix E. OpenStack Cinder Storage Volumes toolkit	69
Installing and configuring	69
Toolkit scenarios	69
Creating storage volume	70
Registering scripts for storage volumes	70

Attaching script package to an image.	71
Unregistering scripts for storage volume.	72
Attaching volumes	72
Detaching volumes	74
Deleting volumes	74
Managing volumes related to a virtual system instance	75
Toolkit developer's reference.	76
Environmental variables	76
Business objects	77
Human services	78
Coach views	79
Business processes	79
Integration services.	80
Samples about how to use the available services and views	83
Troubleshooting	83
Appendix F. OpenStack Services toolkit	85
Configuring	85
Toolkit scenarios.	87
Deploying a Linux virtual machine	87
Deploying a Linux virtual machine with approval and notification.	88
Deploying a Windows virtual machine	89
Deploying a Windows virtual machine with approval and notification.	89

Deploying a LAMP stack using multitiered technology.	90
--	----

Accessibility features for IBM Cloud Orchestrator	93
--	-----------

Notices	95
Programming interface information	97
Trademarks	97
Terms and conditions for product documentation.	97
IBM Online Privacy Statement	98

Glossary	99
A.	99
B	100
C	100
E	100
H	100
K	100
P	101
R	102
S	103
T	103
V	104

Preface

This publication documents how to author content for IBM® Cloud Orchestrator.

Who should read this information

This information is intended for users and cloud administrators who build content by using the IBM Cloud Orchestrator platform for their internal and wider community use.

Chapter 1. Getting started with IBM Cloud Orchestrator content

IBM Cloud Orchestrator can be extended by providing different types of content packages to enable IBM Cloud Orchestrator to use the features that are delivered by external software and infrastructure devices.

The different types of IBM Cloud Orchestrator content that you can create and deploy in the cloud are explained here.

Content packs

Content Packs provide the building blocks for orchestrations that run on the integrated Business Process Manager platform in IBM Cloud Orchestrator.

Content Packs typically consist of a set of process applications and toolkits. Process applications contain ready-to-use orchestrations and share library items from one or more toolkits. Toolkits typically contain building blocks to be used by process applications and other toolkits. They also provide sample processes and human services to demonstrate the usage of those building blocks.

IBM Cloud Orchestrator ships with a number of toolkits that provide building blocks that are related to pattern and image deployment, scripting support, and offerings. These building blocks can be used to create new toolkits or process applications. Refer to Chapter 4, “Developing toolkit and application content,” on page 7.

You can download extra content packs from the IBM Cloud Orchestrator Catalog.

Processes and human services that are contained in process applications or toolkits can then be configured to be run in IBM Cloud Orchestrator in the following way:

Self-service offerings

A self-service offering is a process that is not related to any pattern or instance. For example, add a user. To understand how to create a self-service operation, see the self-service offerings scenario. The offerings are grouped into categories.

For more information about self-service offerings, see the following topics:

- Designing self-service
- Using self-service
- Self-service offerings

Virtual images

Virtual images provide the operating system and product binary files that are required to create a virtual system instance. Images can be extended to customize the virtual images and the operating systems by adding script packages that provide extra functions during provisioning time and software bundles that enhance the content of the image.

Software bundles

Software bundles contain and describe the software that is available for use within a virtual image. They include information about how to install the software, prerequisites of the software, and parameters that are available for customizing the software. They combine the operating system definition and custom software bundles to create virtual bundles that can be provisioned in the cloud.

Chapter 2. IBM Cloud Orchestrator overview

IBM Cloud Orchestrator provides end-to-end service deployment across infrastructure and operating system layers. It provides integrated IT capabilities for orchestration workflow automation and IT governance, resource monitoring, and cost management.

Your data center policies, process, and infrastructure can be integrated with the cloud in a consistent, flexible, and automated way, across various IT domains. You can define and implement business rules and IT policies with an intuitive graphical tool. It helps you to connect the aspects of the different domains into a consistent orchestration of automated tasks and human tasks.

Architecture

IBM Cloud Orchestrator is built on OpenStack and common pattern modeling. Business Process Manager provides workflow modeling and runtime capabilities. It is integrated so that event triggered automation provides the basis for cloud-centric workflow enablement. The contents, toolkits, and other building blocks can be found, used, adapted, and shared in the IBM Cloud Orchestrator Catalog. For more information about the IBM Cloud Orchestrator architecture, see Product architecture

Cloud Provisioning and Orchestration Development Collaboration Community

To evaluate future versions of this product, join the Development Collaboration Community. From the Cloud Provisioning and Orchestration Development Collaboration Community, you can use the hosted beta system and download the beta code. The hosted system has the latest evaluation code. If you want to evaluate in your own environment, download the beta code.

You can learn how to develop orchestration and other content in this community. To access the community, go to <https://www.ibm.com/developerworks/mydeveloperworks/groups/service/html/communityview?communityUuid=e5a54efe-3c9f-491b-af2a-e5400516b5aa>.

Chapter 3. Installing and configuring the IBM Process Designer

You must install and configure the IBM Process Designer, one of the user interfaces for Business Process Manager to start with IBM Cloud Orchestrator content authoring.

Before you begin

This procedure describes the IBM Process Designer installation and configuration on a Windows operating system.

Procedure

1. Install and configure IBM Cloud Orchestrator. Business Process Manager is installed as a product component. For more information, see [Installing](#).
2. Install IBM Process Designer on the Windows operating system on which you perform content development. See [Installing IBM Process Designer](#) in the Business Process Manager knowledge center.
3. Configure IBM Process Designer to connect with Business Process Manager:
 - a. Edit C:\Windows\system32\drivers\etc\hosts.
 - b. On a new line, add the IP address and the host name of the IBM Cloud Orchestrator Server.
 - c. Save the file.
4. Start Process Designer to develop content.

Chapter 4. Developing toolkit and application content

You can develop toolkit and application content.

Implementing workflow orchestration

Workflow orchestration is a Business Process Manager based extension to IBM Cloud Orchestrator which consists of UI panels to collect extra data.

Workflow orchestration is optionally implemented by Business Process Manager human service and a Business Process Manager business process, which define the logical flow of activities of the extension. The extension itself consists of the human service and the business process, which are developed with the standard Business Process Manager designer. After development, it is registered as an extension operation in the Self-Service Offerings of IBM Cloud Orchestrator. After you register it, it is available in IBM Cloud Orchestrator.

The Business Process Manager-based extension type is:

Self-service offerings

A custom operation that is run in the context of the data center. These operations are administrative operations that are used to automate the configuration and user operations. Self-service offerings are used to enhance the catalog of available services with more features. They are classified into different categories. For more information about self-service offerings, see the following topics:

- Self-service offerings
- Designing self-service
- Using self-service

Managing business processes

IBM Business Process Manager Standard is a comprehensive Business Process Management offering. It gives you visibility and insight to manage business processes. It offers tools and runtime environments for process design, execution, monitoring and optimization, and basic system-integration support.

The process application is the fundamental container for business processes and their components in Business Process Manager. It has a name and a unique tag (acronym) that can have a maximum length of 7 characters. Business Process Manager process applications and their related artifacts are stored within a repository, which is hosted and managed by the process center. Process applications can be created either from the web interface of the process center console or by using the Business Process Manager process designer tool.

Important: If you are a new user to Business Process Manager process designer, verify whether you are granted access to the process center repository. The access can be granted either by using the Business Process Manager process designer tools or the Business Process Manager process center console. If you are an administrator user, you can grant rights to other users:

1. Go to the **Admin** tab > **Manage Users**.
2. Click **Add Users/Groups**.

3. In **Add Users and Groups** dialog, enter the first few letters of the user as a search filter in the **Search for Name** panel.
4. Select the user and click **Add selected**.

The user is granted access to the Process Center repository and can use the Process Designer.

Process application

A process application can contain one or more Business Process Definitions, which represent a model, or template, of a business process that is run. The process application consists of the following definitions and activities:

- Input and output variable definitions
- Business object data definitions
- User interface panel definitions
- Activities of the process that include the flow

The flow defines how the various activities of the process are orchestrated. When a process is started, an instance of the process is created from the process definition template.

Toolkit

A toolkit is a container for Business Process Manager artifacts like Business Process Definitions, Business Objects, Human Services, Coaches, and Business Process Definitions. Toolkits are used to package reusable Business Process Manager artifacts as package content that can be used to build other Business Process Definitions. But in contrast to a Process Application, a toolkit cannot be deployed to the Business Process Manager runtime environment as a stand-alone solution. Instead, the artifacts that are contained can be used as building blocks by other Process Applications, but must be declared as a dependency on the toolkit. Multiple Process Applications can have a dependency on the same toolkit and toolkits can have also dependencies on each other. Toolkits and Process Applications can be exported as a .twx file. This file is transferred to other systems that run the Business Process Manager Process Center, or it is published to the IBM Cloud Orchestrator Catalog. You can download from the IBM Cloud Orchestrator Catalog and then import it into another Process Center. In IBM Cloud Orchestrator, do not use the option to export a toolkit or Process Application to a Business Process Modeling Notation (BPMN) file.

Snapshot

The status of a Process Application or a toolkit can be captured by the Process Designer tool or the web console of the Process Center. This snapshot defines a specific version of the toolkit. It consists of all the related artifacts and their content or state at the time when the snapshot is taken. The tip level is the most recent version of a Process Application or toolkit. As a prerequisite, define a snapshot of a Process Application for most of the management actions like export Process application or toolkit, archive, define a dependency on a toolkit.

Business Process Definition

A Business Process Definition is a reusable model of a process that defines what is common to all runtime instances of that process model. Processes are graphically modeled in the Process Designer tool, which captures the flow of steps (activities). This graphical diagram represents activities in the process as boxes and lines connect them to describe the flow of work. Each box represents an activity and is labeled with its description in the diagram. A Business Process Definition must contain a start event, an end event, at least one lane, and one or more activities. Lanes are used to segment the drawing canvas of a Business Process Definition into one or more horizontal sections. For example, when an activity (Business Process Manager artifact) is added to the process diagram, its vertical position determines the lane to which it belongs. Because each lane can be associated with a role, you can observe the association of an activity of a Business Process with the role. For each Business Process Definition that you create, you must declare variables to capture the business data that is passed from activity to activity in your process. Variables can be input to the process (input variables), output from a completed process (output variables), or local variable (private variables).

Related information:

 [Business process definitions \(BPDs\)](#)

Service

The activities that are contained in the Business Process Definition describe the overall flow of the process, but it does not capture the implementation of the process. Services are reusable implementations of activities in a Business Process Definition. When a Business Process Definition starts, the services carry out the required functions.

The following types of services are available in Business Process Manager:

Integration service

Use an integration service when you want to integrate with an external system. An integration service is the only type of service that can contain a Java or web service or REST integration.

General system service

Use a general system service when you must coordinate other nested services or when you must manipulate variable data. General System services cannot include Java, web service, or REST integrations directly.

Human service

Use a human Service when you want an interactive service with UI panels (Coaches). Human Services generate human tasks in the Business Process Manager Process Portal.

Ajax service

Use an Ajax service when you want to include a control in a Coach to implement dynamic data selection. For example, you can automatically populate lists and complete edit boxes. An Ajax service can pull data dynamically from a connected data source such as a database.

Decision service

Use a decision service when you want a condition to determine the implementation.

Web service

Use a web service to implement a web service in Business Process Manager.

Related information:

 [Service types](#)

Participant group

A participant group is used to represent a set of users in your organization or enterprise locally within an application. Participant groups can be found in the Processes category within Business Process Manager Process Designer. The members of a Participant Group are local to the Business Process Application in which it is defined. In contrast, the Security Groups provide a global mechanism to define groups.

Remember: A participant group is a logical group that does not exist outside of the concept of the Business Process Application in which it is defined. After a Business Process Application is installed on a Process Server in a different environment, you must add or remove users in those groups. For this management, use the Business Process Manager Process Admin Console.

Coach validation framework

Business Process Manager includes a server-side validation framework for validating coach data that is entered during run time. This feature simplifies the validation logic that identifies invalid user input and sends notifications during run time. Server script or a service defines the validation logic. Any service, other than Ajax service or a human service can be used as a validation service. Multiple coaches in a human service flow can use the same validation service or server script. The `coachValidation` of type `CoachValidation` business object is a new system variable that contains information about validation errors.

Fire validation

The fire validation, when added to a link, connects a boundary event to the next step to indicate whether the validation must be started for this boundary event.

In a human service diagram, go to the **Properties > Behavior** of a coach. Select **Before or Never** in the **Fire Validation** field. On selection of **Before**, the validation is done before moving to the next step.

Stay on page

The Stay On Page event in a human service flow causes the flow to return to the originating coach. For example, if there is a validation error in the input, the focus is returned to the originating coach to correct the input.

Server script to define validation

Specify validation logic in the **Properties > Implementation** for the server script. Add the error data to the system variable `coachValidation` by using an API. For example, `tw.system.addCoachValidationErrorMessage`.

It takes three parameters. The first parameter is the system variable `coachValidation`. The second parameter is a string that contains full path to the data item that contains the error. The third parameter is the message that must be displayed when there is an error. If the invalid data is not bound to any coach view, then the validation error is not displayed.

Implementing extensions with IBM Cloud Orchestrator

You can build custom extension operations that are based on business processes and human services, for UI extensions, by using the Business Process Manager Process Designer tool. The business logic of the extension is implemented by the Business Process Definition. The association between a UI extension that is based on a human service and business process definition is done during the registration of the extension operation in the instance actions or self-service offerings.

IBM Cloud Orchestrator delivers a Business Process Manager toolkit that is called **SCOrchestrator_Toolkit**. It provides many useful artifacts such as business processes, services, human services, coaches, business object definitions, java classes that assist you to build custom extensions operations for IBM Cloud Orchestrator. Additionally, the SCOrchestrator_Toolkit contains samples that show how to build custom extensions. Other Business Process Manager toolkits are downloadable from the IBM Cloud Orchestrator Catalog. They provide extra content for different areas (networking, storage) that can be used to build IBM Cloud Orchestrator extensions.

Restriction: If a process application or a toolkit in Business Process Manager has more than one defined snapshot, then only the artifacts at the top level can be used to define a new extension in the instance actions or self-service offerings. The artifacts below the tip level are not considered.

Implementing a custom operation for IBM Cloud Orchestrator

The following steps are needed to develop a custom extension operation with or without a user interface for IBM Cloud Orchestrator:

Before you begin

Make sure that the SCOrchestrator_Toolkit is imported into the Business Process Manager Process Center repository. For more information, see Importing toolkits from the Process Center console.

Procedure

1. Create a Business Process Manager Process Application as a container for your extension operation. For information about managing process applications, see Creating new process applications.
2. In the Business Process Manager Process Designer, open your Business Process Application and declare a dependency on the SCOrchestrator_Toolkit.

Note: The toolkit contains useful building blocks for writing extensions.

3. In the toolkits category, click + and add the dependency to the SCOrchestrator_Toolkit.
4. In the same manner, import and add dependencies to other toolkits as needed.
5. Create business process definitions and human services (extensions) for IBM Cloud Orchestrator.

Note: You can use the building blocks available in the SCOrchestrator_Toolkit. To use a building block that is provided by a toolkit, do the following tasks:

- Drag the corresponding building block from the toolkit to the process or service canvas in the Process Designer.
- Connect it with the other steps or activities of the business process or service.

- Map the input and output variables of the building block to the variables of the embedding business process or service.
- 6. Create a snapshot of your business process application. For information about the steps, see [Creating snapshots in the Process Center console](#).
- 7. Register the extension operation in the IBM Cloud Orchestrator UI. For more information about custom operation registration, see [Designing self-service](#).

Results

You can now use the extension operation in IBM Cloud Orchestrator.

Related tasks:

Chapter 8, “Publishing IBM Cloud Orchestrator content to IBM Cloud Orchestrator Catalog,” on page 29

IBM Cloud Orchestrator Catalog is a platform and one-stop-shop for IBM Cloud Orchestrator customers, partners, and IBM employees. The content that you create can be submitted internally or externally from the IBM Cloud Orchestrator Catalog. The primary goal is to have an environment where developers, partners, and IBM Service teams share content with one another. This web-based application is envisaged to federate content from several repositories into a single cloud automation content catalog.

Start a custom extension operation

Starting a custom extension operation is done with or without a user interface. Start the operation and the following activities are automatically run in the specified sequence:

1. During the following conditions, a task with an operation context is created in the task engine with the status as ‘New’:
 - A self-service offering or user action is triggered from within the IBM Cloud Orchestrator UI.
2. If a human service is not registered in the instance actions or self-service offerings for this operation, then this step is skipped.
 - Human service starts with the coach UI panels.
 - It is passed to the operation context ID (task ID) as parameter.
 - The first activity of the human service is to run a REST call against IBM Cloud Orchestrator to get the operation context business object. It is based on the operation context ID that is given as input parameter.
 - The coach gathers the parameters from the user and store the parameters into the output business object of the human service.
 - When all user input information is collected, the last activity in the Business Process Manager human service (ReturnParameters) is run to persist the output business object into the operation context of the task engine. These business objects can be used later by the extension Business Process Manager Business Process. The status of the task in the task engine is set ‘Queued’ so that the task can continue to run.
3. The task engine starts the Business Process Manager process, which gets the operation context as input parameter. For self-service offering and instance actions, the process starts directly after the human service ends. The Business Process Manager process defines an input business object, which contains the parameters that are collected by the user interface of the related human service. The parameter section of the operation context also contains the parameters that are collected by the graphical user interface of the human service.

Using instance data provided by OperationContext

To support the orchestration development, IBM Cloud Orchestrator provides an OperationContext object containing useful information. When using instance actions, the OperationContext object contains information about the instances on which the operation is executed.

There are two types of instance actions: instance actions that are executed on a single instance, and instance actions that are executed on multiple instances (for example, when multiple instances are selected in the resource view).

For instance actions, as part of the OperationContext object, one or multiple instance IDs are passed to the Business Process Manager human service and process. They can be found in OC.parm.instances.

An instance ID has the following format:

`https://<ico_server>/orchestrator/v2/instancetype/<type>/instances/<providerInstanceId>`

where <providerInstanceId> is resource type specific:

- For OpenStack Nova: <region>--<openstackid>
- For OpenStack Heat: <region>--<stackname>--<openstackid>
- For Azure cloud service: <region>--<servicename>
- For Azure deployment: <region>--<servicename>--<deploymentname>--<slot>

The following integration services are provided to extract the IDs and update the instances parameter:

- “GetInstanceIdFromOperationContext” on page 38
- “GetMultipleInstanceIdsFromOperationContext” on page 39
- “UpdateInstancesParmInOperationContext” on page 43

Implementation of a human service as a UI extension

A Business Process Manager human service that can be used as UI extension for IBM Cloud Orchestrator is developed with Business Process Manager Process Designer. Observe the following guidelines:

- Make the human service available in Business Process Manager as URL to the participant group, which is used in IBM Cloud Orchestrator. This activity can be done in the Process Designer as follows:
 1. Go to the **Overview** tab of the design view.
 2. In the Exposing section, select **All Users** in **Expose to Start**.
 3. Register the process in the instance actions or self-service offerings:
 - a. In the IBM Cloud Orchestrator UI, go to **CONFIGURATION > Self-Service Catalog**.
 - b. Create a registry entry for the process you created.
 - c. You can request the new service from self-service offerings with the UI.

The participant group controls the visibility of the extension operation in the instance actions or self-service offerings for the user or user group. It decides which users or user groups are allowed to run the extension.

- The business object defines the interface of the human service and the related business process. This business object contains the parameters to be returned by the human service. The human service acts as an UI extension for IBM Cloud Orchestrator, which are the input parameters of the related business process.

Therefore, a human service defines an output variable with name `outputParameterObject`. It is of type business object that is defined by the creator of the extension.

- Declare operation context ID as an input variable of type string. The human service gets the operation context ID as input variable. For example, `operationContextId`. As first activity of the human service (before it can access the operation context), it must use the activity `GetOperationContext` that is delivered with the `SCOrchestrator_Toolkit`. This activity retrieves the operation context by using the operation context ID.
- Define a UI coach or a sequence of coaches to collect the data that is needed from the user.
- As a last activity, the human service calls the `ReturnParameters` integration service of `SCOrchestrator_Toolkit`. Here, the data that is collected from the user is registered on the operation context. This data is then passed as an input parameter to the associated business process.

Related concepts:

“Coach validation framework” on page 10

Coach validation framework

Implementation a Business Process Manager process as an extension

A business process that can be used as extension operation for IBM Cloud Orchestrator is developed by using Process Designer. It must observe the following guidelines:

- The Process Designer must be made available in the Business Process Manager as URL to the participant group. This group uses it in IBM Cloud Orchestrator. The participant group controls the user or user group visibility for the extension operation. It is made visible in the configurations to be registered as extension operation. It also decides which users or user groups are allowed to run the extension.
- Every Business Process has `inputParameterObject`, which is an input parameter. The type of this variable is same as the output parameter of the associated human service. The first activity of the Business Process is `GetInputParameter` (part of `SCOrchestrator_Toolkit`), which populates the `inputParameterObject`.
- The business process gets the operation context object as an input parameter and must define a corresponding input variable of type `OperationContext`. It is a predefined business object definition that is delivered by `SCOrchestrator_Toolkit`.
- With activities, the business process can access the information that is contained in the operation context. Also, the information is handed over as parameter from the associated human service. It can also use REST calls to retrieve further information from IBM Cloud Orchestrator. REST calls are implementations that are provided by `SCOrchestrator_Toolkit`.

Best practices and guidelines

When you create toolkits or Process Applications, there are some best practices to be followed in naming conventions, structuring, modeling, and error handling.

Guidelines for naming and documenting your toolkit or process application

When you create toolkits, use the following naming conventions:

- Name the toolkit after the utility or services it provides.
- Add words like "Toolkit" or "Framework" so that you can differentiate it from other process applications.
- Avoid long names. You must use fewer than 64 characters.
- White spaces between words can be added if it improves readability.
- Avoid the version number in the name, unless you want to bring attention to the major changes in the solution.
- Add more information about the toolkit in **Description** field.
- Choose an acronym for your toolkit. Do not use the prefix "IC" as it is used for content that is delivered by IBM.
- Name your snapshots according to this scheme: AABB_YYYYMMDD. Exported TWX archives of your toolkit get this snapshot name appended, so you can easily identify the exported snapshots later.

AA The IBM Cloud Orchestrator release that is prerequisite for the toolkit or process application, for example, 25 for IBM Cloud Orchestrator V2.5.

BB Counting up the version of the toolkit, for example. 00 for the first release, and 01 for the second release

YYYYMMDD

The date the snapshot was created

- When updating an existing process application or toolkit, do not change the chosen acronym because it is used to reference the processes in self-service offerings.

Guidelines for creating artifacts in a toolkit

The general best practices are as follows:

- In the documentation field for a Business Process Manager artifact, enter a description of the input and output parameters of that artifact.
- Use the note object of Business Process Manager to improve the readability of complex processes and human services.
- As mentioned in the naming conventions, provide an understandable and meaningful name for your artifacts.
- Keep the interface definition between a Business Process Manager Human Service and its associated Business Process Manager Business Process Definition as short as possible. The interface is defined by a Business Process Manager Business Object. This object is used to correlate a business process with its associated human services in the IBM Cloud Orchestrator Self-service user interface. Use the Business Process Manager human service only to collect the parameters that are needed by its associated business process. Implement the business logic in the business process. It also helps if you enable the business process to be called by using the REST API from an external application, such as a portal application.

- Avoid **Pre** and **Post** execution assignments. Instead, add explicit activities, if needed. The execution assignments are hidden in the Business Process Manager Process Designer, and the logic of the corresponding activity or service becomes difficult to understand. If needed, use the **Pre** and **Post** executions to make simple assignments like initializing the associated Business Process Manager artifact. For example, consider having two consecutive coaches in a human service. In such cases, do not initialize the objects that are used by the second coach as being **Post** execution assignment of the first coach. If needed, do the initialization as a preexecution assignment of the second coach.
- Do not use passwords in environment variables or other artifacts that are visible to everyone.
- When you deliver a solution for IBM Cloud Orchestrator, make sure that there are no validation errors. These errors can be seen in the Process Designer.
- Avoid changing the interface of a building block that is delivered as a part of a toolkit. If you change the interface of building blocks in a toolkit, it becomes cumbersome for all its dependent toolkits or Process Applications. Even changing the name might lead to redoing the mapping for all activities or services that use the building block.

Guidelines to structure your solution

- In general, an extension content solution for IBM Cloud Orchestrator consists of a Business Process Manager Process Application and a Business Process Manager Toolkit.

The basic rule is that a process application contains artifacts that are ready to be used by the user and not meant to be changed or adapted to be useful. All other artifacts are better placed in a toolkit.

- When structuring your solution, always consider the visibility of your artifacts. Artifacts of one process application are not visible by default to another process application.

For example, a Business Process Manager, process A, can be called by another Business Process Manager, process B. The 'Linked Process' activity is used if both are in the same process application or if process A is in a dependant toolkit.

Avoid cyclic dependencies, that is, when toolkit A depends on toolkit B, avoid having a dependency on toolkit A. If such a cyclic dependency occurs, restructure your toolkits to resolve it.

- Use Business Process Manager tags and smart folders to structure your solution to make it more understandable. If you have UI parts that can be used in UI panels, define them as Coach views. These views can be reused in different Coaches. If you must change something later, for example, wording, you change only the reusable Coach view.

Guidelines for handling errors


An IBM developerWorks article explains extensively about exception handling and logging from a business process management perspective. See Related Links. It identifies the types of exceptions that are encountered in a Business Process Manager scenario. Also, it shows you how to handle them using IBM Business Process Manager.

The following are best practices in error handling:

- Define error message as localization resources.
- Raise errors in your integration services or processes by using the Error End Event node.


- Catch raised errors that are raised from integration services by using Intermediate Error Events or Event Subprocesses.
- For Java classes that are used in Business Process Manager processes or human services, define logging framework. For example, `java.util.logging` to log messages to the WebSphere log.
- Use the logging capabilities of Business Process Manager to log messages to the WebSphere log. A good practice is to log in the entry and exit of an activity to support debugging better.

Related information:

 <http://www.scribd.com/doc/92505753/IBM-Impact-2011-Five-Guidelines-to-Better-Process-Modeling-for-Execution-Stuart-and-Zahn>

You can find many documents with guidelines and best practices about business process modeling. One of it is *Five Guidelines to Better Business Process Modeling for Execution* from Jonas A. Zahn and Stuart Jones, which describes the following design guidelines:

- Rule of Seven - limit any view to no more than seven steps for a good fit.
- Activity granularity - activities must be similar in scope at each level. Avoid the String of Pearls pattern, that is, series of activities in the same lane.
- Activity description - use [action verb] + [business object] and avoid vague verbs like 'process' and 'perform'.

 http://www.ibm.com/developerworks/websphere/library/techarticles/1105_ragava/1105_ragava.html

Importing and exporting toolkits and process applications

You can create toolkits or reuse toolkits that are shared by other content developers. To use a toolkit that is created and shared by other content providers, use the import toolkit feature. You can use export and import utilities to move the toolkit from development process server to production process server or between any two process servers.

Importing toolkits

You can import toolkits into the Business Process Manager.

Before you begin

Resolve all the dependencies and then proceed with toolkit synchronization.

Procedure

1. In the **Business Process Manager Designer**, click **Process Center**.
2. In **Toolkits**, click **Import Toolkit**.
3. In **Import Toolkit**, click **Browse** to select the file to import.

Tip: The Toolkit file extension is in `twx` extension format.

4. Click **Ok**.

Related information:

Chapter 5, “Importing by using the command-line interface,” on page 19

Exporting toolkits

You can export a toolkit to a Business Process Manager export file.

Before you begin

Resolve all the dependencies and then proceed with toolkit synchronization.

Procedure

1. Select the **Toolkits** tab.
2. Click the toolkit that you want to export from the list of toolkits.
3. Find the snapshot that you want to export.
4. If a snapshot does not exist, click **Create New Snapshot**.
5. Click the **Export** option for the snapshot.
6. Select BPMN export (.zip) or IBM BPM export (.twx) and click **Export**.
7. Locate the directory to which you want to save the exported file.
8. Name and save the file. The exported file can be imported into any Process Center repository.

Business Process Manager security

For a class of use cases, it is important to understand the security context of a Business Process Manager process or user interface. The activity-based user authentication and authorization are described here.

Business Process Manager user interfaces are run as the currently logged in user and project. They are used as the initial dialog for collecting data in self-service offerings and in instance actions. They are also used for assignments in the My Inbox panel. Retrieve the currently logged in user with this JavaScript (R) expression in Business Process Manager: `tw.system.user_loginName`.

Activities in the system lane of a business process are run in the background. Therefore, there is no currently logged in user. By default, activities are run as an admin user and project. The admin user is `tw_admin` in Business Process Manager.

Additionally, there are use cases where it is important to know which user, domain, and project submitted a self-service offering or orchestration action. This information is found in the variables `user`, `domain`, and `project` of the operation context.

For instance, the `GenericRESTCall` integration service of the `SCOrchestrator_Toolkit` extracts the authentication information from the security context by default. The service then uses the extracted information to construct the security authentication token (simple token) for REST calls to external systems. Because the `tw_admin` is an internal user, `GenericRESTCall` translates `tw_admin` to the default admin user (generally `admin`) for external calls. It is desirable to run these REST calls on behalf of the person who submitted the request. To do that, map the `user`, `domain`, and `project` variables from the operation context to the `GenericRESTCall`.

Chapter 5. Importing by using the command-line interface

You can import IBM Cloud Orchestrator content using the command-line interface.

The following tool supports the import of content by using command-line interfaces:

- “Import content pack into Business Process Manager”

Important: You must have a command shell to run these commands.

Import content pack into Business Process Manager

A Business Process Manager-based extension to IBM Cloud Orchestrator consists of UI panels to collect more data. To import a content pack in to a Business Process Manager, start the command-line interface in interactive mode and then run the import command.

To start the command-line interface in interactive mode, use this syntax - `wsadmin.sh -h <hostname> -user <username> -password <password> -lang jython`. To Import content pack, use this syntax - `>>>AdminTask.BPMImport('[-inputFile <file path>]')`.

The `wsadmin.sh` is in the Business Process Manager server installation directory and its parameters are as follows:

-h <host> or --host

Specifies the host name or IP address of the Business Process Manager. This parameter is required.

-user

Specifies the user ID to authenticate to the Business Process Manager. Use the same user ID and password that you use to log on to the tool. This parameter is optional.

--password

Specifies the password that is used to authenticate to the user. This parameter is optional and is case-sensitive.

```
wsadmin.sh -user admin -password admin -lang jython
wsadmin>AdminTask.BPMImport('[-inputFile /tmp/abc.twx]')
```

For more information about Business Process Manager command-line interface, see `BPMImport` command.

Chapter 6. Example scenarios for workflow-based orchestration

Workflows are driven by the following self-service offering trigger.

A quick start guide is available in the IBM Cloud Orchestrator Catalog for content development. Search for this guide in the IBM Cloud Orchestrator Catalog by using the search filter SCOrchestrator Content QuickStart Guide. For more information about searching and downloading from the IBM Cloud Orchestrator Catalog, see Chapter 8, “Publishing IBM Cloud Orchestrator content to IBM Cloud Orchestrator Catalog,” on page 29.

Self-service offerings scenario

Self-service offerings are custom operations that can be run in the context of the data center. Such offerings can be used to automate configuration tasks or to enhance the catalog of the available services with extra features. This scenario depicts a self-service offering for registering the firewall host device on IBM Cloud Orchestrator by storing the firewall details in the storehouse.

Procedure

1. In the Process Designer, click **Create a New ProcessApp** or **Toolkit** and name it RegisterFirewall.
2. Open the RegisterFirewall application or toolkit in the Process Designer.
3. Create a dependency on the SCO_Orchestrator_Toolkit and the SCOrchestrator_Scripting_Uilities_Toolkit.
 - a. Click the plus sign (+) to the right of **Toolkit**.
 - b. From the **Add dependency** dialog, select **SCO_Orchestrator_Toolkit**.
 - c. Select the latest snapshot.
 - d. Repeat the procedure for **SCOrchestrator_Scripting_Uilities_Toolkit**.
4. Create a human service.
 - a. Click the plus sign (+) to the right of **User Interface**.
 - b. Add new human service and name it Register_Firewall_Host_HS. This human service includes a coach that collects the required input.
 - c. Click the **Variables** tab.

Table 1. Input Variables

Variable	Type	Description
operationContextId	String	The ID contains the operation context object identifier, and it is passed during the start of the operation by the IBM Cloud Orchestrator framework. The identifier is used to fetch the operationContext, which is an object that contains all the data that is related to the execution of an operation. Note: This variable is mandatory for all human services that are run through IBM Cloud Orchestrator.
outputParameterObject	FirewallHost	The output parameter is of type FirewallHost business object that has register firewall host-related parameters such as host IP address, user name, and password.

Table 2. Default values of outputParameterObject

Property	Value
hostIpAddress	""
username	"root"
password	""
commandLine	"iptables -I INPUT -p tcp --dport 80 -j ACCEPT; service iptables save "

- d. Create a business object and name it FirewallHost. Add the following parameters to this business object:

Table 3. Output Variables

Variable	Description
hostIpAddress	IP address of the firewall host.
username	Log in user on the firewall host.
password	Password for the user on the firewall host.
commandLine	Command line to configure firewall.

- e. Create the following coaches:

Table 4. Coach

Coach	Input fields
firewall_host_coach	hostIpAddress
	username
	password
	commandLine

- Map the parameters from the outputParameterObject to these coach fields.
- f. Use the ReturnParameters integration service from the SCOrchestrator_Toolkit before the 'end' activity.
 - 1) Click SCOrchestrator_Toolkit to expand it.
 - 2) Click **Implementation**.
 - 3) Drag and drop the **ReturnParameters** integration service.
 - g. Map the operationContextId and outputParamaterObject variable as an input to this service.
 - 1) Click **ReturnParameters**.
 - 2) Click **local.input**.
 - 3) Map local.input.operationContextId to operationContextId, Local.Input.outputParamaterObject to outputParamaterObject.
 - h. Connect the activities to complete the human service.
5. Create a **Business Process Definition**.
- a. Click the plus sign (+) to the right of **Processes**.
 - b. Create a **Business Process Definition** and name it as Register_Firewall_Host_BPD.
 - c. Click **Variables** tab
 - d. Click **Add Input** and create the following input variables:

Table 5. Input Variables

Variable	Type	Description
operationContext	OperationContext	The operationContext object contains all the data that is related to the execution of an operation.
inputParameterObject	FirewallHost	The input parameter is of type FirewallHost.

Note: Both the input variables are mandatory for all business processes that are run from IBM Cloud Orchestrator.

- e. Add the integration service GetInputParameter from SCOrchestrator_Toolkit as first activity in the business process. Complete the data-mapping for this service:

Input Mapping:
 tw.local.operationContext -> operationContext
 tw.local.inputParameterObject -> inputParameterObject_in

Output Mapping:
 inputParameterObject -> tw.local.inputParameterObject
 - f. Add the integration service SSH_ExecuteCommand from SCOrchestrator_Scripting_Uilities_Toolkit. Complete the data-mapping for this service:

Input Map:
 tw.local.inputParameterObject.hostIpAddress -> remoteMachine
 tw.local.inputParameterObject.username -> username
 tw.local.inputParameterObject.password -> password
 tw.local.inputParameterObject.comandLine -> commandLine
 - g. Connect the activities to complete the business process.
6. Expose the human service to the participant group.
- a. Go to the **Overview** tab of the human service.
 - b. Select all users for **Exposed to** for the option under **Exposing**.

- c. Select **Expose as URL (Available from a URL)**.
7. Expose the business process to the participant group.
 - a. Go to **Overview** tab for the business process.
 - b. Select all users for **Expose to Start** to start option under **Exposing**.
8. Register the operation in the IBM Cloud Orchestrator UI.
 - a. Log in to the IBM Cloud Orchestrator UI.
 - b. Click **CONFIGURATION > Self-Service Catalog**.
 - c. Click **Offerings** and then click **Create Offering** in the **Actions** menu.
 - d. Enter the following details:
 - **Offerings name** - enter the name of the new offering.
 - **Description** - enter the description.
 - **Offering icon** - select the icon from the list.
 - **Process** - select the Register_Firewall_Host_BPD from the list. It is the Business Process Definition that is created in the Process Designer.
 - **User Interface** - select Register_Firewall_Host_HS. The human service is created in the Business Process Designer.
 - e. Click **Create**.
9. Run the Self-Service Offering.
 - a. Go to **Self-Service**.
 - b. Click **Category**.
 - c. Click **Offering**.
 - d. Enter the details and click **OK**.
10. After the operation completes successfully, check that the firewall setting command is run on the host.

Related information:



Managing and using toolkits

Information about how to include integration service, process, coach, and human service inside the toolkit library.

Chapter 7. Automating the creation of categories, offerings, and instance actions

You can automate the creation of categories, offerings, and instance actions by using the `catalogTool.sh` tool.

Before you begin

The `catalogTool.sh` tool is installed together with IBM Cloud Orchestrator. The installation directory of the tool is `/opt/ibm/ico/ccs/catalog`.

The tool selects the JVM in the following order:

1. The JVM used by the Business Process Manager, if it exists
2. The JVM defined in `PATH` variable, if there is any
3. The JVM in the `JAVA_HOME` variable, if it has been defined

The `catalogTool.sh` tool is installed on the Business Process Manager node.

To automate the creation of categories, offerings, and instance actions, you need the following items:

- A configuration file that describes the IBM Cloud Orchestrator environment
- An XML file that represents the resources to be created
- Valid credentials to connect to the IBM Cloud Orchestrator. Admin role is required.

Note: Although, in this topic, the script is `catalogTool.sh`, if you are installing on a Windows operating system, the script is `catalogTool.bat`.

About this task

The following directories contain the following items:

conf

- `log4j.properties`, logs, and trace configuration
- `env.properties`, the configuration file for IBM Cloud Orchestrator (IP address)

samples

Various XML files that can be used as examples

lib Resources that are required by the script

logs Log files

To compile your own XML file, start from one of the provided samples and create your `<category>`, `<offering>`, `<instance-type>`, and `<instance-action>` elements.

For a **category**, specify the name, for example:

```
<category>
  <name>A test category</name>
  <description>An optional description</description>
  <icon>Web</icon>
</category>
```

The description and icon fields are optional. For icon, the following values are allowed:

Cloud Cloud Category Icon

Application

Application Category Icon

Backup

Backup Category Icon

Client Client Category Icon

Configuration

Configuration Category Icon

Job Job Category Icon

Network

Network Category Icon

Server Server Category Icon

Storage

Storage Category Icon

Virtual Machine

Virtual Machine Category Icon

Web Web Category Icon

For **offerings**, specify the following parameters:

```
<offering>
  <name>A test offering</name>
  <description>An optional description</description>
  <icon>Web</icon>
  <category-name>Reference a category by name here</category-name>
  <process toolkit-name="toolkit_name">
    <name>Enter the name of the process you want to link to the offering</name>
  </process>
  <user-interface toolkit-name="toolkit_name">
    <name>Enter the name of the user interface you want to link to the process</name>
  </user-interface>
</offering>
```

You can use the same labels that are displayed in the UI when creating a new self-service offering. The description and icon fields are optional. For icon, the same fields that are described for category are allowed here too.

The attribute **toolkit-name** for the <process> and <user-interface> elements is optional and can be specified to state which toolkit contains the process/user-interface.

For **instance actions**, specify the following parameters:

```
<instance-action selection-type="multiple">
  <name>An instance action</name>
  <description>An optional description</description>
  <icon>Icon</icon>
  <instance-type>Resource type to which you can apply the action</instance-type>
  <process toolkit-name="toolkit_name">
  <user-interface toolkit-name="toolkit_name">
    <name>Enter the name of the process you want to link to the useraction</name>
  </user-interface>
  </process>
  <tags>You can mark the instance with an unlimited amount of tags to filter actions.
```



```

    <tag>Any_tagA</tag>
    <tag>Any_tagB</tag>
  </tags>
</instance-action>

```

The selection-type field can have a value of either "multiple" or "single" depending on if the action is applicable to only one instance or to several instances. The default is "single."

For **instance type**, specify the following parameters:

```

<instance-type>
  <name>A_unique_instance_type_name</name>
  <Display name>The name as it appears in the UI</description>
  <icon>Icon</icon>
  <Provider>The provider</instance-type>
  <details-view>
    <toolkit-name>Toolkit_Name</toolkit-name>
    <user-interface>UI display view of type of resource</user-interface>
  </details-view>
  <keyfields>
    <keyfield>
      <attribute>Instance type attribute</attribute>
      <header>Header of the column that displays the corresponding attribute in the resource list
    </keyfield>
  </keyfields>
  <tags>You can mark the instance with an unlimited amount of tags to filter actions.
    <tag>Any_tagA</tag>
    <tag>Any_tagB</tag>
  </tags>
</instance-type>

```

Keyfields are column headers in the UI. Detail-view is the details view panel of the UI.

Note: If you want to import XML files that are related to version 2.4 toolkits, ensure that your XML has the version attribute set to 2.4 in the catalog tag, for example <catalog version="2.4">.

The overall structure of the XML file is:

```

<catalog>
  <automation-categories>
    your categories here
  </automation-categories>

  <offerings>
    your offerings here
  </offerings>

  <instance-types>
    your instance types here
  </instance-types>

  <instance-actions>
    your instance actions here
  </instance-actions>
</catalog>

```

To create the resources that are described in <yourfile>.xml in the IBM Cloud Orchestrator Self-Service catalog, run the following command:

```
./catalogTool.sh <yourfile>.xml <IC0 username> <IC0 username password>
```

where

- <yourfile>.xml is your catalog file to create your catalog resources
- <ICO username> in a IBM Cloud Orchestrator administrative user name
- <ICO username password> is the password for the user who is specified previously

Note: Being an administrative tool, the tool must not be run in concurrent mode. You cannot run an instance of the tool and catalog administration from the IBM Cloud Orchestrator.

What to do next

Note: The catalog version must NOT be set for version 2.3 XML files.

An extra step is required if the tool is installed as separate component from IBM Cloud Orchestrator, for example, the tool is purchased and downloaded from the IBM Cloud Orchestrator Catalog and installed on a desktop. You must edit the tool configuration file, which is at `conf/env.properties`, with the following content:

```
IWD_HOST = <IP or HOSTNAME of Cloud Orchestrator>
IWD_PROTOCOL = https
IWD_PORT = 8443
```

IWD_PROTOCOL = https and IWD_PORT = 8443 are the default values that you must use.

Exporting by using the `catalogTool.sh` tool

You can export all offerings, categories, instance types, and instance actions to XML using the `catalogTool.sh` tool.

About this task

Use the following commands to export:

```
./catalogTool.sh --export <xmlfile> <ico-user> <ico-password>
```

To export everything.

```
./catalogTool.sh --export <xmlfile> <ico-user> <ico-password>
```

```
--includeCategories=<category_1>,<category_2>
```

```
--includeInstanceTypes=<instanceType_1>,<instanceType_2>
```

Exports all the indicated categories and related offerings. Exports the instance types and the related instance actions.

```
./catalogTool.sh --export <xmlfile> <ico-user> <ico-password>
```

```
--excludeCategories=<category_1>,<category_2>
```

```
--excludeInstanceTypes=<instanceType_1>,<instanceType_2>
```

Exports everything except the indicated categories and related offerings.
Exports everything except the instance types and the related instance actions.

--includeCategories and --includeInstanceTypes can be used either together or alone. The same applies to --excludeCategories and --excludeInstanceTypes.

Chapter 8. Publishing IBM Cloud Orchestrator content to IBM Cloud Orchestrator Catalog

IBM Cloud Orchestrator Catalog is a platform and one-stop-shop for IBM Cloud Orchestrator customers, partners, and IBM employees. The content that you create can be submitted internally or externally from the IBM Cloud Orchestrator Catalog. The primary goal is to have an environment where developers, partners, and IBM Service teams share content with one another. This web-based application is envisaged to federate content from several repositories into a single cloud automation content catalog.

About this task

To download content from the IBM Cloud Orchestrator Catalog, access the IBM Cloud Marketplace and search for a solution.

To upload content to the IBM Cloud Orchestrator Catalog, access the Ready for Cloud and Smarter Infrastructure site and follow the validation process.

For additional information about content development, see the Content Developers Corner wiki.

Appendix A. Base orchestrator toolkit

The Base orchestrator toolkit (from now on, called `SCOrchestrator_Toolkit`) is delivered as a part of IBM Cloud Orchestrator. It provides the essential Business Process Manager building blocks, which are needed to build Business Process Manager business processes and human tasks, which are used as extensions for IBM Cloud Orchestrator.

If you want to write an extension for IBM Cloud Orchestrator, you must perform the following tasks:

- Ensure that the toolkit was imported into Business Process Manager, which is done by the IBM Cloud Orchestrator installer.
- Declare a dependency from your process application or toolkit to this toolkit.

The following building blocks are included with the toolkit:

Table 6. Building blocks

Building block	Section	Function
Business object	Data	Interface between IBM Cloud Orchestrator and Business Process Manager business processes that are provided as business objects within the toolkit. You can use them for parameter and private data declarations with Business Process Manager.
General system services, integration services, and web services	Implementation	Define Business Process Manager process extensions. They have a defined interface (input and output variable definitions) that are based on standard business objects that are provided by Business Process Manager and the <code>SCOrchestrator_Toolkit</code> .
Human services	User Interface	Sample UI extensions for an IBM Cloud Orchestrator extension. A Business Process Manager coach, which is included in a human service, defines a UI panel.

Table 6. Building blocks (continued)

Building block	Section	Function
Set of sample Business Process Manager processes, human services, and other Business Process Manager artifacts	Tagged as SCOrchestrator_Samples, contain the 'Sample' prefix in their name.	Provide samples that show you how to use the building blocks that are delivered with the toolkit. Required to write operation extensions for IBM Cloud Orchestrator. Some samples are ready to use, other must be adapted. The documentation that is provided with every sample artifact includes information about how to use it.

Business objects

The business objects provide a natural representation of the business data for application processing. You can use the business objects included in the SCOrchestrator_Toolkit to create custom objects.

The following business objects are delivered as part of the SCOrchestrator_Toolkit:

- OperationContext
- ServiceInstance
- VirtualMachine
- NetworkInterface

Representation of long types in business objects

Long types are not supported by Business Process Manager. Therefore, the simple type business object LongString is introduced as a part of the SCOrchestrator_Toolkit.

It represents a long type that is encoded as a string. For a new business object that contains a long type, define a new variable with the data type LongString. Ensure that it has the same name as defined by the IBM Cloud Orchestrator object variable. An example is the variable *created* (LongString) found in the CloudGroup. When serializing JSON Objects to BPM business objects, the long typed variables are automatically mapped to the defined Business Object variables of type LongString. When deserializing business objects to JSON objects, the type LongString indicates the mapping to the origin data type Long.

OperationContext

The operation context is an umbrella object that contains all data that is related to the execution of an operation. This operation context object must be defined as an input parameter variable for all business processes that are started as an extension for a IBM Cloud Orchestrator operation. Human services must define the operation context ID as an input parameter and as a first activity, must retrieve the operation context object with its ID.

The structure is as shown in the table:

Table 7. OperationContext Structure

Attribute	Data type	Description
id	String	The unique operation context identifier.
Status	String	The status of the task that is associated with the operation context.
eventTopic	String	The array that contains an object for each role.
description	String	The description of the ID.
Params	Map	The map that contains (key/value) pairs for storing arbitrary parameters. It can be used, for example, to hand over parameters that are collected from UI coaches in an extension Human Service to the associated Business Process. One of the most important keys in Params is the instances key that contains the instance IDs on which the instance actions execute. Use the integration services to fetch the instance IDs.
serviceInstance	ServiceInstance	Deprecated. Use the virtualMachines attribute instead.
user	String	The ID of the user in whose context the operation is executed
domain	String	The name of the domain in whose context the operation is executed
project	String	The name of the project in whose context the operation is executed
virtualMachines	Array of Object	The array contains an object for each managed virtual machine.

Service Instance

Deprecated. Use the virtualMachines attribute instead.

VirtualMachine

The VirtualMachine business object describes a virtual machine as part of a deployment.

It has the following structure:

Table 8. VirtualMachine structure

Attribute	Data type	Description
name	String	Deprecated.
cloudGroup	String	Deprecated.
networkInterfaces	NetworkInterface (List)	A list of network interfaces that the virtual machine is equipped with.
hostname	String	The primary host name for the virtual machine. If there are multiple network interfaces, the primary host name is implementation-dependent.
virtualCpu	Integer	Deprecated.
memory	Integer	Deprecated.
disk	Integer	Deprecated.
imageId	String	Deprecated.
parameters	Map	Deprecated.
Id	String	Deprecated.
keypair	String	The name of the key pair for accessing the virtual machine.

NetworkInterface

The NetworkInterface business object describes a network interface of a virtual machine.

It has the following structure:

Table 9. NetworkInterface Structure

Attribute	Data type	Description
ip	String	The IPv4 address in dotted decimal notation or the IPv6 address.
hostname	String	Deprecated.
ipgroup	String	Deprecated.
type	String	Deprecated.

Services

To simplify parameter handling, most of the REST calls to IBM Cloud Orchestrator have a JSON object that is coded as a string in the input and output parameters. The activities of Business Process Manager Business Processes and Services operate on complex business objects. Therefore, building blocks are needed to convert a business object to a JSON string and vice versa.

The different types of services are general system services, integration services, and web services.

- The general system services used are as follows:
 - CreateBusinessObject_OperationContext
 - GetErrorMessage
 - GetInputParameter
- The integration services used are as follows:
 - CreateBusinessObject_ServiceInstance
 - GenericBPDInvocation
 - GenericRestCall
 - GetInstanceIdFromOperationContext
 - GetMultipleInstanceIdsFromOperationContext
 - ReadBusinessObject
 - ReturnParameters
 - SetOperationContextParameters
 - UpdateInstancesParmInOperationContext
 - WriteValuesToBO
- The web service that is used is GenericBPDInvocationWS.

General system services

The general system services of the Business Process Manager are defined along with their input and output variables.

CreateBusinessObject_OperationContext

The CreateBusinessObject_OperationContext service creates a business object of type OperationContext based on a JSON input string.

Table 10. CreateBusinessObject_OperationContext Input Variables

Input Variable	Data type	Description
operationContextString	String	JSON string that represents the operationContext

Table 11. CreateBusinessObject_OperationContext Output Variables

Output Variable	Data type	Description
operationContext	OperationContext	It is a OperationContext business object

GetErrorMessage

The GetErrorMessage general system service receives as input an error XML element, which is returned by the variable *tw.system.step.error*. It extracts and returns the error message and stack trace from the XML element as output variables.

Table 12. GetErrorMessage Input Variable

Input Variable	Data type	Description
error	XMLElement	The XMLElement variable containing the error exception information as returned by <i>tw.system.step.error</i> variable.

Table 13. GetErrorMessage Output Variable

Output Variable	Data type	Description
errorMessage	String	The error message extracted from the XMLElement input variable.
stackTrace	String	The stack trace extracted from the XMLElement input variable.

GetInputParameter

The GetInputParameter general system service retrieves the inputParameterObject of Business Process Manager business process. The inputParameterObject is used as a workflow extension for IBM Cloud Orchestrator. It must be the first activity in each business process to be used as workflow extension for IBM Cloud Orchestrator.

Table 14. GetInputParameter Input Variable

Input Variable	Data type	Description
operationContext	OperationContext	The operation context object.
inputParameterObject_In	ANY	The inputParameterObject of the process must be mapped to this variable. It is required to enable a process to be called as linked process by another process. In this case, the inputParameterObject_In is not null. In all other cases, the inputParameterObject is null, which indicates that the process is called directly from the Cloud Orchestrator UI.

Table 15. *GetInputParameter* Output Variable

Output Variable	Data type	Description
inputParameterObject	ANY	The inputParameterObject contains the input parameters for the business process as collected by the associated human service.

Integration services

You define the integration services of the Business Process Manager together with their input and output variables.

CreateBusinessObject_ServiceInstance

The CreateBusinessObject_ServiceInstance service creates a business object of type ServiceInstance that is based on a JSON input string.

Table 16. *CreateBusinessObject_ServiceInstance* input variable

Input Variable	Data type	Description
jsonString	String	JSON string that represents the service instance

Table 17. *CreateBusinessObject_ServiceInstance* Output Variable

Output Variable	Data type	Description
serviceInstance	ServiceInstance	Service Instance Business Object

GenericBPDInvocation

The GenericBPDInvocation service is used internally by IBM Cloud Orchestrator to do some housekeeping actions, such as mapping parameters that are encoded as JSON strings to business objects. Finally, it calls the customer extension Business Process.

GenericRestCall

The GenericRestCall service is based on a Java implementation for a generic REST Call on IBM Cloud Orchestrator.

By default, it extracts the authentication information from the security context. The authentication information uses it to construct the security authentication token (Simple Token) for the REST calls that are made to external systems. As `tw_admin` is an internal user, GenericRestCall translates `tw_admin` to a default admin user for external calls. For more information about business process security, see “Business Process Manager security” on page 18. The GenericRestCall also accepts a user name on behalf of which a REST call must be made. It is especially useful for REST calls in the System Lane of a Business Process. They run by default as an admin user. It is desirable to run these REST calls on behalf of the person who submitted the request. To do that, map the user and project variables from the operation context to the GenericRestCall

Table 18. *GenericRestCall* input variables

Input Variable	Data type	Description
restMethod	String	The methods, such as POST, PUT, GET, and DELETE
jsonString	String	The input for PUT or POST encoded as a JSON string
restUrl	String	The REST URL that follows the host name (after http://<hostname>). For example, /resources/clouds.
iwdUser	String	The user name on behalf of which the REST call must be made. If not specified, the following users are considered: <ul style="list-style-type: none"> • If it is run in the context of Business Process Manager user interface, the logged in user is used. • In case it is for activities that run in the System Lane of a business process, admin user is used.
domain	String	The domain name for the REST call context. If not specified, the default domain is used
project	String	The project name for the REST call context. If not specified, the user default project is used

Table 19. *GenericRestCall* output variable

Output Variable	Data type	Description
returnFromRest	String	Result of a REST call that is encoded as a JSON string

GetInstanceIdFromOperationContext

The `GetInstanceIdFromOperationContext` integration service is used to fetch the instance ID from the given `OperationContext` object. For information about using instance data, see “Using instance data provided by `OperationContext`” on page 13.

Table 20. *GetInstanceIdFromOperationContext* input variable

Input Variable	Data type	Description
operationContext	OperationContext	The <code>OperationContext</code> business object.

Table 21. *GetInstanceIdFromOperationContext* output variable

Output Variable	Data type	Description
instanceId	String	The instance ID from the OperationContext object.

GetMultipleInstanceIdsFromOperationContext

The `GetMultipleInstanceIdsFromOperationContext` integration service is used to fetch the instance IDs from the given `OperationContext` object. For information about using instance data, see “Using instance data provided by `OperationContext`” on page 13.

Table 22. *GetMultipleInstanceIdsFromOperationContext* input variable

Input Variable	Data type	Description
operationContext	OperationContext	The <code>OperationContext</code> business object.

Table 23. *GetMultipleInstanceIdsFromOperationContext* output variable

Output Variable	Data type	Description
multipleInstanceId	String (List)	The list of instance IDs from the <code>OperationContext</code> object.

PerformOrchestratorRetCall

The service is based on a Java implementation for a generic REST Call on IBM Cloud Orchestrator.

By default, the service extracts the authentication information from the security context. The authentication information are used to construct the security authentication token (Simple Token) for the REST calls that are made to the external systems. As `tw_admin` is an internal user, this service translates `tw_admin` to a default admin user for external calls. For more information about business process security, refer to the Business Process Manager security.

The service also accepts a user name, password, and HTTP attributes on behalf of which a REST call must be made. The authentication information is fetched by invoking the JAVA implementation for `AuthenticationTokenHelper` on IBM Cloud Orchestrator.

It is especially useful for REST calls in the System Lane of a Business Process. They run by default as an admin user. It is desirable to run these REST calls on behalf of the person who submitted the request.

Table 24. *PerformOrchestratorRetCall* input variables

Input Variable	Data type	Description
restMethod	String	The methods, such as POST, PUT, GET, and DELETE

Table 24. PerformOrchestratorRetCall input variables (continued)

Input Variable	Data type	Description
restEndpoint	String	The REST endpoint (http://<hostname>). For example: http://ico-cs.cil.rtp.raleigh.ibm.com/v3/
jsonString	String	The input for PUT or POST encoded as a JSON string
restPath	String	The REST URL that follows the host name (after http://<hostname>). For example: /resources/clouds.
user	String	The user name on behalf of which the REST call must be made. If not specified, the following users are considered: if it is run in the context of the Business Process Manager user interface, the logged in user is used. In case it is for activities that run in the System Lane of a business process, admin user is used.
domain	String	The domain name for the REST call context. If not specified, the default domain is used.
project	String	The project name for the REST call context. If not specified, the user default project is used.
password	String	The user password for the authentication for the REST call context.
content	String	The HTTP content.
accept	String	The HTTP accept type.
cookies	String	Cookies to send in the HTTP request.
headers	String	The HTTP headers.

Table 24. PerformOrchestratorRetCall input variables (continued)

Input Variable	Data type	Description
parameters	Map	<p>Map of the parameters to be given as input parameter for the associated business process. If a particular business process needs to specify time out parameters, they can be added as part of the parameters.</p> <ul style="list-style-type: none"> • <code>icoRestReadTimeout</code>: parameter used to set the read timeout for the HTTP connection. For example (all in one line): <pre>tw.local.parameters.put("icoRestReadTimeout", "18000");</pre> • <code>icoRestConnectionTimeout</code>: parameter used to set for the HTTP connection time out. For example (all in one line): <pre>tw.local.parameters.put("icoRestConnectionTimeout", "24000");</pre>

Table 25. PerformOrchestratorRetCall output variables

Output Variable	Data type	Description
success	String	Business process is success or not (true or false).
statusCode	String	Result of a REST call status code.
reasonPhrase	String	Error message of the REST call.
message	String	Success message of REST call.
response	String	Result of a REST call that is encoded as a JSON string.
responseHeader	String	HTTP response header.

ReadBusinessObject

The service converts a business object into a JSON string so that it can be used. For example, as parameter for a REST call. It is a generic method that gets ANY business object as input parameter.

Restriction: The business object that is specified as input parameter must not have a parameter of type Map.

Table 26. ReadBusinessObject input variable

Input Variable	Data type	Description
BusinessObject	ANY	The business object to be converted into JSON string

Table 27. ReadBusinessObject output variable

Output Variable	Data type	Description
jsonString	String	Hypervisor Business Object that is converted into a JSON string.

ReturnParameters

The ReturnParameters integration service is used to return parameters that are collected by a human service. The collected parameters are used as input parameters for the associated process. It must be the last activity of a human service that collects parameters for a business process, which is used as a workflow extension for IBM Cloud Orchestrator.

Note: The outParameterObject business object must be of the same business object type as the inputParameterObject of the associated process. The type is used by IBM Cloud Orchestrator to correlate a human service with its associated business process.

Table 28. ReturnParameters input variable

Input Variable	Data type	Description
operationContextId	String	ID of the operation context
outParameterObject	ANY	A business object that contains the parameters that are collected by the human service. They are used as input parameters for the associated business process.

Table 29. ReturnParameters output variable

Output Variable	Data type	Description
returnString	String	Result of REST call that is encoded as JSON String.

SetOperationContextParameters

The SetOperationContextParameters integration service helps to enable the workflow of IBM Cloud Orchestrator extension to access and use parameters that are collected by a human service. It stores the input parameter map as key/value pairs into the parameter section of the operation context for the specified operationContextId. This parameter is later given as input parameter to the Business Process Manager Business Process extension. The parameter 'startTask' specifies whether the Business Process Manager workflow must be triggered by the execution of this integration service (true) or not (false). If false is selected, several consecutive calls can be made by a human service extension to store the

parameters that are collected into the `OperationContext` of the task. This activity is done before the Business Process Manager workflow process starts.

Note: This integration service is used internally to implement the `ReturnParameters` integration service. Normally, it must not be used directly by the human service that is associated with a workflow extension. Instead, the higher-level integration service `ReturnParameters` must be used.

Table 30. SetOperationContextParameters input variable

Input Variable	Data type	Description
operationContextId	String	ID of the operation context
params	Map	Map of parameters to be given as input parameter for the associated business process.
startTask	Boolean	True if the associated business process must be started (default false)

Table 31. SetOperationContextParameters output variable

Output Variable	Data type	Description
returnString	String	Result of REST call that is encoded as JSON String.

UpdateInstancesParmInOperationContext

The `UpdateInstancesParmInOperationContext` integration service must be used after a create operation to update the instances parameter in the `OperationContext` object, so that any subsequent `GetOperationContext` call uses this ID to update instance data such as `virtualMachines`, if applicable. For information about using instance data, see “Using instance data provided by `OperationContext`” on page 13.

Table 32. UpdateInstancesParmInOperationContext input variable

Input Variable	Data type	Description
taskId	String	The ID of the task. For example, the <code>operationContextId</code> .
providerInstanceId	String	The <code>providerInstanceId</code> of the instance.
providerType	String	The type of the resource of the instance. For example: <code>heat</code> , <code>openstackvms</code> .

Table 33. UpdateInstancesParmInOperationContext output variable

Output Variable	Data type	Description
params	Map	The updated instances parameter in the <code>OperationContext</code> object.

WriteValuesToBO

The service gets a JSON string representation of a business object as input variable and generates the corresponding TWObject Business Object.

Restriction: Using 'ANY' for the type of the output variable business object leads to an error in Business Process Manager. The business object type of the output variable (for example, Hypervisor) must be specified explicitly. Therefore, if you are using a Business Object other than 'Hypervisor', the integration service must be copied from the toolkit to your Business Process Application. You must adapt the type of business object to the special context.

Table 34. WriteValuesToBO input variable

Input Variable	Data type	Description
BusinessObject	ANY	The business object to be converted into JSON string

Table 35. WriteValuesToBO output variable

Output Variable	Data type	Description
jsonString	String	The business object that is converted into a JSON string.

Cloud management services

IBM Cloud Orchestrator offers cloud management services, for example, backup services or server restart.

Approval

The cloud management services require approvals. The approval is implemented by the human service and coach of the Business Process Manager technology. It is a building block of the SCOrchestrator_Toolkit.

Table 36. Approval Input Variable

Input Variable	Data type	Description
requester	String	Approval requester
parameters	NameValuePair (List)	List of parameters for approvals. The list of parameters in the form of name-value pairs that are shown as extra information in the approval UI panel.


Table 37. Approval Output Variable


Output Variable	Data type	Description
reason	String	Reason for approval or reject as specified in UI approval panel.
isApproved	Boolean	Approved (true) or not (false)


Instead of Business Process Manager Coach UI technology, the Cloud Orchestrator UI implements the approval panels. These approval panels show only the data

from the Business Process Manager 'Approval' human service. You can use predefined approval building blocks to build your own approval Business Process to display information that the approver needs to make a decision. The sample Business Processes that are delivered as part of the toolkit, such as `Sample_DeployInstanceApproval` and `SampleDeleteInstanceApproval` can be copied and then customized to your needs. This customized approval Business Process can then be enabled by registering it in the **Orchestrator Actions**.

Related information:

 [IBM Business Process Manager documentation](#)

 <http://www.scribd.com/doc/92505753/IBM-Impact-2011-Five-Guidelines-to-Better-Process-Modeling-for-Execution-Stuart-and-Zahn>
IBM Impact 2011 - Five Guidelines to Better Process Modeling for Execution - Stuart and Zahn

 http://www.ibm.com/developerworks/websphere/library/techarticles/1105_ragava/1105_ragava.html
Developer Works: WebSphere Lombardi exception handling and logging - Gopalakrishnan Ragava and Sateesh Balakrishnan

Samples

A number of samples are delivered as part of the `SCOrchestrator_Toolkit`. You must adapt the samples to your needs before they are run. You can find the actual samples documentation by opening the Business Process Manager process center (<http://<ico-server-IP>:9080/ProcessCenter/login.jsp>) and selecting the `SCOrchestrator_Toolkit`.

Note: Refer to the Business Process Manager documentation to know how to configure group membership and swim lane participation to achieve the required rights for the user you want to grant access to for requests like approval or problem handling.

Sample_AssignProblem

This sample process assigns the problem reported with sample process 'Sample_ReportProblem' to a user of participant group 'Problem Administrator'. Make sure that there are users defined for participant group 'Problem Administrator'.

Sample_DeleteInstanceApproval

This sample process includes an approval. It can be embedded in a IBM Cloud Orchestrator deletion workflow. The intention is to approve the deletion process. Make sure to also adapt the participant to the group which should have the approval right.

Sample_DeployInstanceApproval

This sample process includes an approval. It can be embedded in the IBM Cloud Orchestrator deploy instance workflow. The intention is to approve the deploy instance process. Make sure to also adapt the participant to the group which should have the approval right.

Sample_Post_ServiceInstance_Data

This sample process is to post ServiceInstance data, in this case parameters for Service Instance MetaData. The sample process first retrieves the ServiceInstance data as JSON String, creates a ServiceInstance Business Object and finally posts a parameter to the Service Instance MetaData.

Note: To run the process, the default value for the ServiceInstance ID has to be updated with a valid ServiceInstance ID.

Sample_Process_wo_HumanService

This sample extension is a process without human service. It displays the operation context business object in XML representation.

Sample_TestMultiTenancy

This sample is a simple test for REST calls within a process:

1. Using the default admin user context.
2. Using the caller's identity and domain/project scope that are stored in the operationContext.

Template_BusinessProcess

This sample template helps to create a Business Process Manager Business Process that can be used either as orchestrator action (event triggered action or user action) or as self-service offering within IBM Cloud Orchestrator. You can start developing your own business process extension by copying this template and enhancing it with your implementation logic.

Note: The variable type of the input parameter `inputParameterObject` has to be changed from 'ANY' to the concrete Business Object data type to be used. Do not forget to expose the process (Expose to start) to make it visible in the Configuration panel of IBM Cloud Orchestrator.

Appendix B. Scripting utilities toolkit

The `SCOrchestrator_Scripting_Uilities_Toolkit` is delivered as a part of the IBM Cloud Orchestrator product. It provides the essential building blocks of the Business Process Manager. These building blocks are required to build Business Process Manager Business Processes and human tasks, which are used as extensions for IBM Cloud Orchestrator.

When you work with remote devices, such as remote virtual machines and storage hosts, copy and run scripts/batch files on remote servers are frequently needed tasks. The Cloud Orchestrator Scripting Utilities Toolkit provides implementation services that you can drag and drop during IBM Cloud Orchestrator Business Process Manager process creation. These services can be used to copy files to a remote system or execute command lines on remote systems, including uploaded scripts/batch files.

Note: This toolkit is not designed to work with systems configured with restricted shell access, for example `rbash`.

Note: If you want to execute scripts on the deployed virtual machine, you must enable RXA by following the procedure described in Requirements for using Remote Execution and Access (RXA).

Note: Make sure that your script exits with return code 0. If your script exits or terminates with a return code different from 0, the script execution status is set to failure in the request details.

File upload restrictions

You can only upload files that reside within a local file repository such as a local folder or locally mounted folder on the IBM Cloud Orchestrator Server node.

It is defined by the toolkit environment variable `localRepositoryBasePath`. By default, this variable is defined as `/var/ibm/sco/scriptRepo`.

For security reasons, the processes of the toolkit check whether the files to be uploaded are always within the repository base folder.

Note: In an high-availability installation, you must synchronize the directory on both IBM Cloud Orchestrator Servers if you make a change to this directory. For more information, refer to Synchronizing the directory for the scripts.

Public Cloud Gateway restrictions

This topic describes the Public Cloud Gateway restrictions.

Note: The scripting toolkit UI does not provide the capability to specify `userid` and `password` to run commands and scripts on Public Cloud Gateway instances. This capability is instead provided for other managed platforms.

User name and password restrictions

Most of the images provided by Amazon EC2 and some from SoftLayer do not allow default logon to the provisioned virtual machine with user name and password. In general, the remote clouds advise to have keys to log on to internet facing virtual machines.

To enable user name and password for the image and ensure that cloudinit is configured correctly, follow the procedure described in [Creating a supported image](#).

Note: Every new image that is released in the remote cloud might have additional configurations that prevent using user name and password to login through SSH.

Note: If you require root access through SSH using either passwords or keys, you must do additional steps as root access through SSH is disabled in most of the remote cloud images.

Network selection and access to the provisioned virtual machine instances

In case of Public Cloud Gateway, you have the option to either use a private network or a public and private network for your provisioned virtual machine instances. Depending on this setup you might see either private or private and public as the networking option.

It is highly dependent on your network connectivity between IBM Cloud Orchestrator and the remote clouds like Amazon EC2 and SoftLayer which network interface must be used for a successful script execution.

For more information, see [Network planning](#) and [Public Cloud Gateway overview](#).

Script execution for Windows virtual machine instances

Script execution for Windows virtual machine instances requires a working RXA connection between IBM Cloud Orchestrator and the remote virtual machine instance.

Some of the remote clouds place restrictions on RXA connectivity. It might be required to setup VPN connectivity as described in [Network planning](#).

Note: If you require a VPN in your scenario, make sure that the connection timeout setting is high enough so that RXA can complete connecting to the remote virtual machine instance. Keep in mind that RXA through a VPN might be slow.

Note: Make sure that you correctly enabled RXA as described in the note in [Appendix B, "Scripting utilities toolkit,"](#) on page 47. Failure to a correct RXA setup prevents script executing into the remote virtual machine instance.

Business objects

These business objects are delivered as part of the SCOrchestrator_Scripting_Utilities_Toolkit.

Some of the business objects are partly reused within other business objects.

- RemoteMachineCredentials
- UploadFileData
- ExecuteCommandData
- OutputMapValues

ExecuteCommandData

Data to define a command line execution task.

Table 38. ExecuteCommandData

Parameters	Data type	Description	Sample data
commandLine	String	Command to be executed on the remote machine. This might also be a shell or batch script.	"ls -l"
workingDirectory	String	The processes change the current directory to the given working directory, before the command is executed.	"/tmp/test"
operatingSystemType	String	The type of operating system. The allowed values are "Windows" and "Linux/Unix".	
ipAddress	String	IP address of the target machine.	
subnetMask	String	Wildcard Card Entry for IP address.	"*22"
userId	String	UserID of the target machine.	
Password	String	Password of the target machine.	
executionOption	String	A free format string of options that can be passed to the execution script.	
privateKey	String	The private key of the target machine.	
UploadFileLocation	String	The directory where your file gets uploaded.	
ScriptFileName	String	The name of the script.	

Table 38. *ExecuteCommandData* (continued)

Parameters	Data type	Description	Sample data
timeOut	Integer	The duration (in seconds) during which the command or script must be completed. After the timeout has expired, the connection is closed forcefully. To disable the timeout, specify -1.	

OutputMapValues

Data object that holds well-defined return parameters of a called process.

Table 39. *OutputMapValues*

Parameters	Data type	Description	Sample data
errCode	String	Empty string if no error occurred by processing the implementation service. Otherwise an error code is returned.	"CTJCA1699E"
errMsg	String	Empty string if no error occurred by processing the implementation service. Otherwise an error message is returned.	"Details about the problem"
retCode	String	The return code of the called command.	"0"
stdOut	String	The standard output of the called command.	">script called"
stdErr	String	The standard error output of the called command.	"file is not executable"
sessionID	String	The ID of the open session, if there is an open session.	
timeoutExpired	Boolean	Specifies whether the connection for the command or script reached the timeout.	

Implementation services

The toolkit contains a number of implementation services.

The implementation services are described in separate topics. The following implementation services are used only internally and are not described in detail:

- Decrypt_Text
- Encrypt_Text

For more information about the services, see the documentation inside the toolkit or service itself.

The implementation services of the toolkit are as follows:

SSH_UploadFile

This integration or implementation service uploads a file to a remote machine through SSH protocol.

Table 40. SSH_UploadFile

Input parameters	Data type	Expected value	Sample data
remoteMachine	String	Remote machine IP address or host name	"9.120.xxx.xxx"
userName	String	Log in user name for the SSH protocol	"root"
password	String	Login password. Optional. See Table 41.	"*****"
PrivateKey	String	RSA-Key. Optional. See Table 41.	"RSA-Key"
fileName	String	Name of the file to be uploaded from the local file repository	"HelloWorld.sh"
destinationPath	String	Full destination path in the format of the remote machine	"/tmp/destination"
localRepositorySubDir	String	Subdirectory or NULL/Empty string. See "File upload restrictions" on page 47.	"/SubFolder/"
makeFileExecutable	Boolean	Make the file executable for "Implementation-Service" user (ignored on Windows operating systems)	"True"

Table 41. Optional parameters

Optional parameters
Password="value" / PrivateKey=ignored For the connection to the remote machine, the credentials "user" and "password" are used.
Password=empty / PrivateKey=empty If the private key of the Business Process Manager machine is inserted into the trusted list of keys within the remote machine, it is also possible to connect on this trusted relationship. In this case, the password field must be left empty to identify this connection mode.

Table 41. Optional parameters (continued)

Optional parameters			
Password=empty / PrivateKey="value" If another trusted private key must be used to enable the connection than it can be passed as a parameter. For this mode, the password must be left empty.			

Table 42. General output parameter map for most of the implementation services

Output parameters	Data type	Expected value	Sample data
OutputMap	Map (Key-,Value-String)	Results of the specific implementation service	See individual samples

Table 43. Possible keys and their values

Key string or key value	Expected value	Value string or sample data
errCode	SUCCESS string if no error occurred during the execution of the process. Otherwise an error code is returned.	"CTJCA1699E"
errMsg	Error details, if errCode is set	"Source file not available"
RC *	Return code of the called command line	"0"
stdOut *	Standard output of a called command line	"Hello world"
stdErr *	Standard error output of a called command line	"file/command not found"

* These keys are only available for services with command line.

SSH_ExecuteCommand

This integration or implementation service runs a command line on a remote machine through SSH protocol.

Table 44. SSH_ExecuteCommand

Input parameters	Data type	Expected value	Sample data
remoteMachine	String	Remote machine IP address or host name	"9.120.xxx.xxx"
userName	String	Log in user name for the SSH protocol	"root"
password	String	Login password. Optional. See Table 41 on page 51.	"*****"
PrivateKey	String	RSA-Key. Optional. See Table 41 on page 51.	"RSA-Key"
commandLine	String	Command line or script file that must be run on the remote machine	"ls -l"
cmdLineWorkingDirectory	String	Full path in the format of the remote machine	"/tmp/destination"

For the output parameters, see Table 42.

SSH_UploadFileAndExecuteCommand

This integration or implementation service:

- Uploads a file to a remote machine
- Runs a command line on a remote machine
- Through SSH protocol

Table 45. SSH_UploadFileAndExecuteCommand

Input parameters	Data type	Expected value	Sample data
remoteMachine	String	Remote machine IP address or host name	"9.120.xxx.xxx"
userName	String	Log in user name for the SSH protocol	"root"
password	String	Log in password. Optional. See Table 41 on page 51.	"*****"
PrivateKey	String	RSA-Key. Optional. See Table 41 on page 51.	"RSA-Key"
fileName	String	Name of the file to be uploaded from the local file repository	"HelloWorld.sh"
destinationPath	String	Full destination path in the format of the remote machine.	"/tmp/destination"
localRepositorySubDir	String	Subdirectory or NULL/Empty string. See "File upload restrictions" on page 47.	"/SubFolder/"
makeFileExecutable	Boolean	Make file executable for "Implementation-Service" user. Ignored on Windows operating systems	"True"
commandLine	String	Command line or script file that must be run on the remote machine.	"ls -l"
cmdLineWorkingDirectory	String	Full path in the format of the remote machine.	"/tmp/destination"

For the output parameters, see Table 42 on page 52.

Windows_UploadFile

This integration or implementation service:

- Uploads a file to a remote machine
- Through RXA toolkit/protocol

Table 46. Windows_UploadFile

Input parameters	Data type	Expected value	Sample data
remoteMachine	String	Remote machine IP address or host name	"9.120.xxx.xxx"
userName	String	Log in user name for the SSH protocol	"Administrator"
password	String	Log in password	"*****"

Table 46. *Windows_UploadFile* (continued)

Input parameters	Data type	Expected value	Sample data
fileName	String	Name of the file to be uploaded from the local repository.	"HelloWorld.bat"
destinationPath	String	Full destination path in the format of the remote machine	"c:\tmp\destination\"
localRepositorySubDir	String	Subdirectory or null/empty string. See "File upload restrictions" on page 47.	"\SubFolder\"

For the output parameters, see Table 42 on page 52.

Windows_ExecuteCommand

This integration or implementation service runs a command line on a remote machine through RXA toolkit/protocol.

Table 47. *Windows_ExecuteCommand*

Input parameters	Data type	Expected value	Sample data
remoteMachine	String	Remote machine IP address or host name	"9.120.xxx.xxx"
userName	String	Log in user name for the SSH protocol	"Administrator"
password	String	Log in password	"*****"
commandLine	String	Command line or script file that must be run on the remote machine.	"dir"
cmdLineWorkingDirectory	String	Full path in the format of the remote machine	"c:\tmp\destination\"

For the output parameters, see Table 42 on page 52.

Windows_UploadFileAndExecuteCommand

This integration or implementation service:

- Uploads a file to a remote machine
- Runs a command line on a remote machine
- Through RXA toolkit/protocol

Table 48. *Windows_UploadFileAndExecuteCommand*

Input parameters	Data type	Expected value	Sample data
remoteMachine	String	Remote machine IP address or host name	"9.120.xxx.xxx"
userName	String	Log in user name for the SSH protocol	"Administrator"
password	String	Log in password	"*****"
fileName	String	Name of the file to be uploaded from the local file repository	"HelloWorld.bat"

Table 48. *Windows_UploadFileAndExecuteCommand* (continued)

Input parameters	Data type	Expected value	Sample data
destinationPath	String	Full destination path in the format of the remote machine	"c:\tmp\destination\"
localRepositorySubDir	String	Subdirectory or NULL/Empty string. See "File upload restrictions" on page 47.	"\SubFolder\"
makeFileExecutable	Boolean	Make file executable for "Implementation-Service" user. Ignored on Windows operating systems	"True"
commandLine	String	Command line or script file that must be run on the remote machine	"dir"
cmdLineWorkingDirectory	String	Full path in the format of the remote machine	"c:\tmp\destination"

For the output parameters, see Table 42 on page 52.

ExtractOutputMap

This integration or implementation service extracts the OutputMap values (Key-Value-Pairs) to concrete fields of a OutputMapValues object.

Table 49. *ExtractOutputMap*

Input parameters	Data type	Expected value	Sample data
OutputMap	Map of strings	See Table 42 on page 52.	
retValues	OutputMapValues	ErrCode errMsg retCode stdOut stdErr	See the samples in Table 43 on page 52.

For the output parameters, see Table 42 on page 52.

FindVMInstanceForHostname

This integration or implementation service looks into the input VirtualMachine list, for a virtual machine instance with a specified host name (input) for a matching IP address and returns the virtual machine instance.

Table 50. *FindVMInstanceForHostname* Input Parameters

Input parameter	Data type	Expected value	Sample data
virtualMachines	List Of VirtualMachine	At least one virtual machine, with the IP address, host name populated. The KeyPair field is optional.	

Table 50. FindVMInstanceForHostname Input Parameters (continued)

Input parameter	Data type	Expected value	Sample data
hostname	String	Name of the virtual machine.	

Table 51. Output parameters

Output parameter	Data type	Expected value	Sample data
vmInstance	VirtualMachine	A virtual machine with the IP address, host name populated. The KeyPair field is optional.	
successMessage	String	Message when the operation is successful.	
failureMessage	String	Message when the operation fails.	

FindVmInstanceForIPAddress

This integration or implementation service looks into the input VirtualMachine list for a virtual machine instance with a specified IP address (input) for a matching host name and returns the virtual machine instance.

Table 52. FindVmInstanceForIPAddress input parameters

Input parameters	Data type	Expected value	Sample data
virtualMachines	List Of VirtualMachine	At least one virtual machine, with the IP address, host name populated. The KeyPair field is optional.	
ipaddress	String	IP address of the virtual machine.	

Table 53. Output parameters

Output parameters	Data type	Expected value	Sample data
vmInstance	VirtualMachine	A virtual machine, with the IP address, host name populated. The KeyPair field is optional.	
successMessage	String	Message when the operation is successful	
failureMessage	String	Message when the operation fails	

Sample processes

The `SCOrchestrator_Scripting_Uilities_Toolkit` contains sample processes, which are based on tasks, targeted against a single virtual server machine.

For output data for all of the samples, see *General Output Parameters (parameter map)* in “Business objects” on page 49.

For input data for the sample, see the business object references in the following list:

VirtualMachine_ExecuteCommand

For input data, refer to the `ExecuteCommandData`.

Sample UIs

The `SCOrchestrator_Scripting_Uilities_Toolkit` contains sample UIs, which reflect the needs of the corresponding sample processes.

The sample processes and their matching sample UI are related by their corresponding business object.

The type of output parameter of the sample UI must match the type of input parameter of the sample process.

The sample UI is:

VirtualMachine_ExecuteCommand

For input data, refer to the `ExecuteCommandData`.

Appendix C. Support IaaS toolkit

The `SCOrchestrator_Support_IaaS_Toolkit` provides capabilities that enable the user to perform operations on the IaaS layer in IBM Cloud Orchestrator. The toolkit provides two different ways to perform the API calls:

- An integration service "Generic IaaS Rest Call" that can be used to authenticate and perform REST calls against any IaaS services in specified regions. The requests can be performed on behalf of a user in a domain and project. It is the most flexible and documented way to perform REST calls to OpenStack services through an IaaS gateway.
- An integration service "Get IaaS Regions" that retrieves the list of available regions in a multi-region cloud environment. There exists a helper method in the `iaas_util.js` file that implements the authentication routine based on the configured shared secret keys in the WebSphere run time.

The toolkit provides building blocks that can be used in custom toolkits and process applications. These building blocks are required to build Business Process Manager Business Processes and human tasks, and are also used as extensions. IBM Cloud Orchestrator uses these extensions for actions against the IaaS layer. For information about IaaS layer, see APIs provided by OpenStack and IaaS gateway. The toolkit provides an essential Java REST client API and samples that show how to use the API in Business Processes. To write an extension for IBM Cloud Orchestrator, perform these mandatory tasks:

- Import this toolkit into Business Process Manager.
- Declare a dependency from your Process Application / toolkit to this toolkit.

The following building blocks are provided by this toolkit:

- Libraries (in the Files section of the toolkit) - the libraries contain the Java classes that implement the client-side of the OpenStack and Cloud IaaS gateway API. The libraries contain Java implementation of resources and methods. These resource implementations and methods can be used in human services and business processes for actions against the OpenStack and IaaS gateway.
- Samples - the toolkit provides samples that illustrate the usage of the client-side API. For example, the toolkit provides a sample to create and delete a tenant to the OpenStack identity service Keystone. Another example is about the modification of the metadata of a server. The samples are implemented based on default Business Process Manager artifacts. It includes the following elements:
 - Business objects (BOs) (in the Data section of the toolkit) - some sample objects that are used in the sample Business Process Manager Processes that illustrate the flow of data between Business Process Manager and the OpenStack or IaaS gateway. You can easily use them for parameter and private data declarations with the standard Business Process Manager. For example, business objects can represent tenants and servers.
 - Human services (in the User Interface section of the toolkit) - some sample human services and coaches are implemented to illustrate the usage of the API. For example, human services can specify the parameters of a tenant that are required to create the tenant. Another example is to select a server that exists in OpenStack. It shows its details and modifies its metadata.
 - Business Process Manager- some sample processes that illustrate the usage of long-running tasks. These processes include, for example, the creation and deletion of a tenant and the modification of the metadata of a server.

- Integration services - the toolkit provides the following integration services:
 - REST API calls against OpenStack and IaaS gateway
 - Retrieve the list of available OpenStack regions
- Authentication Routines - authenticate with the OpenStack identity service through the IaaS gateway.

Generic IaaS REST call

The IaaS Generic REST call makes REST calls to OpenStack services through the IaaS gateway. You can use the interface to specify the target region and service to start, for example, the compute service Nova in the US-West region. The interface is not typed and therefore expects and returns a string in a JSON format.

This generic interface allows any RESTful calls against the OpenStack APIs by specifying the URL and JSON response.

The authentication of the client is done by using the shared secret key, which is configured at run time of the Business Process Manager. The shared secret is used to generate a token for the admin user, which is used for authentication.

The integration service provides the following input variables:

restMethod

The method of the REST call, supported types are GET, POST, PUT, PATCH, and DELETE. It is a required property.

iaasServiceType

The type of the service that is instantiated and used to start the REST call. Examples for the service types are identity for keystone, compute for Nova, image for glance. It is a required property.

iaasEndpointType

The endpoint of the service to use. It is a required property. The supported values are

- For v2: adminURL, publicURL (default), and internalURL.
- For v3: admin, public (default), and internal.

iaasRegion

The region from which the service is instantiated, for example, RegionOne. It is an optional property. The first region that is found is taken by default.

restUrl

Defines the REST resource that must be started. For example, /tenants or /user/{user-id}. It is a required property.

restJsonContent

The content that must be posted or put to the REST URL. The string must have a valid JSON format as defined in the IaaS API. It is a required property.

iaasUser

The user of this request. It is an optional property. The default is admin.

iaasProject

The project that performs the request. The user is authenticated and the scope of the user for this project is determined. This property is optional. The default is admin.

iaasDomain

The domain of the user is authenticated. The user can only be uniquely determined by specifying the domain. This property is optional. The default is Default.

The output variable of the integration service is

restJsonResponse

The response of the REST call as a string in JSON format.

OpenStack and Cloud API library in Business Process Manager

The OpenStack and Cloud library contain Java classes that can be used to interact with the OpenStack and Cloud IaaS gateway. The Java classes and methods can be used in any Java or JavaScript implementation in any of the Business Process Manager artifacts. The basic concepts of how to use the library within Business Process Manager are covered here.

Note: To allow the usage of Java implementations in JavaScript implementation, the Business Process Manager uses **LiveConnect**. During Java class access, it requires a fully qualified package name because the engine must look up the classes.

For example, to create an instance of the `OpenStackConnectionFactory` for a given host (String), the full package name must follow the `Packages` keyword:

```
var connectionFactory = Packages.com.ibm.openstack.api.OpenStackConnectionFactory.newInstance(host);
```

To simplify the usage and avoid having to add the full package name for each variable, the developer can use the JavaScript by omitting the type definition. In this example, the object of type

`com.ibm.openstack.api.OpenStackConnectionFactory` is stored in the local variable `connectionFactory`. Next, the variable can be used as described in the API reference of the OpenStack and Cloud library. For example, to set up a connection for a user and password (both variables of type String) -

```
var connection = connectionFactory.newConnection(username, password);
```

Follow this pattern and the API reference to implement activities of type Java script in human services or processes to do actions against the IaaS layer. The next sections focus on authentication and actions against OpenStack and IaaS gateway.

Get IaaS Regions

The Get IaaS Regions integration service uses the generic IaaS client to retrieve the list of available regions. These regions can be used to find IaaS services, for example, the compute service in the US-West region.

The integration service does not require any input parameters. The output parameter is

-regions

a list of region names that are available.

Related concepts:

“Generic IaaS REST call” on page 60

The IaaS Generic REST call makes REST calls to OpenStack services through the IaaS gateway. You can use the interface to specify the target region and service to start, for example, the compute service Nova in the US-West region. The interface is

not typed and therefore expects and returns a string in a JSON format.

Samples for the IaaS support toolkit

The sample IaaS support toolkit provides basic samples for the IaaS support toolkit. The samples are available via the **Sample IaaS Support Process App** process application which is installed together with the IaaS support toolkit by default.

The following samples are available:

Sample Create Domain

This sample shows how to create a new domain entity in the v3 API of the identity service in Keystone. It assumes that Keystone v3 is enabled in the service catalog. The samples also demonstrates how to create a new 'Default' project for the domain and adding the admin user as a member to the project.

Sample Delete Domain

This sample shows how to delete a domain entity in the v3 API of the identity service in Keystone. The human service shows how to select a domain out of the list of domains. The process disables the domain and deletes it afterwards.

Sample Create Project

This sample shows how to create a project entity in the v3 API of the identity service in Keystone. The human service shows how to specify the project details and checks whether the project already exists. The process creates a project in Keystone.

Sample Delete Project

This sample show how to delete a project entity in the v3 API of the identity service in Keystone. The human service shows how to select a project out of the list of project . The process deletes the projects in Keystone.

Sample Project On-Boarding

This samples shows how to configure a new or existing project in Cloud Orchestrator. The purpose of the sample is to show the API calls to configure the project to be operational. Therefore, the human services is a wizard that steps the users through a list of steps to:

- Create or select the project entity in Keystone
- Configure the quota of the project
- Add users and roles to project
- Grant access to environment profiles, patterns, and images to the project
- Grant access to self-service offerings and operation actions to the project

Sample Create User

This sample shows how to create a user entity within a domain in the v3 API of the identity service in Keystone. The human service shows how to specify the user details, how to select the domain, and how to select the default domain and role. The human service checks if the user already exists. The process creates the user in the selected domain and assigns the selected project to the user with the role specified.

Sample Delete User

This sample show how to delete a user entity in the v3 API of the identity

service in Keystone. The human service shows how to select a user from the list of users. The process deletes the user in Keystone.

Sample to create a tenant in OpenStack identity service

The sample shows how to create a tenant. A business object is created and it contains the required parameters, such as name and description. It is assumed that the `inputParameterObject` is instantiated with the business object of type `Tenant`.

The code to create a tenant is as follows:

```
// authenticate using the helper method in iaas_utils.js
var connection = authenticateIaaS(tw.env.iaas_host, tw.env.iaas_user,
    tw.env.iaas_password, tw.env.iaas_tenant,
    tw.env.iaas_use_gateway);

// lookup the first identity admin service
var serviceFactory = Packages.com.ibm.openstack.api.
    service.OpenStackServiceFactory.
    newInstance(connection);
var identityAdminServices = serviceFactory.lookupServices
    (Packages.com.ibm.openstack.api.service.
    IIdentityAdminService);
var identityAdminService = identityAdminServices.get(0);

// create a new resource of type tenant
var factory = Packages.com.ibm.openstack.api.model.
    identity.IdentityFactory.eINSTANCE;
var tenant = factory.createTenant();

// initialize the tenant resource object with the
// attributes from the inputParameterObject
tenant.setName(tw.local.inputParameterObject.name);
tenant.setDescription(tw.local.inputParameterObject.description);
tenant.setEnabled(tw.local.inputParameterObject.enabled);

// call the identity admin service to create a new tenant
tenant = identityAdminService.createTenant(tenant);

// store the Id of the new tenant as a reference
// in the inputParameterObject
tw.local.inputParameterObject.id = tenant.getId();

// adapt authentication method -
// authentication based on the webshpere configuration settings
// using the helper method in iaas_utils.js
var connection = authenticateIaaS();
```

Sample to get a list of servers from OpenStack compute service

See how to get a list of servers from OpenStack compute service.

Note: This example is a single region example.

The example code gets the details of a server, and the logic of the steps are as follows:

1. Authenticate to the identity service.
2. Look up the compute service.
3. Get a list of all servers.

When the connection is established, the system must look up a compute service, which is registered in the service catalog of the identity service.

```
// get the service factory based on the connection
// (knows the identity service and the catalog)
var serviceFactory = Packages.com.ibm.openstack.api.service.
    OpenStackServiceFactory.newInstance(connection);
// lookup the compute services
var computeServices = serviceFactory.lookupServices
    (Packages.com.ibm.openstack.api.service.IComputeService);
// pick the first compute service
var computeService = computeServices.get(0);
```

In the next step, you get the list of servers and fill a local list. The list contains Business Process Manager Business Objects (BOs) that are created in the process application or toolkit to hold values. This list can then be used in UI elements, for example, as "selection box" (see the **Sample Play with VM** human service). The business object of type server can be found in the IaaS Support Toolkit. It has the attributes id and name of type string. The following code fills the list of servers with the list of servers that are retrieved from the compute service.

```
// the list of servers from compute service
var servers = computeService.getServers();
// a local variable of type Server in the process or human service
tw.local.servers = new tw.object.listOf.Server();
// iterate through the list of servers retrieved from compute service
for (var i=0; i<servers.size(); i++) {
    var s = new tw.object.Server(); // instantiate a new BO
    s.id =servers.get(i).getId(); // copy the id
    s.name=servers.get(i).getName(); // copy the name
    tw.local.servers.insertIntoList(i, s);
}
// add the new instance to the local list
```

Examples of how to obtain more details of a server can be found in the **Get Server Details** activity in the **Sample Play with VM** human service.

Troubleshooting

Read the following considerations about troubleshooting the toolkit.

The pattern designers use patterns to do actions against an OpenStack service, which has a benefit on simplification and time-to-value. The library provides a typesafe implementation of the OpenStack API, which is less error-prone and more comfortable than using REST calls and parsing JSON responses. However, this pattern has the following implications:

- The Business Process Manager documentation denotes performance implication in the usage of **LiveConnect**.
- Debugging is difficult.
- The Java script editor has no code completion and syntax highlight.

As a workaround for these implications, encapsulate the code in integration services or general system services. They are Java implementations of routines that are provided by Business Process Manager.

Appendix D. Email notification toolkit

The `SCOrchestrator_Email_Notification_Toolkit` is delivered with IBM Cloud Orchestrator. It provides the essential building block for sending email notifications with improved usability.

Note: You must configure an SMTP server via a `100Custom.xml` file. The email notification toolkit uses this configured SMTP server value to send email notifications. For information about how to configure SMTP, see “Configuring an SMTP server” on page 86.

Using the toolkit, you can:

- Send .HTML message files
- Send messagetext in HTML format
- Send messagetext
- Attach a list of files
- Send to a list of recipients' email addresses
- Send to list of cc recipients' email addresses

Note: Defined custom variables are automatically substituted by values.

Business objects

These business objects are delivered as part of the `SCOrchestrator_Email_Notification_Toolkit`.

EmailTemplate

Table 54. EmailTemplate

Parameters	Data type	Description	Sample data
subject	String	Email subject	Message subject text
message	String	Email message text	Filename of html file (" <code>/tmp/email.html</code> "), message-text in html format, or message text
contentType	String	Email type of messagepart	"text/html"
to	String	List of recipients' email addresses	"mailto:localhost"
cc	String	List of cc recipients' email addresses	"mailto:localhost1, mailto:localhost2" (recipients listed with separating comma or space)
from	String	Sender's email address	"root@localhost"
attachmentFiles	String	List of attached files	"/tmp/ibm-logo,/tmp/cloud.jpg"
variablepair (NameValuePair): - name(String) - value(String)	List	List of Email-Variable value pairs for substitution	[{"name": "\$FIRSTNAME", "value": "xxxxx"}, {"name": "&LASTNAME", "value": "yyyyy"}]

tw.local.status

Table 55. Status

Output variable	Data type	Description	Return codes
tw.local.status. tw.local.status == 0	Integer	Indicates the successfully executed building block to send an email. Status of executed process.	-1 missing message template xxx.html -2 missing attachment file -3 missing 'to' recipient email-address -4 missing sender email-address -5 other failures

Implementation services

The toolkit contains a number of implementation services.

The following implementation services are used only internally and are not described in detail:

- Read File
- Send File

For more information about the services, see the documentation inside the toolkit or service itself.

The implementation services of the toolkit are as follows:

Send Email Template

This implementation service prepares the message and message-text and sends it to the recipients' email addresses.

Table 56. Send Email Template

Parameters	Data type	Description	Sample data
subject	String	Email subject	Message subject text
message	String	Email message text	<ul style="list-style-type: none">• Filename of html file: "/tmp/email.html"or• message-text in html format or• message text
contentType	String	Email type of messagepart	"text/html"
to	String	List of recipients' email addresses	"mailto:localhost"
cc	String	List of cc recipients' email addresses	"mailto:localhost,mailto:localhost" (recipients listed with separating comma or space)
from	String	Sender's email address	"root@localhost"
attachmentFiles	String	List of attached files	"/tmp/ibm-logo,/tmp/cloud.jpg"

Sample processes

The SCOrchestrator_Email_Notification_Toolkit contains the sample process **Sample Send Email from Template**.

For output data and input data for **Sample Send Email from Template**, see the **EmailTemplate** business object in “Business objects” on page 65.

Sample UI

The SCOrchestrator_Email_Notification_Toolkit contains sample UIs, which reflect the needs of the corresponding sample processes.

The sample processes and matching sample UIs are determined by their corresponding data object. The type of output parameter of the sample UI must match the type of input parameter of the sample process. The sample UI is **Sample Send Email Notification**. For output data, see the **EmailTemplate** business object in “Business objects” on page 65.

Appendix E. OpenStack Cinder Storage Volumes toolkit

The `SCOrchestrator_OpenstackBlockStorage_Toolkit` is delivered as part of IBM Cloud Orchestrator. Using this toolkit, you can manage the lifecycle of block storage volumes through the OpenStack Cinder service. You can manage the block storage volumes in IBM Cloud Orchestrator deployed instances.

Installing and configuring

A good knowledge of IBM Cloud Orchestrator and of the Business Process Manager programming model is required for using the toolkit as a software development kit (SDK) for building new content.

This toolkit comes installed in IBM Cloud Orchestrator. The related offerings and actions are already available in the IBM Cloud Orchestrator environment.

Before starting to use this content pack, on the OpenStack Controller that manages the region that you are working on, you must ensure that you have:

- The OpenStack Cinder service up and running.
- At least one storage backend that is configured on Cinder and a volume type that maps to this backend. For instructions about how to configure a backend on Cinder and how to create volume types, see the OpenStack documentation.

Toolkit scenarios

A number of cinder-based scenarios are immediately available from the toolkit.

- “Creating storage volume” on page 70
- “Registering scripts for storage volumes” on page 70
- “Attaching script package to an image” on page 71
- “Unregistering scripts for storage volume” on page 72
- “Attaching volumes” on page 72
- “Detaching volumes” on page 74
- “Deleting volumes” on page 74
- “Managing volumes related to a virtual system instance” on page 75

First, you register the required Self-Service Offerings and Instance Actions, based on the configuration parameters specified in each of the scenarios. For more information about operation registration, see the following topics:

- Chapter 7, “Automating the creation of categories, offerings, and instance actions,” on page 25
- Orchestration workflows
- Working with self-service

Alternatively, you can use the Self-Service Catalog Population Tool to automatically register all of them simultaneously.

Creating storage volume

You can create new volumes in your instance.

This use case is available as a self-service offering.

To run this scenario, perform the following steps:

1. Go to the Self-Service Catalog and open the **Deploy cloud services** category.
2. Start the **Create storage volumes** offering.
3. Select the region and availability zone in which you want to create your volume.
4. Select a storage service level (volume type) from among the ones defined in the region that you have selected and click **Next**.
5. Enter the **Volume Name** and an optional **Description**.

Note: If you create more than one volume, from the first volume, an increasing index is added as a suffix to the volume base name.

6. Select or enter the number of instances and volumes.
7. Select or enter **Volume size** in GB.
8. Click **Submit**.

Table 57. Configuration

Name	Create volumes
Category	Deploy cloud services
Process	Create Volumes (SCOrchestrator_OpenstackBlockStorage_Toolkit)
User interface	Create Volumes (SCOrchestrator_OpenstackBlockStorage_Toolkit)

Registering scripts for storage volumes

You can upload custom scripts that can be used for volume-related operations, such as Partitioned Volume, Format Volume, and Mount Volume.

Procedure

1. Go to the Self-Service catalog and open the **Deploy cloud services** category.
2. Start the **Register scripts for storage volume** offering.
3. Enter **Script Package Name**.
4. Select your script files for **Partitioned Volume**, **Format Volume**, and **Mount Volume** functions.
5. Click **Upload**.

This step creates a folder in IBM Cloud Orchestrator Server at this location `/var/ibm/sco/scriptRepo/<scriptpackagename>` wherein the characters "CT" is appended to the `<scriptpackagename>` name. For example, if `ICOScriptPackage` is the name of the Script Package, then the folder name is `ICOScriptPackageCT`.

For Linux, the uploaded script files are renamed as `create_partition.sh`, `format_volume.sh`, and `mount_volume.sh`.

For Windows, the uploaded script files are renamed as `create_partition.ps1`, `format_volume.ps1`, and `mount_volume.ps1`.

Note: The base64Content of the script file is read and written to the files. Do not use single quotes in the custom script files instead use double quotes.

Table 58. Configuration

Name	Register scripts for storage volumes
Category	Deploy cloud services
Process	Script Upload Process (SCOrchestrator_OpenstackBlockStorage_Toolkit)
User interface	Script Upload (SCOrchestrator_OpenstackBlockStorage_Toolkit)

What to do next

Attach the uploaded scripts to an image.

Attaching script package to an image

After you register the customized volume scripts in the form of a script package, attach the script package to an image.

Before you begin

You must first upload your custom script and then attach it to an image. See “Registering scripts for storage volumes” on page 70.

About this task

Customized volume scripts work for a virtual server instance only if they are attached to the respective image. If you do not attach a script package to an image, then only the default scripts work for volume-related functions.

Procedure

1. Go to the **Self-Service Catalog** and open the **Deploy cloud services** category.
2. Start **Attach script package to image** offering.
3. Select the **Package** against the image name you want to attach.
4. Click **OK**.
5. Optional: Go to **Requests** and confirm the success of the operation.

Note: You cannot unregister a script that is attached to an image. To unregister scripts, see “Unregistering scripts for storage volume” on page 72.

Table 59. Configuration

Name	Attach script package to an image
Category	Deploy cloud services
Process	Attach Script to Image (SCOrchestrator_OpenstackBlockStorage_Toolkit)
User interface	Attach Script to Image (SCOrchestrator_OpenstackBlockStorage_Toolkit)

Unregistering scripts for storage volume

When a script package is unregistered, the selected package gets deleted from the `/var/ibm/sco/ScriptRepo` location. After you successfully attach a script to an image, you cannot unregister the script or delink the association of a script package from the image. Before you unregister a script from an image, attach the default package to that image as a prerequisite.

Before you begin

As a prerequisite, run this workaround to remove the association of a script page to an image:

1. Go to the **Self-Service Catalog** and open the **Deploy cloud services** category.
2. Start **Attach script to image** offering.
3. Select Default as **Package** name against the image name you want to attach.
4. Click **OK**.

Note: After you successfully attach the default package to an image, the script that you want to unregister gets detached from the image.

5. Do steps 1 - 4 until all the images that are associated to the script are detached successfully.

Procedure

1. In the **Deploy cloud services** category, go to **Unregister scripts for storage volume** offering.
2. Select the script package and click **Unregister**.

Table 60. Configuration

Name	Unregister scripts for storage volumes
Category	Deploy cloud services
Process	Unregister scripts process (SCOrchestrator_OpenstackBlockStorage_Toolkit)
User interface	Unregister scripts (SCOrchestrator_OpenstackBlockStorage_Toolkit)

Attaching volumes

You can attach a volume in your project to a virtual machine and optionally format it as a single partition and mount it to a given mount point. In addition, even the pre-partitioned volumes can be directly mounted.

This use case is available as instance action applicable to a volume resource instance. The volume instance must be in available status. For more information about volumes, see Working with volumes.

To run this scenario, perform the following steps:

1. Go to the volume resource view by clicking the **RESOURCES** tab and then clicking the **Volumes** icon.
2. Select an available volume.
3. Click the **Attach volume** action.
4. In the Select a Server page, select a virtual machine from the list and click **Next**.
5. In the Configure format and mount options page, enter the following details:

- a. Select the **Virtual machine operating system**: Windows or Linux.
- b. Select whether you want to connect to the virtual machine through credentials or SSH key:
 - In case of SSH, use the key that is stored in the keystore or use your own SSH key. For Windows operating systems, connection by using the SSH key is disabled.
 - In case of credentials, enter the User name and Password.
6. Click **Next**.
7. If you select **Format Volume**, then select the **File system type**.
8. If you select the **Mount Volume**, then enter the **Mount point**.
9. Click **Next**.
10. Check the summary and click **Submit**.

After having attached the volume, its format and mount options (if there are any) are stored in the volume metadata. Volume metadata also contains the file system label and UUID.

Table 61. Configuration

Name	Attach volume
Instance type	volumes
Process	Attach Volumes (SCOrchestrator_OpenstackBlockStorage_Toolkit)
User interface	Attach Volumes (SCOrchestrator_OpenstackBlockStorage_Toolkit)

Note:

- You cannot enable this option for machines that are powered off.
- This option is not supported for AIX instances.
- For Windows instances:
 - The operating system must be enabled for RXA access as described in Requirements for using Remote Execution and Access (RXA).
 - Volume can be mounted on a non-existing drive but not on a directory in the non-existing drive. For example, P:\ is valid but not P:\cinder_vol.
 - The default RXA connection timeout might be too small to allow to the volume to be correctly formatted or mounted. If this problem occurs, connection errors are displayed in the SystemOut.log file in the provisioned virtual machine.

To solve the problem, add the `com.ibm.tivoli.remoteaccess.connection_timeout_default_millis` property to Business Process Manager Java WebSphere configuration by performing the following steps:

1. Log on to `http://<IBM Cloud Orchestrator Server>:9060/admin` as `bpm_admin`.
2. Click **Servers > All servers > SingleClusterMember1 > Java and Process Management > Process definition > Java Virtual Machine > Custom properties**.
3. Click **New** to add the following property:

Property name

`com.ibm.tivoli.remoteaccess.connection_timeout_default_millis`

Value At least 180000 (that is 3 minutes). The time value is in milliseconds.

4. Restart the Business Process Manager to activate the change.

If the provisioned virtual machine resides on a remote cloud that is managed by Public Cloud Gateway, see also Public Cloud Gateway overview and Network planning.

- For Linux operating systems, if you chose to mount the file system on which you have formatted the volume, no record in the `/etc/fstab` file is added. When you restart the machine, the file system is no longer mounted and you must mount it manually.
- If a volume is attached to a multi-disk image, the device data associated to the volume might not display correctly in the Self-service user interface.

Detaching volumes

If necessary, you can umount and detach a volume.

This use case is available as an instance action applicable to a volume resource instance. The volume instance must be in in-use status. For more information about volumes, see Working with volumes.

If the volume has been previously formatted, its filesystem persistent data, such as type, label, and UUID are not deleted from the metadata.

To run this scenario, perform the following steps:

1. Go to the volume resource view by clicking the **RESOURCES** tab and then clicking the **Volumes** icon.
2. Select an in-use volume.
3. Click the **Detach volume** action.
4. If your volume is mounted, specify how you want to connect to the machine. See the options in “Attaching volumes” on page 72.
5. Check the summary and click **Submit**.

Note: Because of an operating system limitation, after you detach a volume from a SLES instance on VMware, you must reboot the SLES instance to complete the detach operation.

Table 62. Configuration

Name	Detach volume
Instance type	volumes
Process	Detach Volumes (SCOrchestrator_OpenstackBlockStorage_Toolkit)
User interface	Detach Volumes (SCOrchestrator_OpenstackBlockStorage_Toolkit)

Deleting volumes

You can delete one or more volumes that are in available status.

This use case is available as an instance action applicable to a volume resource instance. The volume instance must be in available status. For more information on volumes, see Working with volumes.

To run this scenario, perform the following steps:

1. Go to the volume resource view by clicking the **RESOURCES** tab and then clicking the **Volumes** icon.
2. Select one or more available volumes.

3. Click the **Delete volumes** action.
4. Confirm the action.

Table 63. Configuration

Name	Delete volume
Instance type	volumes
Process	Delete Volumes (SCOrchestrator_OpenstackBlockStorage_Toolkit)

Managing volumes related to a virtual system instance

You can attach or delete all the available volumes that are defined in the project that you are currently logged on to. You can detach the volumes in the same project that are in use by the selected virtual system instance.

This use case is available as instance action applicable to an **openstackvms** resource instance.

To run this scenario, perform the following steps:

1. Go to the **RESOURCES** tab and click the **VMs** icon.
2. Select a virtual machine.
3. Click the **Manage volume** action.
4. Select a volume from the ones that are created in the project that you are currently logged on to and in the region where your instance is hosted. When you select a volume, the action buttons are enabled or disabled depending on the status of the selected volume. If the volume is available, you can attach it to a virtual machine or delete it. If the volume is in use, you can detach it from its virtual machine.
5. Click the management action that you want to perform on the selected volume:

Attach

Select the virtual machine where you want to attach the volume. Choose whether to format and mount the volume. You cannot enable this option on machines that are powered off. Select the operating system and configure the access to that machine. If you chose to format and mount the volume, specify the filesystem and mount point. See “Attaching volumes” on page 72. After reviewing the summary, click **Submit**. The volume is attached, formatted, and mounted to the selected machine.

Note: For Linux operating systems, if you chose to mount the file system on which you have formatted the volume, no record in the `/etc/fstab` file is added. When you restart the machine, the file system is no longer mounted and you must mount it manually.

Detach

If the volume is mounted, you must configure access to the virtual machine to unmount it before detachment. Review the summary and click **Submit**. The volume is detached from the attached virtual machine.

Delete

Review the summary and click **Submit**. The volume is deleted and no longer appears in the volume list.

Note: Only volumes that are in "available" state or attached to the selected Virtual System instance are listed in the volume table.

Table 64. Configuration

Name	Manage volume
Instance type	openstackvms
Process	Manage Volumes (SCOrchestrator_OpenstackBlockStorage_Toolkit)
User interface	Manage Volumes (SCOrchestrator_OpenstackBlockStorage_Toolkit)

Toolkit developer's reference

This section contains reference information about the Business Process Manager artifacts exposed in the toolkit contained in the content pack. It is intended to be used by IBM Cloud Orchestrator content developers to extend the already available scenarios or to write new scenarios leveraging the building blocks available from the toolkit.

The following items are the main building blocks of the toolkit:

- "Environmental variables"
- "Business objects" on page 77
- "Human services" on page 78
- "Coach views" on page 79
- "Business processes" on page 79
- "Integration services" on page 80

Environmental variables

Some environment variables are available so you can customize some behaviors when you run the use cases.

Environmental variables are defined in the toolkit in **Toolkit Settings > Environment**.

ATTACH_WAIT_RETRIES

This variable specifies the maximum time, in minutes, to wait for a volume becoming available after attachment. A query is run every minute to check the volume status. The default value is 10.

CREATE_WAIT_RETRIES (formerly known as ATTACH_TIMEOUT)

This variable specifies the maximum time, in minutes, to wait for a volume becoming available after creation. A query is run every minute to check the volume status. If the volume is still not available after the timeout value, an error is issued. The queries are also stopped if the new volume turns into an error status. The default value is 30.

DELETE_WAIT_RETRIES

This variable specifies the maximum time, in minutes, to wait for a volume being deleted. A query is run every minute to check the volume status. The default value is 30.

DETACH_WAIT_RETRIES

This variable specifies the maximum time, in minutes, to wait for a volume becoming detached. A query is run every minute to check the volume status. The default value is 10.

DEFAULT_PACKAGE_NAME

This variable is the name of the default script package that gets called when no script package is attached to the image.

IMAGE_PCK_DB

This variable specifies the namespace that is used to persist the details of custom script package that is associated with an image.

PACKAGE_NAME_POSTFIX

This variable defines the postfix that is appended to the script package name so that you can uniquely identify the script package directories.

Business objects

There are Business Process Manager business objects defined in the toolkit.

AttachVolumesToInstanceRequestData

This business object is used for storing information about the volume which is being attached to a given instance.

CinderBreadcrumbsObject

This object is used for storing the temporary properties to manage the breadcrumb widget that appears on the bottom part of some user interfaces.

VMCredentials

This object stores the virtual machine credentials and in case also public and private ssh keys.

Volume

This is the volume object. It contains info about the volume name, status, volume type, ID, size, attachment metadata, and the server which it is attached to.

Volume_Attachment

This object stores the correspondence between the volume and the server it is attached to.

VolumeManager

This object contains the volume data, the operation which must be carried out on this volume, the attachment data for this volume, the virtual machine object the volume must be attached to and the region which the volume is defined on.

Volume type

This object represents the volume type or Storage Service Level.

Image_Package_Data

This object has the details of the image and the package associated with the image

Script_Upload

This object has the file data of custom scripts that are registered.

Human services

There are a number of Human Services artifacts available in the toolkit.

Attach Volumes

This an entry point user interface for the "Attach volume" instance action.

Detach Volumes

This an entry point user interface for the "Detach volume" instance action.

Create Volumes

This an entry point user interface for the "Create volumes" self-service offering.

Manage Volumes

This an entry point user interface for the "Manage volumes" instance action.

Attach Script to Image

This an entry point user interface for the "Attach Script to Image" self-service offering.

Script Upload

This an entry point user interface for the "Register scripts for storage volumes" self-service offering.

Unregister Scripts

This an entry point user interface for the "Unregister scripts for storage volumes" self-service offering.

Coach views

There are a number of coach view artifacts available in the toolkit.

Breadcrumbs

It is the breadcrumb widget that appears on the bottom part of some user interfaces.

Generic Listener

It is the panel fragment which show the summary for detaching operation.

Password

The Password coach view renders the password as a masked input.

Server Selected Listener

This is the listener on the servers list. It disables the **Next** button if no servers are selected.

Vertical Two RadioButtons

A coach view for implementing a choice between two possibilities.

Volume Selected Listener

This is the listener on the volumes list. It disables the buttons which correspond to action not compatible with the selected volume status. For example, volumes “in-use” can only be detached.

Business processes

There are a number of the business processes available in the toolkit.

Attach Volume

This process attaches a volume to a given virtual server and can, if necessary, format and mount it after the attachment. It returns the volume object after the attachment.

Attach Volume Utility

A wrapper for the Attach Volume process which provides a more usable interface for input parameters.

Create Volume

This process create a new volume of a given size within a given volume type on a specified region. The volume is returned as output value if it has been available after a given timeout.

Create Volume Utility

A wrapper for the Create Volume process which provides a more usable interface for input parameters.

Create Volumes

This process loops over the number of requested volumes calling Create Volume process for creating them.

Detach Volumes

This process detaches a volume and, if it is mounted, unmounts it before detachment.

Delete Volume

This process triggers a volume deletion and wait for it is completed.

Manage Volumes

This process create a new volume or manage an existing volume as stated by an operationid (1=ATTACH, 2=DETACH, 3=DELETE). It is called by “Manage Volumes Offering” and “Manage Volumes Action” processes.

Attach Script to Image

This process persists the association of a custom script package to an image.

Script Upload Process

This process reads the uploaded custom scripts and saves it to the script repo, which is used in the scripting utilities toolkit.

Unregister Scripts Process

This process removes the custom script package from the environment.

Integration services

There are a number of integration services available in the toolkit.

Attach Volume Service

This integration service attaches one or more Cinder Volumes to a server.

Create Volume Service

This integration service creates a volume.

Delete Volume

This integration service deletes a volume.

Detach Volume

This integration service detaches a volume from a server.

Format Volume on Linux

This integration service connects to Linux machine, formats an attached volume, and mounts it to a mount point. The volume is formatted as a single partition and mounted by using the script `configure_volume.sh`, which is contained in the toolkit. It also updates the attachment metadata for the volume. The script is run remotely through SSH by using the management IP address. The SSH connection is tried for each available machine IP until it is successful. The SSH connection can be carried out either by providing credentials or by using a pre-configured `privateKey`.

Format Volume on Windows

This integration service connects to a Windows machine, formats an attached volume, and mounts it to a mount point. The volume is formatted as a single partition and mounted by using the script `configure_volume.ps1`, which is contained in the toolkit. It also updates the attachment metadata for the volume. The script is run remotely through SMB by using the management IP address. The SMB connection is tried for each available machine IP until it is successful.

Get Message

This integration service gets a message from the loaded message bundle.

Get Volume

This integration service retrieves a volume given its ID.

Get Volume Attachment MetaData

This integration service returns the volume attachment data (filesystem type, mount point, filesystem label, and file system UUID) by retrieving them from volume metadata.

List Volume Types

This integration service retrieves all volume types defined on a region.

List Volumes

This integration service gets the list of all volumes defined on a region.

Load Message Bundle

This integration service loads the message bundles.

Set Volume Attachment Metadata

This integration service sets the volume attachment metadata.

Set Volume Metadata

This integration service set a volume metadata.

Umount Attached Volume Partition

This integration service connects to a Windows or Linux machine and unmounts the mounted volume. It also updates the attachment metadata for the volume. The script is run remotely through SMB or SSH by using the management IP address. The SMB or SSH connection is tried for each available machine IP until it is successful. The SMB or SSH connection can be carried out either by providing credentials or by using a pre-configured privateKey (only for Linux).

Unset Volume Metadata

This integration service deletes one volume metadata.

Detect devices on Linux

This integration service detects the available devices on a Linux server.

Detect devices on Windows

This integration service detects the available devices on a Windows server.

Get Attached package List

This integration service gets the list of custom packages that are attached to an image.

Get Disk Info on Linux

This integration service gets the disk information of the specified disk on a Linux server.

Get Disk Info on Windows

This integration service gets the disk information of the specified disk on a Windows server.

Get Image Metadata

This integration service gets metadata of an image.

Get Info and Attach Volume for Windows

This integration service attaches a volume, detects the attached disk, and gets the disk information of the specified disk on a Windows server.

Get List of Packages

This integration service gets the list of all custom script packages.

Partition Format Mount Volume on Linux

This integration service is used to partition, format, and mount the new volume attached to the Linux server. If custom scripts are associated to the image, then they are started to perform the operations. Otherwise, default scripts are started.

Partition Format Mount Volume on Windows

This integration service is used to partition, format, and mount the new volume attached to the Windows server. If custom scripts are associated to the image, then they are started to perform the operations. Otherwise, default scripts are started.

Remove Script Package

This integration service removes the selected custom script package.

Samples about how to use the available services and views

The services that are delivered as part of this content pack can be used as starting points and as samples for developing new content.

Using samples can give you an idea about how to invoke and combine the Integration Services delivered in the content pack. You can clone and adapt them to better fit your needs. Once you are familiar with them, you can start creating your new offerings and actions.

For information about how to develop new content, see [Developing IBM Cloud Orchestrator content](#).

Troubleshooting

There are some known problems and limitations.

- For non-IBM supplied OpenStack, because formatting volumes is not supported, the **Attach volumes** action is supported only if the **Format volume** option is not selected.

Logs and traces

If you receive an error, you can check the Business Process Manager log files `SystemOut.log` and `SystemErr.log` at `/opt/ibm/ico/BPM/v8.5/profiles/Node1Profile/logs/SingleClusterMember1/`.

You can find extra logs on the OpenStack Controller at `/var/log/cinder` and `/var/log/nova`. For errors regarding format and mount operation, a log file `formatFileSystem.log` is present in the `/tmp` directory of the virtual machine. Error messages are logged for all probable errors during deployment.

Error codes

Table 65. Error codes

Error code	Cause
CTJCA2000E	The volume was not available within the defined timeout. See <code>CREATE_WAIT_RETRIES</code> variable. Its status might be <i>creating</i> or <i>error</i> .
CTJCA2001E	A problem occurred while trying to format and mount the attached volume. This is usually due to a connection problem.
CTJCA2002E	A problem occurred while trying to unmount the attached volume. This is usually due to a connection problem.
CTJCA2006E	A problem occurred in updating metadata of the volume.

Appendix F. OpenStack Services toolkit

The SCOrchestrator_OpenStack_Services toolkit is delivered with IBM Cloud Orchestrator. It provides offerings to deploy a predefined Linux or Windows virtual machine, to deploy a Linux or Windows virtual machine with approval and notification, or to deploy a multitiered LAMP stack (Linux and Apache, Linux and MySQL, Linux and PHP).

These offerings refer to operating system images that are not distributed with IBM Cloud Orchestrator. Before using the offerings, you must perform the configuration steps described in “Configuring.”

Configuring

Before using the offerings provided in the OpenStack Services toolkit, you must perform the following configuration steps.

- “Defining a Linux image”
- “Defining a Windows image”
- “Configuring an SMTP server” on page 86
- “(Optional) Modifying the notification message template” on page 86

Defining a Linux image

Before deploying a Linux virtual machine, you must create a Linux image, add it to your OpenStack environment, and create a key pair for accessing the virtual machine. Perform the following steps:

1. Create a Linux image by following the procedure described in Creating Linux base images. Alternatively, for example, you can download a CentOS image from http://cloud.centos.org/centos/6/images/CentOS-6-x86_64-GenericCloud.qcow2.
2. Add the image to your OpenStack environment by following the procedure described in Adding images to your OpenStack environment.

Important: You must specify `linux_img` as name of the image.

3. Create a key pair for accessing the Linux virtual machine by following the procedure described in Registering a key pair.

Important: You must specify `linux_key` as name of the key pair.

Defining a Windows image

Before deploying a Windows virtual machine, you must create a Windows image and add it to your OpenStack environment. Perform the following steps:

1. Create a Windows image that has RXA enabled by following the procedure described in Creating Windows base images.

Note: Make sure that you enable RXA before you run `sysprep.exe`.

Note: Make sure that the image contains the user Admin as the Cloud-init user and that the use of metadata password is enabled.

2. Add the image to your OpenStack environment by following the procedure described in Adding images to your OpenStack environment.

Important: You must specify `windows_img` as name of the image.

Configuring an SMTP server

To use the email notification feature, you must configure Business Process Manager to use an appropriate SMTP relay server inside your organization. Contact your mail server administration team to obtain the host name of the SMTP server.

Business Process Manager does not support any type of SMTP authentication or non standard SMTP ports.

Perform the following steps:

1. Create or update the `100Custom.xml` file in the `/opt/ibm/ico/BPM/v8.5/profiles/DmgrProfile/config/cells/PCCell1/nodes/Node1/servers/SingleClusterMember1/process-center/config` directory with the following content:

```
<properties>
  <server merge="mergeChildren">
    <email merge="mergeChildren">
      <smtp-server merge="replace">${SMTP_HOST_NAME}</smtp-server>
    </email>
  </server>
</properties>
```

where `SMTP_HOST_NAME` is your SMTP server host name.

2. Restart the Business Process Manager server by using the following command:
`systemctl restart bpm`

(Optional) Modifying the notification message template

The email template is by default in English. To modify it, perform the following steps:

1. Log in to the IBM Process Designer with administrative credentials.
2. In the SCOrchestrator OpenStack Services (SC00S) toolkit, click **Manage** and check the **Allow users to update the toolkit** check box.
3. Open the toolkit in designer mode.
4. In the list on the left pane, click **Implementation** and double-click **Send Email Notification** in the **Integration Service** section.
5. Double-click the **Send Approved Email** or **Send Rejected Email** script and edit the HTML content in the `tw.local.messageTemplate` variable.
6. Click **Save**.

Toolkit scenarios

There are a number of scenarios that are available from the toolkit.

Before running the scenarios, ensure you performed the configuration steps described in “Configuring” on page 85

The following scenarios are available:

- “Deploying a Linux virtual machine”
- “Deploying a Linux virtual machine with approval and notification” on page 88
- “Deploying a Windows virtual machine” on page 89
- “Deploying a Windows virtual machine with approval and notification” on page 89
- “Deploying a LAMP stack using multitiered technology” on page 90

Deploying a Linux virtual machine

To deploy a Linux virtual machine, perform the following procedure.

Before you begin

Ensure you have a Linux image and key pair defined as described in “Defining a Linux image” on page 85.

Procedure

1. In the Self-service user interface, click **SELF-SERVICE CATALOG** and open the **Deploy customized cloud services** category.
2. Click the **Deploy single Linux server** offering.
3. If you have more than one region in your OpenStack environment, select the target region for the deployment and click **OK**.
4. If you have more than one network in your selected region, then by default the first network is used for the deployment.
5. If you have more than one availability zone in your selected region, then by default the first availability zone is used for the deployment.

Results

A new virtual machine named `Sample_Linux_Server_XXXXX` (where `XXXXX` is a random five digit number) is created with medium flavor.

To get the virtual machine details, click the `Sample_Linux_Server_XXXXX` name in the list that you can access from **RESOURCES > Virtual Machines**.

To access the virtual machine, use the `linux_key` key pair that was created in the procedure described in “Defining a Linux image” on page 85.

Deploying a Linux virtual machine with approval and notification

To deploy a Linux virtual machine with an approval process and a notification email, perform the following procedure.

Before you begin

Ensure you have a Linux image and key pair defined as described in “Defining a Linux image” on page 85.

Ensure an email address is configured in IBM Cloud Orchestrator for the user that is running the offering.

Ensure an SMTP server is configured as described in “Configuring an SMTP server” on page 86.

Procedure

1. In the Self-service user interface, click **SELF-SERVICE CATALOG** and open the **Deploy customized cloud services** category.
2. Click the **Deploy single Linux server with approval and notification** offering. A request to create a Linux virtual machine is created. The admin user must claim and approve this request. When the request is approved, you receive an email to the address configured for your user.
3. If you have more than one region in your OpenStack environment, select the target region for the deployment and click **OK**.
4. If you have more than one network in your selected region, then by default the first network is used for the deployment.
5. If you have more than one availability zone in your selected region, then by default the first availability zone is used for the deployment.

Results

A new virtual machine named `Sample_Linux_Server_XXXXX` (where `XXXXX` is a random five digit number) is created with medium flavor.

To get the virtual machine details, click the `Sample_Linux_Server_XXXXX` name in the list that you can access from **RESOURCES > Virtual Machines**.

To access the virtual machine, use the `linux_key` key pair that was created in the procedure described in “Defining a Linux image” on page 85.

To modify the notification message template, refer to “(Optional) Modifying the notification message template” on page 86.

Deploying a Windows virtual machine

To deploy a Windows virtual machine, perform the following procedure.

Before you begin

Ensure you defined a Windows virtual machine as described in “Defining a Windows image” on page 85.

Procedure

1. In the Self-service user interface, click **SELF-SERVICE CATALOG** and open the **Deploy customized cloud services** category.
2. Click the **Deploy single Windows server** offering.
3. If you have more than one region in your OpenStack environment, select the target region for the deployment and click **OK**.
4. If you have more than one network in your selected region, then by default the first network is used for the deployment.
5. If you have more than one availability zone in your selected region, then by default the first availability zone is used for the deployment.

Results

A new virtual machine named `Sample_Windows_Server_XXXXX` (where `XXXXX` is a random five digit number) is created with medium flavor.

To get the virtual machine details, click the `Sample_Windows_Server_XXXXX` name in the list that you can access from **RESOURCES > Virtual Machines**.

To access the virtual machine, use username `Admin` and password entered while deploying the virtual machine as credentials. The password must meet complexity requirements.

Deploying a Windows virtual machine with approval and notification

To deploy a Windows virtual machine with an approval process and a notification email, perform the following procedure.

Before you begin

Ensure you defined a Windows virtual machine as described in “Defining a Windows image” on page 85.

Ensure an email address is configured in IBM Cloud Orchestrator for the user that is running the offering.

Ensure an SMTP server is configured as described in “Configuring an SMTP server” on page 86.

Procedure

1. In the Self-service user interface, click **SELF-SERVICE CATALOG** and open the **Deploy customized cloud services** category.
2. Click the **Deploy single Windows server with approval and notification** offering. A request to create a Windows virtual machine is created. The admin

user must claim and approve this request. When the request is approved, you receive an email to the address configured for your user.

3. If you have more than one region in your OpenStack environment, select the target region for the deployment and click **OK**.
4. If you have more than one network in your selected region, then by default the first network is used for the deployment.
5. If you have more than one availability zone in your selected region, then by default the first availability zone is used for the deployment.

Results

A new virtual machine named `Sample_Windows_Server_XXXXX` (where `XXXXX` is a random five digit number) is created with medium flavor.

To get the virtual machine details, click the `Sample_Windows_Server_XXXXX` name in the list that you can access from **RESOURCES > Virtual Machines**.

To access the virtual machine, use username `Admin` and password entered while deploying the virtual machine as credentials. Password must meet complexity requirements.

To modify the notification message template, refer to “(Optional) Modifying the notification message template” on page 86.

Deploying a LAMP stack using multitiered technology

To deploy a LAMP stack with multiple virtual machines and on each virtual machine deploying the specific software for that tier (Apache, MySQL, PHP), perform the following procedure.

Before you begin

Ensure you have a Linux image and key pair configured as described in “Defining a Linux image” on page 85.

Ensure that default LAMP stack Heat template `LAMP stack` is not deleted or incorrectly edited.

Ensure the availability of access to the yum repository from virtual machines for installing the following packages:

- Apache
- MySQL
- PHP

If you have more than one availability zone in your selected region, then the `default_availability_zone` specified in the Nova configuration file is used for the deployment. Ensure that the target OpenStack region has only one network defined. If multiple networks are defined in the region, the offering fails with the error message `Multiple networks found`. Specify a network in the LAMP stack heat template.

The LAMP stack template can be edited using the IBM Cloud Orchestrator Self-service user interface in **CONFIGURATION > Heat Templates** for more customization, for example the networks setting can be specified in the resources section of Heat template.

Important: Do not change LAMP stack as name of the Heat template.

Procedure

1. In the Self-service user interface, click **SELF-SERVICE CATALOG** and open the **Deploy customized cloud services** category.
2. Click the **Deploy LAMP stack** offering.
3. If you have more than one region in your OpenStack environment, select the target region for the deployment and click **OK**.
4. Provide a Stack Name and click **Deploy**.

Results

A LAMP stack is created with two new virtual machine instances with medium flavor. Apache and PHP are installed in one virtual machine, MySQL is installed in the other virtual machine.

To get the stack details, click the stack instance name in the list that you can access from **RESOURCES > Stacks**. To get details on the virtual machine instances, click on each server resource.

To access the virtual machine instances, use the `linux_key` key pair that was created in the procedure described in “Defining a Linux image” on page 85.

Accessibility features for IBM Cloud Orchestrator

Accessibility features help a user who has a physical disability, such as restricted mobility or limited vision, to use software products successfully. The major accessibility features of IBM Cloud Orchestrator are described in this topic.

Accessibility features

The following list includes the major accessibility features in IBM Cloud Orchestrator:

- Keyboard-only operation
- Interfaces that are commonly used by screen readers
- Keys that are discernible by touch but do not activate just by touching them
- Industry-standard devices for ports and connectors
- The attachment of alternative input and output devices

Note: The default configuration of JAWS screen reader does not read tooltips. JAWS users must enable their current mode to read tooltips by selecting **Utilities > Settings Center > Speech Verbosity > Verbosity Level > Configure Verbosity Levels**.

User documentation is provided in HTML and PDF format. Descriptive text is provided for all documentation images.

The knowledge center, and its related publications, are accessibility-enabled.

Related accessibility information

You can view the publications for IBM Cloud Orchestrator in Adobe Portable Document Format (PDF) using the Adobe Reader. PDF versions of the documentation are available in the knowledge center.

IBM and accessibility

See the IBM Human Ability and Accessibility Center for more information about the commitment that IBM has to accessibility.

Notices

This information was developed for products and services offered in the US. This material might be available from IBM in other languages. However, you may be required to own a copy of the product or product version in that language in order to access it.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

For license inquiries regarding double-byte character set (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

*Intellectual Property Licensing
Legal and Intellectual Property Law
IBM Japan Ltd.
19-21, Nihonbashi-Hakozakicho, Chuo-ku
Tokyo 103-8510, Japan*

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some jurisdictions do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM websites are provided for convenience only and do not in any manner serve as an endorsement of those

websites. The materials at those websites are not part of the materials for this IBM product and use of those websites is at your own risk.

IBM may use or distribute any of the information you provide in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

*IBM Director of Licensing
IBM Corporation
North Castle Drive, MD-NC119
Armonk, NY 10504-1785
US*

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

The performance data discussed herein is presented as derived under specific operating conditions. Actual results may vary.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to actual people or business enterprises is entirely coincidental.

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

Programming interface information

This publication primarily documents information that is NOT intended to be used as Programming Interfaces of IBM Cloud Orchestrator. This publication also documents intended Programming Interfaces that allow the customer to write programs to obtain the services of IBM Cloud Orchestrator. This information is identified where it occurs, either by an introductory statement to a chapter or section or by the following marking: *Programming Interface information*.

Trademarks

IBM, the IBM logo, and ibm.com are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the web at "Copyright and trademark information" at www.ibm.com/legal/copytrade.shtml.

Terms and conditions for product documentation

Permissions for the use of these publications are granted subject to the following terms and conditions.

Applicability

These terms and conditions are in addition to any terms of use for the IBM website.

Personal use

You may reproduce these publications for your personal, noncommercial use provided that all proprietary notices are preserved. You may not distribute, display or make derivative work of these publications, or any portion thereof, without the express consent of IBM.

Commercial use

You may reproduce, distribute and display these publications solely within your enterprise provided that all proprietary notices are preserved. You may not make derivative works of these publications, or reproduce, distribute or display these publications or any portion thereof outside your enterprise, without the express consent of IBM.

Rights

Except as expressly granted in this permission, no other permissions, licenses or rights are granted, either express or implied, to the publications or any information, data, software or other intellectual property contained therein.

IBM reserves the right to withdraw the permissions granted herein whenever, in its discretion, the use of the publications is detrimental to its interest or, as determined by IBM, the above instructions are not being properly followed.

You may not download, export or re-export this information except in full compliance with all applicable laws and regulations, including all United States export laws and regulations.

IBM MAKES NO GUARANTEE ABOUT THE CONTENT OF THESE PUBLICATIONS. THE PUBLICATIONS ARE PROVIDED "AS-IS" AND WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING BUT NOT LIMITED TO IMPLIED WARRANTIES OF MERCHANTABILITY, NON-INFRINGEMENT, AND FITNESS FOR A PARTICULAR PURPOSE.

IBM Online Privacy Statement

IBM Software products, including software as a service solutions, ("Software Offerings") may use cookies or other technologies to collect product usage information, to help improve the end user experience, to tailor interactions with the end user or for other purposes. In many cases no personally identifiable information is collected by the Software Offerings. Some of our Software Offerings can help enable you to collect personally identifiable information. If this Software Offering uses cookies to collect personally identifiable information, specific information about this offering's use of cookies is set forth below.

Depending upon the configurations deployed, this Software Offering may use session and persistent cookies that collect each user's user name, or other personally identifiable information for purposes of session management, enhanced user usability, single sign-on configuration. These cookies cannot be disabled.

If the configurations deployed for this Software Offering provide you as customer the ability to collect personally identifiable information from end users via cookies and other technologies, you should seek your own legal advice about any laws applicable to such data collection, including any requirements for notice and consent.

For more information about the use of various technologies, including cookies, for these purposes, See IBM's Privacy Policy at <http://www.ibm.com/privacy> and IBM's Online Privacy Statement at <http://www.ibm.com/privacy/details> the section entitled "Cookies, Web Beacons and Other Technologies" and the "IBM Software Products and Software-as-a-Service Privacy Statement" at <http://www.ibm.com/software/info/product-privacy>.

Glossary

This glossary includes terms and definitions for IBM Cloud Orchestrator.

The following cross-references are used in this glossary:

- See refers you from a term to a preferred synonym, or from an acronym or abbreviation to the defined full form.
- See also refers you to a related or contrasting term.

To view glossaries for other IBM products, go to www.ibm.com/software/globalization/terminology (opens in new window).

A

account code

A code that uniquely identifies an individual, billing, or reporting entity within chargeback and resource accounting.

account code conversion table

An ASCII text file that contains the definitions that are required to convert the identifier values defined by the account code input field to the user-defined output account codes.

account report

A report that is used to show account level information for usage and charge.

availability zone

A logical group of OpenStack Compute hosts. It provides a form of physical isolation and redundancy from other availability zones, such as by using separate power supply or network equipment.

B

Bill program

A program that performs cost extensions within SmartCloud Cost Management and summarizes cost and resource utilization by account code. The Bill program uses the rate code table that is assigned to the client to determine the amount to be charged for each resource consumed.

building block

The model of an image that is created by combining models of a base operating system and software bundles. Each building block contains a semantic and functional model that describes the contents of the components, for example, the installed products, supported operating systems, prerequisites, and requirements.

business object

A software entity that represents a business entity, such as an invoice. A business object includes persistent and nonpersistent attributes, actions that can be performed on the business object, and rules that the business object is governed by.

business process

A defined set of business activities that represent the required steps to achieve a business objective. A business process includes the flow and use of information and resources.

C

chargeback identifier

A label, which is often tied to an algorithm or set of rules, that is not guaranteed to be unique, but is used to identify and distinguish a specific chargeback item or chargeback entity from others.

compute node

A node that runs a virtual machine instance, which provides a wide range of services, such as providing a development environment or performing analytics.

consolidation process

A process during which the data collectors process the nightly accounting

and storage files that were created by the data collection scripts and produce an output CSR file.

conversion mapping

An entry in a mapping table which allows you to map identifiers to accounts or other identifiers.

custom node

A virtual image part that provides an unconfigured node for a pattern that has a deployment manager or a control node as its base.

E

exception file

A file that contains a list of records with identifier names that do not have a matching Parameter IdentifierName attribute value.

exception processing

A process in which the system writes all records that do not match an entry in the account code conversion table to an exception file.

H

human service

An activity in the business process definition that creates an interactive task that the process participants can perform in a web-based user interface.

hypervisor

Software or a physical device that enables multiple instances of operating systems to run simultaneously on the same hardware.

K

kernel The part of an operating system that contains programs for such tasks as input/output, management and control of hardware, and the scheduling of user tasks.

P

parameter (parm)

A value or reference passed to a function, command, or program that serves as input or controls actions. The value is supplied by a user or by another program or process.

parm See parameter.

performance counter

A utility that provides a way for software to monitor and measure processor performance.

primary key

In a relational database, a key that uniquely identifies one row of a database table.

process application

A container in the Process Center repository for process models and supporting implementations. A process application typically includes business process definitions (BPDs), the services to handle implementation of activities and integration with other systems, and any other items that are required to run the processes. Each process application can include one or more tracks.

proration

A process that distributes the overall or individual resources of an account and the cost of those resources across multiple accounts at a specified percentage.

proration table

An ASCII text file that defines the identifier values and rate codes that are used in the proration process.

R

rate code

The identifier of a rate that is used to link a resource unit or volume metric with its charging characteristics.

rate group

A group of rate codes that is used to create rate subtotals in reports, graphs, and spreadsheets.

registry

A repository that contains access and configuration information for users, systems, and software.

S

service operation

A custom operation that can be run in the context of the data center. These operations are typically administrative operations and are used to automate the configuration. Service operations can also be used to enhance the catalog of available services with extra functionality.

software bundle

A collection of software installation files, configuration files, and metadata that can be deployed on a virtual machine instance.

T

toolkit

A container where artifacts can be stored for reuse by process applications or other toolkits.

V

virtual machine (VM)

An instance of a data-processing system that appears to be at the exclusive disposal of a single user, but whose

functions are accomplished by sharing the resources of a physical data-processing system.

VM See virtual machine.



Product Number: 5725-H28

Printed in USA