Creating workflows

OnCommand Workflow Automation

NetApp
October 09, 2025

Table of Contents

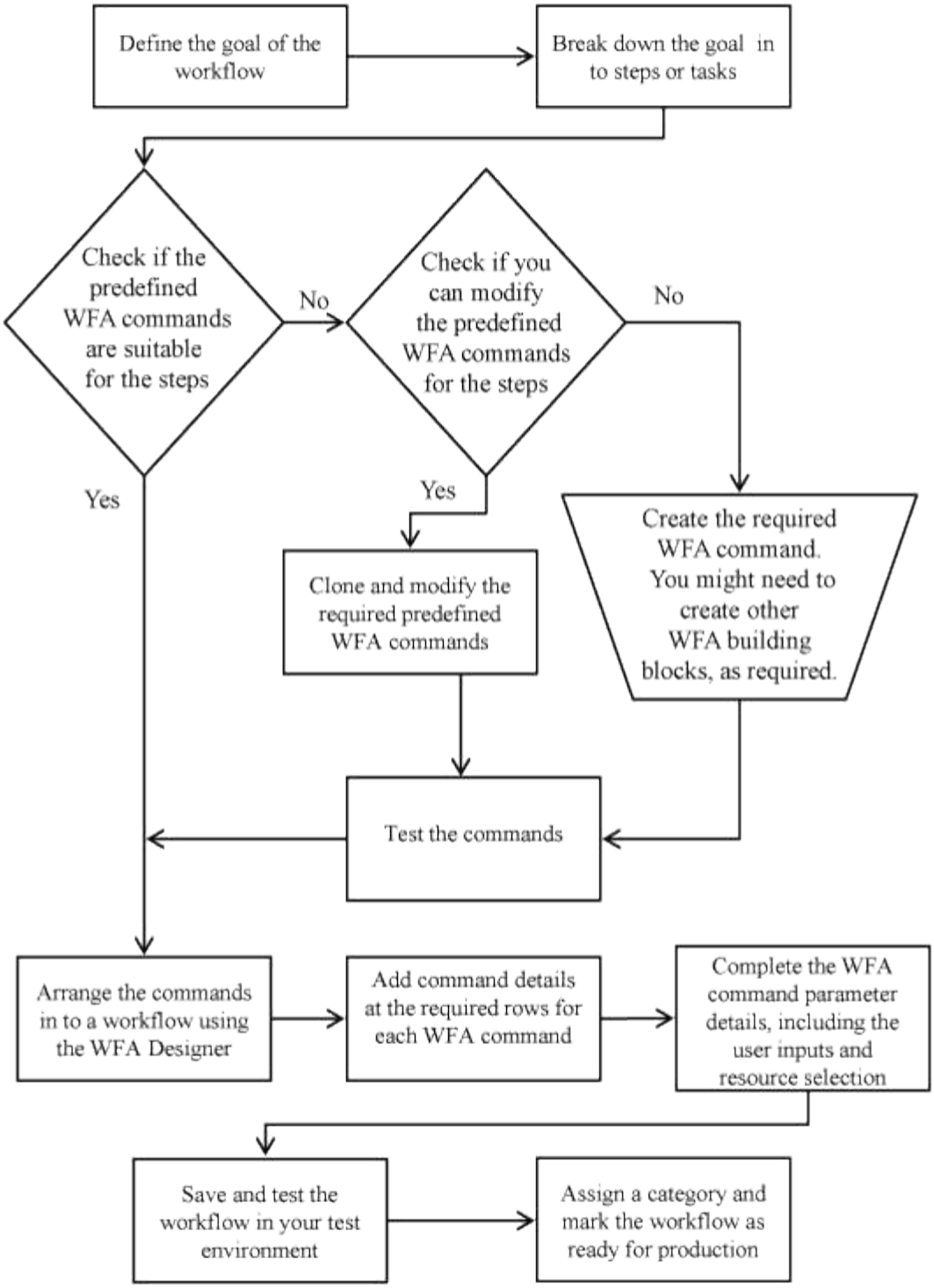
Creating workflows

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 through15.

- **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	You can validate the user inputs type so that only valid values are entered by users: • 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	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.
Marking as mandatory	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.

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

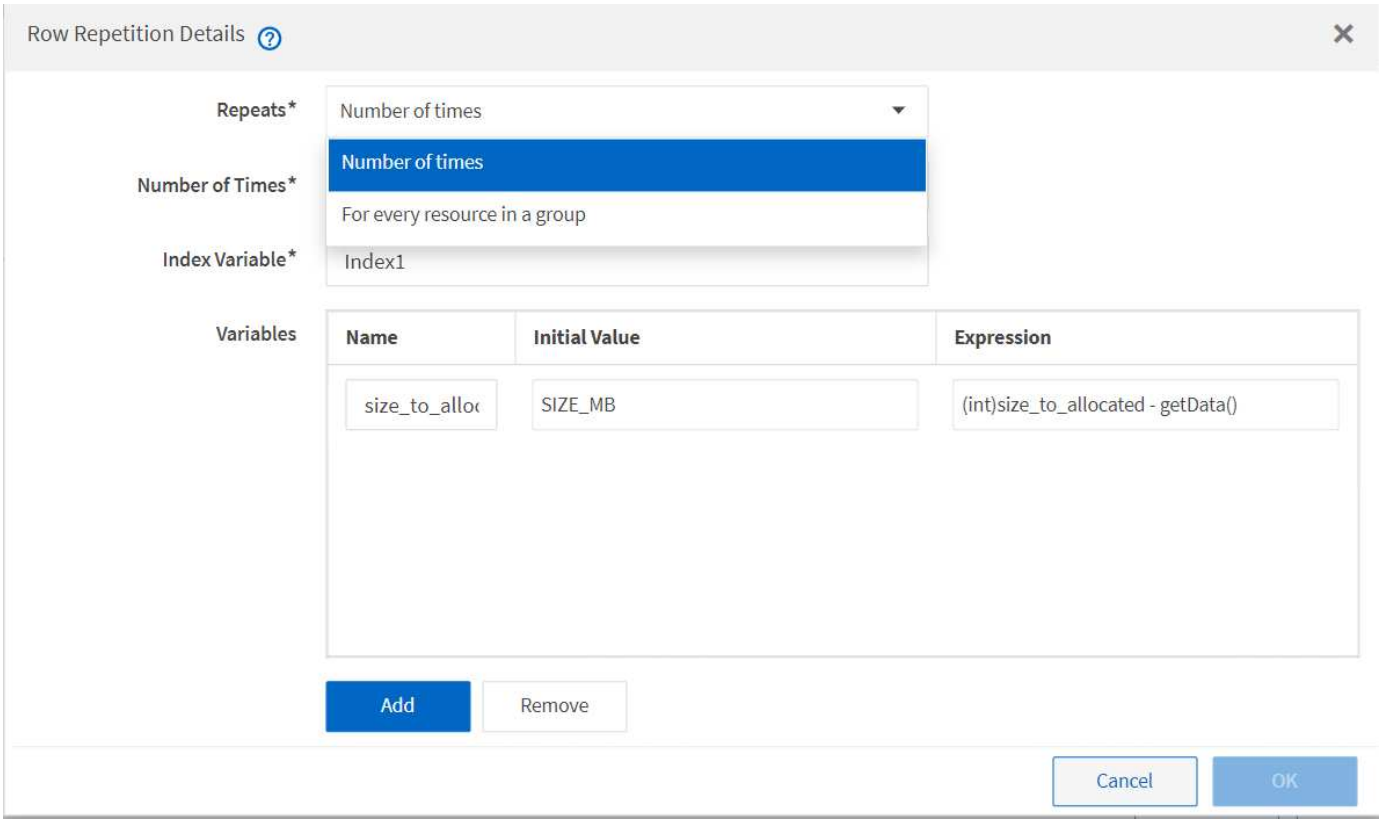
A workflow contains commands and command details arranged in rows. You can specify the commands in a row to be repeated for a fixed number of iterations or dynamic number of iterations based on the results of search criteria.

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 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.

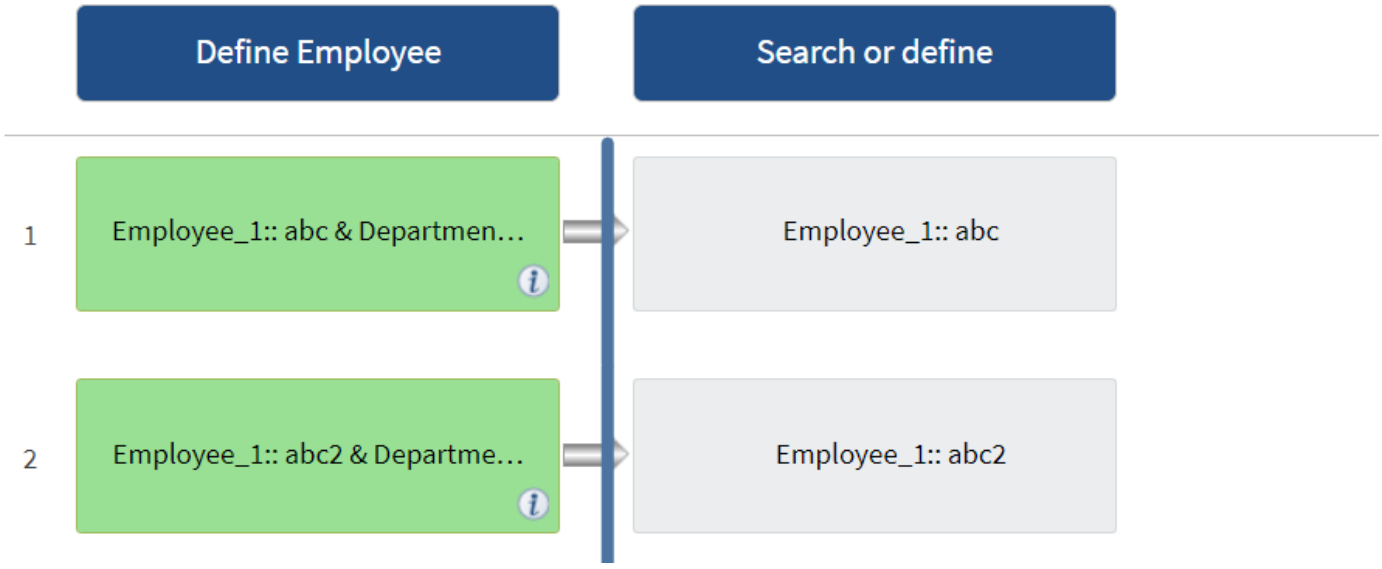
The following illustration shows the repeat row options and an example of a row repetition variable:



Row repetition 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.

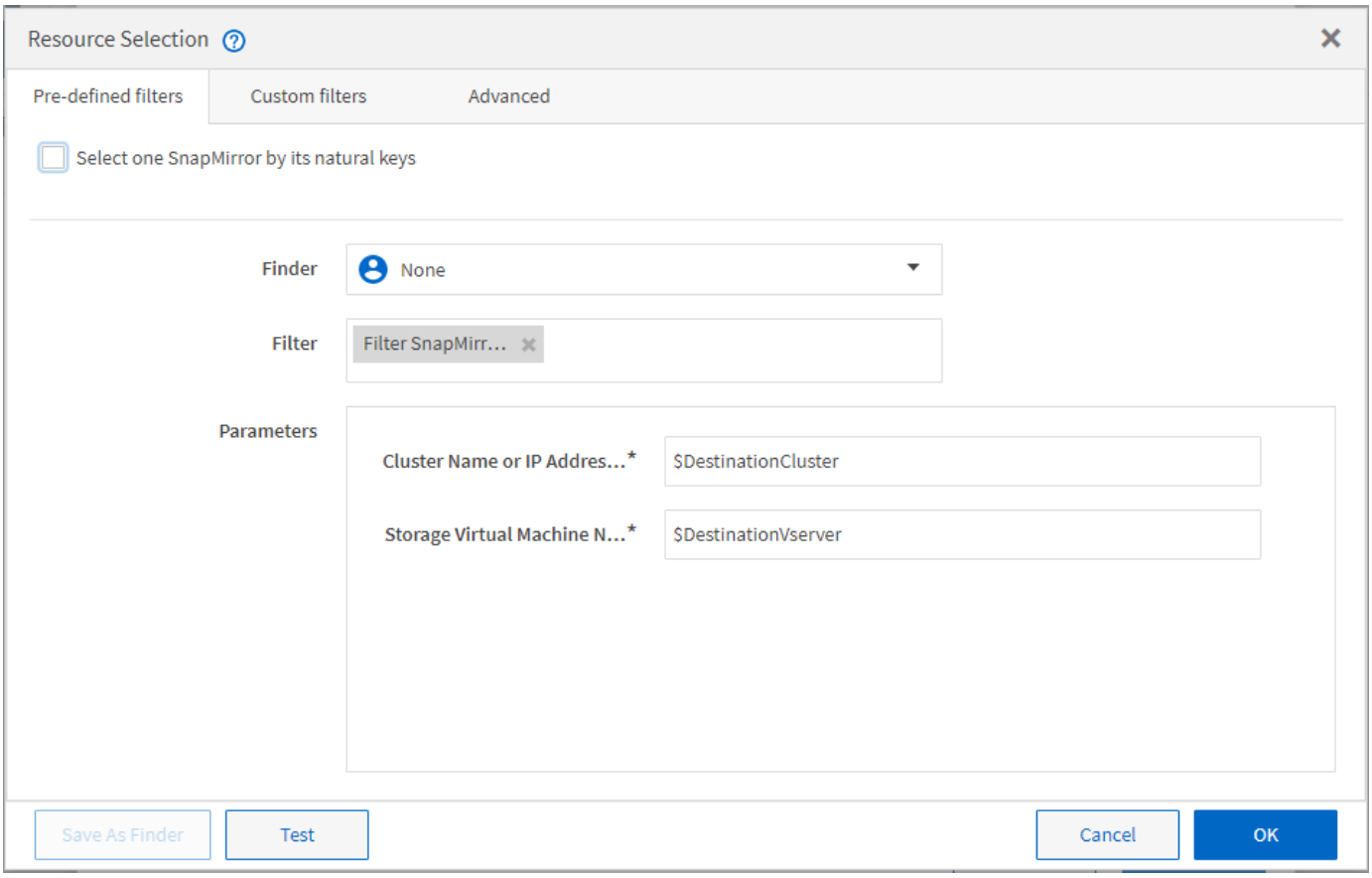
WFA selects dictionary entry resources—such as vFiler units, aggregates, and virtual machines—using search algorithms. The selected resources are then used for executing the workflow. The WFA search algorithms are part of the WFA building blocks, and include finders and filters. To locate and select the required resources, the search algorithms search through the data that is cached from different repositories, such as Active IQ 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

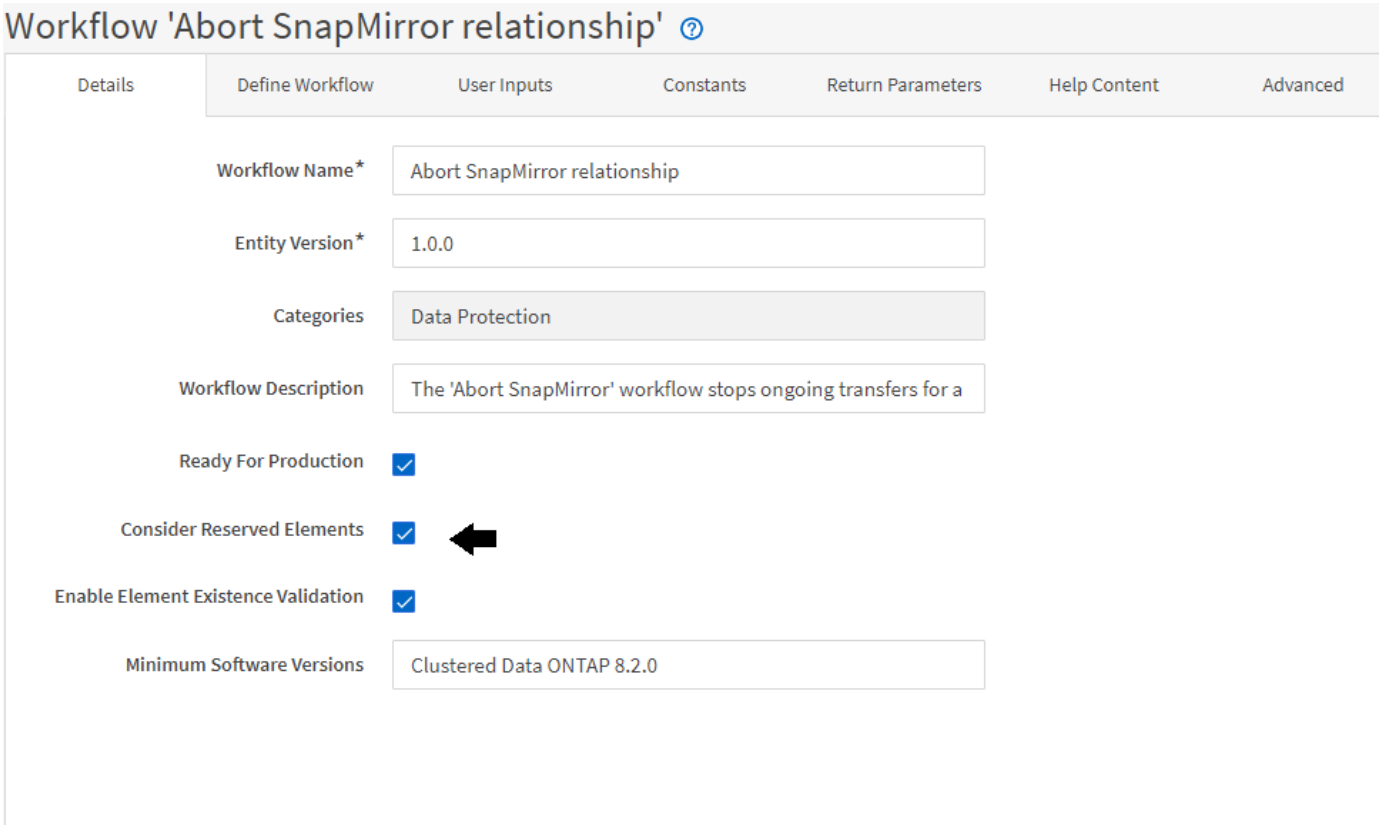
OnCommand Workflow Automation resource reservation capability reserves the required resources to ensure that the resources are available for successful execution of workflows.

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 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:

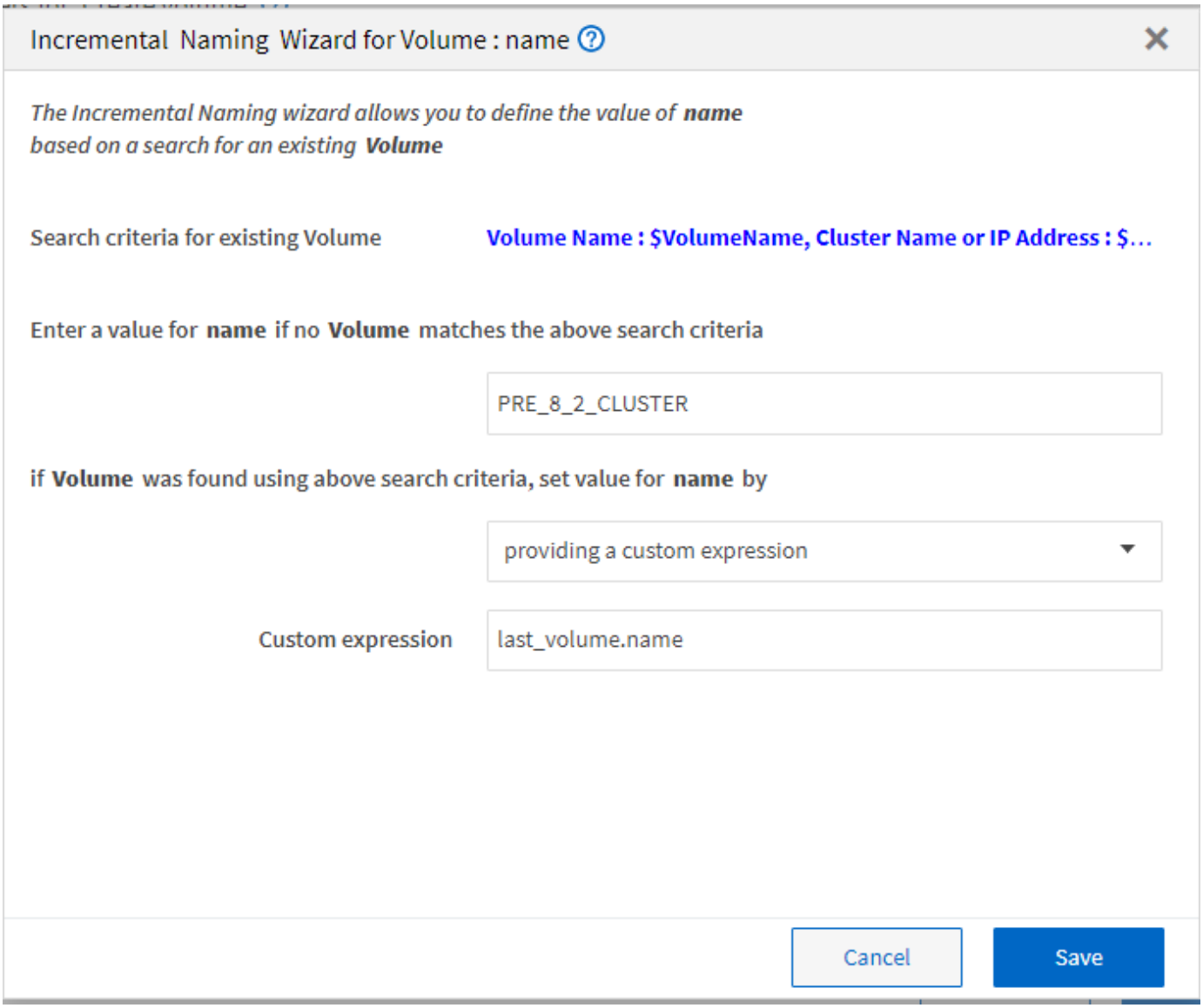
- Providing an increment value and suffix

  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:



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	`volume1.name`, which is an attribute of the “volume name” variable
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

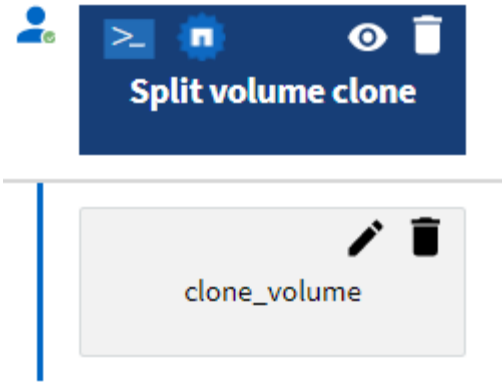
If you want to understand how return parameters are specified, you can open the following predefined workflows in the Designer and review the specified return parameters:

- 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.

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.
High-level description of how the workflow works	• 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	• Use the **Create CM Volume** command • Command details for **Create CM Volume**: ◦ 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. The volume will be mounted in the SVM namespace as `/volname` (under the root namespace). ◦ Use the **actualVolumeSize** function because the snap reserve will be 5%. ◦ SVM reference is defined with the following resource selection logic: ▪ 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): ▪ 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

Name	Type	Description (data values, validation, and so on)
Cluster	Locked query (tabular)	• Cluster hosting the SVM • Query can be tabular display with primary address and name of the cluster • Sort alphabetically by name
SVM	Locked query	• SVM in which the volume is provisioned • Query should only display SVM names belonging to the cluster chosen in the previous input Show only cluster type SVMs, not admin or node (type column of cm_storage.vserver) • Sort alphabetically
Volume	String	• Name of the volume to be created
Size in GB	Integer	• 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 **Workflow Design** > **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.

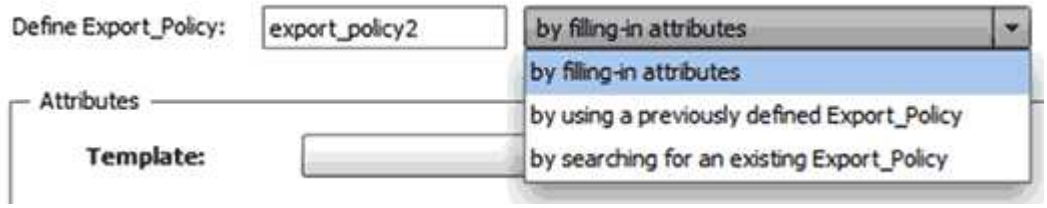
   b. **Optional:** Click to add the required number of rows, which are used to specify details for execution of steps.

      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.
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	Enter the value for attributes using the following options: • 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 *object*	Select the previously defined `object` in the box before the option list.

For…	Do this…

by searching for an existing *object*	i. Click **Enter search criteria** to search for the object using the resource selection capability.
	ii. Select one of the required options for execution if the required object is not found:
	◦ Abort workflow
	This option aborts the workflow execution if the specific object is not found.
	◦ Disable this command
	`This option disables only the current step and executes the workflow.`
	◦ Fill-in attributes for `object` and execute the command
	`This option enables you to enter the required attributes and execute the workflow.`

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

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

   b. **Optional:** Specify the entity version.
   c. **Optional:** Clear the **Consider Reserved Elements** check box if you do not want to use the reservation capability.
   d. **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:
   a. Click the **User Inputs** tab.
   b. Double-click the user input you want to edit.
   c. In the **Edit Variable: <user input>** dialog box, edit the user input.
8. If you want to add constants, perform the following steps
   a. 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.

   b. 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.

   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"]`

If the parameter name is…	Specify as…
`ChildWorkflow1.value$`	`ChildWorkflow1.value$`
`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.

   `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.

Copyright information

Trademark information