

Oracle® Cloud

Using Oracle Orchestration



E78055-11
August 2019



Oracle Cloud Using Oracle Orchestration,

E78055-11

Copyright © 2017, 2019, Oracle and/or its affiliates. All rights reserved.

This software and related documentation are provided under a license agreement containing restrictions on use and disclosure and are protected by intellectual property laws. Except as expressly permitted in your license agreement or allowed by law, you may not use, copy, reproduce, translate, broadcast, modify, license, transmit, distribute, exhibit, perform, publish, or display any part, in any form, or by any means. Reverse engineering, disassembly, or decompilation of this software, unless required by law for interoperability, is prohibited.

The information contained herein is subject to change without notice and is not warranted to be error-free. If you find any errors, please report them to us in writing.

If this is software or related documentation that is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, then the following notice is applicable:

U.S. GOVERNMENT END USERS: Oracle programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, delivered to U.S. Government end users are "commercial computer software" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the programs, including any operating system, integrated software, any programs installed on the hardware, and/or documentation, shall be subject to license terms and license restrictions applicable to the programs. No other rights are granted to the U.S. Government.

This software or hardware is developed for general use in a variety of information management applications. It is not developed or intended for use in any inherently dangerous applications, including applications that may create a risk of personal injury. If you use this software or hardware in dangerous applications, then you shall be responsible to take all appropriate fail-safe, backup, redundancy, and other measures to ensure its safe use. Oracle Corporation and its affiliates disclaim any liability for any damages caused by use of this software or hardware in dangerous applications.

Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other names may be trademarks of their respective owners.

Intel and Intel Xeon are trademarks or registered trademarks of Intel Corporation. All SPARC trademarks are used under license and are trademarks or registered trademarks of SPARC International, Inc. AMD, Opteron, the AMD logo, and the AMD Opteron logo are trademarks or registered trademarks of Advanced Micro Devices. UNIX is a registered trademark of The Open Group.

This software or hardware and documentation may provide access to or information about content, products, and services from third parties. Oracle Corporation and its affiliates are not responsible for and expressly disclaim all warranties of any kind with respect to third-party content, products, and services unless otherwise set forth in an applicable agreement between you and Oracle. Oracle Corporation and its affiliates will not be responsible for any loss, costs, or damages incurred due to your access to or use of third-party content, products, or services, except as set forth in an applicable agreement between you and Oracle.

Contents

1 Getting Started with Oracle Orchestration

About Oracle Orchestration	1-1
Before You Begin Using Oracle Orchestration	1-2
Target Audiences	1-2
Useful Concepts	1-2
Integration with Other Oracle Management Cloud Components	1-4
Understanding Oracle Orchestration Roles and Privileges	1-5
Configure Oracle Orchestration	1-6
Perform Pre-requisite Setup Tasks	1-6
Set Up Credentials for Cloud-based Entities	1-6

2 Author Workflows

About Workflows	2-1
Author Workflows Using UI	2-3
Author Workflows Using V2.0 Syntax	2-4
Author Workflows Using V1.0 Syntax	2-6
Workflow Variables and Data Passing	2-9

3 Submit Workflows

Submit a Workflow Using the Orchestration UI	3-1
Submit Workflows Using REST-API	3-2

4 Schedule Workflows

Schedule Workflows Using UI	4-1
CRON Expressions	4-2

5 Manage Orchestration Workflows

6 Integrate Orchestration with Log Analytics

7 Monitor a Workflow Submission

8 Simplifying IT Management

Performing Bulk Configuration Management	8-1
Automating Provisioning and Deployment	8-1
Managing Incidents with Remediation Actions	8-2
Remediation Action Workflow Creation and Submission	8-2
REST API Use Cases	8-13

A Appendix A: Workflow Samples

Example 1: Starting a Web Logic Server	A-2
Example 2: Shutting Down a Web Logic Server	A-2
Example 3: Patch an Oracle Database Server(s) with Input Parametrization	A-3
Example 4: Adding Wait Time in Seconds to a Workflow	A-4
Example 5: Single Step REST Call	A-5
Example 6: Single Step REST Call with Variable Inputs	A-6
Example 7: Single Step REST call with Workers	A-6
Example 8: Two Step Workflow: Execute and Validate REST Calls	A-7
Example 9: Nested Workflows	A-8
Example 10: Remediation Workflow	A-10

Abstract

Documentation for Oracle Orchestration that describes how to use orchestration workflows to automate complex tasks and processes.

Preface

Oracle Orchestration automates tasks and processes within your enterprise.

Topics:

- [Audience](#)
- [Related Resources](#)
- [Conventions](#)

Using Oracle Orchestration describes how to use this service to automate business processes and tasks.

Audience

Using Oracle Orchestration is intended for administrators who want to automate processes and tasks across cloud and on-premises environments.

Related Resources

For more information about Oracle Management Cloud see:

- [Oracle Cloud](#)
- [Getting Started With Oracle Management Cloud](#)
- [Get started with Log Analytics](#)

Conventions

Table Text Conventions

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
<i>italic</i>	Italic type indicates the book titles, emphasis, or placeholder variables for which you supply particular values.
monospace	Monospace type indicates commands within a paragraph. URLs, code in examples, text that appears on the screen, or text that you enter.

1

Getting Started with Oracle Orchestration

Oracle Orchestration is a Platform-as-a-Service (PaaS) solution that lets you schedule, track, and execute *workflow* scripts on hosts or call Web service endpoints, all from a single location. You can schedule recurring maintenance tasks across your on-premises and cloud infrastructure environments.

Oracle Orchestration allows developers, system administrators and DevOps engineers to easily Author, Manage, Submit and Monitor workflows via a comprehensive easy to use UI and integrated JSON editor. Developers can use this Oracle Management Cloud service to integrate a scheduler into their applications. DevOps engineers can use Oracle Orchestration to automate and schedule deployments. System administrators can use Orchestration to rapidly and efficiently respond in an automatic fashion to service interruptions, backups, schedule downtime and restarts. Oracle Orchestration can also provide a simple REST API that allows easy integration with existing tools, and scripts to provision new environments and applications.

Note:

For this release, Oracle Orchestration REST API documentation access will be limited to approved customers. Contact your Oracle Support or Sales Representative for more information about accessing and using the Oracle Orchestration REST API. In addition, see the My Oracle Support note *Orchestration Master Note* (Doc ID 2229523.1) for the latest information.

About Oracle Orchestration

Orchestration is a set of activities performed on multiple entities or hosts to achieve a specific purpose. With Oracle Orchestration, these activities can be performed as a single step across large number of systems.

Oracle Orchestration provides you with a workflow automation tool in Oracle Cloud for system administrators and developers to automate processes and tasks across cloud and on-premises environments. You can define schedules for workflow executions repeatedly or in the future. This service uses the Oracle Management Cloud platform to deliver a highly scalable service that's integrated with other Oracle Management Cloud feature-rich solutions for an on-premises data center or public cloud task automation.

Benefits

- Single solution for on-premises and cloud
- Centralized operations
- Reduced complexity
- Lower investment
- Enhanced capacity prediction and planning

- Customized analysis to produce business insights

Before You Begin Using Oracle Orchestration

Before you use Oracle Orchestration, you should be familiar with the basic concepts of orchestration and setting up your environment.

This chapter covers the following topics:

- [Target Audiences](#)
- [Useful Concepts](#)
- [Integration with Other Oracle Management Cloud Components](#)

Target Audiences

Oracle Orchestration simplifies common tasks performed by IT personnel. Typically, job functions performed by IT personnel can be categorized as: DevOps, system administrators, and developers. Depending on the role that you have within your organization, your approach to learning and using Oracle Orchestration may differ.

The following examples illustrate how the three categories of IT personnel can use Oracle Orchestration to automate and simplify their typical job responsibilities.

DevOps

- Automate configuration tasks by running commands or scripts across distributed environments.
- Provision, patch, deploy, and maintain the IT infrastructure.
- Respond to critical incidents quickly and reduce IT support costs.

System Administrators

- Schedule, organize, and manage jobs or tasks.
- Automate configuration tasks by running commands or scripts across distributed environment.
- Provision, patch, and maintain the IT infrastructure.
- Respond to critical incidents quickly and reduce IT support costs.

Developers

- Automate application deployment.
- Enable continuous integration to trigger the post-build deployment process, including verification.

Useful Concepts

Before using the Oracle Orchestration, you should be familiar with the following concepts:

- **Orchestration:** A set of activities performed on multiple entities or hosts to achieve a specific purpose. With Oracle Orchestration Cloud Service, these activities can be performed as a single step across a large number of systems.

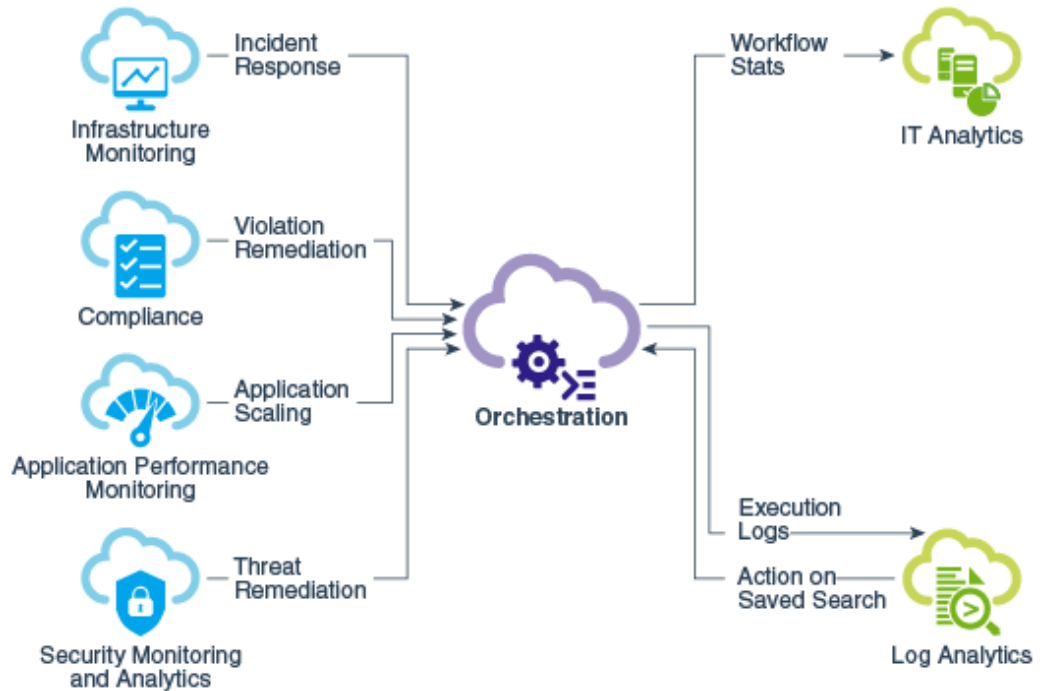
- **Workflow:** The specific pattern of activity that needs to be performed on a system or a set of systems. A workflow can be viewed as a fundamental building block for a business activity that can be automated by Oracle Orchestration Cloud Service. Workflows allow you to automate tasks and activities. For example, creating an instance on Amazon Web Services (AWS) to spin an Infrastructure-as-a-Service (IaaS) instance is a workflow. A workflow is formatted in JSON and submitted via the Orchestration UI. To view the different workflow elements and JSON examples please refer to [Author Workflows Using V2.0 Syntax](#).
- **Workflow Execution:** A workflow execution is a collection of multiple *workers* based on the worker specification in the workflow submission. Oracle Orchestration Cloud Service creates an instance of a workflow execution based on the schedule specified in the workflow submission.
- **Workflow Submission:** The instance of a workflow that's submitted using the *Submission* API of the Oracle Orchestration Cloud Service. A workflow submission payload consists of *plan*, *steps*, *input*, *workers*, *credential reference*, *schedule* and *notification*. Orchestration uses this definition of workflow to spawn one or more executions based on its schedule across multiple entities. These operations could be scheduled to run at the earliest time (or immediately), at a specified time, or at repeat specified intervals. The orchestration engine uses an appropriate scheduler to calculate the necessary *expected* execution time (cloud-based CRON). The workflow submission can be used to group and analyze multiple past executions and the executions scheduled in the future.
- **Workflow Plan:** A plan is a structured list of steps, where each step is a task that needs to be performed on an entity. All workflows from single step to multistep have a plan section. Steps are a set of instructions within the plan that are executed by the workers.
- **Workflow Step:** The workflow step is the unit of activity that executes on the destination entity or web-service endpoint based on the step type. A step within a workflow has a name, type (REST, ENTITY, WAIT, WORKFLOW), input parameters, body, and authentication details. Steps are conformed by four parts PRE, OPERATION, POST and PUBLISH. When the step is executed, it generates a result that indicates its success or failure status along with output.
 - **REST Step Type:** This step type lets you call a RESTful web-service endpoint using Oracle Orchestration Cloud Service. You specify the *URI*, and request input and authorization details in the workflow definition. For instance, you might retrieve the list of objects or files from the Oracle Storage Service. For a REST-based step, the HTTP status code determines the result of the step, while the output is determined by the HTTP response body.
 - **ENTITY Step Type:** This step type lets you execute host-based operations within an Oracle Management Cloud-discovered entity using Oracle Management Cloud agent. For instance, execute a Chef recipe to update a server's configuration or patch Oracle database servers. The definition of the operation to be performed (program location, arguments, credentials, stdin, etcetera) would be under your control. You would specify the command and arguments used to spawn the process; the destination host where the process needs to be spawned; the host credentials that the process needs to execute as and also pass input via the process' standard input. The host of the process that may be spawned could either be the host that monitors the entity or the host that the monitoring agent of the entity resides on. The entity step type allows certain contextual information of the entity to be passed on to the spawned process using the command, arguments or the

standard input of the process. For an entity step, the output is the stdout of the spawned process on the host.

- **WAIT Step Type:** This step type allows you to add an intended delay in the execution flow with the command *waitInSeconds*. A minimum wait time of 5 seconds and maximum wait time of 1800 seconds (or 30 minutes) can be entered. Decimal and partial seconds are not allowed, expressions must always be expressed in seconds and not minutes. WAIT step types have no output.
- **WORKFLOW Step Type:** This step type allows you to execute nested workflows, or workflows within workflows in a Parent-child manner. The nested workflow to be called from the parent will be delineated within the WORKFLOW step type, and will require a name for the nested workflow, the name of the workflow in the workflow library and a version and inputs parameters to be sent to the child workflow.
- **Workflow Worker:** Workers define the level of parallelism for the entire workflow. Each workflow worker runs its own instance of the workflow in parallel. There is no "crosstalk" or synchronization of data between workflow worker instances. A workflow may instantiate one or more workflow workers. Consider an ENTITY use case where a set of steps needs to be executed across 10 different hosts. Similarly, consider a use case, where a REST call needs to be made to 10 different endpoints. For both cases, the same set of steps needs to be executed with different host names/port numbers. Each execution is a *worker*.
- **Input Parameters:**
 - **String:** Text based input, it is conformed by a set of characters like letters, numbers and spaces.
 - **Boolean:** Input based on only two values, true and false. Does not allow any other character.
 - **Integer:** Number based input, is comprised of whole (no decimal place) numbers.
 - **Sensitive:** Allows for sensitive information to be sent to either an agent (ENTITY) or thru a REST step type. Sensitive information should be encrypted using a symmetric key provided on the agent or target. Sensitive input parameters are only allowed in *stdin* and not allowed under arguments for security reasons.
- **Oracle Management Cloud Agent:** A Cloud Agent collects metrics for targets and enables workflow entity steps to be executed on the target hosts.

Integration with Other Oracle Management Cloud Components

Oracle Orchestration is tightly integrated with Oracle Log Analytics. Orchestration workflow executions produce logs and these logs provide valuable information for understanding failures and forecasting execution times. Oracle Orchestration automatically retains workflow logs and generated output for seven days. For additional retention and analysis, logs can optionally be sent directly to Oracle Log Analytics where they're automatically parsed and efficiently stored. You can view Oracle Orchestration workflow logs alongside other log sources to understand and correlate outside factors affecting performance parameters such as execution time and service failures.



Understanding Oracle Orchestration Roles and Privileges

Once you are an Oracle Cloud customer and you create an Oracle Management Cloud instance, the following user roles are provisioned:

- Oracle Management Cloud Administrator
- Oracle Management Cloud User

Table 1-1 Roles for Oracle Orchestration

Role	Tasks
Oracle Management Cloud Administrator	<p>With this role you can perform the following tasks:</p> <ul style="list-style-type: none"> • Author, Publish and Manage Workflows in the Library • Submit a Workflow • View and Monitor Workflow Submissions and associated Workflow Executions, including output results.

For more information about the tasks that the users assigned with the above roles can perform, see *Adding Users and Assigning Roles* in *Getting Started with Oracle Management Cloud*.

Currently in Orchestration, no objects are shared across tenants. Workflow submissions cannot be seen by users on other tenants.

Configure Oracle Orchestration

To use Oracle Orchestration, you need to perform some initial setup.

Topics:

- [Perform Pre-requisite Setup Tasks](#)
- [Set Up Credentials for Cloud-based Entities](#)

Perform Pre-requisite Setup Tasks

To get started with Oracle Orchestration, you must enable the license editions and install the cloud agents. Cloud agents are installed on both physical and virtual hosts and are used to collect data and allow workflows to properly execute. To complete these tasks, you must have the Oracle Management Cloud Administrator role.

1. Install a gateway on a host in your data center (the host should have internet access to Oracle Management Cloud). This is an optional task.

A gateway acts as a channel between Oracle Management Cloud and all other Oracle Management Cloud agents. See *Install a Gateway in Installing and Managing Oracle Management Cloud Agents*.

2. Install cloud agents on the hosts you want to monitor or perform administration tasks.

Cloud agents collect availability, performance and configuration data from the entities that run on those hosts.

3. Add entities

Add the entities that your cloud agent will monitor. Oracle Orchestration performs workflow tasks on entities discovered within your enterprise. See *Adding Entities for Infrastructure Monitoring in Using Oracle Infrastructure Monitoring*.

4. Assign licenses

Orchestration is part of Oracle Management Cloud Enterprise Edition. Oracle Orchestration needs the Enterprise Edition enabled on entities (at least the host entity) to properly execute workflows. See *Auto-Assign License Editions to All Entities in Getting Started with Oracle Management Cloud*.

Option 1: Enable license editions from the Oracle Management Cloud console. Assign the Enterprise Edition automatically to all your entities. See *Auto-Assign License Editions to All Entities in Getting Started with Oracle Management Cloud*.

Option 2: If you do not enable auto-assignment of license editions, assign the Enterprise Edition license to selected entities. Assign this license to least to the host entity you will be running workflows on. See *Assign License Editions to Selected Entities in Getting Started with Oracle Management Cloud*.

Set Up Credentials for Cloud-based Entities

Oracle Orchestration allows for a large host of credentials to be setup. There are two entity types that you can define credentials for: Cloud-based entities and Agent-based entities.

Set Up Credentials for Cloud-based Entities:

- Oracle Management Cloud Credentials Store securely stores credentials for operations against cloud-based entities. These credentials require appropriate authorization tokens for the cloud-based endpoints to acknowledge the request and perform the operation appropriately. Each endpoint potentially may support different forms of authorization schemes and each authorization scheme would require different types of credentials for the authorization. To create and manage credentials in the OMC credentials store, navigate to the OMC Global Menu, select *Administration*, navigate to the *Credential Store* and click on *New Credential*.

Set up Credentials for Agent-based Entities:

- Entity-based workflows execute host-based scripts that may run shell scripts on the given host entity that may further call SQL*Plus on a database. These operations require you to provide the reference of entity `HostSSHCreds` credentials in the workflow definition so that Oracle Orchestration can perform required authentication on your behalf as the one who submitted the operation. `HostSSHCreds` credentials are highly sensitive and can't be stored in Oracle Cloud, so maintenance of these credentials must be handled locally at each Cloud agent. A credential definition on the agent has a unique name and can be specified in the context of a single target as key/value pairs. The cloud agent can add, update, and remove a credential. The agent also can add an alias or remove an alias. To create and manage credentials in a cloud agent credential store, see *Manage Agent Credentials* in *Installing and managing Oracle Management Cloud Agents*

2

Author Workflows

A workflow defines a series of tasks or activities that can be automated using Oracle Orchestration. A workflow may consist of one or more steps and may use multiple workers or call upon other workflows to complete its tasks.

To learn more about workflows, how they are structured and how they work; see [About Workflows](#).

Note:

This book has two versions of workflow syntax: V1.0 and V2.0. Currently V1.0 is only included in this document for remediation action support. Oracle Orchestration requires the use of V2.0 syntax and definitions going forward for all workflows authored.

- [Author Workflows Using UI](#)
- [Author Workflows Using V2.0 Syntax](#)
- [Author Workflows Using V1.0 Syntax](#)
- [Workflow Variables and Data Passing](#)

About Workflows

Depending on the complexity of the activity, a workflow may consist of one or more steps. The general execution model for workflows combines parallel and serial actions to allow for any level of serial/parallel actions to be combined.

Some Important Terms

Before looking at how workflows are structured, you need to understand some workflow-related terms and concepts.

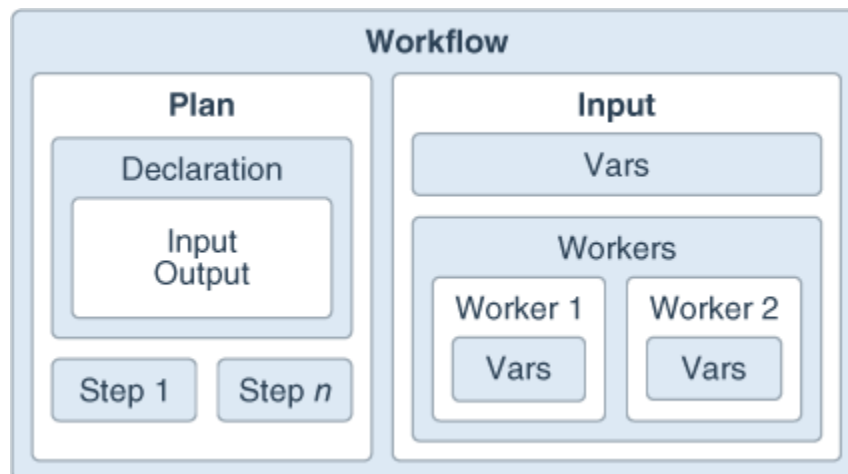
- **Workflow** : Defines a set of orchestrated tasks or activities that can be automated using Oracle Orchestration, being executed by one or more workers.
- **Workflow Name**: This name is an identifier you can use to search for a workflow submission for status and results. All workflows in Orchestration must have a unique name.
- **Workflow Library**: A persistent storage area where you can store and retrieve workflows. Workflows can have one or more versions associated with them, and can also be in a *published* or *draft* state. Workflow Library supports the complete life cycle management of workflows enabling the users to reuse the workflows by publishing and managing these workflows. Workflows can be updated and published as a newer version; only the latest version will be active, meaning there can only be one active version of a workflow. Workflows are accessed via the Workflow Library UI .

- **Workflow Submission:** An immediate or scheduled execution of a specific workflow.
- **Workflow Submission Name:** Users can provide a custom name for the workflow submission, helping identify it. The workflow submission name is not the name of the workflow itself and does not need to be unique.
Example: *Run the Patch DBCS workflow*
- **Workflow Declaration:** The workflow declaration contains the code that defines the workflow's logical execution. Defining them as input and output definitions, including their validation rules., with appropriate properties, which can then be referenced within the plan section using UEL expressions.
- **Plan :** Describes all the steps and the input for a workflow. Supports the user, abstracting out the run time context of a workflow.
- **Step:** Defines an individual unit of activity (step) that executes as part of the workflow execution. Example steps types include ENTITY, REST and WAIT.
 - **Operation Stage:** Defines the main action of the step.
 - **PRE, POST, OPERATION, PUBLISH:** Optional stages defined as part of the Step Execution Flow.

Workflow Structure

Workflows are expressed as a single JSON files, stored in the **Workflow Library** or as an inline **Workflow Submission**. The following JSON structure defines a single workflow/plan. The following graphic shows how a workflow is organized.

Figure 2-1 Orchestration Workflow Structure



Workflow Execution

Workflows executions combine parallel and serial actions to allow for any level of serial/parallel step action. Additionally, workflow steps may reference other workflows, which lets you componentize or modularize workflows; and for a workflow administrator to assemble complex activities from smaller well-defined workflow *building blocks*.

Workflow Characteristics and Behaviors

- A workflow may instantiate one or more **workflow workers**. The number of workflow workers defines the level of parallelism for the entire workflow.
- Each workflow worker runs its own instance of the workflow in **parallel**. There is no "crosstalk" or synchronization of data between workflow worker instances.
- Each workflow may have one or more steps.
- Each step runs in **serial** with every other step in the same workflow plan.
- Each step may have one or more **step workers** (like a workflow).
- Each step worker runs its own instance of the step in **parallel**. There is no crosstalk or synchronization of data between other step instances.
- Steps may combine the results of all completed steps per step worker into one single result.
- Workflows may combine the results of all completed workflows into one single result.
- Workflows may publish the results of the entire workflow (this is critical for embedded workflows, the result of a workflow publish is synonymous with the results of a published step).

Author Workflows Using UI

A workflow defines tasks or activities that can be automated using Oracle Orchestration.

To create a workflow using the Oracle Management Cloud UI, follow these steps:

1. Navigate to the Workflow Library on the left hand hamburger menu. The main Workflow Library screen will appear.
2. Click on the plus "+" sign on the left hand corner. A new screen will appear with the workflow editor.
3. Select a unique name for your workflow and enter a brief description of what your workflow does. Once done, click Next
4. In the JSON definition screen, enter your workflow code in JSON format.
5. Once completed you can save the workflow as either *Draft* or *Publish*. To submit or run a published workflow please view [Manage Orchestration Workflows](#).

To learn more about workflows, how they're structured and how they work. See: [About Workflows](#). For information on authoring UI enabled workflows see: [Author Workflows Using V2.0 Syntax](#)

Note:

It is recommended that you author workflows using the V2.0 syntax. V1.0 syntax is currently deprecated and only noted for legacy purposes. V1.0 workflows **cannot** be created using the UI interphase.

Author Workflows Using V2.0 Syntax

A workflow is the fundamental building block for a business activity that can be automated using Orchestration. Workflows allow you to automate tasks and activities; in this topic we will show you the main elements of a workflow, how a workflow is structured and JSON code examples. Providing you with the ability to:

- Define multiple steps within a single workflow plan. See: Example 8: Two Step Workflow Execute and Validate REST Calls
- Increase workflow re-usability, allowing the use of input variables. See: Example 6: Single Step REST Call with Variable Inputs
- Allow the workflow to reference variables via [Workflow Variables and Data Passing](#).
- Chain and nest workflows, users can call another workflow from within a step execution.

V2.0 Workflow Elements

A workflow is composed of several elements, the following definitions explain in hierarchical order the primary elements used to define a workflow.

1. **Workflow Name:** A name given to the workflow. This name is a unique identifier of the workflow in the library.
2. **Plan:** A plan is a list of steps where each step is a task that needs to be performed on an ENTITY. A plan can also contain an input declaration to define workflow inputs.
Input Declaration: Allows the workflow author to define the workflow inputs to be referenced within the workflow through parametrization. The workflow author can also, specify the input types like *string*, *integer* and *boolean*. Input types can also be marked as required based on the author's needs.

Types of Steps:

- **REST:** Used for REST URI workflows. This step type has 3 parameters:
 - *URI*, the REST URI to use, this is a **required** parameter.
 - *Method*, either GET or POST, this is a **required** parameter.
 - *Headers* are a map of key/value pairs, this is an optional parameter.

For REST sample workflows view: Example 5: Single Step REST Call and Example 7 Single Step REST call with Workers

- **ENTITY:** A type of step that executes actions within a discovered entity. Its parameters have the following properties:
 - *Credential:* Credentials are required to authenticate against the entity where the command will be executed. You will need to reference the name of the credential created in the *agent credential store* associated with the entity.
 - *Command:* To be executed on agent.
 - *Arguments:* Inputs to be sent to the command. Multiple inputs can be sent as comma separated values.
 - *stdin* : Input sent to the command via standard input.

- *Entity*: where the command needs to be run. An agent monitored host or database.
- *stdout*: standard output from the command execution.
- *commandTimeout* is used when a custom timeout in an entity workflow is needed to terminate a step. Time is expressed in left-to-right granularity *h*: hours, *m*: minutes and *s*: seconds; valid examples: 5m, 1h 10m, 30s. Timeouts cannot exceed 24 hours in length, must not have negative components, or spaces between number and unit.

For ENTITY sample workflows view: Example 1: Starting a Web Logic Server and Example 2: Shutting Down a Web Logic Server

- **WAIT**: A wait type allows a user intended delay in the execution flow with the command.
 - *waitInSeconds*. Minimum wait time of 5 seconds and maximum wait time of 1800 seconds can be entered (decimals are not allowed).

For a WAIT sample workflow view: Example 4: Adding Wait Time in Seconds to a Workflow

- **WORKFLOW**: A workflow type allows workflows to be nested in a parent-child manner. A nested workflow needs the following design specifications:
 - *name*: Name of the nested workflow (different from parent name).
 - *workflowName*: Name of the workflow to be called in the Workflow Library
 - *workflowVersion*: Version of the workflow in the Workflow Library. This parameter is optional.
 - *input*: Input to be sent to the nested workflow. This input is needed when variables from the parent workflow need to be passed to the child workflow. Input variables entered here will be overridden by variables defined in the Workflow Library.

For a nested Parent-Child sample workflow view: Example 9: Nested workflows Parent Child

Step Components:

- **PRE**: Prerequisites that need to be completed or guaranteed for the step to start execution. Supported commands:
 - *Skip*, if the prerequisites are not met a step can be skipped.
 - **POST**: Validates step condition, by verifying if the step successfully did what it was intended to do. Supported Commands:
 - *Retry* retries the step until successful execution.
 - *Abort* aborts the step if the post condition is not met.
 - *Skip* skips the step entirely if the step conditions are not met.
 - **OPERATION**: Actions to be performed by the step. This is based on the step type defined (*REST*, *ENTITY*, *WAIT* or *WORKFLOW*).
 - **PUBLISH**: Information to be passed on to the subsequent steps to use via variables.
3. **Input**: The input section allows the author to specify any default inputs to the workflow corresponding with the workflow input declaration. User inputs provided

during workflow submission will be automatically passed upon workflow submission. Input may contain two elements:

- *vars*: Is a JSON object for defining variables, whose keys will be made available as global variables for the workflow.
- *workerVars*: Is a JSON object, the number of workers and their names are defined by the keys. Every key must have a value. *workerVars* has the same structure as *vars* defining local worker variables. The values defined in *workerVars* override the values defined in the global *vars* object for any key that shares the same name. An empty object must be provided if there are no intended overrides.

For a sample workflow using input please view: Example 3: Patch an Oracle Database Server with Input Parametrization

4. **Workers:** Each workflow worker runs its own instance of the workflow in parallel. There is no "crosstalk" or synchronization of data between workflow worker instances. For a sample worker workflow please view: Example 7: Single Step REST call with Workers
5. **Notification:** A list of email addresses that will receive workflow status updates.

For sample workflows view Appendix A: Workflow Samples.

Author Workflows Using V1.0 Syntax

A workflow is the fundamental building block for a business activity that can be automated using Orchestration it allows you to automate tasks and activities.

Note:

Currently V1.0 syntax is only included in this book for remediation action support. Oracle requires using V2.0 syntax and definitions going forward for all workflows authored. **Version 1.0 is not supported via the UI.**

The following is the standard V1.0 JSON structural format to define a workflow:

```
name
workflow
  plan
    name
    steps
      step
        step payload
        uploadToLA
        logSource
    input
      workflow workers (multiple workers are supported)
  workflowProperties
    property 1
    property 2
    ...
    property n
```

schedule
 notification

Table 2-1 Workflow Elements



Element	Description
Name	<p>A name given to the workflow submission. The name is an identifier you can use to search for a workflow submission for status and results.</p> <div data-bbox="1162 541 1458 856" style="border: 1px solid #0070C0; padding: 10px; background-color: #E6F2FF;"> <p> Note:</p> <p>The workflow submission name does not have to be unique within a given tenant scope.</p> </div>
Plan	<p>A list of steps where each step is a task that needs to be performed on an entity or a REST call.</p> <ul style="list-style-type: none"> • Each step has a name, type (REST/ ENTITY) and step-specific input. • Step-specific input can reference the variables declared in the <i>input</i> section of the workflow. <p>When defining a step, you can also specify that workflow details be sent to Oracle Log Analytics by setting the <i>uploadToLA</i> property to <i>true</i>. Additionally, you can have the uploaded logs associated with an existing log source known to Log Analytics by setting the <i>logSource</i> property.</p>

Table 2-1 (Cont.) Workflow Elements

Element	Description
Workflow Properties	<p>A map of key-value pairs that can be associated with the workflow submission. These key-value pairs let you associate keywords that will help in searching for the workflow submission at some later point in time.</p> <div data-bbox="1084 548 1380 890" style="border: 1px solid #0070C0; padding: 10px; background-color: #E6F2FF;"> <p> Note:</p> <p>You can specify up to five tags using the names <i>property1</i>, <i>property2</i>, <i>property3</i>, <i>property4</i>, and <i>property5</i>.</p> </div>
Input	The input for the workflow. Input may consist of <i>vars</i> and <i>workerVars</i> .
Workers	<p>Workers define the level of parallelism for the entire workflow:</p> <p>Consider a use case (ENTITY) where a set of steps needs to be executed across 10 different hosts. Similarly, consider a use case (REST) where a REST call needs to be made to 10 different endpoints. For both cases, the same set of steps needs to be executed with different host names/port numbers. Each execution is a <i>worker</i>.</p> <p>All steps specified in the plan (for the current release, only one step is permitted) would be executed for each worker using the specified input values.</p>
Schedule	<p>The time when a workflow should be executed.</p> <p>For example, say a workflow needs to be executed every day at 9:00 PM or every Saturday at 10:30 PM. For both cases, schedule information is passed in the workflow submission so that the workflow can be executed at the scheduled time.</p>
Notification	A list of email addresses that will receive workflow status updates.

Example 1: V1.0 Host Patching Sample Workflow

```
{
  "name": "Patch Hosts Demo",
  "workflow": {
```

```

    "plan": {
      "name": "Patch_Hosts_WF",
      "steps": [{
        "name": "patch",
        "type": "ENTITY",
        "input": {
          "command": "/bin/sh",
          "arguments": ["/scratch/patch.sh"],
          "credential": "emcosTestCred"
        }
      }
    ]
  },
  "input": {
    "workers": {
      "thread1": {
        "entity": {
          "entityByName": {
            "name": "example.com",
            "type": "omc_host_linux"
          }
        }
      }
    }
  }
}

```

Workflow Variables and Data Passing

Workflows allow the use of variables both with internal data and from external output data.

Templating Language UEL 3.0

To facilitate specifying references to internal data and data structures, the JSON workflow specification supports expanding dynamic data values via a templating language (UEL 3.0). Templating language supports basic object navigation and expressions, like:

```
"${workflow.omc.startTime}"
```

The following examples illustrate templating language expressions:

Example: Map and Array Navigation

```
"${steps['REST step'].result.body}"
"${steps.provisioning.result}"
"${retry[0].vars.result}"
```

Example: Expressions

```
"${2 + 2}" // => 4
"${'foo' + 'bar'}" // => myoutput
```

Data Passing and Data Extraction from JSON Variables

When a step (REST, ENTITY or WORKFLOW) is executed the output produced can be recovered by the workflow in the form of a single variable. To be able to extract the required variable from the step result you will need to use the following command:

- For JSON result types:

```
"${steps.step1.result.myPublish.<json node name>}"
```

- For REGEX case:

```
"${steps.step1.result.myPublish[0]}"
```

3

Submit Workflows

 **Note:**

For this release, Oracle Orchestration REST API documentation access will be limited to approved customers. Contact your Oracle Support or Sales Representative for more information about accessing and using the Oracle Orchestration REST API. In addition, see the My Oracle Support note *Oracle Orchestration Master Note* (Doc ID 2229523.1) for the latest information.

When you submit a workflow, one or more instances of that workflow are executed immediately or based on the workflow schedule (if defined). Each execution runs the steps using workers, both are defined in the workflow JSON.

There are two ways to submit a workflow in Orchestration:

- [Submit Workflows Using UI](#)
- [Submit Workflows Using REST-API](#)

Submit a Workflow Using the Orchestration UI

Submitting workflows using the new Orchestration UI is quite simple and intuitive.

To submit a workflow using the new UI interface, you need to navigate to the Workflow Library on the left-hand menu.

1. From the main workflow table, select the workflow you wish to submit and click on the right side drop down menu. Scroll to submit and click; this will take you to the workflow submission screen.
2. Submission Name will be defaulted to <Workflow Name>_<current timestamp>
3. Under the Global Inputs section, you will be prompted to provide the user inputs based on workflow declaration.
4. You can add workers to your workflow submission to create parallel workflow instances. Click on add to add the required number of workers and press the modify values button to set values for each worker.
5. Schedule, will allow you to set the time frame and period for the workflow to run. It can be set up as a onetime run, recurring or you can set a complex run schedule using CRON expressions.
6. To schedule, set the start date, time and time zone. If recurring set the Repeat schedule by selecting from the drop-down menu. Finally set the end time for the workflow.
7. Once all parameters are entered click on the Submit button in the top right.

Submit Workflows Using REST-API

REST API availability in Orchestration is currently limited to a select number of customers, for more information about REST API availability, see the My Oracle Support note *REST API for Oracle Management Cloud Orchestration Cloud Service* (Doc ID 2243981.1).

4

Schedule Workflows

As a cloud-based system, Oracle Orchestration is easier to manage and provides a more secure replacement for a Linux CRON utility, or as a replacement for legacy tools such as Control-M or HP Operations Orchestration. Oracle Orchestration gives you a simple UI interface to run both on-demand and scheduled operations tasks.

Oracle Orchestration's workflow scheduling capability lets you:

- Schedule and organize tasks that were previously triggered manually.
- Provide visibility into workflows that were previously run through CRON or other custom tools.
- Replace expensive and hard to manage legacy workflow schedulers.
- Seamlessly execute jobs across both cloud and on-premises IT data center assets.
- As part of Oracle Management Cloud, provide rich integration with Oracle Log Analytics Cloud Service and Oracle IT Analytics Cloud Service.

Examples of Scheduled Workflows

- Operating System: Log rotation, backups.
- Shut down development compute instances at night.
- Database Automatic Workload Repository (AWR) Generation, Data Guard Monitor, Log File Manager, Gather Statistics.

Schedule Workflows Using UI

Scheduling workflows, will allow you to set the time frame and period for the workflow to run. It can be set up as a onetime run, recurring or you can set a complex run schedule using CRON expressions.

To schedule a workflow in the Workflow Library, start by selecting a workflow and clicking Submit from the drop down menu. This will take you to the Workflow Submission screen, here you can set the schedule; set the start date, time and time zone. You can set your schedule to be One time, Recurring or CRON expression. For recurring you can choose from 15 minutes, 30 minutes, Hourly, Daily, Weekly, Monthly or Custom Interval. If recurring set the Repeat schedule by selecting from the drop-down menu. Finally set the end time for the workflow. Once all parameters are entered click on the Submit button in the top right.

To set up a CRON expression, select CRON and enter the character string as outlined in [CRON Expressions](#).

CRON Expressions

A CRON expression is a string that specifies a time schedule for triggering a workflow in Oracle Orchestration. It consists of six required fields and one optional field, each separated by a space, which together specify when to trigger the event.

Table 4-1 CRON Expression Syntax:

Field	Values	Special Characters
Seconds	0-59	, - * /
Minutes	0-59	, - * /
Hours	0-23	, - * /
Day of Month	1-31	, - * ? / L W
Month	1-12 or JAN-DEC	, - * /
Day of Week	1-7 or SUN-SAT	, - * ? / L #
Year (optional)	empty or 1970-2199	, - * /

CRON Syntax Special Characters

- * : Selects all values within a field. For example, every minute, every day or every hour.
- ? : No specific value, it is used when no specific value is needed for a field. If the workflow needs to run on the 15th of each month, but the day of the week is not important; “?” would be the valued for my day of the week field and 15 would be the value for my day of the month field.
- L : Last, “L” in the day-of-month field means the last day of the month. In the day-of-week field by itself, it means Saturday (7). If the value L is used in the day of the week field after another numerical value, it means the last specified day of the month.
- W : Weekday, specifies the weekday (Monday-Friday) nearest the given day. The full syntax is “number”W. Meaning it will run the nearest weekday to “number”, for example 5W.
- - : Specifies a set range of time. For example a 10-12 hour window starting at 10, ending at 12.
- , : Specifies additional values. For example, “MON,WED,FRI”.
- / : Specifies incrementing values. Full syntax is: “number”/ “number”, the first value indicates the start and the second value the increment. 0/20 in the seconds field, for example, means “0, 20, 40”, and 1/3 in the day field means that it will run every 3 days from the first of the month.
- # : Specifies the “nth” day of the month. Full syntax is “number”#“number”, where the first number is the day of the week (0: Sunday, 7:Saturday) and the second number is the month. For example 5#2 would run on the second Friday of the month.

CRON Expression Format and Examples

As we previously mentioned a CRON expression is made up of seven fields, six are mandatory and an optional field. The fields can contain any allowed value established in the table at the top, as well as any special character allowed in that field. The CRON string is expressed by separating the fields using a single space and entering the values for each field together interacting special characters and number into a sub string.

Here are some examples for CRON Expressions

- Run daily at 6am: 0 0 6 * * ?
- Run daily at 6am and 6pm: 0 0 6,18 * * ?
- Run every 15 minutes: 0 0/15 0 * * ?
- Run at 6:30pm on the last day of the month: 0 30 18 L * ?
- Run on the 3rd Friday of the month at 4:30am: 0 30 4 ? * 6#3
- Run on the last day of the month at 2pm: 0 0 14 L * ?

5

Manage Orchestration Workflows

When managing Orchestration workflows a user can Create, Read, Update or Delete (CRUD) a workflow or set of workflows. These CRUD operations support the complete lifecycle management of workflows. Further management actions such as Deactivate, Schedule and Download can also be performed.

These operations, can be achieved within the main Workflow Library page under Orchestration.

- **Author (Create):**
To author a new workflow in the Workflow Library, select the My Workflows tab. This tab will show all your Active and Inactive workflows. Click on the + icon on the top left corner and follow the steps in [Author Workflows Using UI](#) to create a new workflow.
- **Submit (Run):**
To submit a workflow in the Workflow Library, select the My Workflows tab. Search for the workflow you wish to run, click the drop-down menu on the left-hand side and click on Run.
- **Publish:**
Workflows in draft first need to be published before they can be submitted for execution. Navigate to the workflow library and select the *Drafts* tab. Select the workflow to publish, click the drop-down menu on the left-hand side and click on Publish. The workflow will now be visible under the *My Workflows* tab ready to be submitted.
- **Update:**
To update a workflow, navigate to the workflow library and select the workflow you wish to edit. Click on the drop down menu on the left hand side and click on Update. This will take you to the Workflow Editor page where you will perform the updates. Be aware that you will also need to update the version number within the JSON Definition to be able to save or submit the updated workflow.
- **Deactivate:**
To deactivate a workflow in the Workflow Library, click on the Deactivate command by clicking the drop-down menu on the left-hand side. This disables the user from making any future submissions. A deactivated workflow will need to be activated in order to allow any future submission.
- **Delete:**
To delete a workflow in the Workflow Library, select the My Workflows tab and click the drop-down menu on the left-hand side of the workflow to delete and select Delete. Deletions are permanent and unrecoverable.
- **Download:**
To download a workflow in the Workflow Library, select the workflow to download by clicking the drop-down menu on the left-hand side and click on Download. This will download the JSON syntax.
- **Schedule:**
To schedule a workflow in the Workflow Library, start by selecting a workflow and clicking Submit from the drop down menu. This will take you to the Workflow

Submission screen, here you can set the schedule to be One time, Recurring or CRON expression. Recurring you can choose from 15 minutes, 30 minutes, Hourly, Daily, Weekly, Monthly or Custom Interval. For CRON expression details review: [CRON Expressions](#).

6

Integrate Orchestration with Log Analytics

Oracle Log Analytics lets you monitor, aggregate, index, analyze, search, explore, and correlate all log data from your applications and system infrastructure. Oracle Orchestration can automatically upload workflow execution logs to Log Analytics for further analysis, allowing administrators to fine tune workflows and extract important execution information.

 **Note:**

To successfully integrate Orchestration with Log Analytics, as a pre-requisite discovered entities to be analyzed must be licensed with Log Analytics. For further information on Log Analytics, please view [Get Started With Log Analytics](#).

To enable Log Analytics data gathering, navigate to the [Workflow Submission](#) page, scroll down and expand the *Log Analytics* section and check mark *Upload output to Log Analytics*. Once complete click the *submit* button on the top right corner. The *uploadToLA* flag will now be added to the JSON code and you can run the workflow with Log Analytics data gathering enabled.

 **Note:**

Log Analytics can generate a large amount of data, before you enable the *uploadToLA* flag make sure there is enough available disk space in your log source location.

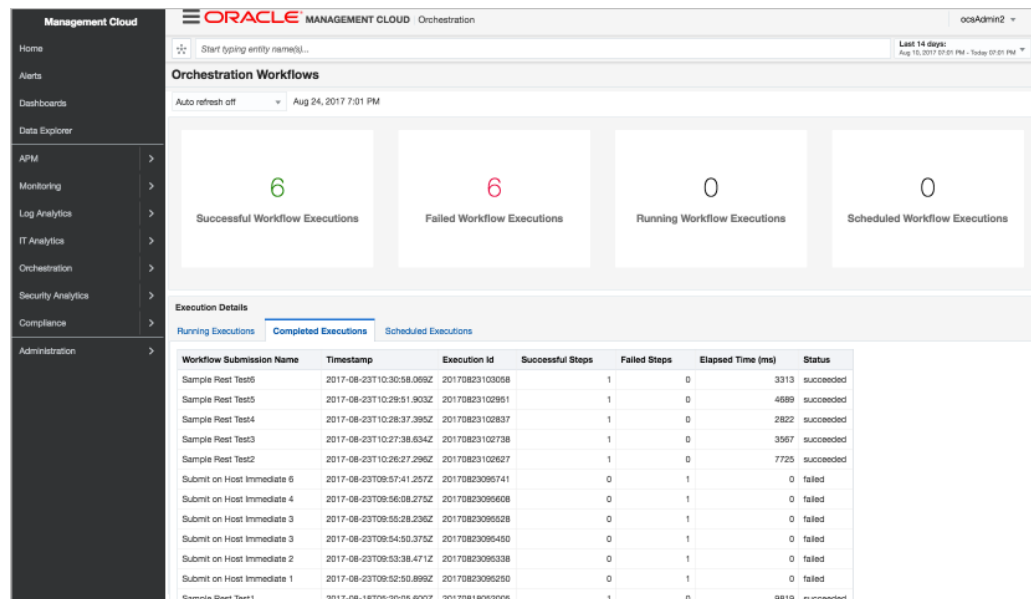
Once the *Upload output to Log Analytics* has been turned on, the *uploadToLA* flag will be included in the JSON code at the **submission** level.

Once the log analytics enabled workflow has successfully executed, its details can be viewed on the Log Analytics page. For further information on Log Analytics, please view [Get Started With Log Analytics](#).

7

Monitor a Workflow Submission

Oracle Orchestration gives you an intuitive way to view and analyze both the current and historic status of workflow submissions using the Orchestration Workflows dashboard. The Orchestration Workflows dashboard provides an overview of all workflow submissions and execution activity. Each widget and data grid provides valuable information and insight into the status and health of your workflows.



The information displayed in a widget or data grid pertains to a specific date range, which lets you view workflow behavior more accurately. Each widget and data grid has its own time selector.

The *summary* region provides a count of the total number of workflow executions as well as a breakdown of the total by status (successful, failed or running).

Execution Details contains detailed information for *Running*, *Completed* and *Scheduled* workflow executions.

- Running Executions displays the status and step details for workflow executions that are executing but not yet completed.
- Completed Executions displays detailed information about workflow executions that have completed all steps included in the overall elapsed time and step count by status.
- Scheduled Executions displays a list of scheduled workflow submissions including their next scheduled execution time and the user who submitted the workflow.

Viewing Workflow Execution Details

You can drill down to view detailed information about a specific workflow execution by clicking on the workflow *Execution ID* link in the Orchestration Workflows dashboard.

From this page, you can view the workflow's execution summary. The execution summary shows you information such as the name, status, submitted by, total workers, total steps, start & end time, and target entities. The execution is displayed in a tree view, which gives you an easy way to drill-down further into individual step-level details such as the corresponding input, output and results. It also provides convenient access to the execution logs as well as a direct link to Oracle Log Analytics.

Viewing Workflow Execution Details for Nested Workflows

Nested workflows have the additional ability to show execution details for both the Parent and Child workflows. You can view detailed information about a Parent workflow execution by clicking on the workflow *Execution ID* link in the Orchestration Workflows dashboard. Within the *Execution Details* tree for the parent workflow a *Workflow* step type will show an *Operation* stage. This *Operation* stage is a clickable link; it will open a new browser tab with the *Execution Details* for the Child workflow where you can view the exact same information and details as any other workflow.

8

Simplifying IT Management

Oracle Orchestration is much more than a simple way to automate IT administrative tasks. The service's comprehensive orchestration capabilities let you to plan, organize, and automatically implement complex administrative and developmental workflows that are typically carried out in IT and development environments. The following use cases illustrate the flexibility of Oracle Orchestration and how it can save you significant time and effort.

- [Schedule Workflows](#)
- [Performing Bulk Configuration Management](#)
- [Automating Provisioning and Deployment](#)
- [Managing Incidents with Remediation Actions](#)
- [REST API Use Cases](#)

Performing Bulk Configuration Management

Using Oracle Orchestration as a central engine to orchestrate specialized scripts, you can easily deploy or scale entire applications in the cloud with minimal effort. Oracle Orchestration can seamlessly execute complex workflows across your on-premises and cloud resources. For example, the service can run a command to list all running Amazon Elastic Compute Cloud (EC2) instances under your Amazon Web Services (AWS) account and narrow it down by region or with a special tag like *test*. Oracle Orchestration can then execute a command to shut down or terminate instances that have the *test* tag defined.

Oracle Orchestration lets you:

- Execute workflow operations in bulk across multiple resources either on-demand or schedule.
- Configure and store your cloud credentials securely.

Example Bulk Configuration Management Jobs

- Cloud resources: Create new Oracle Compute Cloud Service, Oracle Database Cloud Service, Oracle Java Cloud Service resources.
- Component applications: Provision new instances of applications such as DB, Middleware, or Web servers.
- Provision entire application stacks: Seamlessly provision complete topologies.

Automating Provisioning and Deployment

You can use Oracle Orchestration as a central engine to orchestrate specialized scripts to deploy or scale entire applications in the cloud. The service also can serve as the fabric to bind the scripts from disparate tools like Chef, Puppet, and so on. to build or update development and QA environments. For example, after building a

software release, Oracle Orchestration can automatically deploy the latest build to machines within your IT environment. Some service benefits are:

- Lets your organization move toward continuous delivery (CD) by having your build or continuous integration (CI) server trigger an Oracle Orchestration workflow to execute the post-build deployment process and deployment verification testing.
- Lets you reuse those same jobs to deploy your software across all environments automatically (zero-click) or with a manual trigger (one-click).
- Oracle Orchestration exposes its complete functionality using the REST API interface, making it easy to integrate workflows with existing processes in your organization.

Example Provisioning and Deployment Jobs

- Cloud resources: Create new Oracle Compute Cloud Service, Oracle Database Cloud Service, Oracle Java Cloud Service resources.
- Component applications: Provision new instances of database, middleware, web servers, and so on.
- Provision entire application stacks: Wire all topologies together.

Managing Incidents with Remediation Actions

Oracle Orchestration can execute immediate and automated actions as a result of an event generated by any discovered entity within your environment. Oracle Management Cloud can execute automated actions via Oracle Orchestration with predetermined workflows to target specific issues, such as patching and remediation actions.

Example Remediation/Preventative Action Jobs

- Automatically scale applications based on performance issues detected by Oracle Application Performance Monitoring.
- Resize cloud resources based on capacity constraints detected by Oracle Infrastructure Monitoring.
- Generate alerts from a saved search (an error in a workflow log, for example) in Oracle Log Analytics.

Remediation Action Workflow Creation and Submission

With Orchestration you can automate remediation actions using workflows. The following is a short introductory guide on how to create and submit a Remediation Action Workflow.

Note:

Remediation Workflow submission require that workflows be authored using V1.0 syntax, for more information on V1.0 syntax please see [Author Workflows Using V1.0 Syntax](#)

Table 8-1 Remediation Prerequisites and Actions Required

Task	Description	Configuration Information
Prerequisite: Add Credentials if required	For ENTITY workflows add a SSH host credential, <code>HostSSHCreds</code> type, to the agent running on the target host. For REST workflows add a REST credential.	Manage Agent Credentials in <i>Installing and Managing Oracle Management Cloud Agents</i>
Author and test the workflow in Orchestration	Create your ENTITY or REST type workflow using V1.0 syntax.	See: Author Workflows Using V1.0 Syntax

Table 8-1 (Cont.) Remediation Prerequisites and Actions Required


Task	Description	Configuration Information
Upload and deploy the workflow script to the host	Submit your workflow using the Workflow Submission API .	 N o t e : O r a c l e O r c h e s t r a t i o n R E S T A P I d o c u m e n t a t i o n a c c e s

Table 8-1 (Cont.) Remediation Prerequisites and Actions Required

Task	Description	Configuration Information
		 swill be limited to approved customers. Please contact your Oracle

Table 8-1 (Cont.) Remediation Prerequisites and Actions Required

Task	Description	Configuration Information
		 e S u p p o r t o r S a l e s R e p r e s e n t a t i v e · F o r m o r e i n f o r m a t i o n s e e t h e M

Table 8-1 (Cont.) Remediation Prerequisites and Actions Required

Task	Description	Configuration Information
		 y O r a c l e S u p p o r t n o t e O r c h e s t r a t i o n M a s t e r N o t e (D o c I D 2 2 2 9 5 2 3

Table 8-1 (Cont.) Remediation Prerequisites and Actions Required

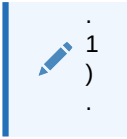
Task	Description	Configuration Information
Embed the workflow in remediation action definition and perform appropriate variable substitutions	Create your remediation action and embed the workflow created above modifying the variable fields where applicable. For example, the host name should be a variable, so a script can run on any host, depending on what host name is being passed on from an alert.	 Example 10: Remediation Workflow

Table 8-1 (Cont.) Remediation Prerequisites and Actions Required


Task	Description	Configuration Information
Create Remediation Action	Publish your remediation action using the Event Service API.	<div style="border-left: 2px solid #0070C0; padding-left: 10px;">  N o t e : O r a c l e O r c h e s t r a t i o n R E S T A P I d o c u m e n t a t i o n a c c e s </div>

Table 8-1 (Cont.) Remediation Prerequisites and Actions Required

Task	Description	Configuration Information
		 swill be limited to approved customers . Please contact your Oracle

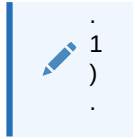
Table 8-1 (Cont.) Remediation Prerequisites and Actions Required

Task	Description	Configuration Information
		 e S u p p o r t o r S a l e s R e p r e s e n t a t i v e · F o r m o r e i n f o r m a t i o n s e e t h e M

Table 8-1 (Cont.) Remediation Prerequisites and Actions Required

Task	Description	Configuration Information
		 y O r a c l e S u p p o r t n o t e O r c h e s t r a t i o n M a s t e r N o t e (D o c I D 2 2 2 9 5 2 3

Table 8-1 (Cont.) Remediation Prerequisites and Actions Required

Task	Description	Configuration Information
Create an Alert Rule and select the appropriate Remediation Action which was published	Create an alert rule to be notified when something happens, and under remediation action select the action you just published.	 <p>When an alert is triggered you can:</p> <ul style="list-style-type: none"> View alert details by clicking on the alert message in the alerts page See a history of what happened when the alert was triggered. Remediation actions triggered will show as annotations within the alert history. Under the Orchestration Workflows see the Execution details of your workflow, see Monitor a Workflow Submission for more details Monitor the alert, it should clear when the issue is resolved

REST API Use Cases

Oracle Orchestration use cases have been tested and are fully documented in the Oracle Orchestration REST API. This API provides explicit instructions on how to perform actions such as:

- Running a Script On Multiple Hosts.
- Debug Failed Threads of an Execution.
- Running an Ad-hoc SQL Script On Multiple Databases.
- Running a Create Instance on Oracle Cloud To Spin A JCS Instance.

Note:

For this release, Oracle Orchestration REST API documentation access will be limited to approved customers. Contact your Oracle Support or Sales Representative for more information about accessing and using the Oracle Orchestration REST API. In addition, see the My Oracle Support note *Orchestration Master Note* (Doc ID 2229523.1) for the latest information.

A

Appendix A: Workflow Samples

The following are JSON workflow code samples to help get you started using Orchestration:

Related Topics

- [Example 1: Starting a Web Logic Server](#)
In this V2.0 syntax example we see how to start up a web logic server using an ENTITY step type.
- [Example 2: Shutting Down a Web Logic Server](#)
In this V2.0 syntax example we see how to shut down a web logic server using an ENTITY step type.
- [Example 3: Patch an Oracle Database Server with Input Parametrization](#)
In this V2.0 syntax example we patch an Oracle Database server using input parametrization. This is achieved by listing the host name, host credentials and script path within the *declaration* section of the workflow.
- [Example 4 Adding Wait Time in Seconds to a Workflow](#)
In this V2.0 example we add wait time in seconds to a workflow using the WAIT type step and the *waitInSeconds* parameter.
- [Example 5: Single Step REST Call](#)
In this V2.0 example we look at one of the simplest forms of a workflow, executing a single step REST call.
- [Example 6: Single Step REST Call with Variable Inputs](#)
In this V2.0 syntax example we execute a single step rest call using a variable to call upon the REST *uri*.
- [Example 7: Single Step REST call with Workers](#)
In this V2.0 syntax example we execute a single step REST call using workers to execute the *uri* command.
- [Example 8 Two Step Workflow Execute and Validate REST Calls](#)
In this V2.0 syntax example we demonstrate passing information from one step to another step. This example is also a multistep workflow sample.
- [Example 9 Nested workflows Parent Child](#)
In this V2.0 syntax example we will look at 2 JSON workflow code snippets. The first sample is that of a Parent workflow calling on the second Child workflow. Nested workflows work by having one workflow call upon a second workflow.
- [Example 10 Remediation Workflow](#)
In this example we see an example of a Remediation workflow for restarting a WebLogic server and it's corresponding Remediation Action to start a Weblogic server from an alert.

Example 1: Starting a Web Logic Server

In this V2.0 syntax example we see how to start up a web logic server using an ENTITY step type.

```
{
  "steps": [
    {
      "name": "start_weblogic",
      "operation": {
        "type": "ENTITY",
        "parameters": {
          "entity": {
            "entityByName": {
              "name": "host01.example.com",
              "type": "omc_host_linux"
            }
          },
          "command": "/bin/sh",
          "arguments": [
            "/u01/app/oracle/Middleware/Oracle_Home/
user_projects/domains/medrec/startWebLogic.sh"
          ],
          "credential": "MySSHCred"
        }
      }
    }
  ],
  "input": {
    "vars": {},
    "workerVars": {
      "worker1": {}
    }
  }
}
```

Example 2: Shutting Down a Web Logic Server

In this V2.0 syntax example we see how to shut down a web logic server using an ENTITY step type.

```
{
  "name": "Shutdown WebLogic Server",
  "version": "1.0.2",
  "description": "This is an example of a workflow to shutdown a
WebLogic server",
  "source": "WebLogic",
  "plan": {
    "steps": [
      {
        "name": "stop_weblogic",
        "operation": {
```



```

        "type": "ENTITY",
        "parameters": {
            "entity": {
                "entityByName": {
                    "name": "host01.example.com",
                    "type": "omc_host_linux"
                }
            },
            "command": "/bin/sh",
            "arguments": [
                "/u01/app/oracle/Middleware/Oracle_Home/
user_projects/domains/medrec/bin/stopWebLogic.sh"
            ],
            "credential": "MySSHCred"
        }
    }
}
},
"input": {
    "vars": {},
    "workerVars": {
        "worker1": {}
    }
}
}
}

```

Example 3: Patch an Oracle Database Server(s) with Input Parametrization

In this V2.0 syntax example we patch an Oracle Database server using input parametrization. This is achieved by listing the host name, host credentials and script path within the *declaration* section of the workflow.

```

{
    "name": "Oracle Database Patch Workflow",
    "version": "3.0.1",
    "description": "This is an example of a workflow used to patch an
Oracle database server",
    "source": "Custom",
    "plan": {
        "declaration": {
            "input": {
                "required": [
                    "HostName",
                    "HostCredential",
                    "ScriptPath"
                ],
                "properties": {
                    "HostName": {
                        "type": "string",
                        "maxLength": 30,
                        "description": "The name of the Host"
                    }
                }
            }
        }
    }
}

```

```

    },
    "HostCredential": {
      "type": "string",
      "maxLength": 40,
      "description": "Credential required to access the
host"
    },
    "ScriptPath": {
      "type": "string",
      "minLength": 1,
      "description": "Full path to the script on the
Oracle database server"
    }
  },
  "steps": [
    {
      "name": "Patch",
      "operation": {
        "type": "ENTITY",
        "parameters": {
          "entity": {
            "entityByName": {
              "name": "${workflow.vars.HostName}",
              "type": "omc_host_linux"
            }
          },
          "command": "sh",
          "arguments": [
            "${workflow.vars.ScriptPath}"
          ],
          "credential": "${workflow.vars.HostCredential}"
        }
      }
    }
  ],
  "input": {
    "vars": {
      "HostName": "",
      "HostCredential": "",
      "ScriptPath": ""
    }
  }
}

```

Example 4: Adding Wait Time in Seconds to a Workflow

In this V2.0 example we add wait time in seconds to a workflow using the *WAIT* type step and the *waitInSeconds* parameter.

```

{ "name": "WaitTime_Sample",
  "version": "1.0",

```

```

"description": "How to add a wait time using waitInSeconds function"
"workflow": {
  "source": "Custom",
  "version": "1.0",
  "name": "WaitTime_Sample",
  "plan": {
    "steps": [
      {
        "name": "WaitStepSample",
        "operation": {
          "type": "WAIT",
          "parameters": {
            "waitInSeconds": 240
          }
        }
      }
    ]
  },
  "input": {
    "vars": {},
    "workerVars": {
      "worker1": {
        .....
      }
    }
  }
}

```

Example 5: Single Step REST Call

In this V2.0 example we look at one of the simplest forms of a workflow, executing a single step REST call.

```

{ "name": "Submit simple REST workflow",
  "version": "1.0",
  "workflow": {
    "name": "Single step REST workflow",
    "version": "1.0",
    "plan": {
      "steps": [
        { "name": "rest step",
          "operation": {
            "type": "REST",
            "parameters": {
              "method": "GET",
              "uri": "http://www.somewhere.com/rest/api"
            }
          }
        }
      ]
    }
  }
}

```

```

    }
  }
}

```

Example 6: Single Step REST Call with Variable Inputs

In this V2.0 syntax example we execute a single step rest call using a variable to call upon the REST *uri*.

```

{ "name": "Submit simple REST workflow",
  "version": "1.0",
  "workflow": {
    "name": "Single step REST workflow with input uri",
    "version": "1.0",
    "plan": {
      "steps": [
        {
          "name": "rest step",
          "operation": {
            "type": "REST",
            "parameters": {
              "method": "GET",
              "uri": "${workflow.vars.uri}"
            }
          }
        }
      ]
    },
    "input": {
      "vars": {
        "uri": "http://www.somewhere.com/rest/api"
      }
    }
  }
}

```

Example 7: Single Step REST call with Workers

In this V2.0 syntax example we execute a single step REST call using workers to execute the *uri* command.

```

{ "name": "Submit simple REST workflow",
  "version": "1.0",
  "workflow": {
    "name": "Single step REST workflow with input uri",
    "version": "1.0",
    "plan": {
      "steps": [
        {
          "name": "rest step",
          "operation": {
            "type": "REST",
            "parameters": {

```



```

    {
      "name": "validate",
      "operation": {
        "type": "REST",
        "parameters": {
          "method": "GET",
          "uri": "${steps.provision.result.uri}"
        }
      }
    }
  ],
  "publish": {
    "result": "done"
  }
},

"input": {
  "vars": {
    "REST_URL": "http://host01.example.com/provision/compute",
  },
}
}

```

Example 9: Nested Workflows

In this V2.0 syntax example we will look at 2 JSON workflow code snippets. The first sample is that of a Parent workflow calling on the second Child workflow. Nested workflows work by having one workflow call upon a second workflow.

Example 9.1: Parent Workflow

This is the first of two examples explaining nested workflows. This example shoes a parent workflow calling a child workflow.

```

{"name": "SampleParentWorkflow",
 .....
 "workflow": {
 .....
 "plan": {
 "steps": [
 {
 "name": "MyParentFirstStep",
 .....
 "publish":{
 "MyParentFirstStepVar1":"MyParentFirstStepVar1Value",
 }
 }
 {
 "name": "MyParentWorkflowStep",
 "vars": {
 .....
 "variable6": "var6",

```



```
        "entityByName": {
            "name": "${entity.hostName}",
            "type": "omc_host_linux"
        }
    }
},
"responseBody": {
    "annotation": "${'Submitted workflow 'response.name'. Submission Id =
'response.submissionId}'"
}
```