



Workflows

OnCommand Workflow Automation 5.0

NetApp
April 19, 2024

This PDF was generated from <https://docs.netapp.com/us-en/workflow-automation-50/workflows/concept-overview-of-oncommand-workflow-automation.html> on April 19, 2024. Always check docs.netapp.com for the latest.

Table of Contents

- Workflows 1
 - Overview of OnCommand Workflow Automation 1
 - Understanding Workflow Automation designer 2
 - Managing workflows 11
 - Creating building blocks for workflows 46
 - Coding guidelines for WFA 58
 - Reserved words 89
 - How you use REST APIs 90
 - References to learning material 91
 - Related documentation for OnCommand Workflow Automation 93

Workflows

Overview of OnCommand Workflow Automation

OnCommand Workflow Automation (WFA) is a software solution that helps to automate storage management tasks, such as provisioning, migration, decommissioning, data protection configurations, and cloning storage. You can use WFA to build workflows to complete tasks that are specified by your processes.

A workflow is a repetitive and procedural task that consists of sequential steps, including the following types of tasks:

- Provisioning, migrating, or decommissioning storage for databases or file systems
- Setting up a new virtualization environment, including storage switches and datastores
- Setting up storage for an application as part of an end-to-end orchestration process

Storage architects can define workflows to follow best practices and meet organizational requirements, such as the following:

- Using required naming conventions
- Setting unique options for storage objects
- Selecting resources
- Integrating internal configuration management database (CMDB) and ticketing applications

WFA features

- Designer portal to build workflows

The designer portal includes several building blocks, such as commands, templates, finders, filters, and functions, that are used to create workflows. The designer enables you to include advanced capabilities to workflows such as automated resource selection, row repetition (looping), and approval points.

The designer portal also includes building blocks, such as dictionary entries, cache queries, and data source types, for caching data from external systems.

- Execution portal to execute workflows, verify status of workflow execution, and access logs
- Administration/Settings option for tasks such as setting up WFA, connecting to data sources, and configuring user credentials
- Web services interfaces to invoke workflows from external portals and data center orchestration software
- Storage Automation Store to download WFA packs

WFA license information

No license is required for using the OnCommand Workflow Automation server.

Understanding Workflow Automation designer

You create workflows in the Workflow Automation (WFA) designer using the building blocks such as finders, filters, and commands. Understanding the building blocks and the workflow creation process is important before you start creating your workflows.

Working with the building blocks in OnCommand Workflow Automation

The Workflow Automation (WFA) workflows consist of several building blocks and WFA includes a library of the predefined building blocks. You can use the building blocks that WFA provides to create workflows that match the requirements of your organization.

WFA provides the structure for storage automation processes. WFA's flexibility is based on how the workflows are constructed by using the workflow building blocks.

The WFA building blocks are as follows:

- Dictionary entries
- Commands
- Filters
- Finders
- Functions
- Templates

You should understand how the building blocks are used in WFA to help you in creating the workflows.

What data sources are

A data source is a method for establishing a connection to other systems, files and databases in order to extract data. For example, a data source can be a connection to an OnCommand Unified Manager database or OnCommand Unified Manager 9.4 data source type.

You can add a custom data source to OnCommand Workflow Automation (WFA) for data acquisition after defining the required data source type by associating the caching scheme, the required port, and the acquisition method with the data source type.

WFA caches information through various data sources. WFA collects resource information from the data sources and formats it for the caching scheme. The cache tables, which are the tables inside the caching schemes, are formatted to match the dictionary entry objects. When you use a finder in workflows, it returns a dictionary object and the data from the dictionary object is populated from the cache tables. The process of acquiring data from the data sources is known as *data source acquisition*. You can use either a script-based method or a driver-based method for data source acquisition. The sources can be different from each other and data source acquisition might sample them at different time intervals. WFA then merges that information in to the database and superimposes the reservation data to maintain updated resource information in the database.

The WFA database includes several different caching schemes. A caching scheme is a set of tables and each table includes information from a certain dictionary entry type; however, the tables might include combined information from multiple sources of a specific data source type. WFA uses the database information to

understand the status of the resources, perform calculations, and execute commands on the resources.

What dictionary entries are

Dictionary entries are one of the OnCommand Workflow Automation (WFA) building blocks. You can use dictionary entries to represent object types and their relationships in your storage and storage-related environments. You can then use filters in workflows to return the value of the natural keys of the dictionary entries.

A dictionary entry is the definition of an object type that is supported by WFA. Each dictionary entry represents an object type and its relationship in the supported storage and storage-related environments. A dictionary object consists of a list of attributes, which might be type checked. A dictionary object with complete values describes an object instance of a type. In addition, the reference attributes describe the relationship of the object with the environment; for example, a volume dictionary object has many attributes, such as name, size_mb, and volume_guarantee. In addition, the volume dictionary object includes references to the aggregate and the array containing the volume in the form of array_id and aggregate_id.

The cache table of an object is a database containing a few or all of the dictionary entry's attributes that are marked for caching. For a dictionary entry to include a cache table, at least one of the dictionary entry's attributes must be marked for caching. Dictionary entries include natural keys, which are unique identifiers for the objects; for example, 7-Mode volumes are identified uniquely by their name and the IP address of the array containing them. Qtrees are identified by the qtree name, the volume name, and the array IP address. You must identify the dictionary attributes that are part of the dictionary entry's natural keys when creating dictionary entries.

How commands work

OnCommand Workflow Automation commands are the execution blocks for workflows. You can use a command for each step in your workflow.

WFA commands are written using PowerShell and Perl scripts. PowerShell commands use the Data ONTAP PowerShell toolkit and VMware PowerCLI, if the package is installed. Perl commands use the Perl distribution and Perl modules installed on the WFA server. If you include multiple scripting languages in a command, such as PowerShell and Perl, the appropriate script is chosen by WFA based on the operating system on which it is installed and the preferred order of language you have specified in the WFA configuration menu.

The scripts for the WFA commands include several parameters. These parameters might be mapped to dictionary entry attributes.

Note that each WFA command can include several Data ONTAP commands.

Some of the WFA commands are known as *wait* commands because they can wait for long-running operations and poll periodically—for example, the **Wait for multiple volume moves** command. The waiting interval at which the polling command is executed can be configured to check if the operation has been completed.

A WFA command is initiated by WFA while the workflow is in its execution phase. WFA executes the commands serially, in left-to-right and top-to-bottom order. The planning of the workflow confirms the availability and validity of the parameters that are supplied to the command. The WFA server supplies all the parameters required for the commands before the commands are executed.

Parameters to commands are finalized during workflow planning. The workflow then passes these parameters to the commands during execution time. The commands cannot pass parameters back to the workflow. However, if you want to exchange information that is obtained during execution time between commands in a workflow, you can use the designated WFA PowerShell cmdlets or Perl functions.

WFA PowerShell commands do not use the `-ErrorAction stop` flag for the PowerShell cmdlets; therefore, workflow executions continue even when the cmdlets fail because of an error. If you want the `-ErrorAction stop` flag to be included in a specific command, you can clone the command and modify the PowerShell script to add the flag.

The following are the PowerShell cmdlets and Perl functions that are included in WFA to enable exchange of information between commands:

PowerShell cmdlets	Perl functions
Add-WfaWorkflowParameter	addWfaWorkflowParameter
Get-WfaWorkflowParameter	getWfaWorkflowParameter

Parameters added by the “add” cmdlets or functions to a command can be retrieved by a command that is executed subsequently and uses the “get” cmdlets or functions. For example, in a PowerShell WFA command, you can use the following in the code to add a parameter named `volumeId`: `Add-WfaWorkflowParameter -Name "VolumeUUID" -Value "12345" -AddAsReturnParameter $true`. Then, you can use the following in a subsequent command to retrieve the value of `volumeId`: `$volumeId = Get-WfaWorkflowParameter -Name volumeId`.

WFA commands can query the WFA database and obtain the required result. This enables you to construct a command without using filters and finders. You can use the following functions to query the database:

PowerShell cmdlet	Perl function
<code>Invoke-MySQLQuery</code> For example: <code>Invoke-MySQLQuery -Query "SELECT cluster.name AS 'Cluster Name' FROM cm_storage.cluster"</code>	<code>invokeMySQLQuery</code>

What filters are

You can use WFA filters in your workflows to select the required resources.

A WFA filter is an SQL-based query that works on the WFA database. Each filter returns a list of elements of a specific dictionary type. The returned elements are based on the selection criteria specified in the SQL query. You must be aware of SQL syntaxes to create or edit a filter.

What finders are

A finder is a combination of one or more filters that are used together to identify common results. You can use a finder in your workflows to select the required resources for workflow execution.

Finders might apply a sorting order to differentiate the applicable results. Finders return the best resource based on the selection criteria and sorting.

Finders return either one result or no result; therefore, they can be used to verify the existence of certain storage elements. However, when a finder is used as part of a repeat row definition, the result sets are used to form the list of members in the group. Filters that are used in finders return the natural key of the dictionary type, at a minimum, but might return additional fields, whose value can be referenced. A sorting order might be applied to any returned field of a filter's SQL query.

You can test the results of a finder. When testing a finder, you can view the common results of all the WFA filters, where the effective result of the finder is highlighted in the results. When using a finder in a workflow, you can create a customized error message to convey meaningful information to the storage operator.

What functions are

You can use a function in your workflows for a complex task that has to be completed during the planning phase of the workflow.

You can write functions by using the MVFLEX Expression Language (MVEL). You can use functions to put together commonly used logic as well as more complex logic in a named function and reuse it as values for command parameters or filter parameters. You can write a function once and use it across workflows. You can use functions to handle repetitive tasks and tasks that might be complex, such as defining a complex naming convention.

Functions might use other functions during their execution.

What schemes are

A scheme represents the data model for a system. A data model is a collection of dictionary entries. You can define a scheme and then define a data source type. The data source defines how the data is acquired and the scheme is populated. For example, a vc scheme acquires data about your virtual environment, such as virtual machines, hosts, and datastores.

Schemes can also be populated directly with data through workflows that are customized to solve specific problems.

Dictionary entries are associated with an existing scheme when the dictionary entries are created. Dictionary entries are also associated with cache queries, and cache queries include SQL queries.

Schemes can acquire data using either script based data source type or SQL data source type. The scripts are defined while creating the data source type and SQL queries are defined in the cache queries.

The following schemes are included in WFA:

- **7-Mode (storage)**

Scheme to acquire data through OnCommand Unified Manager from Data ONTAP operating in 7-Mode.

- **Clustered Data ONTAP (cm_storage)**

Scheme to acquire data through OnCommand Unified Manager from clustered Data ONTAP.

- **7-Mode Performance (performance)**

Scheme to acquire performance data of Data ONTAP operating in 7-Mode through Performance Advisor.

- **Clustered Data ONTAP Performance (cm_performance)**

Scheme to acquire performance data of clustered Data ONTAP through Performance Advisor.

- **VMware vCenter (vc)**

Scheme to acquire data from VMware vCenter.

- **Playground (playground)**

Scheme that can directly populate with data.

What remote system types are

OnCommand Workflow Automation (WFA) communicates with remote system types. A remote system type specifies the type of remote systems with which WFA can communicate. You can configure remote system types in WFA. For example, Data ONTAP system can be configured as a remote system type.

A remote system type has the following attributes:

- Name
- Description
- Version
- Protocol
- Port
- Timeout

You can have a Perl script for each remote system type to validate the credentials of the remote system. You can store the credentials for the remote systems configured on WFA. You can add or edit a new custom remote system type. You can also clone an existing remote system type. You can delete a remote system type only if no systems are associated with it.

How you use templates

You can use WFA templates in your workflows as a reference or for adhering to usage policies.

A WFA template acts as a blueprint of an object definition. You can define a template by including the properties of an object and the values for the object's properties. Then, you can use the template for populating the properties of an object definition in your workflows.

When you use a template, you cannot edit the fields that include the values that are obtained from the template. Therefore, you can use templates for setting up usage policies and creation of objects. If you remove the association of a template with the workflow after you have applied the template, the values populated from the template remain, but you can edit the fields.

How you use categories

You can categorize your workflows to better organize the workflows and to apply access

control capability on the workflows.

You can categorize workflows such that they appear in specific groups on the WFA portal. You can also apply access control capability on workflow categories. For example, you can allow only certain storage operators or approvers to view certain categories of workflows. Storage operators or approvers can execute only the workflows within the category for which they have been granted access rights.

Active Directory groups also can be used for access control to categories.

How entity versioning works

The OnCommand Workflow Automation (WFA) entities, such as commands and workflows, are versioned. You can use the version numbers to easily manage changes to the WFA entities.

Each WFA entity includes a version number in the *major.minor.revision* format—for example, 1.1.20. You can include up to three digits in each part of the version number.

Before modifying the version number of a WFA entity, you must be aware of the following rules:


- Version numbers cannot be changed from the current version to an earlier version.
- Each part of the version must be a number from 0 through 999.
- New WFA entities are versioned as 1.0.0, by default.
- An entity's version number is retained when cloning or using **Save As** to save a copy of the entity.
- Multiple versions of an entity cannot exist in a WFA installation.

When you update the version of a WFA entity, the version of its immediate parent entity is updated automatically. For example, updating the version of the **Create Volume** command updates the **Create an NFS Volume** workflow, because the **Create an NFS Volume** workflow is an immediate parent entity of the **Create Volume** command. The automatic update to versions is applied as follows:

- Modifying the major version of an entity updates the minor version of its immediate parent entities.
- Modifying the minor version of an entity updates the revision version of its immediate parent entities.
- Modifying the revision version of an entity does not update any part of the version of its immediate parent entities.

The following table lists the WFA entities and their immediate parent entities:

Entity	Immediate parent entity
Cache query	<ul style="list-style-type: none">• Data source type
Template	<ul style="list-style-type: none">• Workflow

Entity	Immediate parent entity
Function	<ul style="list-style-type: none"> • Workflow • Template <div>  <p>If a function contains special or mixed case characters, the version of its immediate parent entities might not be updated.</p> </div>
Dictionary	<ul style="list-style-type: none"> • Template • Filter • Cache query • Command • Data source types which are using script method
Command	<ul style="list-style-type: none"> • Workflow
Filter	<ul style="list-style-type: none"> • Finder • Workflow
Finder	<ul style="list-style-type: none"> • Workflow
Data source type	None
Workflow	None

You can search for an entity in WFA either using the parts of the version number or the complete version number.

If you delete a parent entity, the child entities are retained and their version is not updated for the deletion.

How versioning works when importing entities

If you import entities from versions earlier than Workflow Automation 2.2, the entities are versioned as 1.0.0, by default. If the imported entity is already present in the WFA server, the existing entity is overwritten with the imported entity.

The following are the potential changes to WFA entities during an import:

- Upgrade of entities

The entities are replaced with a later version.

- Rollback of entities

The entities are replaced with an earlier version.



When you perform a rollback of an entity, the version of its immediate parent entities are updated.

- Import of new entities



You cannot selectively import entities from a .dar file.

If a later version of an entity is imported, the version of its immediate parent entities is updated.

If there are multiple child entities to the imported parent entity, only the highest degree of change (major, minor, or revision) to the child entities is applied to the parent entity. The following examples explain how this rule works:

- For an imported parent entity, if there is one child entity with a minor change and another child entity with a revision change, the minor change is applied to the parent entity.

The revision part of the parent's version is incremented.

- For an imported parent entity, if there is one child entity with a major change and another child entity with a minor change, the major change is applied to the parent entity.

The minor part of the parent's version is incremented.

Example of how the versions of imported child entities affect the parent's version

Consider the following workflow in WFA: "Create Volume and export using NFS - Custom" 1.0.0.

The existing commands included in the workflow are as follows:

- "Create Export Policy - Custom" 1.0.0
- "Create Volume - Custom" 1.0.0

The commands included in the .dar file, which is to be imported, are as follows:

- "Create Export Policy - Custom" 1.1.0
- "Create Volume - Custom" 2.0.0

When you import this .dar file, the minor version of the "Create Volume and export using NFS - Custom" workflow is incremented to 1.1.0.

What a playground database is

The playground database is a MySQL database, which is included in the Workflow Automation (WFA) server installation. You can add tables to the playground database to include information, which can be used by filters and SQL queries for user inputs.

The playground database is a schema that cannot be accessed through the WFA web portal. You can use a MySQL client, such as SQLyog, Toad for MySQL, and MySQL Workbench or a command-line interface (CLI), to access the database.

You must use the following credentials to access the playground database:

- User name: wfa
- Password: Wfa123

The credentials provide complete access to the playground database and read-only access to other schemas defined in the WFA MySQL database. You can create the required tables in the playground database.

You can add the tags or metadata that you are using for storage objects in your environment to a table in the playground database. The tags or metadata can then be used along with the information in other WFA cache tables by WFA filters and user input queries.

For example, you can use the playground database for the following use cases:

- Tagging aggregates with business unit (BU) name and allocating volumes to the BUs based on these tags
- Tagging vFiler units with BU names
- Adding geography or location details to storage objects
- Defining access of database admins to databases

For example, if you are using the name of the BU as a tag for the storage objects, such as aggregates and vFiler units, you can create a table in the playground database that includes the name of the BU. The BU name can then be used by filters and user input queries for your workflows.

The following is an example playground database table (playground.volume_bu):

array_ip	volume_name	BU
10.225.126.23	data_11	Marketing
10.225.126.28	arch_11	HR

The following is an example SQL query that you can use to filter volumes by BU:

```
SELECT
    vol.name,
    array.ip AS 'array.ip'
FROM
    storage.volume AS vol,
    storage.array AS array,
    playground.volume_bu AS vol_bu
WHERE
    vol.array_id = array.id
    AND array.ip = vol_bu.array_ip
    AND vol.name = vol_bu.volume_name
    AND vol_bu.bu = '{$bu}'
```

Related information

SQLyog: www.webyog.com

Managing workflows

You can customize predefined workflows or create new workflows as part of managing your workflows. You must also understand the relevant concepts before you start managing your workflows.

Customize predefined workflows

You can customize a predefined Workflow Automation (WFA) workflow if there is no predefined workflow that is suitable for your requirement.

What you'll need


You must have identified the required modifications for the predefined workflow.

About this task

Questions and support request for the following must be directed to the WFA community:

- Any content downloaded from the WFA community
- Custom WFA content that you have created
- WFA content that you have modified

Steps

1. Click **Designer > Workflows**.
2. Select the predefined workflow that closely matches your requirement, and then click  on the toolbar.
3. In the workflow designer, make the required changes in the appropriate tabs, such as editing the description, adding or deleting a command, modifying the command details, and modifying the user input.
4. Click **Preview**, enter the required user inputs to preview the workflow execution, and then click **Preview** to view the planning details of the workflow.
5. Click **OK** to close the preview window.
6. Click **Save**.

After you finish

You can test the workflow that you modified in your test environment, and then mark the workflow as ready for production.

Customize the Create a Volume and a CIFS Share workflow

You can customize your workflows based on your requirements. For example, you can modify the predefined *Create a Volume and a CIFS Share* workflow to include deduplication and compression.




About this task

The customization and illustrations in this task are examples; you can modify the WFA workflows based on your requirements.

Steps

1. Click **Designer > Workflows**.

2. Select the **Create a Volume and a CIFS Share** workflow, and then click  on the toolbar.

Workflows		
Certified	Name	Scheme
	Create a Qtree CIFS Share in a vFiler	storage
	Create a Volume and a CIFS Share	storage
	Create a Volume and Qtrees	storage

3. Click the **Details** tab and edit the description of the workflow in the **Workflow name** field.

4. Click the **Workflow** tab, expand the **storage** schema, and then drag and drop the **Setup deduplication and compression** command in between the **Create volume** and **Create CIFS share** commands.



5. Place your mouse cursor below the **Setup deduplication and compression** command on the first row and then click .

6. In the **Volume** tab of the **Parameters for 'Setup deduplication and compression'** dialog box, select the **by using a previously defined Volume** option, and then select the **share_volume** option in the **Define Volume** field, which is the Volume object variable created by the **Create Volume** command in the workflow.

7. Click the **Other Parameters** tab and perform the following steps:

- Select **true** in the **StartNow** field.
- Select **Inline** in the **Compression** field.

- c. Enter 'sun-sat@1' expression in the **Schedule** field, which schedules deduplication and compression on all days of the week at 1 a.m.

Parameters for 'Setup deduplication and compression' ⓘ

Volume Other Parameters **Advanced**

StartNow *: true Compression: Inline

Schedule: 'sun-sat@1'

8. Click **OK**.
9. Click **Preview** to ensure that the planning of the workflow is completed successfully, and then click **OK**.
10. Click **Save**.

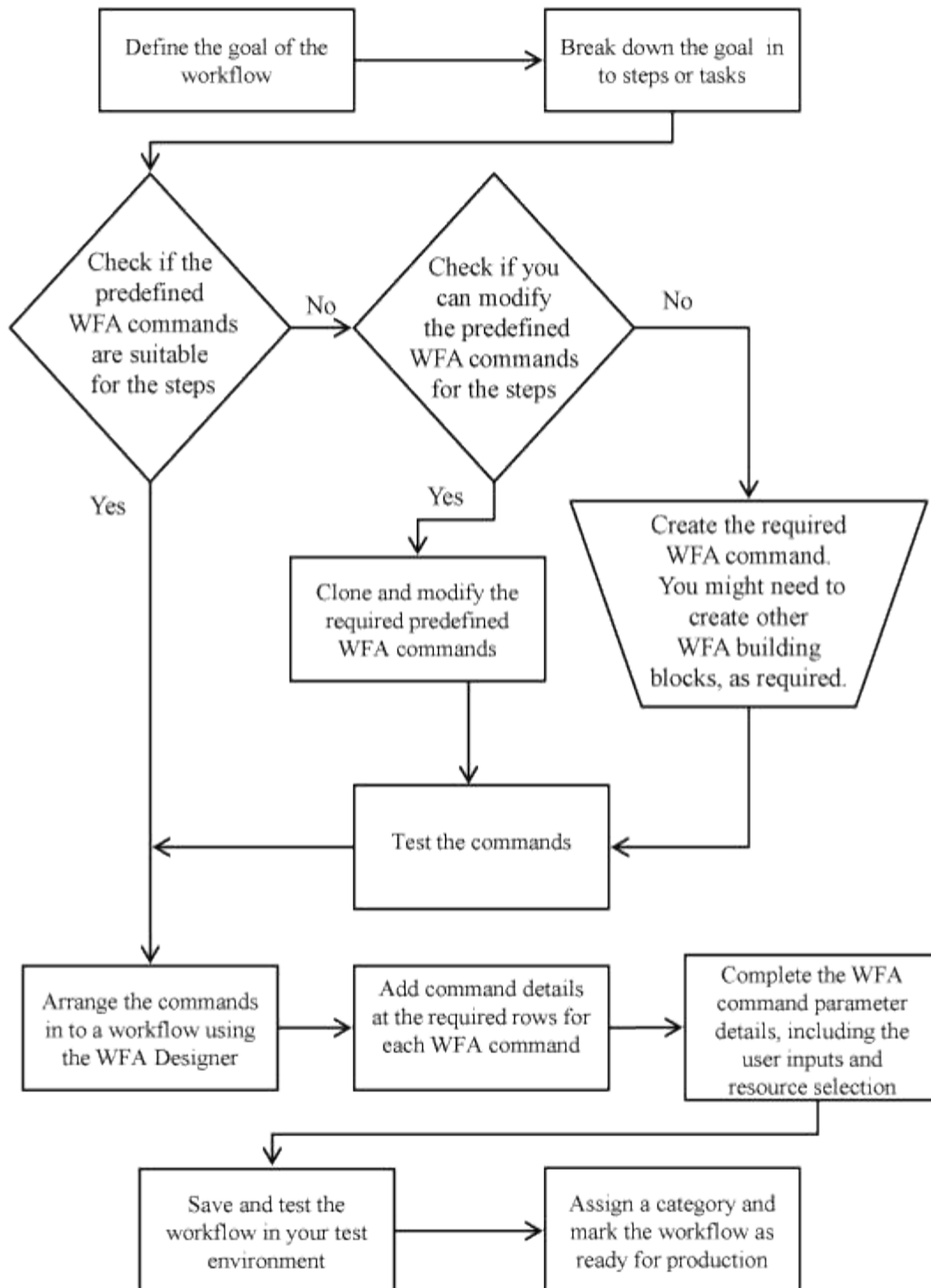
Creating workflows

If the predefined workflows do not match your requirements, you can create the required workflow. Before you create your workflows, you should understand the capabilities available in the WFA designer and create a workflow checklist.

Tasks involved in creating workflows

Creating storage automation workflows in OnCommand Workflow Automation (WFA) includes defining the steps to be performed by a workflow and creating the workflow using the WFA building blocks, such as commands, finders, filters, and dictionary entries.

The following flowchart illustrates the workflow creation process:



How you define workflows

You must break down the goal of a workflow into the steps that should be executed by the workflow. You can then arrange the steps to complete your workflow.

A workflow is an algorithm that includes a series of steps that are required to complete an end-to-end process. The scope of the process might vary, depending on the goal of the workflow. The goal of a workflow might be defined to handle only storage operations or more complex processes such as handling networking, virtualization, IT systems, and other applications as part of a single process. OnCommand Workflow Automation (WFA) workflows are designed by storage architects and are executed by storage operators.

Defining your workflow includes breaking down the goal of your workflow into a series of steps—for example, creating an NFS volume includes the following steps:

1. Creating a volume object
2. Creating a new export policy and associating the policy with the volume

You can use a WFA command or a workflow for each step in your workflow. WFA includes predefined commands and workflows, which are based on common storage use cases. If you do not find a predefined command or workflow that can be used for a particular step, you can do one of the following:

- Choose a predefined command or workflow that closely matches the step, and then clone and modify the predefined command or workflow according to your requirements.
- Create a new command or workflow.

You can then arrange the commands or workflows in a new workflow to create the workflow that accomplishes your goal.

At the beginning of the workflow execution, WFA plans the execution and verifies that the workflow can be executed using the input to the workflow and the commands. When planning the workflow, all resource selection and user input are resolved to create an execution plan. After planning is completed, WFA executes the execution plan, which consists of a set of WFA commands with applicable parameters.

How user inputs are defined

The OnCommand Workflow Automation (WFA) user inputs are data input options that are available during the execution of workflows. You must define the user input parameters for your workflows to enhance the flexibility and usability of your workflows.

User inputs are shown as input fields, which can be filled out with relevant data when previewing or executing workflows. You can create a user input field when specifying the command details in a workflow by prefixing a label or variable with the dollar sign (\$). For example, `$VolumeName` creates a `Volume Name` user input field. WFA automatically populates the User Inputs tab in the Workflow <workflow name> window with the user input labels that you have created. You can also define the type of the user input and customize the input fields by modifying the user input attributes, such as type, display name, default values, and validation values.

User input type options

- **String**

You can use a regular expression for valid values—for example, `a*`.

Strings, such as `0d` and `0f`, are evaluated as numbers similar to `0d` evaluated as 0 of type double.

- **Number**

You can define a numerical range that can be selected—for example, `1 through 15`.

- **Enum**

You can create enumeration values that can be selected when filling the user input field using the enum type. You can optionally lock the enum values that you have created to ensure that only the values you have created are selected for the user input.

- **Query**

You can select the query type when you want the user input to be selected from the values available in the WFA cache. For example, you can use the following query to automatically populate the user input fields with the IP address and name values from the WFA cache: `SELECT ip, name FROM storage.array`. You can optionally lock the values retrieved by a query so that only the results retried by the query are selected.

- **Query (multi-select)**

The query (multi-select) type, which is similar to the query type, enables the selection of multiple values during the execution of the workflow. For example, users can select multiple volumes or a volume together with its shares and exports. You can allow the users to select multiple rows, or restrict the selection to a single row. Selecting a row selects the values from all the columns of the selected row.

You can use the following functions when using the query (multi-select) type of user input:

- `getSize`
- `getValueAt`
- `getValueAt2D`
- `getValueFrom2DByRowKey`

- **Boolean**

You can use the Boolean type to display a check box in the user input dialog box. You must use the Boolean type for user inputs that have “true” and “false” as the possible values.

- **Table**

You can use the table type of user input to specify the column headers of a table that can be used to enter multiple values during the execution of the workflow. For example, a table that can be used to specify a list of node names and port names. You can also specify one of the following user input types for the column headers to validate the values that are entered during run time:

- String
- Number
- Enum
- Boolean
- Query String is the default user input type for the column headers. You must double-click the Type column to specify a different user input type.

You can open the Create SnapMirror policy and rules workflow in the Designer to see how the user input types are used in the “SnapMirrorPolicyRule” user input.

You can use the following functions when using the table type of user input:

- `getSize`
- `getValueAt`
- `getValueAt2D`
- `getValueFrom2DByRowKey`

You can open the **Create and configure a Storage Virtual Machine with Infinite Volume** workflow in the Designer to see how the table type is used.

• Password

You can use the password type for user inputs that are meant for entering passwords. The password entered by the user is encrypted and displayed as a sequence of asterisk characters across the WFA application and in the log files. You can use the following functions to decrypt the password, which can then be used by the command:

- For Perl commands: `WFAUtil::getWfaInputPassword ($password)`
- For PowerShell commands: `Get-WfaInputPassword -EncryptedPassword $password`

Here, `$password` is the encrypted password that is passed by WFA to the command.

• Dictionary

You can add the table data for the selected dictionary entry. The dictionary entry attribute selects the attribute that is to be returned. You can select a single value or multiple values while executing the workflow. For example, you can select a single volume or multiple volumes. By default, single values are selected. You can also select Rules for filtering. A rule consists of a dictionary entry attribute, an operator, and a value. The attribute can also include attributes of its references.

For example, you can specify a rule for aggregates by listing all aggregates with name starting with the string “aggr” and have an available size greater than 5 GB. The first rule in the group is the attribute `name`, with the operator `starts-with`, and the value `aggr`. The second rule for the same group is the attribute `available_size_mb`, with the operator `>` and the value `5000`.

The following table lists the options that you can apply to the user input types:

Option	Description
Validating	<p>You can validate the user inputs type so that only valid values are entered by users:</p> <ul style="list-style-type: none">• The string and number types of user input can be validated with the values entered during run time of the workflow.• The string type can also be validated with a regular expression.• The number type is a numeric floating-point field and can be validated using a specified numeric range.
Locking values	<p>You can lock the values of the query and enum types to prevent the user from overwriting the drop-down values and to enable the selection of only the displayed values.</p>
Marking as mandatory	<p>You can mark user inputs as mandatory so that the users must enter certain user inputs in order to continue with the execution of the workflow.</p>

Option	Description
Grouping	You can group related user inputs and provide a name for the user input group. The groups can be expanded and collapsed in the user input dialog box. You can select a group that should be expanded by default.
Applying conditions	With the conditional user input capability, you can set the value of a user input based on the value that is entered for another user input. For example, in a workflow that configures the NAS protocol, you can specify the required user input for protocol as NFS to enable the “Read/Write host lists” user input.

How you map command parameters

The parameters in Workflow Automation (WFA) commands are mapped to specific attributes and dictionary entry references based on certain rules. You must be aware of the rules to map command parameters when you create or edit a WFA command.

Command parameter mapping defines how command details are defined in the workflows. Mapped command parameters of a command are displayed in tabs when you are specifying the command details for commands in workflows. The tabs are named based on the group name specified in the Object Name column of the Parameters Mapping tab. The parameters that are not mapped are displayed in the Other Parameters tab when you are specifying the command details in workflows.

The rules for command parameter mapping are applicable based on the command category and how the commands are represented in the workflow editor.

The following are the command categories:

- Commands that create objects
- Commands that update objects
- Commands that remove objects
- Commands that deal with optional parent and child objects
- Commands that update associations between objects

The rules are listed below for each category:

All command categories

When mapping a command parameter, you should use the natural path based on how the command is used in workflows.

The following examples show how you can define a natural path:

- For the `ArrayIP` parameter, depending on the command, you should use the `aggregate.array.ip` attribute of the `Volume` dictionary entry and not the `array.ip` attribute.

This is important when a workflow creates a volume and then performs an additional step with the created

volume by referring to it. The following are similar examples:

- `volume.aggregate.array.ip` of the `Qtree` dictionary entry
- `volume.aggregate.array.ip` of the `LUN` dictionary entry
- For `Cluster` used in commands, you should use one of the following:
 - `vserver.cluster.primary_address` of the `Volume` dictionary entry
 - `volume.vserver.cluster.primary_address` of the `Qtree` dictionary entry

Commands that create objects

This category of commands is used for one of the following:

- Finding a parent object and defining new objects
- Searching for an object and creating the object if the object does not exist

You should use the following parameter mapping rules for this category of commands:

- Map the relevant parameters of the object that is created to the object's dictionary entry.
- Map the parent object through the references of the dictionary entry that is created.
- Ensure that the relevant attribute is present in the dictionary entry when adding a new parameter.

The following are the exception scenarios for this rule:

- Some objects that are created do not have a corresponding dictionary entry and only the parent object is mapped to the relevant parent dictionary entry—for example, the **Create VIF** command—in which only an array can be mapped to array dictionary entry.
- Parameter mapping is not required

For example, the `ExecutionTimeout` parameter in the **Create or resize aggregate** command is an unmapped parameter.

The following certified commands are examples for this category:

- Create Volume
- Create LUN

Commands that update objects

This category of commands is used to find an object and update the attributes.

You should use the following parameter mapping rules for this category of commands:

- Map the objects that are updated to the dictionary entry.
- Do not map the parameters that are updated for the object.

For example, in the **Set Volume State** command, the `Volume` parameter is mapped but the new `State` is unmapped.

Commands that remove objects

This category of commands is used to find an object and delete it.

You should map the object that is deleted by the command to its dictionary entry. For example, in the **Remove Volume** command, the Volume to be deleted is mapped to the relevant attributes and references of the Volume dictionary entry.

Commands that deal with optional parent and child objects

You should use the following parameter mapping rules for this category of commands:

- Do not map any mandatory parameter of a command as a reference from an optional parameter of the command.

This rule is more relevant when a command deals with optional child objects of a specific parent object. In this case, the child and parent object should be mapped explicitly. For example, in the **Stop Deduplication Jobs** command, the command stops a running deduplication job on a specific volume when specified along with `Array` or on all volumes of the given `Array`. In this case, the array parameter should be mapped directly to the `array` dictionary entry and not to `Volume.Array` because `Volume` is an optional parameter in this command.

- If a parent and child relationship exists between dictionary entries at the logical level but not between the actual instances in a specific command, then those objects should be mapped separately.

For example, in the **Move Volume** command, `Volume` is moved from its current parent aggregate to a new destination aggregate. Therefore, `Volume` parameters are mapped to a `Volume` dictionary entry and the destination aggregate parameters are mapped separately to the `Aggregate` dictionary entry but not as `volume.aggregate.name`.

Commands that update associations between objects

For this category of commands, you should map both the association and the objects to relevant dictionary entries. For example, in the **Add Volume to vFiler** command, the `Volume` and `vFiler` parameters are mapped to the relevant attributes of the `Volume` and `vFiler` dictionary entries.

How you define constants

You can create and use constants to define a value, which can be used across a single workflow. Constants are defined at a workflow level.

The constants used in the workflow and their value are displayed in the monitoring window of the workflow during planning and execution. You must use unique names for constants.

You can use the following naming conventions to define constants:

- Uppercase for the first letter of each word, without underscores or spaces between words

All terms and abbreviations should use upper case—for example, `ActualVolumeSizeInMB`.

- Uppercase for all letters

You can use underscores to separate words—for example, `AGGREGATE_USED_SPACE_THRESHOLD`.

You can include the following as values for workflow constants:

- Numbers
- Strings
- MVEL expressions

Expressions are evaluated during the planning and execution phases of the workflows. In the expressions, you must not reference variables that are defined in a loop.

- User inputs
- Variables

How repeat row works

A workflow contains commands and command details arranged in rows. You can specify the commands in a row to be repeated for a fixed number of iterations or dynamic number of iterations based on the results of a search criteria.

The command details in a row can be specified to repeat a certain number of times or when the workflow is designed. The workflow can also be designed such that the number of times the row must repeat can be specified when the workflow is executed or scheduled for an execution. You can specify a search criteria for an object and the commands in a row can be set to repeat as many times as the objects are returned by the search criteria. Rows can also be set to repeat when certain conditions are met.

Row repetition variables

You can specify variables in the variable list that can be manipulated during the row iterations. For the variables, you can specify a name, a value with which the variables are initialized, and an MVFLEX Expression Language (MVEL) expression that is evaluated after every iteration of the row repetition.

The following illustration shows the repeat row options and an example of a row repetition variable:

Row Repetition Details ?

Repeats: Number of times

Number of Times: For every resource in a group

Index Variable: Index1

Name	Initial Value	Expression
size_to_allocated	SIZE_MB	(int)size_to_allocated - getDate

Add Remove

Ok Cancel

Row repetition with approval points

When you have specified iterations of repeat rows for commands and included approval points, all the iterations of the commands before an approval point are executed. After you approve the approval point, the execution of all iterations of the successive commands continues until the next approval point.

The following illustration shows how the iterations of repeat rows are executed when an approval point is included in a workflow:



Repeat row examples in predefined workflows

You can open the following predefined workflows in the Designer to understand how repeat rows are used:

- Create a Clustered Data ONTAP NFS Volume
- Create VMware NFS Datastore on Clustered Data ONTAP Storage
- Establish Cluster Peering
- Remove a Clustered Data ONTAP Volume

How resource selection works

OnCommand Workflow Automation (WFA) uses search algorithms to select storage resources for workflow execution. You should understand how resource selection works in order to design workflows efficiently.

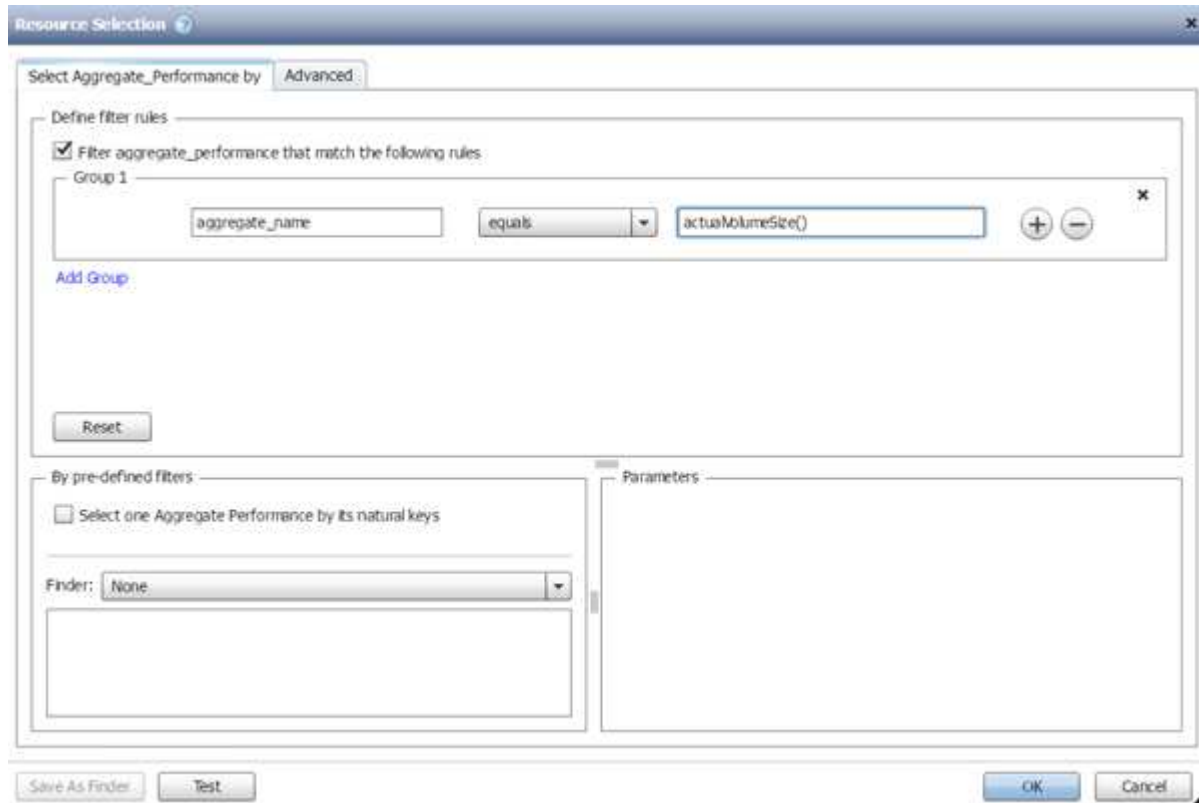
WFA selects dictionary entry resources—such as vFiler units, aggregates, and virtual machines—using search algorithms. The selected resources are then used for executing the workflow. The WFA search algorithms are part of the WFA building blocks, and include finders and filters. To locate and select the required resources, the search algorithms search through the data that is cached from different repositories, such as OnCommand Unified Manager, VMware vCenter Server, and a database. By default, a filter is available for every dictionary entry for searching a resource based on its natural keys.

You should define the resource selection criteria for each command in your workflow. In addition, you can use a finder to define the resource selection criteria in each row of your workflow. For example, when you are creating a volume that requires a specific amount of storage space, you can use the “Find aggregate by available capacity” finder in the “Create Volume” command, which selects an aggregate with a specific amount of available space and creates the volume on it.

You can define a set of filter rules for dictionary entry resources, such as vFiler units, aggregates, and virtual machines. Filter rules can contain one or more groups of rules. A rule consists of a dictionary entry attribute, an operator, and a value. The attribute can also include attributes of its references. For example, you can specify a rule for aggregates as follows: List all aggregates that have names starting with the string “aggr” and have more than 5 GB of available space. The first rule in the group is the attribute “name”, with the operator “starts-with”, and the value “aggr”. The second rule for the same group is the attribute “available_size_mb”, with the operator “>”, and the value “5000”. You can define a set of filter rules along with public filters. The Define filter rules option is disabled if you have selected a finder. The Save as Finder option is disabled if you have selected the Define filter rules check box.

In addition to the filters and finders, you can use a search or define command to search for available resources. The search or define command is the preferred option over the No-op commands. The search and define command can be used to define resources of both the certified dictionary entry type and the custom dictionary entry type. The search or define command searches for resources but does not perform any action on the resource. However, when a finder is used to search for resources, it is used in the context of a command, and the actions defined by the command are executed on the resources. The resources returned by a search or define command are used as variables for the other commands in the workflow.

The following illustration shows that a filter is used for resource selection:



Resource selection examples in predefined workflows

You can open the command details of the following predefined workflows in the Designer to understand how resource selection options are used:

- Create a Clustered Data ONTAP NFS Volume
- Establish Cluster Peering
- Remove a Clustered Data ONTAP Volume

How reservation works

OnCommand Workflow Automation resource reservation capability reserves the required resources to ensure that the resources are available for successful execution of workflows.

WFA commands can reserve the required resources and remove the reservation after the resource is available in the WFA cache database, typically after a cache acquisition. The reservation capability ensures that the reserved resources are available for the workflow until the reservation expiration period that you have configured in the WFA configuration settings.

You can use the reservation capability to exclude resources reserved by other workflows during resource selection. For example, if a workflow that has reserved 100 GB of space on an aggregate is scheduled for execution after a week, and you are executing another workflow that uses the **Create Volume** command, the workflow that is executing does not consume the space reserved by the scheduled workflow to create a new volume. In addition, the reservation capability enables workflows to be executed in parallel.

When previewing a workflow for execution, the WFA planner considers all the reserved objects, including the existing objects in the cache database. If you have enabled reservation, the effects of the scheduled workflows and the workflows that are executing in parallel, and the existence of storage elements are considered when planning the workflow.

The arrow in the following illustration shows that reservation is enabled for the workflow:



Reservation examples in predefined workflows

You can open the following predefined workflows in the Designer to understand how reservation is used:

- Clone Environment
- Create a Clustered Data ONTAP Volume
- Establish Cluster Peering
- Remove a Clustered Data ONTAP Volume

What incremental naming is

Incremental naming is an algorithm that enables you to name the attributes in a workflow based on the search results for a parameter. You can name the attributes based on an incremental value or a custom expression. The incremental naming functionality helps you implement a naming convention based on your requirement.

You can use the incremental naming functionality when designing workflows to dynamically name the objects created by the workflow. The functionality enables you to specify a search criteria for an object using the resource selection feature and the value returned by the search criteria is used for the object's attribute. In addition, you can specify a value for the attribute if no object was found with the specified search criteria.

You can use one of the following options for naming the attributes:

- Providing an increment value and suffix

You can provide a value that should be used along with the value of the object found by the search criteria

and increment with the number you specify. For example, if you want to create volumes with the naming convention of *filer name_unique number_environment*, you can use a finder to find the last volume by its name prefix and increment the unique number by 1, as well as add the suffix name to the volume name. If the last volume name prefix found was *vf_023_prod* and you are creating three volumes, the names for the volumes created are *vf_024_prod*, *vf_025_prod*, and *vf_026_prod*.

- Providing a custom expression

You can provide a value that should be used along with the value of the object found by the search criteria and add additional values based on the expression you enter. For example, if you want to create a volume with the naming convention of *last volume name_environment name padded with 1*, you can enter the expression `last_volume.name + '_' + nextName("lab1")`. If the last volume name found was *vf_023*, the name for the volume created is *vf_023_lab2*.

The following illustration shows how a custom expression can be provided to specify a naming convention:

The screenshot shows a dialog box titled "Incremental Naming Wizard for Volume : name". It contains the following text and fields:

*The Incremental Naming wizard allows you to define the value of **name** based on a search for an existing **Volume***

Search criteria for existing Volume: **Array IP or Name** : 10.25.85.45
 Volume Name Prefix : vf
 vFiler Name : labvFiler

Enter a value for **name** if no **Volume** matches the above search criteria:

If **Volume** was found using above search criteria, set value for **name** by:

Custom expression

At the bottom right are "Save" and "Cancel" buttons.

What conditional execution is

Conditional execution helps you to design workflows that can execute commands when specified conditions are met.

Execution of commands in a workflow can be dynamic. You can specify a condition for the execution of each command or a row of commands in your workflow. For example, you might want the "Add volume to dataset" command to be executed only when a specific dataset is found and you do not want the workflow to fail if the dataset is not found. In this case, you can enable the "Add volume to dataset" command to search for a specific dataset and if it is not found, you can disable the command in the workflow.

Options for conditional execution of commands are available in the *Dictionary object* tab and the Advanced tab of the Parameters for *commands* dialog box.

You can abort a workflow or disable a specific command in the workflow. In addition, you can set a command to be executed using one of the following options:

- Without any condition
- When the variables you have specified are found
- When the variables you have specified are not found
- When the expression you have specified is true

You can also set a command to wait for a specific time interval.

Conditional execution examples in predefined workflows

You can open the command details of the following predefined workflows in the Designer to understand how conditional execution of commands are used:

- Create a basic Clustered Data ONTAP Volume
- Create a Clustered Data ONTAP NFS Volume

How return parameters work

Return parameters are parameters that are available after the planning phase of a workflow. The values returned by these parameters are useful in debugging a workflow. You should understand how return parameters work and what parameters can be used as return parameters to debug workflows.

You can designate a set of parameters, such as variable attributes, expressions, and user input values, in a workflow as return parameters. During workflow execution, the values of the designated parameters are populated in the planning phase and execution of the workflow starts. The values of these parameters are then returned the way they were calculated in that specific execution of the workflow. If you want to debug the workflow, you can refer to the values that were returned by the parameters.

You can specify the required return parameters in a workflow when you want to see what are the calculated or selected values for those parameters. For example, when using resource selection logic to select an aggregate in a workflow, you can specify `aggregate` as the return parameter so that you can see which aggregate was selected during the planning of the workflow.

Before referring to the values of the return parameters for debugging your workflow, you should confirm that the execution of the workflow is complete. The return parameter values are set for each workflow execution. If you have added a return parameter after several executions of a workflow, the value of that parameter is available only for executions after the addition of the parameter.

Parameters that can be used as return parameters

Return parameters	Example
Variable attributes that are scalar	<code>volume1.name</code> , which is an attribute of the “volume name” variable

Return parameters	Example
Constants	MAX_VOLUME_SIZE
User inputs	\$clusterName
MVEL expressions that involve variable attributes, constants, and user inputs	volume1.name+'-'+\$clusterName
The return parameter that a command adds during execution	The \$volumeUUID parameter is added as a return parameter when you use the following line in a PowerShell command: Add-WfaWorkflowParameter -Name "VolumeUUID" -Value "12345" -AddAsReturnParameter \$true.

Examples of return parameters in predefined workflows

If you want to understand how return parameters are specified, you can open the following predefined workflows in the Designer and review the specified return parameters:

- Create an NFS Volume in a vFiler
- Create a Qtree CIFS Share in a vFiler
- Create a Clustered Data ONTAP Volume CIFS Share

What approval points are

Approval points are check points used in a workflow to pause the workflow execution and resume it based on a user approval.

The blue vertical bar shown in the following illustration is an approval point:



You can use approval points for incremental execution of a workflow, where sections of the workflow should be executed only after a certain condition is met. For example, when the next section has to be approved or when successful execution of the first section is validated. Approval points do not handle any process between pausing and resuming of a workflow. Email and SNMP notifications are sent, as specified in the WFA configuration, and the storage operator can be asked to perform certain actions upon receiving the workflow pause notification. For example, the storage operator can send planning details to admin, approver, or operator for approval and resume the workflow when the approval is received.

Approvals might not be required at all times. In some scenarios, the approval might be required only if a particular condition is met and the conditions can be configured when an approval point is added. For example, consider a workflow that is designed to increase the size of a volume. You can add an approval point at the beginning of the workflow for the storage operator to obtain approval from the managers when the increase in the volume size results in an 85% usage of the space in the aggregate that contains the volume. During the workflow execution and on selecting a volume that results in this condition, the execution is stopped until it is approved.

The condition that is set up for the approval point can have one of the following options:

- Without any condition
- When the variable you have specified is found
- When the variable you have specified is not found
- When the expression you have specified evaluates to true

There is no limitation on the number of approval points in a workflow. You can insert approval points before commands in a workflow and set the commands after the approval point to wait for approval before execution. Approval points provide information, such as time of change, user, and comments, allowing you to see when and why the workflow execution was paused or resumed. The approval point comments can include MVEL expressions.

Approval point examples in predefined workflows

You can open the following predefined workflows in the Designer to understand how approval points are used:

- Remove a Clustered Data ONTAP Volume
- Controller and shelf upgrade of an HA pair
- Migrate Volumes

How you execute custom REST end points

OnCommand Workflow Automation (WFA) provides a mechanism to configure the custom REST end points to execute the workflows. Custom REST end points help an architect to configure easy-to-understand, intuitive, and uniform resource identifiers (URIs) to execute workflows, which follow the REST conventions of POST, PUT, or DELETE based on the workflow semantics. These URIs ease the client code development for client developers.

WFA enables you to configure a custom URI path for workflow execution through the API calls. Each segment in the URI path can be a string or a valid name of the user input of the workflow in brackets, for example, `/devops/{ProjectName}/clone`. The workflow can be invoked as a call to `https://WFAServer:HTTPS_PORT/rest/devops/Project1/clone/jobs`.

Validation for the URI path is as follows:


- The REST path must start with “/”.
- The characters allowed are alphabets, digits, and underscore.
- The user input name must be surrounded by “{}”.



You must check that the value surrounded by “{}” is a valid user input name.

- There should be no empty path segments, for example, //, /{}/, and so on.
- The HTTP method configuration and custom URI path configuration should either both be configured or neither configured.

How continue on failure works

The continue on failure feature helps you to configure a step in a workflow so that the workflow execution can continue even if the step fails. You can address the failed steps and resolve the issue that caused the step to fail by accessing the `wfa.log` file or by clicking the  icon.

A workflow that has one or more such failed steps is in the Partially Successful state after the execution is complete. You can configure a step so that the workflow execution continues even if the step fails by selecting the required option in the Advanced tab of the Parameters for `<command_name>` dialog box.

If a step is not configured to continue on failure, the workflow execution is aborted if the step fails.

If a step that is configured to continue on failure fails, you can set the workflow to be executed by using one of the following options:

- Abort workflow execution (default option)
- Continue execution from the next step
- Continue execution from the next row

Sample workflow requirements checklist

A workflow requirements checklist includes detailed requirements—such as commands, user input, and resources—for a planned workflow. You can use the checklist to plan your workflows and identify the gaps in the requirements.

Requirements checklist example

The following sample workflow requirements checklist lists the requirements for the “Create a Clustered Data ONTAP Volume” workflow. You can use this sample checklist as a template to list your workflow requirements.

Workflow details

Requirement	Description
Workflow name	Create a Clustered Data ONTAP Volume
Category	Storage provisioning
Description	The workflow creates a new volume in a specific SVM. This workflow is meant for a scenario where a volume is provisioned and delegated for later usage.

Requirement	Description
High-level description of how the workflow works	<ul style="list-style-type: none"> • The SVM that contains the volume is specified by the user (cluster, SVM names). • A volume is created based on the specified size. • The configuration of the volume is described in a template.

Requirement	Description
Details	<ul style="list-style-type: none"> • Use the Create CM Volume command • Command details for Create CM Volume: <ul style="list-style-type: none"> ◦ Execution is set as always ◦ Volume details are specified by filling in attributes for the volume ◦ Use the Space Guaranteed Settings template for configuring the volume ◦ Volume name and size are provided by user. <p>The volume will be mounted in the SVM namespace as /volname (under the root namespace).</p> <ul style="list-style-type: none"> • Use the actualVolumeSize function because the snap reserve will be 5%. • SVM reference is defined with the following resource selection logic: <ul style="list-style-type: none"> ◦ CM SVM by key — searches for SVM by name and the cluster, which is provided by the user ◦ CM SVM by type — only data SVMs (type = cluster) ◦ SVM by state — (state = running) • Aggregate reference is defined with the resource selection logic as a predefined finder (CM Aggregate by space thresholds and RAID Type): <ul style="list-style-type: none"> ◦ CM Aggregate by available capacity (capacity = size of volume to be provisioned, cluster given by user) ◦ CM Aggregate by delegation to SVM ◦ CM Aggregate by RAID Type (RAID-DP) ◦ CM Aggregate not aggr0 ◦ CM Aggregate by used size % (threshold = 90, spaceToBeProvisioned = size provided, since guarantee is volume) ◦ CM Aggregate by over commitment (threshold = 300, spaceToBeAllocated = Size of volume being provisioned) ◦ Select the aggregate with maximum free space

User inputs

Name	Type	Description (data values, validation, and so on)
Cluster	Locked query (tabular)	<ul style="list-style-type: none"> Cluster hosting the SVM Query can be tabular display with primary address and name of the cluster Sort alphabetically by name
SVM	Locked query	<ul style="list-style-type: none"> SVM in which the volume is provisioned Query should only display SVM names belonging to the cluster chosen in the previous input <p>Show only cluster type SVMs, not admin or node (type column of cm_storage.vserver)</p> <ul style="list-style-type: none"> Sort alphabetically
Volume	String	<ul style="list-style-type: none"> Name of the volume to be created
Size in GB	Integer	<ul style="list-style-type: none"> Size of the volume to be provisioned Data size (snap reserve should be considered)

Commands

Name	Description	Status
Create CM Volume	Creates a volume in the SVM	Existing

Return parameters

Name	Value
Volume name	Name of the provisioned volume
Aggregate name	Name of the selected aggregate
Node name	Name of the node
Cluster name	Name of the cluster

Gaps and issues

1.	
2.	
3.	

4.	
5.	

Create a workflow

You can use Workflow Automation (WFA) to create workflows for tasks such as provisioning, migrating, and decommissioning storage for databases or file systems. You should create workflows when the predefined WFA workflows do not match your requirements.




What you'll need

- You must have understood the concepts for WFA building blocks.
- You must have understood capabilities such as repeat row, approval points, and resource selection that are required for your workflow.
- You must have completed the planning required for your workflow, including the workflow requirement checklist.
- You should have created the help content, which provides information about the workflow to storage operators.


About this task

The construction of each workflow might vary based on the goal and requirement of the workflow. This task does not provide instructions for a specific workflow, but provides general instructions for creating a workflow.


Steps

1. Click **Designer > Workflows**.
2. Click  on the toolbar.
3. In the **Workflow** tab, perform the following steps:
 - a. Expand the required schema, and then double-click the required  (command) or  (workflow) from the **Available Steps** list.

You can repeat this step as required. You can drag-and-drop steps to rearrange the steps in the workflow editor.

- b. **Optional:** Click  to add the required number of rows, which are used to specify details for execution of steps.

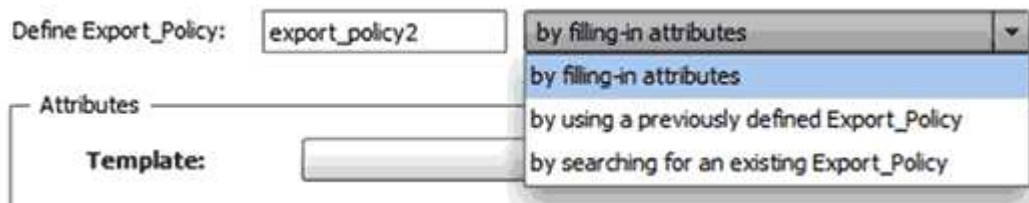
Each step is executed based on the specified step details at the specified row and column. The steps are executed from left to right and in the top to bottom order.

- c. Position your cursor below the step you have added and click  to add step details for the step execution, at the required row.


For this step...	Do this...
Workflow	Enter the required user inputs in the Workflow tab and the required condition in the Advanced tab.

For this step...	Do this...
Command	In the Parameters for <command> tab, click each object tab, select the required option to define the object attributes, and then enter the required details in the Advanced tab and the Other Parameter tab.
Search or define	Select the dictionary entry object that should be searched for or defined.

The following illustration shows the available options for defining the object attributes:




Choose the appropriate action:

For...	Do this...
by filling-in attributes	<p>Enter the value for attributes using the following options:</p> <ul style="list-style-type: none"> • Expressions • Variables • User inputs • Resource selection • Incremental naming You must position your cursor over the attribute fields and click  to use the resource selection or incremental naming capabilities.
by using a previously defined <i>object</i>	Select the previously defined <i>object</i> in the box before the option list.

For...	Do this...
by searching for an existing <i>object</i>	<p>i. Click Enter search criteria to search for the object using the resource selection capability.</p> <p>ii. Select one of the required option for execution if the required object is not found:</p> <ul style="list-style-type: none"> ◦ Abort workflow <p>This option aborts the workflow execution if the specific object is not found.</p> <ul style="list-style-type: none"> ◦ Disable this command <p>This option disables only the current step and executes the workflow.</p> <ul style="list-style-type: none"> ◦ Fill-in attributes for object and execute the command <p>This option enables you to enter the required attributes and execute the workflow.</p>

4. If you want to insert an approval point, click  and enter the required comment for the approval point.

Approval point comments can include MVEL expressions.

5. Click  that is next to the row numbers to perform the following:

- Insert a row.
- Copy the row.
- Repeat the row.

You can use one of the following options to specify repetition of the command parameters:

- Number of times

You can use this option to repeat the command execution for the number of repetitions you specify. For example, you can specify that the “Create qtree” command should be repeated three times to create three qtrees.

You can also use this option for a dynamic number of command executions. For example, you can create a user input variable for the number of LUNs to be created and use the number specified by the storage operator when the workflow is executed or scheduled.

- For every resource in a group

You can use this option and then specify a search criteria for an object. The command is repeated as many times as the object is returned by the search criteria. For example, you can search for the nodes in a cluster and repeat the “Create iSCSI Logical Interface” command for each node.

- Add a condition for execution of the row.

- Remove the row.

6. In the Details tab, perform the following steps:

- Specify the required information in the **Workflow name** and **Workflow Description** fields.

The workflow name and description must be unique for each workflows.

- Optional:** Specify the entity version.
- Optional:** Clear the **Consider Reserved Elements** check box if you do not want to use the reservation capability.
- Optional:** Clear the **Enable element existence validation** check box if you do not want to enable validation for elements that exist with the same name.

7. If you want to edit the user inputs, perform the following steps:

- Click the **User Inputs** tab.
- Double-click the user input you want to edit.
- In the **Edit Variable: <user input>** dialog box, edit the user input.

8. If you want to add constants, perform the following steps

- Click the **Constants** tab, and then add the required constants for your workflow by using the **Add** button.

You can define constants when you are using a common value for defining the parameters for multiple commands. For example, see the AGGREGATE_OVERCOMMITMENT_THRESHOLD constant used in the “Create, map and protect LUNs with SnapVault” workflow.

- Enter the name, description, and value for each constant.

9. Click the **Return Parameters** tab, and then add the required parameters for your workflow by using the **Add** button.

You can use return parameters when the workflow planning and execution must return some calculated or selected values during planning. You can view the calculated or selected values in the Return Parameters tab of the monitoring window in the workflow preview or after the workflow execution is complete.

Example

Aggregate: You can specify aggregate as a return parameter to see which aggregate was selected using the resource selection logic.

If you have included a child workflow in your workflow and if the child workflow return parameter names contain a space, dollar sign (\$), or a function, you should specify the return parameter name within square brackets in the parent workflow to view the child workflow return parameter value in your parent workflow.

If the parameter name is...	Specify as...
ChildWorkflow1.abc\$value	ChildWorkflow1["abc\$"+"value"]
ChildWorkflow1.\$value	ChildWorkflow1["\$"+"value"]
ChildWorkflow1.value\$	ChildWorkflow1.value\$

If the parameter name is...	Specify as...
ChildWorkflow1.P N	ChildWorkflow1["P N"]
ChildWorkflow1.return_string("HW")	ChildWorkflow1["return_string(\\"HW\\")"]

10. **Optional:** Click the **Help Content** tab to add the help content file you have created for the workflow.
11. Click **Preview** and ensure that the planning of the workflow is completed successfully.
12. Click **OK** to close the preview window.
13. Click **Save**.

After you finish

Test the workflow in your test environment, and then mark the workflow as ready for production in **WorkflowName > Details**.

Create workflow help content

OnCommand Workflow Automation (WFA) admins and architects who design workflows can create help content for the workflows and include it in the workflow.

What you'll need

You must be aware of how to create web pages using HTML.

About this task

The help should provide information about the workflow and the user inputs for the workflow to the storage operator who executes the workflow.

Steps

1. Create a folder with the following name: workflow-help.
2. Author the help content using an HTML editor or a text editor and save it as an `index.htm` file in the workflow-help folder.

You must not include JavaScript files as part of the help content. The following are the supported file extensions:

- .jpg
- .jpeg
- .gif
- .png
- .xml
- .thmx
- .htm
- .html
- .css

You can also include the `Thumbs.db` file, which is created by Windows.

3. Verify that the `index.htm` file and other files associated with the help content, such as images, are available in the `workflow-help` folder.
4. Create a `.zip` file of the folder and ensure that the size of the `.zip` file is not more than 2 MB.

Example

Create an NFS `volume-help.zip`

5. Edit the workflow for which you have created the help content, and then click **Setup > Help Content > Browse** to upload the `.zip` file.

Create WFA workflow packs

You can create workflow packs on OnCommand Workflow Automation (WFA) for your storage automation and integration requirements.

Steps

1. Log in to the **WFA** window through a web browser.
2. Click **Designer > Pack**.
3. Click the **New Pack** icon.
4. In the **New Pack** dialog box, enter values for the **Name**, **Author**, **Version**, and **Description** fields.
5. Click **Save**.
6. Verify that the new pack is created in the **Packs** window.

Add entities to WFA workflow packs

You can add one or more entities to a workflow pack in OnCommand Workflow Automation (WFA) for your storage automation and integration requirements.

About this task

You can remove a pack from the following entities:

- Workflow
- Finders
- Filters
- Commands
- Functions
- Templates
- Schemes
- Dictionary
- Data Sources Types
- Remote System Types
- Cache Queries

- Categories

Steps

1. Log in to the **WFA** window through a web browser.
2. Click **Designer > <Entities>**.
3. In the Entity window, select the entity you want to add to the pack.
4. Click the **Add To Pack** icon.

“Add To Pack” is enabled only for entities for which the certification is set to **None**.

5. In the **Add To Pack <Entity>** dialog box, from the **Available Packs** drop-down list, select the pack to which you want to add the entity.
6. Click **OK**.

Delete OnCommand Workflow Automation packs

You can delete a pack from OnCommand Workflow Automation (WFA) if you no longer require it. If you delete a pack, all the entities associated with the pack are deleted.

About this task


- You cannot delete a pack if there are any dependencies on the entities that are part of the pack.

For example, if you attempt to delete a pack that includes a command that is part of a custom workflow, the delete operation will fail because the custom workflow is dependent on the pack. You can delete the pack only after you delete the custom workflow.

- Entities that are part of a pack cannot be deleted individually.

To delete an entity that is part of a pack, you must delete the pack containing that entity. If an entity is part of multiple packs, the entity will be deleted from the WFA server only when all the packs containing that entity are deleted.

Steps

1. Log in to WFA through a web browser as an admin.
2. Click **Designer > Packs**.
3. Select the pack you want to delete and click .
4. In the **Delete Pack** confirmation dialog box, click **OK**.

Export OnCommand Workflow Automation content

You can save user-created OnCommand Workflow Automation (WFA) content as a .dar file and share the content with other users. The WFA content can include the entire user-created content or specific items such as workflows, finders, commands, and dictionary terms.




What you'll need

- You must have access to the WFA content that you want to export.
- If content that is to be exported contains references to certified content, the corresponding certified content

packs must be available on the system when the content is imported.

These packs can be downloaded from the Storage Automation Store.

About this task

- You cannot export the following types of certified content:
 -  - NetApp-certified content
 -  - content developed by Professional Services (PS), which is available only on custom installations made by PS
 -  - packs developed by users
- All of the objects that are dependent on the exported object are also exported.


For example, exporting a workflow also exports the dependent commands, filters, and finders for the workflow.

- You can export locked objects.

The objects remain in the locked state when they are imported by other users.

Steps

1. Log in to WFA through a web browser.
2. Export the necessary content:

If you want to...	Do this...
Export all user-created content as a single .dar file	<ol style="list-style-type: none">a. Click Settings, and under Maintenance click Export All Workflows.b. Specify a file name for the .dar file, and then click Export.
Export specific content	<ol style="list-style-type: none">a. Navigate to the window from which you want to export content.b. Select one or more items from the window, and then click .c. In the Export As dialog box, specify a file name for the .dar file, and then click Export.

3. In the **Save As** dialog box, specify the location where you want to save the .dar file, and then click **Save**.

Import OnCommand Workflow Automation content

You can import user-created OnCommand Workflow Automation (WFA) content such as workflows, finders, and commands. You can also import content that is exported from another WFA installation, content that is downloaded from the Storage Automation Store or the WFA community, as well as packs, including Data ONTAP PowerShell toolkits and Perl NMSDK toolkits.

What you'll need

- You must have access to the WFA content that you want to import.
- The content that you want to import must have been created on a system that is running the same version or an earlier version of WFA.

For example, if you are running WFA 2.2, you cannot import content that was created using WFA 3.0.

- If the `.dar` file references NetApp-certified content, the NetApp-certified content packs must be imported.

The NetApp-certified content packs can be downloaded from the Storage Automation Store. You must refer to the documentation of the pack to verify that all requirements are met.

Steps

1. Log in to WFA through a web browser.
2. Click **Settings**, and under **Maintenance** click **Import Workflows**.
3. Click **Choose File** to select the `.dar` file that you want to import, and then click **Import**.
4. In the **Import Success** dialog box, click **OK**.

Related information

[NetApp community: OnCommand Workflow Automation](#)

Import WFA workflow packs

You can import workflow packs from the server to OnCommand Workflow Automation (WFA) for your storage automation and integration requirements.

What you'll need

You must have access to the WFA content in the server that you want to import.

Steps

1. Log in to the **WFA** window through a web browser.
2. Click **Designer > Pack**.
3. Click the **Import From Server** icon.
4. In the Import From Server Folder dialog box, in the **Folder location at server system** field, enter the location of the pack in the server in a string format, for example, `C:\work\packs\test`.
5. Click **OK**.
6. Verify that the pack is imported in the **Packs** window.

Considerations while importing OnCommand Workflow Automation content

You must be aware of certain considerations when you import user-created content, content that is exported from another OnCommand Workflow Automation (WFA) installation, or content that is downloaded from the Storage Automation Store or the WFA community.

- WFA content is saved as a `.dar` file and can include the entire user-created content from another system or specific items such as workflows, finders, commands, and dictionary terms.

- When an existing category is imported from a `.dar` file, the imported content is merged with the existing content in the category.

For example, consider there are two workflows WF1 and WF2 in category A in the WFA server. If workflows WF3 and WF4 in category A are imported to the WFA server, category A will contain workflows WF1, WF2, WF3, and WF4 after the import.

- If the `.dar` file contains dictionary entries, then the cache tables corresponding to the dictionary entries are automatically updated.

If the cache tables are not updated automatically, an error message is logged in the `wfa.log` file.

- When importing a `.dar` file that has a dependency on a pack that is not present in the WFA server, WFA tries to identify whether all the dependencies on the entities are met.
 - If one or more entities are missing or if a lower version of an entity is found, the import fails and an error message is displayed.

The error message provides details of the packs that should be installed in order to meet the dependencies.

- If a higher version of an entity is found or if the certification has changed, a generic dialog box about the version mismatch is displayed, and the import is completed.

The version mismatch details are logged in a `wfa.log` file.

- Questions and support requests for the following must be directed to the WFA community:
 - Any content downloaded from the WFA community
 - Custom WFA content that you have created
 - WFA content that you have modified

Pack identification during upgrade

During the upgrade process, OnCommand Workflow Automation (WFA) identifies and classifies the entities into a pack. If you had deleted any entity of a pack before the upgrade, the pack will not be identified during the upgrade.

During the upgrade process, WFA compares the packs in the database with the list of packs that were released in the Storage Automation Store to identify the packs that were installed before the upgrade. Pack identification thus classifies existing packs in the database.

WFA performs the following processes to identify and classify packs:

- Maintains a list of packs released in the Storage Automation Store to compare and identify the packs that were installed before the upgrade.
- Classifies the entities in a pack as part of the Storage Automation Store synchronization, if Storage Automation Store is enabled.
- Classifies the entities into packs using the updated list.

Pack identification is applicable only to NetApp-certified packs that were downloaded from the Storage Automation Store.

If a pack is not identified during upgrade, you can re-import the pack to get it identified in WFA. The wfa.log files provide details about the entities that were not identified as a pack during the upgrade.

Integrating WFA workflow packs with the SCM repository

You can integrate the OnCommand Workflow Automation (WFA) pack with the Source Control Management (SCM) repository.

You must have admin or architect credentials.

SCM tools such as GitHub, Perforce, and SVN require you to map a local directory to check out the code from the SCM repository server. This local directory mapping is called the *SCM client location*. You must set up the SCM client with a file system location as a client area.

You can set up the SCM client on a WFA server system. You must have access to the WFA server system for SCM operations

Check in a new workflow pack to SCM

You can create a new workflow pack with OnCommand Workflow Automation (WFA) and check it in to Source Control Management (SCM).

What you'll need

SCM must be set up and you must have admin or architect credentials.

Steps

1. Log in to the **WFA** window through a web browser.
2. Create a new workflow pack.

[Create a workflow automation pack](#)

3. Add entities to the pack you created.

[Add entities to an OnCommand Workflow Automation pack](#)

4. Click the **Export To Server** icon.
5. In the **Export To Server Folder** dialog box, in the **Folder location at server system** field, enter the file system location where the pack is to be saved in the server containing the SCM client.

To edit or re-export the pack or the contents, click the **Unlock** icon.

6. In the SCM client location, check in the pack content to the SCM server.

Check in a new version of a WFA workflow pack

You can update the version of a pack in OnCommand Workflow Automation (WFA), and then check in the updated pack to a new location in the Source Control Management (SCM) server.

What you'll need

The SCM must be set up and you must have admin or architect credentials.

Steps

1. Log in to the **WFA** window through a web browser.
2. Click **Designer > Pack**.
3. Click the **Edit Pack** icon.
4. In the **Pack <pack name>** dialog box, in the **Version** field, update the version of the pack.
5. Click **Save**.
6. Click the **Export To Server** icon at the pack level.
7. In the Export To Server Folder dialog box, in the **Folder location at server system** field, enter a new file system location.

If the pack was previously saved in the C:\p4\cdot\1.0.0 file system location, now save it in the C:\p4\cdot\2.0.0 location.

8. In the SCM client location, check in the pack content to a new location in the SCM server.

If the pack was previously saved in the //depot/wfa/packs/cdot/1.0.0 path in the SCM server, you can save it in another location, such as //depot/wfa/packs/cdot/2.0.0.

Update WFA workflow packs from the SCM server

You can update a pack in the Source Control Management (SCM) server, and then import the updated pack to OnCommand Workflow Automation (WFA).

What you'll need

The SCM must be set up and you must have admin or architect credentials

About this task

If you make any changes or updates to a pack in the SCM server, the admin or architect needs to resolve the conflicts, if any, using the SCM-provided diff tools. WFA tailors the XML diff file to show only the relevant changes.

Before importing the pack, you are notified about the changes going in to the WFA pack content.

Steps

1. Log in to the **WFA** window through a web browser.
2. Import the updated pack to WFA.

Import WFA workflow packs



If the WFA database already contains the same pack, then the pack content will be overwritten.

Check in existing WFA workflow packs to the SCM server

You can check in already existing packs to the Source Control Management (SCM) server from OnCommand Workflow Automation (WFA).

What you'll need

The SCM must be set up and you must have admin or architect credentials.

Steps

1. Log in to the **WFA** window through a web browser.
2. Click **Designer > Pack**.
3. Click the **Export To Server** icon.
4. In the Export To Server Folder dialog box, in the **Folder location at server system** field, enter the server folder location where the pack is saved in the server.

This exports the pack in an exploded form in the file system where the SCM client is created.

5. In the SCM client location, check in the pack content to the SCM server.
6. Use the SCM-provided diff tools to verify the changes against the SCM version of the pack.

Remove WFA workflow packs from entities

You can remove a pack from the entities in OnCommand Workflow Automation (WFA) and check in the updated pack to the Source Control Management (SCM) server.

What you'll need

The SCM must be set up and you must have admin or architect credentials.

About this task

You can remove a pack from the following entities:

- Workflow
- Finders
- Filters
- Commands
- Functions
- Templates
- Schemes
- Dictionary
- Data Sources Types
- Remote System Types
- Cache Queries
- Categories

Steps

1. Log in to the **WFA** window through a web browser.
2. Click **Designer > <Entity>**.
3. Click the **Remove From Pack** icon.
4. In the **Remove From Pack <Entity>** dialog box, select the pack you want to delete from that entity.
5. Click **OK**.

6. Click the **Packs** tab.
7. Click the **Export To Server** icon.
8. In the **Export To Server Folder** dialog box, in the **Folder location at server system** field, enter the server folder location where the pack is saved in the server.

This exports the pack in exploded form in the file system where the SCM client is created.

9. In the SCM client location, check in the pack content to the SCM server.
10. Use the SCM-provided diff tools to verify the changes against the SCM version of the pack.

Roll back a WFA workflow pack to its previous version in SCM

You can roll back a pack to the previous version in Source Control Management (SCM) and import it to OnCommand Workflow Automation (WFA).

What you'll need

The SCM must be set up and you must have admin or architect credentials.

Steps

1. In the SCM client location, roll back the pack to a previous version in the file system location using SCM tools.

The SCM client gets synced to the exact change number that you are interested in.

2. Log in to the **WFA** window through a web browser.
3. Import the updated pack to WFA.

Import WFA workflow packs



This rolls back the WFA database to the previous version.

Creating building blocks for workflows

Workflow Automation (WFA) includes several building blocks, which are used to construct workflows. You can create the WFA buildings blocks that are required for your workflows.

Create a data source type


You must create a data source type to enable data acquisition from a data source, which is not predefined in OnCommand Workflow Automation (WFA).

What you'll need

- You must have created the required dictionary entry and scheme if you are creating a custom data source type that is not predefined in WFA.
- You must be aware of PowerShell scripting to create a data source type that uses the script method.

Steps

1. Click **Designer > Data Source Types**.


2. Click  on the toolbar.
3. In the **New Data Source Type** dialog box, enter or select the required details in the **Data source**, **Data source version**, and **Scheme** fields.
4. In the **Default port** field, enter the port number.

Example

2638

The port number you have entered is populated when you add this data source type for data acquisition. By default, the port is used by WFA to communicate with the data source and the port should be open on the data source server.

5. From the **Method** list, select the method that WFA must use to acquire data:

If you have selected...	Then...
SQL	<p>From the Driver type list, select one of the following drivers that is appropriate for the data source:</p> <ul style="list-style-type: none"> • Sybase jConnect3 • MySQL Connector/J • MySQL Server JDBC Driver 3.0 • Oracle JDBC Driver 11.2.0.3
SCRIPT	<p>In the Script field, enter the PowerShell script that is used to connect and retrieve data from the data source.</p> <div>  <p>The data in the dictionary entry equivalent CSV files should include tabs as field separator. For example, see the PowerShell script for the VMware vCenter data source type.</p> </div>

6. Click **Save**.


Create a command

You can create a WFA command to complete a specific task in your workflow if there is no predefined WFA command that is suitable for the task.

What you'll need

You must know how to write the required code for the command using either PowerShell or Perl.

Steps

1. Click **Designer > Commands**.
2. Click  on the toolbar.

3. In the **Properties** tab of the **New Command Definition** dialog box, enter or select the required details in the **Name**, **Description**, and **Timeout** fields.
 - a. In the **String Representation** field, enter a string representation for the command using MVEL syntax.

Example

```
VolumeName + "=>" + SnapshotName
```

The string representation for a command is used to display the information you want to see in the workflow design during planning and execution. You must use only the parameters of the command in string representation for a command.

- b. **Optional:** If you are creating a wait command, select **Wait for condition** in the **Command type** section, and set the required value in the **Waiting interval (s)** field.
4. In the **Code** tab, perform the following steps:
 - a. Select the required scripting language for the command from the **Script Language** list.

You can click + and select an additional language for the command.

- b. Enter the appropriate code for the command in the selected language tab.

If you want to use password type for user inputs in the PowerShell script, you must create an alias for the parameter and include `_Password` in the attribute. For Perl script, you can specify the type as **Password** in the **Parameters Definition** tab.

Example

```
param (  
    [parameter(Mandatory=$false, HelpMessage="Specify an AD  
administrator password.")]  
    [Alias("ADAdminPassword_Password")] [string]$ADAdminPassword  
)
```

5. In the **Parameters Definition** tab, perform the following steps:
 - a. Click **Discover Parameters** to populate the parameters definition table.

The parameters and their attributes are extracted from the code and displayed in the table. For example, the `Array` and `VolumeName` parameters are extracted from the following code:

```
param (  
    [parameter(Mandatory=$true, HelpMessage="Array name or IP  
address")]  
    [string]$Array,  
  
    [parameter(Mandatory=$true, HelpMessage="Volume name")]  
    [string]$VolumeName,  
)
```

- b. Click the description column of the parameters to edit the description.

You cannot edit any other field in this tab.

6. In the **Parameters Mapping** tab, perform the following steps for each parameter:

- a. From the **Type** column, select the appropriate dictionary object.
- b. In the **Attribute** column, enter or select the appropriate attributes for the dictionary object from the list.

After entering an attribute, you can enter a period (.) and include another attribute of that object.

Example

Enter `cm_storage.volume` as type and `aggregate.name` as the attribute for the `AggregateName` parameter.

- c. In the **Object Name** column, enter an object name.

The object name is used for grouping the parameters under a tab in the Parameters for <command> dialog box when you are specifying the command details in a workflow.

The unmapped parameters are displayed in the **Other parameters** tab of the Parameters for <command> dialog box when you are specifying the command details in a workflow.

7. **Optional:** In the **Reservation** tab, enter a reservation script using SQL queries to reserve the resources that are required by the command during a scheduled workflow execution:
 - a. In the **Reservation Representation** field, enter a string representation for the reservation using MVEL syntax.

Example

```
"Add rule for SnapMirror label " + SnapMirrorLabel + " to the SnapMirror policy "
+ PolicyName + """
```

The string representation is used to display the details of the resources reserved in the Reservations window.



The reservation script must not perform any operation on databases except the `cm_storage`, `cm_performance`, `storage`, `performance`, `vc`, and custom schemes.

8. **Optional:** In the **Verification** tab, enter an SQL query to verify whether the command has affected the data sources and the WFA cache as expected so that the reservation can be removed.

The SQL query that you enter can only consist of SQL SELECT statements.

- a. Click **Test Verification** to test the verification script.
- b. In the **Verification** dialog box, enter the required test parameters.
- c. If you do not want to use the reservation data to test the verification script, clear the **Use reservation data in test** field.
- d. Click **Test**.
- e. After reviewing the test result, close the dialog box.

9. Click **Test** to test the command.
10. In the Testing Command <command name> dialog box, click **Test**.

The result of the test is displayed in the Log messages section of the dialog box.

11. Click **Save**.

Test the reservation script for commands

You can test the reservation scripts you have written for OnCommand Workflow Automation (WFA) commands on the playground database to ensure that the scripts are working fine and not affecting the WFA database tables.

About this task

The default WFA installation path is used in this procedure. If you changed the default location during installation, you must use the changed WFA installation path.

Steps

1. Open a command prompt on the WFA server and change directories to the following location:
`c:\Program Files\NetApp\WFA\mysql\bin`
2. Create a dump of the WFA database using the following command:

```
mysqldump -u wfa -pWfa123 --single-transaction --skip-add-drop-table  
database_tables> dump_location
```

Example

Command to create a dump of the cm_storage database tables:

```
mysqldump -u wfa -pWfa123 --single-transaction --skip-add-drop-table  
cm_storage> c:\tmp\cmSt2.sql
```

3. Restore the dump you have created on to the WFA playground database using the following command:

```
mysql -u wfa -pWfa123 playground < dump_location
```

Example

```
mysql -u wfa -pWfa123 playground < c:\tmp\cmSt2.sql
```

4. Create or edit a WFA command and write the reservation script in the **Reservation** tab.

You must ensure that the reservation and verification scripts use only the playground database.

5. Create or edit a workflow, include the command in the workflow, and then execute the workflow.
6. Verify that the reservation and verification scripts are working as expected.

The WFA data source acquisition process does not update the playground database. You must remove the reservations created by the command manually.



Create a finder

You can create a WFA finder that can search for resources if there is no predefined WFA finder that is suitable for searching the required resources.


What you'll need

You must have created the required filters that are used in the finder.

Steps

1. Click **Designer > Finders**.
2. Click  on the toolbar.
3. In the **Properties** tab of the **New Finder** dialog box, enter or select the required details in the **Name**, **Type**, and **Description** fields.
4. In the **Filters** tab, select the required filters from the **Available Filters** list and click .

You can add or remove filters based on your requirement.

5. In the **Returned Attributes** tab, select the required attributes for the filter from the **Available** list and click .
6. **Optional:** Click **Test** to test the finder.
 - a. In the **Test Finder <FinderName>** dialog box, enter the required test parameters.
 - b. Clear the **Use reservation data in test** check box if you do not want to use the reservation data for testing the finder.
 - c. Click **Test**.

The result of the test is displayed.

- d. Close the dialog box.
7. Click **Save**.


Create a filter

You can create a WFA filter that can search for resources if there is no predefined WFA filter that is suitable for the task.

What you'll need

You must know the appropriate SQL syntaxes to create the filter.

Steps

1. Click **Designer > Filters**.
2. Click  on the toolbar.
3. In the **Properties** tab of the **New Filter** dialog box, enter or select the required details in the **Name**, **Dictionary type**, and **Description** fields.
4. In the **Query** tab, enter the appropriate SQL query for the filter.

You must enter a single SQL query and optionally use input parameters. You should use the following

syntax to use an input parameter: `${ParameterName}`.

```
SELECT
    array.ip
FROM
    storage.array
WHERE
    array.name = '${ArrayName}'
```

5. Click **Refresh** to populate the **Input Parameters** table and the **Returned Attributes** list.

This information is obtained from the SQL query that you have entered. For example, if you use the SQL query example from the previous step, ip is displayed in Returned Attributes and ArrayName is displayed in Input Parameters. You can edit the entries in the **Label** and **Description** columns.

6. **Optional:** Click **Test** to test the filter.
 - a. In the Test Filter <FilterName> dialog box, enter the required test parameters.
 - b. Clear the **Use reservation data in test** check box if you do not want to use the reservation data for testing the filter.
 - c. Click **Test**.


The test result is displayed.

- d. Close the dialog boxes.
7. Click **Save**.

Create a dictionary entry

You can create a WFA dictionary entry when you want to define a new object type and its relationship in your storage environment.

Steps

1. Click **Designer > Dictionary**.
2. Click  on the toolbar.
3. In the **New Dictionary Entry** dialog box, enter the required details in the **Name of object type** and the **Description** fields.
4. For the **Scheme** field, perform one of the following actions:
 - Select one of the available scheme from the list.
 - Click **Add New Scheme**, enter the required **Scheme Name** in the New Scheme dialog box, and then click **Add**.
5. Click **Add row**, and perform the following steps to describe the attribute:
 - a. Click the Name column and enter the name of the attribute.
 - b. From the Type column, select the required type.

The `String Length` column is populated and is editable if you selected string as the type. Also, the

Values column is editable if you selected **enum** as the type.

- c. Select the appropriate check boxes for the attribute from the **Natural Key**, **To be Cached**, and **Can be Null** columns.

If you have selected the **Natural Key** check box, you cannot select the **Can be Null** check box.

- d. Add the required attributes for the dictionary object.

- e. **Optional:** Select the **Values in natural key columns are case sensitive** check box if you want the natural keys to be case-sensitive.

6. Click **Save**.

Create a function

You can create a WFA function that can be used as a utility, if there is no predefined WFA function that is suitable for your task.

What you'll need

You must know MVFLEX Expression Language (MVEL) syntaxes to create a function.

About this task


You must include the following for the function definitions:

- Name: name of the function

You must not use a reserved word in MVEL syntax. Each function must have a unique name.

- MVEL definition: a string specifying the MVEL syntax of the function definition

Steps

1. Click **Designer > Functions**.
2. Click  on the toolbar.
3. In the **New Function** dialog box, enter or select the required details in the **Function description** and **Function definition** fields.

Example

```
def actualVolumeSize(data_size, snap_pct)
{
    if (snap_pct < 0 ) {
        snap_pct = 0;
    } else if (snap_pct > 99) {
        snap_pct = 99;
    }

    div = 1 - (snap_pct/100);
    return (int)(data_size/div);
}
```

The **Function name** field is populated from the data that is used in the MVEL syntax.

4. **Optional:** Click **Test** to test the function:

- a. In the **Expression** section of the **Test** dialog box, enter the required expression of function.

Example

```
actualVolumeSize(600, 1)
```

- b. Click **Test**.

The test result is displayed.


- c. Close the dialog box.

5. Click **Save**.

Create a template

You can create a template that can be used as a blueprint for filling up attributes in command details.

Steps

1. Click **Designer > Templates**.
2. Click  on the toolbar.
3. In the **New Template** dialog box, enter or select the required details in the **Name**, **Type**, and **Description** fields.

The Attributes table is populated based on the dictionary object you have selected in the **Type** field.

4. Click the value column of each attribute and perform one of the following:
 - Enter or select the required value from the list.
 - Enter a user input entry—for example, `$size` for `size` user input.
5. Click **Save**.


Create a cache query

You can define a cache query when you want to cache information about a dictionary object in the WFA database from a data source type. You can create a cache query and associate it with a dictionary entry and one or more data source types, such as OnCommand Unified Manager 6.1.

What you'll need

You must know the appropriate SQL syntaxes to create a cache query.

Steps

1. Click **Designer > Cache Queries**.
2. Click  on the toolbar.
3. In the **Add Cache Query** dialog box, select the required dictionary entry and data source type.

4. In the “SQL select query” section, enter the appropriate SQL query.

Example

The following SQL query caches information about the disk dictionary object from the OnCommand Unified Manager 6.1 data source type:

```
SELECT
    disk.objId AS id,
    disk.name AS NAME,
    disk.uid AS uid,
    disk.effectiveInterfaceType AS TYPE,
    disk.rpm AS rpm,
    disk.homeNodeId AS home_node_id,
    disk.ownerNodeId AS owner_node_id,
    disk.model AS model,
    disk.serialNumber AS serial_number,
    disk.totalBytes/1024/1024 AS size_mb,
    disk.shelf AS shelf,
    disk.shelfBay AS shelf_bay,
    disk.pool AS pool,
    disk.vendor AS vendor,
    LOWER(disk.raidPosition) AS raid_position,
    disk.containerTypeRaw AS container_type,
    disk.clusterId AS cluster_id
FROM
    netapp_model_view.disk disk
```

5. If you want to test the SQL query, click **Test**.

If you have selected more than one data source type, the Test Cache Query dialog box opens and enables you to select the required data source type.

The test result is displayed.

6. Close the dialog box.

7. Click **Save**.


Create recurring schedules

OnCommand Workflow Automation (WFA) provides two scheduling options for workflows. You can either schedule a workflow to execute once at a specific time or you can create recurring schedules and associate the schedules to workflows so that the workflows are executed routinely.

About this task

A schedule that you have created can be reused and associated with several workflows.

Steps

1. Click **Execution > Schedules**.
2. Click  on the toolbar.
3. In the **New Schedule** dialog box, enter or select a name, description, and frequency for the schedule.

For frequency, you must enter time in 24-hour format. The WFA server time is applied to the schedules.

4. Click **OK**.
 - You can associate the schedule to a workflow when you execute the workflow by using the **Execute recurrently** option.
 - You can view the details of a workflow and its association with a schedule by clicking **Execution > Recurring Schedules**.

The resource and execution planning for the workflows that are scheduled to execute once are done immediately when the workflows are schedules. However, the resource and execution planning for the workflows with recurring schedule occur at the scheduled time and not when the schedule is associated with a workflow.

Define filter rules

You can define a set of rules for filtering dictionary entry resources such as vFiler units, aggregates, and virtual machines. You can customize the rules for existing workflows and for new workflows while you are creating them.







Steps

1. Log in to WFA through a web browser as an admin.
2. Click **Designer > Workflows**.
3. In the **Workflows** window, double-click the workflow that you want to modify.

The Workflow <workflow name> window is displayed.

4. Define a set of rules by choosing one of the following options:

If you want to...	Then do this...
Search for resources when the commands in a row are repeated	<ol style="list-style-type: none">a. Click a row number and select Repeat row.b. In the Row Repetitions dialog box, select the For every resource in the group option from the Repeats drop-down list.c. Select a resource type.d. Click the Enter search criteria link.

If you want to...	Then do this...
Search for resources required in command inputs	a. Click  . b. In the Parameters for <command_name> dialog box, select the by searching for an existing <dictionary object> option from the Define <dictionary object> drop-down list. c. Click the Enter search criteria link.
Search for resources referenced by variables in command inputs	a. Click  . b. In the Parameters for <command_name> dialog box, select the by filling in attributes option from the Define <dictionary object> drop-down list. c. Click  for a field marked with  .
Name command inputs of String type	a. Click  . b. In the Parameters for <command_name> dialog box, select the by filling in attributes option from the Define <dictionary object> drop-down list. c. Click  for a string field.

5. In the **Resource Selection** dialog box, select the **Define filter rules** check box.

If you have selected one of the options from the Finder drop-down of Resource Selection dialog box, the Define filter rules check box is disabled. The value for the finder must be set to “None” for the Define filter rules to be enabled.

6. Enter the attribute, operator, and value for the rule.

The value must be provided within single quotation marks. The filter rules can contain one or more groups.


7. Click **OK**.

Add approval points

You can add an approval point as a checkpoint in a workflow to pause the workflow execution and resume it based on your approval. You can use approval points for incremental execution of a workflow, where sections of the workflow are executed only after a certain condition is met—for example, when the next section has to be approved or when successful execution of the first section is validated.

Steps

1. Log in to WFA through a web browser as an architect or an admin.
2. Click **Designer > Workflows**.
3. In the **Workflows** window, double-click the workflow that you want to modify.

4. In the **Workflow <workflow name>** window, click the  icon to the left of the step for which you want to add the approval point.

You can add approval points for one or more steps.

5. In the **New Approval Point** dialog box, provide the comment and condition details.
6. Click **OK**.

Coding guidelines for WFA

You should understand the general OnCommand Workflow Automation (WFA) coding guidelines, naming conventions, and recommendations on creating various building blocks such as filters, functions, commands, and workflows.

Guidelines for variables

You must be aware of the guidelines for PowerShell and Perl variables in OnCommand Workflow Automation (WFA) when you create a command or a data source type.

PowerShell variables

Guidelines	Example
For script input parameters: <ul style="list-style-type: none">• Use Pascal case.• Do not use underscores.• Do not use abbreviations.	<code>\$VolumeName</code> <code>\$AutoDeleteOptions</code> <code>\$Size</code>
For script internal variables: <ul style="list-style-type: none">• Use Camel case.• Do not use underscores.• Do not use abbreviations.	<code>\$newVolume</code> <code>\$qtreeName</code> <code>\$time</code>
For functions: <ul style="list-style-type: none">• Use Pascal case.• Do not use underscores.• Do not use abbreviations.	<code>GetVolumeSize</code>
Variable names are not case-sensitive. However, to improve readability, you should not use different capitalization for the same name.	<code>\$variable</code> is the same as <code>\$Variable</code> .
Variable names should be in plain English and should be related to the functionality of the script.	Use <code>\$name</code> and not <code>\$a</code> .

Guidelines	Example
Declare the data type for each variable, explicitly.	[string]name [int]size
Do not use special characters (! @ # & % , .) and spaces.	None
Do not use PowerShell reserved keywords.	None
Group the input parameters by placing the mandatory parameters first followed by the optional parameters.	<pre>param ([parameter (Mandatory=\$true)] [string]\$Type, [parameter (Mandatory=\$true)] [string]\$Ip, [parameter (Mandatory=\$false)] [string]\$VolumeName)</pre>
Comment all input variables using HelpMessage annotation with a meaningful help message.	<pre>[parameter (Mandatory=\$false, HelpMe ssage="LUN to map")] [string]\$LUNName</pre>
Do not use "Filer" as a variable name; use "Array" instead.	None
Use ValidateSet annotation in cases where the argument gets enumerated values. This automatically translates to Enum data type for the parameter.	<pre>[parameter (Mandatory=\$false, HelpMe ssage="Volume state")] [ValidateSet ("online", "offline", "r estricted")] [string]\$State</pre>

Guidelines	Example
Add an alias to a parameter that ends with “_Capacity” to indicate that the parameter is of capacity type.	<p>The “Create Volume” command uses aliases as follows:</p> <pre>[parameter(Mandatory=\$false,HelpMessage="Volume increment size in MB")] [Alias("AutosizeIncrementSize_Capacity")] [int]\$AutosizeIncrementSize</pre>
Add an alias to a parameter that ends with “_Password” to indicate that the parameter is of password type.	<pre>param ([parameter(Mandatory=\$false, HelpMessage="In order to create an Active Directory machine account for the CIFS server or setup CIFS service for Storage Virtual Machine, you must supply the password of a Windows account with sufficient privileges")] [Alias("Pwd_Password")] [string]\$ADAdminPassword)</pre>

Perl variables

Guidelines	Example
For script input parameters: <ul style="list-style-type: none"> • Use Pascal case. • Do not use underscores. • Do not use abbreviations. 	<pre>\$VolumeName \$AutoDeleteOptions \$Size</pre>
Do not use abbreviations for script internal variables.	<pre>\$new_volume \$qtrees_name \$time</pre>
Do not use abbreviations for functions.	<pre>get_volume_size</pre>

Guidelines	Example
Variable names are case-sensitive. To improve readability, you should not use different capitalization for the same name.	<code>\$variable</code> is not the same as <code>\$Variable</code> .
Variable names should be in plain English and should be related to the functionality of the script.	Use <code>\$name</code> and not <code>\$a</code> .
Group the input parameters by placing the mandatory parameters first, followed by the optional parameters.	None
In <code>GetOptions</code> function, explicitly declare the data type of each variable for input parameters.	<pre>GetOptions ("Name=s"=>\\$Name, "Size=i"=>\\$Size)</pre>
Do not use “Filer” as a variable name; use “Array” instead.	None
Perl does not include the <code>ValidateSet</code> annotation for enumerated values. Use explicit “if” statements for cases where argument gets enumerated values.	<pre>if (defined\$SpaceGuarantee&&!(\$SpaceG uaranteeeq'none' \$SpaceGuaranteee q'volume' \$SpaceGuaranteeeq'file')) { die'Illegal SpaceGuarantee argument: \''.\$SpaceGuarantee.'\''; }</pre>
All Perl WFA commands must use the “strict” pragma to discourage the use of unsafe constructs for variables, references, and subroutines.	<pre>use strict; # the above is equivalent to use strictvars; use strictsubs; use strictrefs;</pre>

Guidelines	Example
<p>All Perl WFA commands must use the following Perl modules:</p> <ul style="list-style-type: none"> • Getopt <p>This is used for specifying input parameters.</p> <ul style="list-style-type: none"> • WFAUtil <p>This is used for utility functions that are provided for command logging, reporting command progress, connecting to array controllers, and so on.</p>	<pre>use Getopt::Long; use NaServer; use WFAUtil;</pre>

Guidelines for indentation

You must be aware of the guidelines for indentation when writing a PowerShell or Perl script for OnCommand Workflow Automation (WFA).

Guidelines	Example
A tab is equal to four empty spaces.	
Use tabs and braces to show the beginning and end of a block.	<p>PowerShell script</p> <pre>if (\$pair.length-ne 2) { throw "Got wrong input data" }</pre> <p>Perl script</p> <pre>if (defined \$MaxDirectorySize) { # convert from MBytes to Bytes my \$MaxDirectorySizeBytes = \$MaxDirectorySize * 1024 * 1024; }</pre>

Guidelines	Example
Add blank lines between sets of operations or chunks of code.	<pre>\$options=\$option.trim(); \$pair=\$option.split(" "); Get-WFALogger -Info -messages \$("split options: "+ \$Pair)</pre>

Guidelines for comments

You must be aware of the guidelines for PowerShell and Perl comments in your scripts for OnCommand Workflow Automation (WFA).

PowerShell comments

Guidelines	Example
Use the # character for a single line comment.	<pre># Single line comment \$options=\$option.trim();</pre>
Use the # character for an end of line comment.	<pre>\$options=\$option.trim(); # End of line comment</pre>
Use the <# and #> characters for a block comment.	<pre><# This is a block comment #> \$options=\$option.trim();</pre>

Perl comments

Guidelines	Example
Use the # character for single line comment.	<pre># convert from MBytes to Bytes my \$MaxDirectorySizeBytes = \$MaxDirectorySize * 1024 * 1024;</pre>
Use the # character for end of line comment.	<pre>my \$MaxDirectorySizeBytes = \$MaxDirect orySize * 1024 * 1024; # convert to Bytes</pre>
Use the # character in every line with an empty # at the beginning and end to create a comment border for multi-line comments.	<pre># # This is a multi-line comment. Perl 5, unlike # Powershell, does not have direct support for # multi-line comments. Please use a '\#' in every line # with an empty '#' at the beginning and end to create # a comment border #</pre>
Do not include commented and dead code in WFA commands. However, for testing purposes, you can use the Plain Old Documentation (POD) mechanism to comment out the code.	<pre>=begin comment # Set deduplication if (defined \$Deduplication && \$Deduplication eq "enabled") { \$wfaUtil- >sendLog("Enabling Deduplication"); } =end comment =cut</pre>

Guidelines for logging

You must be aware of the guidelines for logging when writing a PowerShell or Perl script

for OnCommand Workflow Automation (WFA).

PowerShell logging

Guidelines	Example
Use the Get-WFALogger cmdlet for logging.	<pre>Get-WFALogger -Info -message "Creating volume"</pre>
Log every action that requires interaction with internal packages such as Data ONTAP, VMware, and PowerCLI. All the log messages are available in Execution Logs in the execution status history of workflows.	None
Log every relevant argument that is passed to internal packages.	None
Use appropriate log levels when using the Get-WFALogger cmdlet, depending on the usage context. -Info, -Error, -Warn, and -Debug are the various available log levels. If a log level is not specified, then the default log level is Debug.	None

Perl logging

Guidelines	Example
Use the WFAUtil sendLog for logging.	<pre>my wfa_util = WFAUtil->new(); eval { \$wfa_util->sendLog('INFO', "Connecting to the cluster: \$DestinationCluster"); }</pre>
Log every action that requires interaction with anything external to the command such as Data ONTAP, VMware, and WFA. All the log messages that you create using the WFAUtil sendLog routine are stored in the WFA database. These log messages are available for the executed workflow and command.	None
Log every relevant argument passed to the routine that was called.	None

Guidelines	Example
Use appropriate log levels. -Info, -Error, -Warn, and -Debug are the various available log levels.	None
When logging at the -Info level, be precise and concise. Do not specify implementation details such as class name and function name in log messages. Describe the exact step or the exact error in plain English.	<p>The following code snippet shows an example of a good message and a bad message:</p> <pre>\$wfa_util->sendLog('WARN', 'Removing volume: ' . \$VolumeName); # Good Message</pre> <pre>\$wfa_util->sendLog('WARN', 'Invoking volume- destroy ZAPI: ' . \$VolumeName); # Bad message</pre>

Guidelines for error handling

You must be aware of the guidelines for error handling when writing a PowerShell or Perl script for OnCommand Workflow Automation (WFA).

PowerShell error handling

Guidelines	Example
<p>Common parameters added to cmdlets by PowerShell runtime include error handling parameters such as <code>ErrorAction</code> and <code>WarningAction</code>:</p> <ul style="list-style-type: none"> The <code>ErrorAction</code> parameter determines how a cmdlet should react to a non-terminating error from the command. The <code>WarningAction</code> parameter determines how a cmdlet should react to a warning from the command. Stop, SilentlyContinue, Inquire, and Continue are the valid values for the <code>ErrorAction</code> and <code>WarningAction</code> parameters. <p>For more information, you can use the <code>Get-Help about_CommonParameters</code> command in PowerShell CLI.</p>	<p>ErrorAction: the following example shows how to handle a non-terminating error as a terminating error:</p> <pre>New-NcIgroup-Name \$IgroupName- Protocol \$Protocol-Type\$OSType- ErrorActionstop</pre> <p>WarningAction</p> <pre>New-VM-Name \$VMName-VM \$SourceVM- DataStore\$DataStoreName- VMHost\$VMHost- WarningActionSilentlyContinue</pre>

Guidelines	Example
Use the general “try/catch” statement if the type of the incoming exception is unknown.	<pre> try { "In Try/catch block" } catch { "Got exception" } </pre>
Use the specific “try/catch” statement if the type of the incoming exception is known.	<pre> try { "In Try/catch block" } catch[System.Net.WebExceptional], [System.IO. IOException] { "Got exception" } </pre>
Use the “finally” statement to release resources.	<pre> try { "In Try/catch block" } catch { "Got exception" } finally { "Release resources" } </pre>

Guidelines	Example
<p>Use PowerShell automatic variables to access information about exceptions.</p>	<pre> try { Get-WFALogger -Info -message \$("Creating Ipspace: " + \$Ipspace) New-NaNetIpspace-Name \$Ipspace } catch { Throw "Failed to create Ipspace. Message: " + \$_.Exception.Message; } </pre>

Perl error handling

Guidelines	Example
<p>Perl does not include native language support for try/catch blocks. Use eval blocks for checking and handling errors. Keep eval blocks as small as possible.</p>	<pre> eval { \$wfa_util->sendLog('INFO', "Quiescing the relationship : \$DestinationCluster://\$Destination Vserver /\$DestinationVolume"); \$server->snapmirror_quiesce('destination-vserver' => \$DestinationVserver, 'destination-volume' => \$DestinationVolume); \$wfa_util->sendLog('INFO', 'Quiesce operation started successfully.');</pre> <pre> \$wfa_util->checkEvalFailure("Failed to quiesce the SnapMirror relationship \$DestinationCluster://\$Destination Vserver /\$DestinationVolume", \$@); </pre>

General PowerShell and Perl conventions for WFA

You must understand certain PowerShell and Perl conventions that are used in WFA to create scripts that are consistent with existing scripts.

- Use variables that help to clarify what you want the script to do.
- Write readable code that can be understood without comments.
- Keep the scripts and commands as simple as possible.
- For PowerShell scripts:
 - Use cmdlets whenever possible.
 - Invoke .NET code when there is no cmdlet available.
- For Perl scripts:
 - Always end “die” statements with newline characters.

In the absence of a newline character, the script line number is printed, which is not useful for debugging Perl commands executed by WFA.

- In the “GetOpt” module, make the string arguments to a command mandatory.

Perl modules bundled with Windows

Some Perl modules are bundled with the Windows Active state Perl distribution for OnCommand Workflow Automation (WFA). You can use these Perl modules in your Perl code for writing commands, only if they are bundled with Windows.

The following table lists the Perl database modules that are bundled with Windows for WFA.

Database module	Description
DBD::mysql	Perl5 database interface driver that enables you to connect to the MySQL database.
Try::Tiny	Minimizes common mistakes with evaluation blocks.
XML::LibXML	Interface to libxml2 that provides XML and HTML parsers with DOM, SAX, and XMLReader interfaces.
DBD::Cassandra	Perl5 database interface driver for Cassandra that uses the CQL3 query language.

Considerations for adding custom PowerShell and Perl modules

You must be aware of certain considerations before adding custom PowerShell and Perl modules to OnCommand Workflow Automation (WFA). Custom PowerShell and Perl modules enable you to use custom commands for creating workflows.

- During the execution of WFA commands, all custom PowerShell modules are added to the WFA install directory */Posh/modules* are automatically imported.
- All custom Perl modules added to the *WFA/perl* directory are included in the *@INC* library.
- Custom PowerShell and Perl modules are not backed up as part of the WFA backup operation.
- Custom PowerShell and Perl modules are not restored as part of the WFA restore operation.

You must manually back up custom PowerShell and Perl modules in order to copy them to a new WFA installation.

The folder name in modules' directory must be same as that of the module name.

WFA cmdlets and functions

OnCommand Workflow Automation (WFA) provides several PowerShell cmdlets as well as PowerShell and Perl functions that you can use in your WFA commands.

You can view all the PowerShell cmdlets and functions provided by the WFA server using the following

PowerShell commands:

- `Get-Command -Module WFAWrapper`
- `Get-Command -Module WFA`

You can view all the Perl functions provided by the WFA server in the `WFAUtil.pm` module. The help sections, WFA PowerShell cmdlets help and WFA Perl methods help, of the WFA Help module Support Links enables access to all the PowerShell cmdlets and functions and the Perl functions.

PowerShell and Perl WFA modules

You must be aware of the PowerShell or Perl modules for OnCommand Workflow Automation (WFA) to write scripts for your workflows.


PowerShell modules

Guidelines	Example
Use the Data ONTAP PS Toolkit to invoke APIs whenever the toolkit is available.	The <code>Add VLAN</code> command uses the toolkit as follows: <code>Add-NaNetVlan-Interface \$Interface-Vlans\$VlanID</code>
If there are no cmdlets available in the Data ONTAP PS Toolkit, use the <code>Invoke-SSH</code> command to invoke the CLI on Data ONTAP.	<code>Invoke-NaSsh-Name \$ArrayName-Command "ifconfig -a"-Credential \$Credentials</code>

Perl modules

The `NaServer` module is used in WFA commands. The `NaServer` module allows the invocation of Data ONTAP APIs, which are used in active management of Data ONTAP systems.

Guidelines	Example
<p>Use the NaServer module to invoke APIs whenever the NetApp Manageability SDK is available.</p>	<p>The following example shows how the NaServer module is used for a resume SnapMirror operation:</p> <pre> eval { \$wfa_util->sendLog('INFO', "Connecting to the cluster: \$DestinationCluster"); my \$server = \$wfa_util- >connect(\$DestinationClusterIp, \$DestinationVserver); my \$sm_info = \$server- >snapmirror_get('destination-vserver' => \$DestinationVserver, 'destination-volume' => \$DestinationVolume); my \$sm_state = \$sm_info- >{'attributes'}->{'snapmirror- info'}->{'mirror-state'}; my \$sm_status = \$sm_info- >{'attributes'}->{'snapmirror- info'}->{'relationship-status'}; \$wfa_util->sendLog('INFO', "SnapMirror relationship is \$sm_state (\$sm_status)"); if (\$sm_status ne 'quiesced') { \$wfa_util->sendLog('INFO', 'The status needs to be quiesced to resume transfer.');</pre>
	<pre> } else { my \$result = \$server- >snapmirror_resume('destination-vserver' => \$DestinationVserver, 'destination-volume' => \$DestinationVolume); \$wfa_util->sendLog('INFO', "Result of resume: \$result"); } }</pre>

Guidelines	Example
<p>If a Data ONTAP API is not available, invoke the Data ONTAP CLI using the executeSystemCli utility method.</p> <div>  <p>executeSystemCli is not supported and is currently available only for Data ONTAP operating in 7-Mode.</p> </div>	None

Considerations while converting PowerShell commands to Perl

You must be aware of certain important considerations when you convert PowerShell commands to Perl because PowerShell and Perl have different capabilities.

Command input types

OnCommand Workflow Automation (WFA) allows workflow designers to use arrays and hash as input for the command when defining a command. These input types cannot be used when the command is defined using Perl. If you want a Perl command to accept array and hash inputs, you can define the input as a string in the designer. The command definition can then parse the input, which is passed to create an array or hash as required. The description for the input describes the format in which the input is expected.

```
my @input_as_array = split(',', $InputString); #Parse the input string of
format val1,val2 into an array

my %input_as_hash = split /[;=]/, $InputString; #Parse the input string of
format key1=val1;key2=val2 into a hash.
```

PowerShell statement

The following examples show how an array input can be passed into PowerShell and Perl. The examples describe the input CronMonth, which specifies the month when the cron job is scheduled to run. The valid values are integers -1 to 11. A value of -1 indicates that the schedule executes every month. Any other value denotes a specific month, with 0 being January and 11 being December.

```
[parameter(Mandatory=$false, HelpMessage="Months in which the schedule
executes. This is a comma separated list of values from 0 through 11.
Value -1 means all months.")]
[ValidateRange(-1, 11)]
[array]$CronMonths,
```

Perl statement

```

GetOptions(
    "Cluster=s"           => \$Cluster,
    "ScheduleName=s"      => \$ScheduleName,
    "Type=s"              => \$Type,
    "CronMonths=s"        => \$CronMonths,
) or die 'Illegal command parameters\n';

sub get_cron_months {
    return get_cron_input_hash('CronMonths', \$CronMonths, 'cron-month',
-1,
        11);
}

sub get_cron_input_hash {
    my $input_name = shift;
    my $input_value = shift;
    my $zapi_element = shift;
    my $low = shift;
    my $high = shift;
    my $exclude = shift;

    if (!defined $input_value) {
        return undef;
    }

    my @values = split(',', $input_value);

    foreach my $val (@values) {
        if ($val !~ /^[+-]?[0-9]+$/) {
            die
                "Invalid value '$input_value' for $input_name: $val must
be an integer.\n";
        }
        if ($val < $low || $val > $high) {
            die
                "Invalid value '$input_value' for $input_name: $val must
be from $low to $high.\n";
        }
        if (defined $exclude && $val == $exclude) {
            die
                "Invalid value '$input_value' for $input_name: $val is not
valid.\n";
        }
    }
    # do something
}

```

Command definition

A one-line expression in PowerShell using a pipe operator might have to be expanded into multiple blocks of statements in Perl in order to achieve the same functionality. An example from one of the wait commands is shown in the following table.

PowerShell statement	Perl statement
<pre># Get the latest job which moves the specified volume to the specified aggregate. \$job = Get-NcJob -Query \$query where {\$_ .JobDescription -eq "Split" + \$VolumeCloneName} Select-Object -First 1</pre>	<pre>my \$result = \$server- >job_get_iter('query' => {'job-type' => 'VOL_CLONE_SPLIT'}, 'desired-attributes' => { 'job-type' => '', 'job-description' => '', 'job-progress' => '', 'job-state' => '' }); my @jobarray; for my \$job (@{ \$result- >{'attributes-list'}}) { my \$description = \$job->{'job- description'}; if(\$description =~ /\$VolumeCloneName/) { push(@jobarray, \$job) } }</pre>

Guidelines for WFA building blocks

You must be aware of the guidelines for using Workflow Automation building blocks.

Guidelines for SQL in WFA

You must be aware of the guidelines for using SQL in OnCommand Workflow Automation (WFA) to write SQL queries for WFA.

SQL is used in the following places in WFA:

- SQL queries to populate user inputs for selection
- SQL queries for creating filters to filter objects of a specific dictionary entry type
- Static data in tables in the playground database

- A custom data source type of SQL type where the data has to be extracted from an external data source such as a custom configuration management database (CMDB).
- SQL queries for reservation and verification scripts

Guidelines	Example
SQL reserved keywords must be in uppercase characters.	<pre>SELECT vserver.name FROM cm_storage.vserver vserver</pre>
Table and column names must be in lowercase characters.	<p>Table: aggregate</p> <p>Column: used_space_mb</p>
Separate words with an underscore (_) character. Spaces are not allowed.	array_performance
Table name is defined in singular. A table is a collection of one or more entries.	“function”, not “functions”
Use table aliases with meaningful names in SELECT queries.	<pre>SELECT vserver.name FROM cm_storage.cluster cluster, cm_storage.vserver vserver WHERE vserver.cluster_id = cluster.id AND cluster.name = '\${ClusterName}' AND vserver.type = 'cluster' ORDER BY vserver.name ASC</pre>

Guidelines	Example
<p>If you have to refer to a filter input parameter or user input parameter in a filter query or user query, use the syntax as '\${inputVariableName}'. You can also use the syntax to refer to a command definition parameter in reservation scripts and verification scripts.</p>	<pre> SELECT volume.name AS Name, aggregate.name as Aggregate, volume.size_mb AS 'Total Size (MB) ', voulme.used_size_mb AS 'Used Size (MB) ', volume.space_guarantee AS 'Space Guarantee' FROM cm_storage.cluster, cm_storage.aggregate, cm_storage.vserver, cm_storage.volume WHERE cluster.id = vserver.cluster_id AND aggregate.id = volume.aggregate_id AND vserver.id = voulme.vserver_id AND vserver.name = '\${VserverName}' AND cluster.name = '\${ClusterName}' ORDER BY volume.name ASC </pre>
<p>Use comments for complex queries. Some of the supported comment styles in queries are as follows:</p> <ul style="list-style-type: none"> • “--” until the end of the line <p>A space is mandatory after the second hyphen in this comment style.</p> <ul style="list-style-type: none"> • From a “#” character until the end of the line • From a “/*” to the following “*/”sequence 	<pre> /* multi-line comment */ --line comment SELECT ip as ip, # comment till end of this line NAME as name FROM --end of line comment storage.array </pre>

Guidelines for WFA functions

You can create functions to encapsulate commonly used and more complex logic in a named function, and then reuse the function as command parameter values or filter parameters values in OnCommand Workflow Automation (WFA).

Guidelines	Example
Use Camel case for a function name.	calculateVolumeSize
Variable names should be in plain English and related to the functionality of the function.	splitByDelimiter
Do not use abbreviations.	calculateVolumeSize, <i>not</i> calcVolSize
Functions are defined using MVFLEX Expression Language (MVEL).	None
The function definition should be specified according to the official Java Programming Language guidelines.	None

Guidelines for WFA dictionary entries

You must be aware of the guidelines for creating dictionary entries in OnCommand Workflow Automation (WFA).

Guidelines	Example
Dictionary entry names must contain only alphanumeric characters and underscores.	Cluster_License Switch_23
Dictionary entry names must start with an uppercase character. Begin every word in the name with an uppercase character and separate each word with an underscore (_).	Volume Array_License
Dictionary entry attribute names should not include the name of the dictionary entry.	None
Attributes and references in a dictionary entry must be in lowercase characters.	aggregate, size_mb
Separate words with an underscore. Spaces are not allowed.	resource_pool

Guidelines	Example
<p>Dictionary entries cannot include references that are from a different scheme.</p> <p>When a dictionary entry requires cross-reference to an object in a different scheme, ensure that all the natural keys of the object being referred to are present in the dictionary entry.</p>	<p>Array_Performance dictionary entry requires all the natural keys of the Array dictionary entry as direct attributes in it.</p>
Use appropriate data types for attributes.	None
Use Long data type for size or space-related attributes.	size_mb and available_size_mb in storage.Volume dictionary entry
Use Enum when an attribute has a fixed set of values.	raid_type in storage.Volume dictionary entry
<p>Set “To be Cached” as true for an attribute or reference when a data source provides value for that attribute or reference.</p> <p>For OnCommand Unified Manager data source, add cacheable attributes if the data source can provide the value to it.</p>	None
Set “Can be Null” as true if the data source providing the value for this attribute or reference can return NULL.	None
<p>Provide a meaningful description to each attribute and reference.</p> <p>The description is displayed in command details when designing a workflow.</p>	None
<p>Do not use “id” as the name of an attribute in dictionary entries.</p> <p>It is reserved for internal WFA usage.</p>	None

Guidelines for commands

You must be aware of the guidelines for creating commands in OnCommand Workflow Automation (WFA).

Guidelines	Example
Use an easily identifiable name for commands.	Create Qtree

Guidelines	Example
Use spaces to delimit words and each word must start with an uppercase character.	Create Volume
Provide a description to explain the functionality of the command, including the expected outcome of the optional parameters.	None
By default, the timeout for standard commands is 600 seconds. The default timeout is set while creating the command. Change the default value only if the command might take a longer time to complete.	Create Volume command
In case of long-running operations, create two commands—one to invoke the long-running operation and another to report the progress of the operation periodically. The first command should be a Standard Execution command type and the second should be Wait for Condition command type.	Create VSM and Wait for VSM commands
Prefix the Wait for condition command names with “Wait” for easy identification.	Wait for CM Volume Move
Use an appropriate waiting interval for the “Wait for condition” commands. The specified value governs the interval at which the polling command gets executed to check if the long-running operation is complete.	60s sampling interval for the Wait for VSM command
For the Wait for condition commands, use an appropriate timeout based on the expected time for the long-running operation to complete. The expected time might be considerably longer if the operation involves data transfer over a network.	A VSM baseline transfer can take many days to complete. Therefore, the specified timeout is 6 days.

String representation

The string representation for a command displays the details of a command in a workflow design during planning and execution. Only the command parameters can be used in the string representation for a command.

Guidelines	Example
Avoid using attributes that do not have any value. An attribute without a value is displayed as NA.	VolName 10.68.66.212 [NA] aggr1/testVol17
Separate different entries in the string representation using the following delimiters: [], / :	ArrayName [ArrayIp]

Guidelines	Example
Provide meaningful labels to every value in string representation.	Volume name=VoumeName

Command definition language

Commands can be written using the following supported scripting languages:

- PowerShell
- Perl

Command parameter definition

The command parameters are described by Name, Description, Type, a default value for the parameter, and whether the parameter is mandatory. The parameter type can be String, Boolean, Integer, Long, Double, Enum, DateTime, Capacity, Array, Hashtable, Password, or an XmlDocument. While the values for most of the types are intuitive, the values for Array and Hashtable should be in a particular format as described in the following table:

Guidelines	Example
Ensure that the value for an Array input type is a list of values, separated by comma.	<pre>[parameter(Mandatory=\$false, HelpMessage="Months in which the schedule executes.")] [array]\$CronMonths</pre> <p>Input is passed as following: 0,3,6,9</p>
Ensure that the value for a Hashtable input type is a list of key=value pairs, separated by semicolon.	<pre>[parameter(Mandatory=\$false, HelpMessage="Volume names and size (in MB) ")] [hashtable]\$VolumeNamesAndSize</pre> <p>Input is passed as following: Volume1=100;Volume2=250;Volume3=50</p>

Guidelines for workflows

You must be aware of the guidelines for creating or modifying a predefined workflow for OnCommand Workflow Automation (WFA).

General guidelines

Guidelines	Example
Name the workflow such that it reflects the operation that is executed by the storage operator.	Create a CIFS Share
For workflow names, capitalize the initial letter of the first word and every word that is an object. Capitalize letters for abbreviations and acronyms.	Volume Qtree Create a Clustered Data ONTAP Qtree CIFS Share
For workflow descriptions, include all of the important steps of the workflow, including any prerequisites, result of the workflow, or conditional aspects of execution.	See the description of the sample workflow Create VMware NFS Datastore on Clustered Data ONTAP Storage, which includes the prerequisites.
Set “Ready For Production” to true only when the workflow is ready for production and can be displayed in the portal page.	None
By default, set “Consider reserved elements” to true . When previewing a workflow for execution, the WFA planner considers all of the objects that are reserved along with the existing objects in the cache database. Effects of other scheduled workflows or workflows executing in parallel are considered when planning a specific workflow if this option is set to true .	<ul style="list-style-type: none"> Scenario 1 Workflow 1 creates a volume, and is scheduled to execute one week later. Workflow 2 creates qtrees or LUNs in volumes that are searched for, and if workflow 2 is executed within a day or so, you should turn off “Consider reserved elements” for workflow 2 to prevent it from considering the volume that is to be created in a week. Scenario 2 Workflow 1 uses the <code>Create Volume</code> command. If there is a scheduled workflow 2 that consumes 100 GB from an aggregate, then workflow 1 must consider the requirements for workflow 2 during planning.

Guidelines	Example
By default, “Enable element existence validation” is set to true .	<ul style="list-style-type: none"> Scenario 1 <p>If you create a workflow that first removes a volume by name using the command <code>Remove Volume</code> only if the volume exists, and the re-creates it using another command such as <code>Create Volume</code> or <code>Clone Volume</code>, then the workflow should not use this flag. The effect of removing the volume will not be available to the <code>Create volume</code> command, thereby causing the workflow to fail.</p> <ul style="list-style-type: none"> Scenario 2 <p>The <code>Create Volume</code> command is used in a workflow with a specific name as “vol198”.</p> <p>If this option is set to true, WFA planner checks during planning to see if a volume by that name exists in the given array. If the volume exists, the workflow fails during planning.</p>
When the same command is selected more than once in a workflow, provide appropriate display names for the command instances.	The “Create, map, and protect LUNs with SnapVault” sample workflow uses the <code>Create Volume</code> command twice. However, it uses the display names as <code>Create Primary Volume</code> and <code>Create Secondary Volume</code> appropriately for the primary volume and the mirrored destination volume.

User inputs

Guidelines	Example
Names: <ul style="list-style-type: none"> Start the name with the “\$” character. Use an uppercase letter at the beginning of each word. Use uppercase letters for all terms and abbreviations. Do not use underscores. 	<code>\$Array</code> <code>\$VolumeName</code>

Guidelines	Example
<p>Display names:</p> <ul style="list-style-type: none"> • Use an uppercase letter at the beginning of each word. • Separate words with spaces. • If inputs have specific units, specify the unit in brackets in the display name directly. 	<p>Volume Name</p> <p>Volume Size (MB)</p>
<p>Descriptions:</p> <ul style="list-style-type: none"> • Provide a meaningful description for each user input. • Provide examples when required. <p>You should do this especially when the user input is expected to be in a specific format.</p> <p>The user input descriptions are displayed as tooltips for the user inputs during workflow execution.</p>	<p>Initiators to be added to an “iGroup”. For example, IQN or WWPN of the initiator.</p>
<p>Type: Select Enum as the type if you want to restrict the input to a specific set of values.</p>	<p>Protocol: "` iscsi`, “fc”, “mixed”</p>
<p>Type: Select Query as the type when the user can select from values available in the WFA cache.</p>	<p>\$Array: QUERY type with query as follows:</p> <pre>SELECT ip, name FROM storage.array</pre>
<p>Type: Mark the user input as locked when the user input should be restricted to the values that are obtained from a query or should be restricted to only the supported Enum types.</p>	<p>\$Array: Locked Query type: Only arrays in the cache can be selected.\$Protocol: Locked Enum type with valid values as iscsi, fcp, mixed. No other value than the valid value is supported.</p>
<p>Type: Query TypeAdd additional columns as return values in the query when it helps the storage operator to make the right choice of user input.</p>	<p>\$Aggregate: Provide name, total size, available size so that the operator knows the attributes before selecting the aggregate.</p>

Guidelines	Example
<p>Type: Query TypeSQL query for user inputs can refer to any other user inputs preceding it. This can be used to limit the results from a query based on other user inputs such as vFiler units of an array, volumes of an aggregate, LUNs in a storage virtual machine (SVM).</p>	<p>In the sample workflow Create a Clustered Data ONTAP Volume, the query for VserverName is as follows:</p> <pre data-bbox="820 296 1481 877"> SELECT vserver.name FROM cm_storage.cluster cluster, cm_storage.vserver vserver WHERE vserver.cluster_id = cluster.id AND cluster.name = '\${ClusterName}' AND vserver.type = 'cluster' ORDER BY vserver.name ASC </pre> <p>The query refers to \${ClusterName}, where \$ClusterName is the name of the user input preceding the \$VserverName user input.</p>
<p>Type:</p> <p>Use Boolean type with values as “true, false” for user inputs that are Boolean in nature. This helps in writing internal expressions in the workflow design using the user input directly. For example, \$UserInputName rather than \$UserInputName == “Yes”.</p>	<p>\$CreateCIFSShare: Boolean type with valid values as “true” or “false”</p>
<p>Type:</p> <p>For string and number type, use regular expressions in the values column when you want to validate the value with specific formats.</p> <p>Use regular expressions for IP address and network mask inputs.</p>	<p>Location-specific user input can be expressed as “[A-Z][A-Z]\-0[1-9]”. This user input accepts values such as “US-01”, “NB-02”, but not “nb-00”.</p>
<p>Type:</p> <p>For number type, a range-based validation can be specified in the values column.</p>	<p>For Number of LUNs to be created, the entry in the Values column is 1-20.</p>

Guidelines	Example
<p>Group:</p> <p>Group related user inputs into appropriate buckets and name the group.</p>	<p>“Storage Details” for all storage-related user inputs. “Datastore Details” for all VMware-related user inputs.</p>
<p>Mandatory:</p> <p>If the value of any user input is necessary for the workflow to execute, mark the user input as mandatory. This ensures that the user input screen mandatorily accepts that input from the user.</p>	<p>“\$VolumeName” in the “Create NFS Volume” workflow.</p>
<p>Default value:</p> <p>If a user input has a default value that can work for most of the workflow executions, provide the values. This helps in allowing the user to provide fewer inputs during execution, if the default serves the purpose.</p>	<p>None</p>

Constants, variables, and returns parameters

Guidelines	Example
<p>Constants: Define constants when using a common value for defining parameters to multiple commands.</p>	<p><i>AGGREGATE_OVERCOMMITMENT_THRESHOLD</i> in the <i>Create, map, and protect LUNs with SnapVault</i> sample workflow.</p>
<p>Constants: Names</p> <ul style="list-style-type: none"> • Use an uppercase letter at the beginning of each word. • Use uppercase letters for all terms and abbreviations. • Do not use underscores. • Use uppercase letters for all letters of constant names. 	<p><i>AGGREGATE_USED_SPACE_THRESHOLD</i></p> <p><i>ActualVolumeSizeInMB</i></p>
<p>Variables: Provide a name to an object defined in one of the command parameter boxes. Variables are automatically generated names and can be changed.</p>	<p>None</p>
<p>Variables: Names Use lowercase characters for variable names.</p>	<p>volume1</p> <p>cifs_share</p>

Guidelines	Example
Return parameters: Use return parameters when the workflow planning and execution should return some calculated or selected values during planning. The values are made available in the preview mode when the workflow is executed from a web service as well.	Aggregate: If the aggregate is selected using the resource selection logic, then the actual selected aggregate can be defined as a return parameter.

Guidelines for creating validation scripts for remote system types

You must be aware of the guidelines for creating validation scripts that are used to test the remote system types that you define in OnCommand Workflow Automation (WFA).

- The Perl script that you create must be similar to the sample script provided in the Validation Script window.
- The output of your validation script must be similar to that of the sample script.

Sample validation script

```
# Check connectivity.
# Return 1 on success.
# Return 0 on failure and set $message
sub checkCredentials {
my ($host, $user, $passwd, $protocol, $port, $timeout) = @_ ;
#
# Please add the code to check connectivity to $host using $protocol here.
#
return 1;
}
```

Guidelines for creating data source types

You must be aware of the guidelines for creating data source types that are used to define custom data sources for OnCommand Workflow Automation (WFA).

You can define a data source type by using one of the following methods:

- SQL: You can use the WFA SQL guidelines to define select queries from data sources based on an external database.
- SCRIPT: You can write a PowerShell script that provides the data for a specific scheme of dictionary entries.

The guidelines for creating data source types are as follows:

- You should use PowerShell language must be used to create script.
- The PowerShell script should provide the output for each dictionary entry in its current working directory.
- The data files should be named `dictionary_entry.csv`, where the name of the dictionary entry should be in lower-case characters.

The predefined data source type that collects information from Performance Advisor uses a SCRIPT-based data source type. The output files are named `array_performance.csv` and `aggregate_performance.csv`.

- The `.csv` file should include the content in the exact order as that of the dictionary entry attributes.

A dictionary entry includes attributes in the following order: `array_ip`, `date`, `day`, `hour`, `cpu_busy`, `total_ops_per_sec`, `disk_throughput_per_sec`.

The PowerShell script adds data to the `.csv` file in the same order.

```
$values = get-Array-CounterValueString ([REF]$data)
Add-Content $arrayFile ([byte[]][char[]] "`n"
t$arrayIP't$date't$day't$hour't$values'n")
```

- You should use Encoding to ensure that the data output from the script is loaded into the WFA cache accurately.
- You should use `\N` while entering a Null value in the `.csv` file.

Reserved words

OnCommand Workflow Automation (WFA) includes some reserved words. You must not use the reserved words in workflows for any attribute or parameters such as variable names, user input, constants, and return parameters.

The following is a list of the reserved words in WFA:

<ul style="list-style-type: none"> • and • Array • assert • Boolean • boolean • Byte • byte • char • Character • CharSequence • Class • ClassLoader • Compiler • contains • convertible_to • def • do • Double • double • else • empty • false 	<ul style="list-style-type: none"> • Float • float • for • foreach • function • if • import • import_static • in • instanceof • int • Integer • is • isdef • Long • long • Math • new • null • Number • Object • or 	<ul style="list-style-type: none"> • proto • return • Runtime • SecurityManager • Short • short • soundslike • StrictMath • String • StringBuffer • StringBuilder • strsim • switch • System • Thread • ThreadLocal • true • until • var • Void • while • with
--	---	--

How you use REST APIs

You can use REST APIs provided by Workflow Automation (WFA) to invoke workflows from external portals and the data center orchestration software. WFA supports XML and JSON content types for all REST APIs.

WFA allows external services to access various resource collections, such as workflows, users, filters, and finders, through URI paths. The external services can use HTTP methods, such as GET, PUT, POST, and DELETE, on these URIs to perform CRUD operations on the resources.

You can perform several actions through the WFA REST APIs, including the following:

- Access workflow definitions and metadata.
- Execute workflows and monitor their execution.
- View users and roles, and change passwords.
- Execute and test resource selection filters.
- Execute and test resource finders.

- Manage credentials of storage or other data center objects.
- View data sources and data source types.

REST documentation has more information about REST APIs:

`https://wfa_server_ip:port/rest/docs`

`wfa_server_ip` is the IP address of your WFA server and *port* is the TCP port number you have used for the WFA server during installation.

References to learning material

You should be aware of certain scripting and programming practices to create advanced Workflow Automation (WFA) workflows. You can use reference material to learn about the required options before creating WFA building blocks or workflows.

Windows PowerShell

WFA uses PowerShell scripts for workflow operations. The following table includes references to learning material for PowerShell:

Getting Started with Windows PowerShell	http://msdn.microsoft.com/en-us/library/windows/desktop/aa973757(v=vs.85).aspx
PowerShell Development — Integrated Scripting Environment (ISE)	https://docs.microsoft.com/en-us/powershell/scripting/windows-powershell/ise/introducing-the-windows-powershell-ise?view=powershell-7.2#:~:text=The%20Windows%20PowerShell%20Integrated%20Scripting,Windows%2Dbased%20graphic%20user%20interface
.NET Framework Naming Guidelines	http://msdn.microsoft.com/en-us/library/xzf533w0%28v=vs.71%29.aspx
PowerShell code style	http://get-powershell.com/post/2011/04/13/Extra-Points-for-Style-when-writing-PowerShell-Code.aspx
PowerShell Try/Catch Finally	http://technet.microsoft.com/en-us/library/dd315350.aspx
PowerShell Automatic Variables	http://technet.microsoft.com/en-us/library/dd347675.aspx
PowerShell Error Reporting	https://docs.microsoft.com/en-us/powershell/scripting/developer/cmdlet/error-reporting-concepts?view=powershell-7.2

PowerShell Common Parameters	https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_commonparameters?view=powershell-7.2
------------------------------	---

Data ONTAP PowerShell toolkit

The Data ONTAP PowerShell toolkit is bundled along with WFA. You can use the PowerShell toolkit cmdlets to invoke Data ONTAP commands from a PowerShell script. For more information, see the *Data ONTAP PowerShell Toolkit Help*, which you can access from the following location:

WFA_install_location\WFA\PoSH\Modules\DataONTAP\webhelp\index.html.

WFA_install_location is the WFA installation directory, and C:\Program Files\NetApp is the default installation directory.

The following table includes references to information about the Data ONTAP PowerShell toolkit:

ONTAP PowerShell Toolkit Articles	https://community.netapp.com/t5/Tech-OnTap-Articles/The-Data-ONTAP-PowerShell-Toolkit/ta-p/85933
ONTAP PowerShell Toolkit NetApp Community	https://community.netapp.com/t5/forums/filteredbylabelpage/board-id/microsoft-cloud-and-virtualization-discussions/label-name/powershell%20toolkit

Perl

WFA supports Perl commands for workflow operations. When you install WFA, the required Perl and Perl modules are installed on the WFA server.

ActivePerl User Guide

You can also access the *ActivePerl User Guide* from the following location:
WFA_install_location\WFA\Perl64\html\index.html.

WFA_install_location is the WFA installation directory, and C:\Program Files\NetApp is the default installation directory.

WFA uses Perl scripts for workflow operations. The following table includes references to learning material for Perl:

Modern Perl: 2014	http://modernperlbooks.com/books/modern_perl_2014/index.html
Perl programming documentation	http://perldoc.perl.org/
Perl programming language	http://www.perl.org/

NetApp Manageability SDK

The required Perl modules of the NetApp Manageability SDK are bundled along with WFA. These Perl modules are required for using the Perl commands in WFA. For more information, see the NetApp Manageability SDK documentation, which you can access from the following location:

`WFA_install_location\WFA\perl\NMSDK\html.`

`WFA_install_location` is the WFA installation directory, and `C:\Program Files\NetApp` is the default installation directory.

Structured Query Language (SQL)

The SQL SELECT syntax is used in filters and to populate user inputs.

[MySQL SELECT syntax](#)

MVFLEX Expression Language (MVEL)

You can use MVEL expression syntaxes in WFA workflows—for example, in functions and variables.

For more information, see the *MVEL Language Guide*.

Regular expressions

You can use regular expression (regex) in WFA.

[ActionScript 3.0 Using regular expressions](#)

Related documentation for OnCommand Workflow Automation

There are additional documents and tools to help you learn to perform more advanced configuration of your OnCommand Workflow Automation (WFA) server.

Other references

The Workflow Automation space within the NetApp community provides additional learning resources, including the following:

- **NetApp community**

[NetApp community: Workflow Automation \(WFA\)](#)

Tool references

- **Interoperability Matrix**

Lists supported combinations of hardware components and software versions.

[Interoperability Matrix](#)

Copyright information

Copyright © 2024 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.