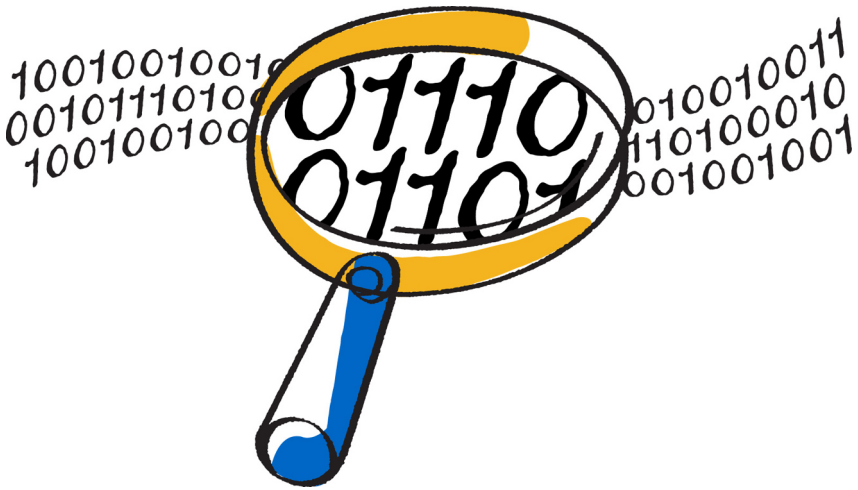
OnCommand® Unified Manager 6.1

API Developer's Guide



NetApp, Inc.
495 East Java Drive
Sunnyvale, CA 94089
U.S.

Telephone: +1 (408) 822-6000
Fax: +1 (408) 822-4501
Support telephone: +1 (888) 463-8277
Web: www.netapp.com
Feedback: doccomments@netapp.com

Part number: 215-08263_A0
December 2013

Contents

API and protection service concepts

Unified Manager provides a client application developer access to protection services through a set of APIs. These services include defining protection configurations, creating replication copies, and performing restore operations.

What Unified Manager protection APIs are

The Unified Manager protection APIs are interfaces between a client application and the Unified Manager server that permit the client application to issue requests to the Unified Manager server for tasks supporting the execution, configuration, management, and monitoring of volume data protection and destination provisioning.

In response to the client application requests, the Unified Manager protection API transmits those requests to the Unified Manager server, which either executes those requests directly or issues supporting requests to the Data ONTAP API for execution at that level. In addition, the Unified Manager protection API returns status or other data for each request to the client application.

Resource pools and the Unified Manager APIs

A client application can call Unified Manager APIs to configure storage services provisioned with specified resource pools that are already set up by a storage administrator in the main Unified Manager user interface.

Resource pools are groups of aggregates that are created by a storage administrator using the main Unified Manager user interface to provide provisioning to partner applications for backup management.

The Unified Manager administrator might pool resources based on attributes such as performance, cost, physical location, or availability. By grouping related resources into a pool, the Unified Manager administrator can treat the pool as a single unit for monitoring and provisioning.

Snapshot copies and the Unified Manager APIs

A client application can call Unified Manager APIs to start mirror or backup vault jobs that transfer Snapshot copies of data from source volumes to destination volumes. Other Unified Manager APIs can be called to start the restore of specified Snapshot copies from mirror or backup vault destination volumes to original or alternate primary locations.

When mirror or backup vault jobs are run, the protected data is transferred and stored in the form of Snapshot copies.

Volumes and the Unified Manager APIs

A client application calls Unified Manager APIs to carry out mirror and backup vault protection of entire volumes of data, rather than of smaller units of data such as directories or files.

Volumes are data containers that enable a cluster or virtual server administrator to partition and manage data. Volumes are the highest-level logical storage object.

Unlike aggregates, which are composed of physical storage resources, volumes are completely logical objects.

Data ONTAP provides two types of volumes: FlexVol volumes and Infinite Volumes, but the Unified Manager APIs implement storage service protection only for FlexVol volumes.

How APIs use resource keys

Resource keys are unique identifiers that client applications and related OnCommand management tools use to unambiguously specify the objects of their API calls.

A resource key is an identification string unique to an object, formulated by a syntax unique to that object type. For example, the syntax of a volume type object's resource key is a combination of the UUID of its containing cluster and its own UUID:

```
<cluster_uuid>:type=volume,uuid=<uuid>
```

The resulting volume resource key is a string similar to the following:

```
abcd7215-1f6b-11e1-a744-123478563412
:type=volume,uuid=14477215-1f6b-11e1-a744-12347856abcd
```

An API call to configure, update, manage, associate, delete, or destroy an object often includes the object's resource key to identify it.

API calls that return resource keys

Some Unified Manager APIs can be included in a client application to retrieve the object-related resource keys that must be specified for other configuration and management API calls.

The following sections list API calls that return resource keys for object types that might require specification in other API calls.

Detailed input and output descriptions of all API calls are provided in the *OnCommand Core Package API Documentation* downloadable compressed file. This file is accessed through the **NMSDK API Documentation** link in the Download section of the *NetApp Developer Community* site:

http://developer.netapp.com

APIs calls that list objects and their resource keys

The following API calls list objects of a particular type, along with their associated resource keys:

aggregate-iter	Lists aggregates and their resource keys.
cluster-iter	Lists clusters and their resource keys.
cluster-node-iter	Lists cluster nodes and their resource keys.
dp-relationship-iter	Lists data protection relationships and their resource keys.
disk-iter	Lists disks and their resource keys.
igroup-iter	Lists igroups and their resource keys.
lun-iter	Lists LUNs and their resource keys.
net-interface-iter	Lists networked interfaces and their resource keys.
qtree-iter	Lists qtrees and their resource keys.
resource-pool-iter	Lists resource pools and their resource keys.
snapshot-get-location	Lists Snapshot resource keys.
storage-class-iter	Lists storage classes and their resource keys.
vserver-iter	Lists and their resource keys.

API calls that look up or search objects and their resource keys

The following API calls can return the resource keys for objects specified by fully qualified name or by search strings:

resource-lookup	Returns the resource key of an object specified by its fully qualified name.
resource-search	Lists and returns the resource keys for objects of a specified type and character string.

What storage services are

A storage service is a preconfigured set of automated data protection and backup provisioning services to which multiple volumes can be subscribed.

When volumes are subscribed to a storage service, OnCommand Unified Manager configures secondary storage, SnapMirror or SnapVault relationships, and creates replication copies of primary data according to the properties defined in the storage service.

The subscribed volumes for which the storage service provides protection are called "root members." The destination volumes and the relationships between the root member volumes and destination volumes are called "protection artifacts." The root member volumes and the destination volumes, together are the "storage service members."

API support for storage services

Unified Manager provides a set of storage APIs that a developer can use to enable creation, management, and monitoring of storage services in a client application.

Functions that the APIs support include the following:

- Creating, configuring, modifying, cleaning up, and deleting storage services
- Configuring storage service protection topology
- Subscribing and unsubscribing volumes as root members of a storage service
- Initializing storage service protection
- Assigning context identification to sets of subscribed volumes
- Importing existing protection relationships into a storage service
- Enforcing storage system and member conformance to storage service requirements
- Listing storage services
- Listing storage service members
- Tracking all operations related to the storage service through jobs and tasks

Summary of basic API calls

Seven common API calls are used by the client application developer to implement the basic Unified Manager protection functions.

Although additional API calls are also necessary, the API calls most important to implementing storage service-based data protection are the following:

`storage-service-create`	Creates and configures a storage service. The inputs to this API call create a storage service and define its topology, which includes the type of data replication connection (mirror or vault), the location of the destination volumes, the provisioning requirements of the destination volumes, lag warning and error thresholds, and the resource pool for provisioning the destination volumes.
`storage-service-subscribe`	Subscribes volumes as root members to a storage service.
`storage-service-import`	Imports existing mirror or vault relationships that are currently not included in a storage service into a Unified Manager storage service.
`storage-service-conform`	Executes the cluster configuration of storage service members and connections to match the specifications made with the `storage-service-create`, `storage-service-subscribe`, and `storage-service-import` API calls.

`storage-service-update-start`	After the storage service is created, new members imported or subscribed, connections configured, and destination volumes generated, starts either the initial replication or the update replication of data from the root member volumes to the destination volumes.

What a storage service topology is

A storage service topology is the model that the Unified Manager server uses to configure data protection services. It contains the properties of the source node, the destination nodes, and the connections that theUnified Manager server uses to generate replicated data protection components for the subscribed root members of a storage service.

If a volume is subscribed as a root member to a storage service that is configured to provide mirror or vault protection, the Unified Manager server uses the storage service topology to generate destination volumes to contain the replicated data and to establish SnapMirror or SnapVault relations with those destination volumes.

If an existing data protection relationship, consisting of source volumes and destination volumes that are not currently managed by the Unified Manager server, is imported into a storage service, the Unified Manager server attempts to fit the source volumes, destination volumes into the storage service topology.

How a storage service topology is specified in the API

The Unified Manager API set enables the client application to specify a variety of storage service topologies to use for configuring and providing mirror or backup vault protection.

Inputs to the `storage-service-create` API call enable the client application to define the storage service topology by defining the topology's connections and nodes.

The client application developer defines a connection by specifying a source node name, a mirror or vault connection type, and a destination node name.

The developer further defines a destination node by specifying one or more resource pools from which the destination node is provisioned and a storage service workflow that specifies additional provisioning configuration requirements.

By defining and combining one or more connections for a storage service topology, the client application developer can create a variety of storage service topologies, from simple one-hop mirror or vault topologies, to fanout topologies, to cascading topologies.

After a storage service topology is defined, inputs to the `storage-service-modify` API enable the client application to modify it.

Unified Manager server supports all protection topologies that are supported in Data ONTAP. The APIs might allow the creation of unvalidated topologies, but only those topologies that Data ONTAP specifies as supported should be used.

Connection types supported between source and destination volumes

The Unified Manager API set supports two basic connection types between a source volume and a destination volume.

The `storage-service-create` API provides input for the client application developer to specify one of two connection types used in a storage service topology:

mirror connection
A mirror connection enables exact replication of all data in the source data object to the destination volume. After a transfer, all active data and Snapshot copies of that data at the source location are also included at the destination. All Snapshot copy retention policies applied to the source object also apply to the destination volume. Mirror connection destination volumes can be provisioned by the SnapMirror Destination storage service workflow.

vault connection
A vault connection enables replication of specified Snapshot copies of data from the source data object to the destination volume. The destination volume can retain a different number of Snapshot copies and follow different retention policies. Vault connection destination volumes can be provisioned by one of the following storage service workflows:

- SnapVault Destination
- SnapVault Destination with Deduplication
- SnapVault Destination with Deduplication and Compression

Destination provisioning storage service workflow support

A storage service workflow is a preconfigured set of actions that is specified by a client application through APIs to provide provisioning for destination data objects. Destination volumes are generated by Unified Manager to provide data protection for root member volumes that are newly subscribed to a storage service.

Preconfigured storage service workflows ensure that destination volumes are optimally provisioned to support a specified set of features. You can use them only as provided; you cannot modify them. The following storage service workflows are available:

SnapMirror Destination
Provisions a SnapMirror destination volume by selecting from the available resource pools an aggregate with the appropriate free space and thresholds.

When deduplication, compression, or both are enabled on the source, the data sent for replication over the network remains compressed, deduplicated, or both, resulting in the savings being inherited at the destination.

SnapVault Destination
Provisions a SnapVault destination volume by selecting from the available resource pools an aggregate with the appropriate free space and thresholds.

	No deduplication or compression is enabled on the destination. When deduplication, compression, or both are enabled on the source, the data sent for replication over the network remains compressed, deduplicated, or both, resulting in the savings being inherited at the destination. This storage service workflow is recommended when deduplication or compression is enabled on the primary volume.
SnapVault Destination with Deduplication	Provisions a SnapVault destination volume by selecting from the available resource pools an aggregate with the appropriate free space and thresholds. On the destination, deduplication is enabled, and compression is disabled. When deduplication, compression, or both are enabled on the source, the data sent for replication over the network remains compressed, deduplicated, or both, resulting in the savings being inherited at the destination. This storage service workflow is recommended when deduplication is disabled on the primary volume.
SnapVault Destination with Deduplication and Compression	Provisions a SnapVault destination volume by selecting from the available resource pools an aggregate with the appropriate free space and thresholds. Deduplication and compression are enabled on the destination volume. Enabling compression on a secondary volume is strongly discouraged, because storage efficiency on the source volume is not preserved during replication, and an offline storage efficiency scanner must be run for compression and deduplication to achieve storage savings. Additional compression on the destination uses more resources. In environments where much dense data exists on the source, such as virtualized environments employing file clones, data inflation during a transfer might cause failed backups due to lack of space on the secondary volume.

Simple one-hop storage service topology API call example

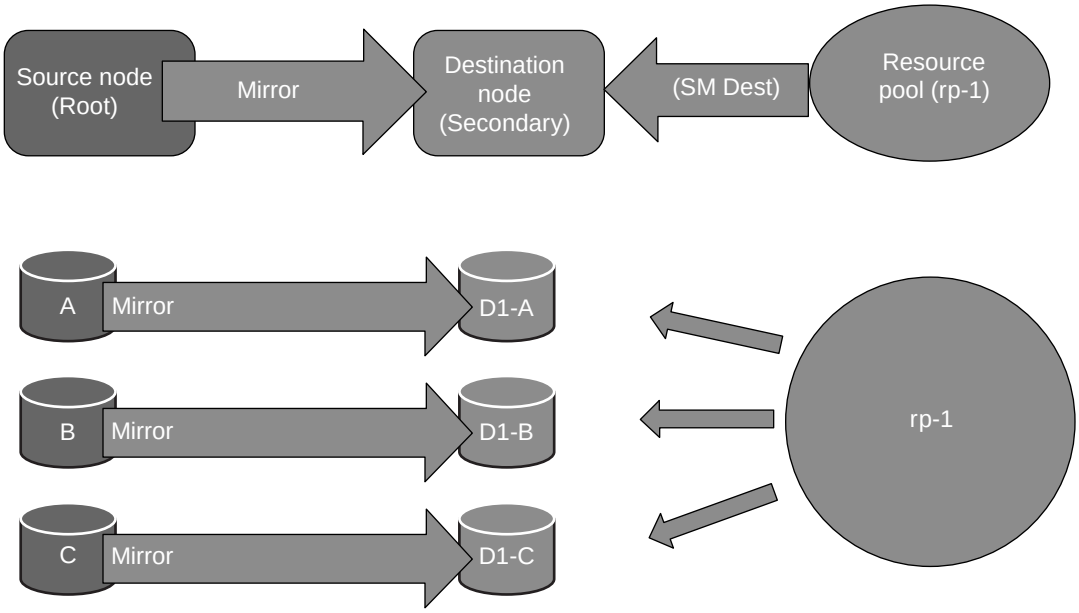
One of the simplest storage service topologies that the `storage-service-create` API enables the client application developer to specify is a one-hop, storage service topology consisting of a single mirror or vault type connection.

In this type of topology the source node defined for the single connection functions as the root node and the destination node functions as the secondary node.

Subscription example

Based on this topology, when primary storage volumes are subscribed as root members to the storage service, the Unified Manager server creates a mirror relationship with a newly-generated destination volume for each newly-subscribed root member. This relationship enables a mirror replication of the data in the root member volumes to their destination volumes. The destination volumes are

provisioned from resource pool rp-1 according to the requirements specified in storage service workflow SnapMirror Destination.



XML call example

The following XML code example shows a `storage-service-create` API call that specifies simple storage service topology with the following components: a single mirror type connection consisting of a source node (Root) and a destination node (Secondary), a resource pool (rp-1), and a storage service workflow (SnapMirror Destination).

```
  <storage-service-create>
    <storage-service-name>SimpleService</storage-service-name>
    <storage-service-client-tag>App-xz</storage-service-client-tag>
    <storage-service-topology-info>
       <nodes>
          <storage-service-topology-node-info>
             <node-name>primary</node-name>
          </storage-service-topology-node-info>
          <storage-service-topology-node-info>
             <node-name>Secondary</node-name>
             <resource-pools>
                <resource-key>063da002-97ac-43ca-
b43b-264a7d256017:app_type=OCUM,type=resource_pool,uuid=e98c5a56-21a3-40c
```

```
1-9ade-59e17b859639</resource-key>
             </resource-pools>
             <service-workflow-resource-key>063da002-97ac-43ca-
b43b-264a7d256017:app_type=OCUM,type=service_workflow,uuid=d22126c0-
e8b4-4eab-a2ef-d7243effe630</service-workflow-resource-key>
           </storage-service-topology-node-info>
         </nodes>
       <connections>
         <storage-service-topology-connection-info>
           <connection-type>mirror</connection-type>
           <destination-node-name>secondary</destination-node-name>
           <source-node-name>Root</source-node-name>
         </storage-service-topology-connection-info>
       </connections>
    </storage-service-topology-info>
  </storage-service-create>
```

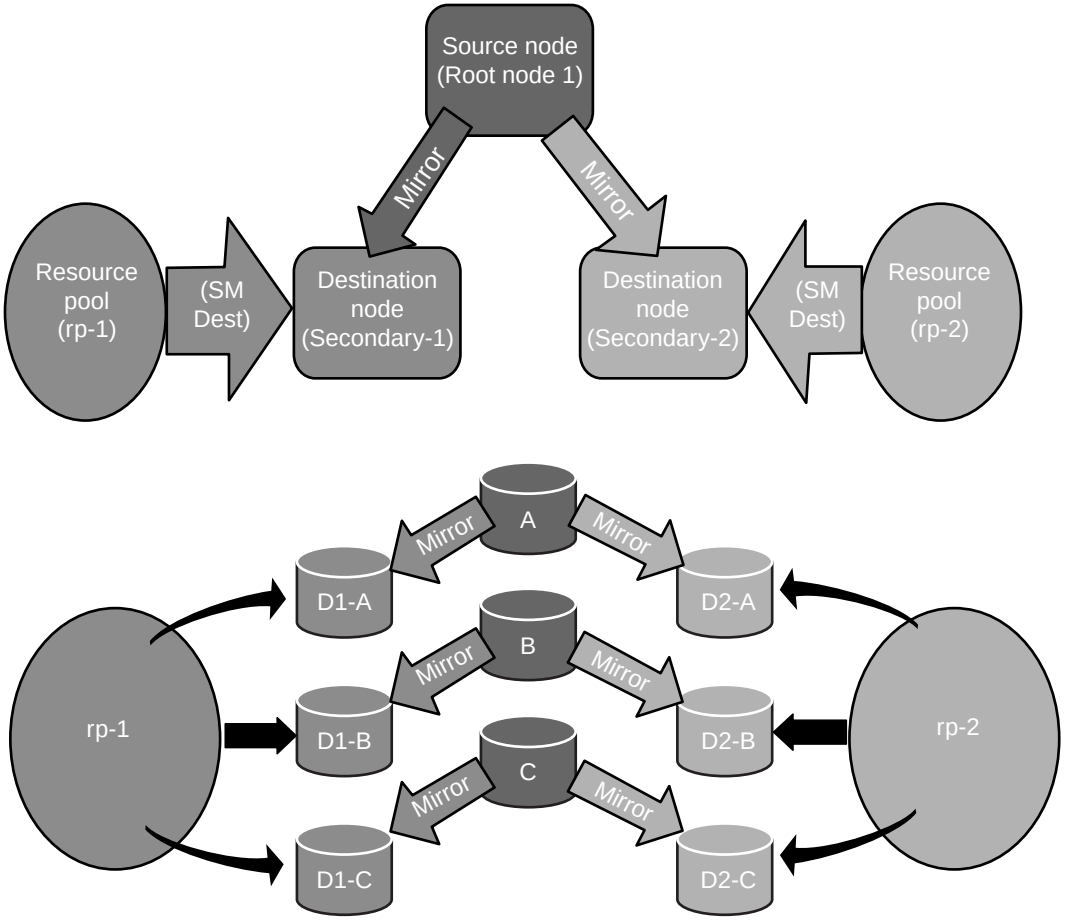
Fanout storage service topology API call example

The `storage-service-create` API enables the client application developer to specify a fanout storage service topology in which one node is included as the source node in several connections to as many different destination nodes.

In this type of topology the source node of all the connections functions as the root node, and the destination nodes of the connections function as fanout secondary nodes. The data in a single source data object is protected by simultaneous replication through multiple mirror or vault connections to multiple destination volumes.

Subscription example

Based on this fan out topology, when primary storage volumes are subscribed as root members to the storage service, the Unified Manager server creates two mirror relationships with two newly generated destination volumes for each newly subscribed root member. These relationships enable a mirror replication of the data in each root member volume to its two destination volumes. The destination volumes are provisioned from resource pools rp-1 and rp-2 according to the requirements specified in storage service workflow SnapMirror Destination.



XML call example

The following XML code example shows a `storage-service-create` API call that specifies a fanout storage service topology example with the following components: two mirror type connections with the same source node (Root-1) and two different destination nodes (Secondary-1 and Secondary-2), two resource pools (rp-1 and rp-2) that provision destination nodes Secondary-1 and Secondary-2, respectively, and one storage service workflow (SnapMirror Destination) applied to both destination nodes.

```
<storage-service-create>
<storage-service-name>SSFanout</storage-service-name>
<storage-service-topology-info>
  <connections>
   <storage-service-topology-connection-info>
     <connection-type>mirror</connection-type>
     <destination-node-name>Secondary-1</destination-node-name>
```

```
      <source-node-name>Root-1</source-node-name>
    </storage-service-topology-connection-info>
    <storage-service-topology-connection-info>
      <connection-type>mirror</connection-type>
      <destination-node-name>Secondary-2</destination-node-name>
      <source-node-name>Root-1</source-node-name>
    </storage-service-topology-connection-info>
  </connections>
  <nodes>
   <storage-service-topology-node-info>
    <node-name>Secondary-1</node-name>
    <resource-pools>
      <resource-key>063da002-97ac-43ca-
b43b-264a7d256017:app_type=OCUM,type=resource_pool,uuid=e98c5a56-21a3-40c
1-9ade-59e17b859639</resource-key>
    </resource-pools>
    <service-workflow-resource-key>
     063da002-97ac-43ca-
b43b-264a7d256017:app_type=OCUM,type=service_workflow,uuid=d22126c0-
e8b4-4eab-a2ef-d7243effe630
    </service-workflow-resource-key>
   </storage-service-topology-node-info>
   <storage-service-topology-node-info>
    <node-name>Secondary-2</node-name>
    <resource-pools>
      <resource-key>063da002-97ac-43ca-
b43b-264a7d256017:app_type=OCUM,type=resource_pool,uuid=e98c5a56-21a3-40c
1-9ade-59e17b859640</resource-key>
    </resource-pools>
    <service-workflow-resource-key>
     063da002-97ac-43ca-
b43b-264a7d256017:app_type=OCUM,type=service_workflow,uuid=d22126c0-
e8b4-4eab-a2ef-d7243effe631
    </service-workflow-resource-key>   </storage-service-topology-node-
info>
  </nodes>
</storage-service-topology-info>
</storage-service-create>
```

Cascade storage service topology API call example

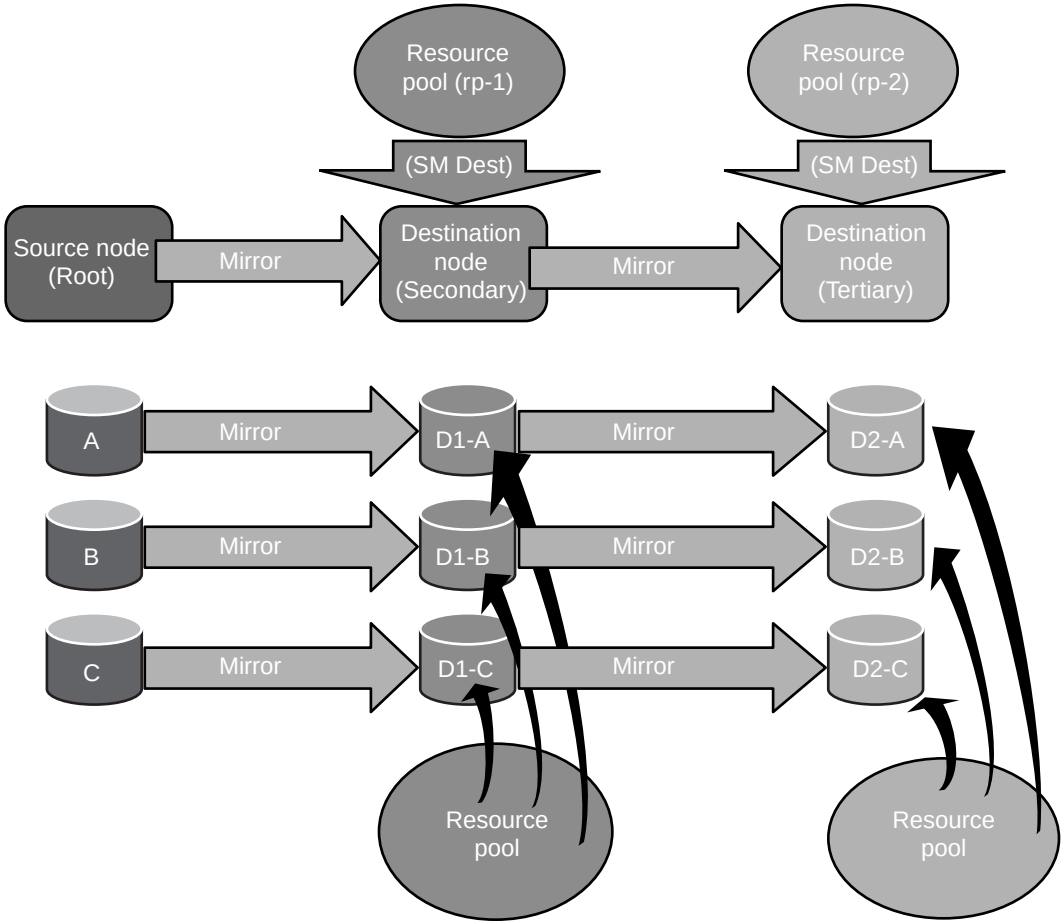
The `storage-service-create` API enables the client application developer to specify a cascade storage service topology that enables secondary and tertiary protection of primary data.

In this type of topology, the connections are chained so that the source node of the first connection functions as the root node, the destination node of the first connection functions as the secondary node, and the destination node of the second connection functions as the tertiary node.

Data in a primary source object is protected by replication through a mirror or vault connection to a destination volume in a secondary location, which is, in turn, replicated to another destination volume in a tertiary location. The intention is to provide source data with two or more levels of destination protection.

Subscription example

Based on this topology, when primary storage volumes are subscribed as root members to the storage service, the Unified Manager server creates two mirror relationships with two newly generated destination volumes for each newly subscribed root member. These relationships enable a mirror replication of the data in each root member volume to its two destination volumes. The destination volumes are provisioned from resource pools rp-1 and rp-2 according to the requirements specified in storage service workflow SnapMirror Destination.



XML call example

The following XML code example shows a `storage-service-create` API call that defines a mirror-mirror cascade storage service topology that specifies as components: one mirror type connection consisting of a source node (Root) and a destination node (Secondary) and second mirror type connection consisting of Secondary as the source node and a destination node (Tertiary), two

resource pools (rp-1 and rp-2) provisioning destination nodes Secondary and Tertiary and a storage service workflow (SnapMirror Destination) applied to both destination nodes.

```
<storage-service-create>
<storage-service-name>SSCascade</storage-service-name>
<storage-service-topology-info>
  <nodes>
    <storage-service-topology-node-info>
        <node-name>Root</node-name>
    </storage-service-topology-node-info>
    <storage-service-topology-node-info>
      <node-name>Secondary</node-name>
      <resource-pools>
       <resource-key>063da002-97ac-43ca-
b43b-264a7d256017:app_type=OCUM,type=resource_pool,uuid=e98c5a56-21a3-40c
1-9ade-59e17b859639</resource-key>
      </resource-pools>
      <service-workflow-resource-key>
       063da002-97ac-43ca-
b43b-264a7d256017:app_type=OCUM,type=service_workflow,uuid=d22126c0-
e8b4-4eab-a2ef-d7243effe630
      </service-workflow-resource-key>
    </storage-service-topology-node-info>
    <storage-service-topology-node-info>
      <node-name>Tertiary</node-name>
      <resource-pools>
       <resource-key>063da002-97ac-43ca-
b43b-264a7d256017:app_type=OCUM,type=resource_pool,uuid=e98c5a56-21a3-40c
1-9ade-59e17b859640</resource-key>
      </resource-pools>
      <service-workflow-resource-key>
       063da002-97ac-43ca-
b43b-264a7d256017:app_type=OCUM,type=service_workflow,uuid=d22126c0-
e8b4-4eab-a2ef-d7243effe631
      </service-workflow-resource-key>
    </storage-service-topology-node-info>
  </nodes>
  <connections>
   <storage-service-topology-connection-info>
     <connection-type>mirror</connection-type>
     <destination-node-name>Secondary</destination-node-name>
     <source-node-name>Root</source-node-name>
   </storage-service-topology-connection-info>
   <storage-service-topology-connection-info>
     <connection-type>mirror</connection-type>
     <destination-node-name>Tertiary</destination-node-name>
     <source-node-name>Secondary</source-node-name>
   </storage-service-topology-connection-info>
  </connections>
</storage-service-topology-info>
</storage-service-create>
```

API task flows

API task flows provide client application developers with examples of how Unified Manager APIs can be called by an client application to execute common protection-related functions. Task flows summarize API requests made by a client application and responses returned by Unified Manager server during execution of common protection jobs.

Where to find detailed API descriptions

Fully detailed API input and output descriptions necessary to implement the API calls that are summarized in the API task flows are provided in the "OnCommand Core Package API Documentation" downloadable compressed file.

This compressed file is accessed through the NMSDK API Documentation link in the Download section of NetApp Developer Community site:

http://developer.netapp.com

Creation of a storage service

The client application can call the `resource-pool-iter`, `storage-service-workflow-list-info`, and `storage-service-create` APIs to create a storage service. After a storage service is created, it can provide preconfigured and automated services, such as data protection, to volumes that subscribe to it.

Scenario

In this scenario, the client application operator uses the client application to create and configure a storage service capable of providing data protection to subscribed volumes.

This scenario is based on the assumption that the client application has the following items available:

- A storage service workflow that supports destination provisioning
- One or more resource pools with one or more aggregates and

Task flow

1. The storage administrator starts creation of a storage service.
2. The client application requests from Unified Manager a list of storage service workflows that support destination provisioning by calling the following API:

```
storage-service-workflow-list-info
```

3. Unified Manager returns a list of storage service workflows to the client application in the following element:

```
storage-service-workflows
```

4. The client application requests a list of resource pools that can provide provisioning by calling the following API:

```
resource-pool-iter
```

When issuing these calls, the client application enables the `is-provisionable` flag.

5. Unified Manager returns the list of available resource pools in the following element:

```
records
```

6. The backup administrator uses the client application user interface to define a storage service policy with at least the following parameters:
   - A storage service topology source node
   - One or more storage service topology destination nodes, associated resource pool, and storage service workflow
   - One or more connections between the source and one or more destination nodes:
     - The connection type specifies whether the relationship between two nodes is a mirror relationship or a vault relationship.
     - The lag threshold

       Specifies the amount of time, in seconds, after which the system generates a warning event if the relationship is not updated.
     - The maximum transfer rate specifies the upper boundary at which data is transferred for relationships created in this connection.
   - Additional general storage service attributes, such as owner, contact, and description
7. The client application requests that Unified Manager create the storage service with the selected policy parameters by calling the following API:

```
storage-service-create
```

When issuing the `storage-service-create` API, the client application specifies elements that reflect the selected policy parameters, resource pools, and topology.

8. Unified Manager creates a storage service and returns a storage service resource key to the client application in the following element:

```
storage-service-resource-key
```

Subscription of volumes to an existing storage service

A client application can call the `storage-service subscribe` and the `storage-service-conform` APIs to subscribe data objects to an existing storage service. The subscribed data objects then receive data protection or whatever other service that the storage service is configured to provide.

Scenario

In this scenario, an object or set of objects needing protection, such as a virtual machine (VM), database, NFS export, CIFS share, or volume are mapped by the client application to one or more Data ONTAP volumes, and then those volumes are subscribed to a storage service.

At the end of this task flow protection relationships are set up from the newly subscribed volumes to newly-generated destination volumes. Actual data replication is enabled, but has not yet taken place.

Requirements

In Unified Manager, you must have a storage service available.

Task flow

1. The client application lists existing volumes in need of storage service based protection by calling the following API:

   ```
   volume-iter
   ```

   OnCommand Unified Manager returns a list of volumes along with their resource keys.

2. The client application lists existing storage services by calling the following API:

   ```
   storage-service-iter
   ```

   OnCommand Unified Manager returns a list of storage services along with their resource keys.

3. The backup administrator selects one or more volumes to protect and associates them with a specific protection topology.

4. The client application determines the storage footprint of the object being protected and then chooses the OnCommand Unified Manager storage service that best matches the selected protection topology.
   The client application then subscribes the volumes to that storage service by calling the following API:

   ```
   storage-service-subscribe
   ```

   When issuing this call the client application includes the following inputs:

- Either the appropriate context tag or the resource keys of the volumes being subscribed
- The resource key of the storage service to which the volumes are being subscribed

5. Unified Manager starts the storage service subscription job and returns a job identifier in the following element:

```
job-id
```

6. The client application starts monitoring for completion of the job associated with the job ID by calling the following API:

```
job-wait-for-state
```

When issuing this call, the client application specifies a timeout value.

7. The client application requests Unified Manager to start the storage service conformance and configuration job by calling the following API:

```
storage-service-conform
```

When issuing this call the client application includes the following inputs:

- Either the appropriate context tag or the resource keys of the volumes being subscribed
- The resource key of the storage service to which the volumes are being subscribed

8. Unified Manager starts the storage service conformance job and returns a job identifier using the following element:

```
job-id
```

9. The storage service conformance job checks all volumes associated with the specified subscription-context input, and starts tasks for bringing the specified data objects into conformance.
In the case of a storage service that provides data protection, these tasks might include provisioning destination data objects and creating SnapMirror relationships.

10. The client application performs the following process loop to monitor the progress of the job:

    a. The application starts monitoring for completion of the job associated with the job ID by calling the following API:

    ```
    job-wait-for-state
    ```

    When issuing this call, the client application specifies a timeout value.

    b. If the `job-wait-for-state` call times out before the job finishes, the client application optionally tracks the progress of the job by calling the following API:

    ```
    job-task-iter
    ```

    c. This loop repeats until the job finishes.

d. When the job has completed, the `job-wait-for-state` API returns the following element:

```
job-ID
```

11. The client application checks the success or failure of the completed job and retrieves any error messages by performing the task flow *Detection and troubleshooting of a failed job* on page 37.
12. If the storage service conformance job is successful, the specified data objects are successfully subscribed to the storage service, and no errors or failures are returned.

Replication of local Snapshot copies to remote nodes

A client application can call the `storage-service-protection-update-start` and the `snapshot-get-location` APIs to replicate local Snapshot copies to remote nodes.

Task flow

Replication of local Snapshot copies to remote nodes proceeds as follows:

1. The client application begins SnapMirror updates for the root member volumes by calling the following API:

   ```
   storage-service-protection-update-start
   ```

   When making this call, the client application specifies the resource key of the affected storage service. If this is the only input, the protection relationships of all members subscribed to this storage service are updated by Snapshot copy replication.
   Additional optional inputs restrict the scope of the Snapshot replication relationship updates within the storage service:

   - Specifying resource keys of one or more storage service root member volumes restricts the update to relationships in which those volumes are the root member.
   - Specifying resource keys for Snapshot copies of one or more root members restricts the update to those relationships in which those volumes are the root members.
   - Specifying a context ID restricts the scope of the update to those relationships associated with that context ID.
   - Specifying a storage service “source node” resource key restricts the scope of the update to the volumes on the specified storage service node.
   - Specifying a storage service “destination node” resource key restricts the scope of the update to the volumes on the storage service node root only up to the volumes on the specified storage service node.
2. The Unified Manager server starts a SnapMirror update from source to destination volumes, and returns a job identifier in the element:

   ```
   job-ID
   ```

3. The client application performs the following process loop to monitor the progress of the job:

a. The application starts monitoring for completion of the job associated with the job ID by calling the following API:

```
job-wait-for-state
```

When issuing this call, the client application specifies a timeout value.

b. If the `job-wait-for-state` call times out before the job finishes, the client application optionally tracks the progress of the job by calling the following API:

```
job-task-iter
```

c. This loop repeats until the job finishes.

d. When the job has completed, the `job-wait-for-state` API returns the following element:

```
job-ID
```

4. The client application checks the success or failure of the completed job and retrieves any error messages by performing the task flow *Detection and troubleshooting of a failed job* on page 37.

5. When the job finishes, the client application retrieves the locations of the destination Snapshot copies by calling the following API:

```
snapshot-get-location
```

When issuing this call, the client application specifies either a Snapshot copy UUID or the original volume name plus the name of its local Snapshot copy.

6. Unified Manager returns the requested information in the following element:

```
snapshot-location-results
```

7. The client application records the destination location Snapshot copies in the backup catalog.

On-demand conformance of a storage service

The client application can call the `storage-service-conform` API to perform an on-demand storage service conformance check of failed protection relationships and to attempt repair of those relationships.

Scenario

In this scenario, an attempt by the backup administrator to subscribe a set of volumes to a storage service has failed due to improper network configuration.

Informed by the backup administrator of the failure, the system administrator has remedied the configuration problem and instructed the backup administrator to attempt an on-demand automated storage service conformance check to repair the missing relationships.

This task flow description is based on the following assumptions:

- In Unified Manager, the backup administrator has a storage service available.
- In Unified Manager, the backup administrator has the following credentials for the source cluster and, if applicable, the destination cluster:
  - IP address
  - Administrator rights

Task flow

The on-demand attempt to automatically confirm or establish storage service conformance proceeds as follows:

1. The backup administrator, through the client application interface, requests on-demand storage service conformance, causing the client application to signal Unified Manager to start the storage service conformance job by calling the following API:

   ```
   storage-service-conform
   ```

   When issuing this API call, the client application includes the following inputs:

   - The context string or the resource keys of the volumes whose protection relationship failed
   - The resource key of the affected storage service
2. Unified Manager starts the storage service conformance job and returns a job identifier for this check in the following element:

   ```
   job-id
   ```

3. The storage service conformance job confirms whether all volumes associated with the specified context ID conform to the storage service's protection requirements.
4. If the specified objects are not yet in conformance with the storage service membership requirements, the storage service conformance job starts tasks for bringing the specified objects into conformance.
5. The client application performs the following process loop to monitor the progress of the job:
   a. The application starts monitoring for completion of the job associated with the job ID by calling the following API:

      ```
      job-wait-for-state
      ```

      When issuing this call, the client application specifies a timeout value.

   b. If the `job-wait-for-state` call times out before the job finishes, the client application optionally tracks the progress of the job by calling the following API:

      ```
      job-task-iter
      ```

   c. This loop repeats until the job finishes.

d. When the job has completed, the `job-wait-for-state` API returns the following element:

```
job-ID
```

6. The client application checks the success or failure of the completed job and retrieves any error messages by performing the task flow *Detection and troubleshooting of a failed job* on page 37.
7. The client application checks the success or failure status of the completed storage service conformance job by calling the following APIs:

```
job-iter
```

8. If the storage service conformance job is successful, no errors or failures are returned.

Data restoration from a source to destination location

The client application can call the `snapshot-get-location` and `dp-restore-start` APIs to restore Snapshot copy data from a source location to a destination location.

Scenario

In this scenario, data stored in a Snapshot copy in a backup location, needs to be restored to its original location or another non-backup location where it is active and accessible by authorized users. In this task flow, the backup location is the source and the non-backup location is the destination. No membership in a storage service is necessary.

This task flow description is based upon the following assumptions:

- In Unified Manager, the operator has the following credentials for the source cluster and, if applicable, the destination cluster:
  - IP address
  - Admin rights
  - NDMP
- The SnapRestore feature is licensed on the source cluster and, if applicable, the destination cluster.

Task flow

In this task flow, a data restore job from the restore job's source location to its destination location is set up, executed, and tracked.

1. The backup administrator starts the restore operation by requesting, through the client application, a list of replicated Snapshot copies in backup storage locations that contain the data to be restored.
2. If information about Snapshot copies in backup storage locations is not maintained in the client application database, this information is retrieved from Unified Manager, as follows:

a. The client application requests a list of the appropriate Snapshot copies from Unified Manager by calling the following API:

```
snapshot-get-location
```

When issuing this call, the client application includes one of the following inputs:

- The Snapshot copy name plus the resource key of the volume
- The resource key of the copy itself

b. Unified Manager returns the requested information in the following element:

```
snapshot-location-results
```

3. The client application displays to the backup administrator the list of replicated Snapshot copies in backup locations.

4. The backup administrator uses the client application user interface to specify the restoration parameters.

5. The client application requests Unified Manager to restore data by calling the following API:

```
dp-restore-start
```

When issuing this call, the client application specifies the following information:

- The resource key of the Snapshot copy that contains the data to be restored
- The path within that copy to the volume, directory, or file to be restored
- The path to the location where the data is to be restored
- If the restore is to an alternate location and the `use-snapshot-restore-volume` parameter is false, the destination volume name or resource key

6. Depending on the specific requirements of the `dp-restore-start` request Unified Manager starts the appropriate Data ONTAP operation for restoring the source Snapshot copy to the specified destination path and returns a job identifier in the element:

```
job-id
```

7. The client application performs the following process loop to monitor the progress of the job:

a. The application starts monitoring for completion of the job associated with the job ID by calling the following API:

```
job-wait-for-state
```

When issuing this call, the client application specifies a timeout value.

b. If the `job-wait-for-state` call times out before the job finishes, the client application optionally tracks the progress of the job by calling the following API:

```
job-task-iter
```

c. This loop repeats until the job finishes.

d. When the job has completed, the `job-wait-for-state` API returns the following element:

```
job-ID
```

8. The client application checks the success or failure of the completed job and retrieves any error messages by performing the task flow *Detection and troubleshooting of a failed job* on page 37.

9. If the `dp-restore-start` job is successful, no errors or failures are returned.

Destruction of a storage service

The client application can use `storage-service-destroy` API to destroy a storage service that is no longer required. Destroying a storage service does not destroy the subscribed root members, but it does destroy the protection artifacts, such as the Snapshot copies, protection relationships, and destination volumes that the destroyed storage service created to support its services.

Scenario

In this scenario, a storage administrator decides to destroy a storage service and its associated protection artifacts to free cluster and Unified Manager resources because the subscribed root members of the storage service no longer require the protection services that the storage service provides.

This task flow is based upon the assumption that the client application operator, normally a storage administrator for this task, has the following credentials for the source cluster and, if applicable, the destination cluster:

- IP address
- Administrator rights

Task flow

The destruction of a storage service proceeds as follows:

1. The storage administrator, through the client application interface, requests destruction of a specified storage service, causing the client application to signal Unified Manager to start the destroy job by calling the following API:

   ```
   storage-service-destroy
   ```

   When issuing this API call, the client application specifies the resource key of the storage service to destroy.

2. Unified Manager starts the job and returns a job identifier in the following element:

   ```
   job-id
   ```

3. The job begins the destruction of all protection artifacts (the Snapshot copies, protection relationships, and the secondary volumes) that the specified storage service generated when volumes were subscribed to it.
4. The client application starts a process loop to monitor the progress of the job:
   a. The client application starts monitoring for completion of the job associated with the job-ID by calling the following API:

      ```
      job-wait-for-state
      ```

      When issuing this call, the client application specifies a timeout value.
   b. If the `job-wait-for-state` call times out before the job finishes, the client application optionally tracks the progress of the job by calling the following API:

      ```
      job-task-iter
      ```

   c. The client application repeats this loop until the storage service destroy job achieves a completed state.
5. When the storage service destroy job achieves a completed state, the `job-wait-for-state` API returns with the following element:

   ```
   job-id
   ```

6. The client application checks the success or failure status of the completed job by calling the following API:

   ```
   job-iter
   ```

7. At the end of the success or failure status check, the client application does one of the following:
   - If the job is successful, no errors or failures are returned and the client applicationdisplays feedback that the operation is successful.
   - If the storage service destroy job is not successful, the client application executes an error condition routine.

Error condition

An error condition routine, executed if the storage service destroy job is unsuccessful, returns error message information that might help the client application operator with troubleshooting information. The routine includes the following actions:

1. The client application checks the state of the storage services in Unified Manager by calling the following API:

   ```
   storage-service-iter
   ```

2. Unified Manager returns storage service state information in the following element:

   ```
   records
   ```

   If the target storage service remains undeleted, the `records` element `is-storage-service-marked-for-deletion` flag is enabled.

3. The client application retrieves any related error messages by calling the following API:

   ```
   job-task-iter
   ```

   When making these API calls, the client application specifies the job-id value.

4. The Unified Manager server returns task message information in the following element:

   ```
   job-task-info
   ```

   This element contains timestamp information, type information, and the reason for failure of the job.

5. To facilitate further troubleshooting, if necessary, the client application can provide an HTTP link to information on a failed job on the server task details page of the Unified Manager server.
   The HTTP link is specified by the following syntax:

   ```
   https://<name_address_um_srvr>:8443
   /#job-details:details-place-object-id=
   <job-id_with_colons_replaced_by_%253A>
   ```

   For example, if the IP address of the Unified Manager server is 10.229.155.49:8443, and the job-id value is bb9a5ff37d10cb4e:e2c16a8:13a8da5bf0d:5671, then the client application generates a link such as the following:

   ```
   https://10.229.155.49:8443/
   #job-details:details-place-object-id=
   bb9a5ff37d10cb4e%253Ae2c16a8%253A13a8da5bf0d%253A5671
   ```

Import of existing storage relationships into a storage service

To provide managed protection to existing unmanaged relationships, a client application can call the `storage-service-import` API to import them into a storage service.

Requirements

The client application operator, normally a backup administrator for this task, must have the following credentials for the source cluster and, if applicable, the destination cluster:

- IP address

- Administrator rights

In Unified Manager, the backup administrator must have a storage service and an existing relationship available.

Task flow

The relationship import operation proceeds as follows:

1. The storage administrator, through the client application interface, requests that one or more existing relationships be imported into a specified storage service, causing the client application to signal the Unified Manager server to start the import job by calling the following API:

   ```
   storage-service-import
   ```

   When issuing this API call, the client application specifies the following information:

   - The resource key of the storage service into which a relationship is to be imported
   - For each relationship, a storage-service-import-info element that contains the following:
     - The resource key of the storage service connection to which the relationship is to be mapped.
       This information can be found by using the `storage-service-iter` or `resource-lookup` APIs.
     - The resource key of either the relationship or the relationship's destination volume.
       This information can be found by using the `dp-relationship-iter`, `volume-iter`, or `resource-lookup` APIs.
   - The subscription context (required unless the source volume of the relationship being imported is already a member of the storage service).

2. The Unified Manager server starts the storage service import job and returns a job identifier in the element:

   ```
   job-ID
   ```

3. The storage service import job begins to import the relationship into the storage service.
4. The client application performs the following process loop to monitor the progress of the job:
   a. The application starts monitoring for completion of the job associated with the job ID by calling the following API:

      ```
      job-wait-for-state
      ```

      When issuing this call, the client application specifies a timeout value.

   b. If the `job-wait-for-state` call times out before the job finishes, the client application optionally tracks the progress of the job by calling the following API:

      ```
      job-task-iter
      ```

c. This loop repeats until the job finishes.

d. When the job has completed, the `job-wait-for-state` API returns the following element:

```
job-ID
```

5. The client application checks the success or failure of the completed job and retrieves any error messages by performing the task flow *Detection and troubleshooting of a failed job* on page 37.
6. After the import job succeeds, a storage service conformance check must be performed to ensure that the imported relationships and members conform to the properties of the storage service. This might include provisioning and creating new relationships to match the storage service topology. The client application calls the following API:

   ```
   storage-service-conform
   ```

   When issuing this API call, the client application specifies the resource key of the storage service on which to run the conformance check.
7. The Unified Manager server starts the storage service conform job and returns a job identifier in the following element:

   ```
   job-ID
   ```

8. The storage service conform job begins to check conformance for all subscribed members of the storage service.
9. The client application starts a process loop to monitor the progress of the storage service conform job, checks the success or failure status of the completed job, and retrieves error messages if the job has failed, as detailed in Steps 4 and 5.

Example: Import two relationships into a fan-out when the shared source volume is a storage service member

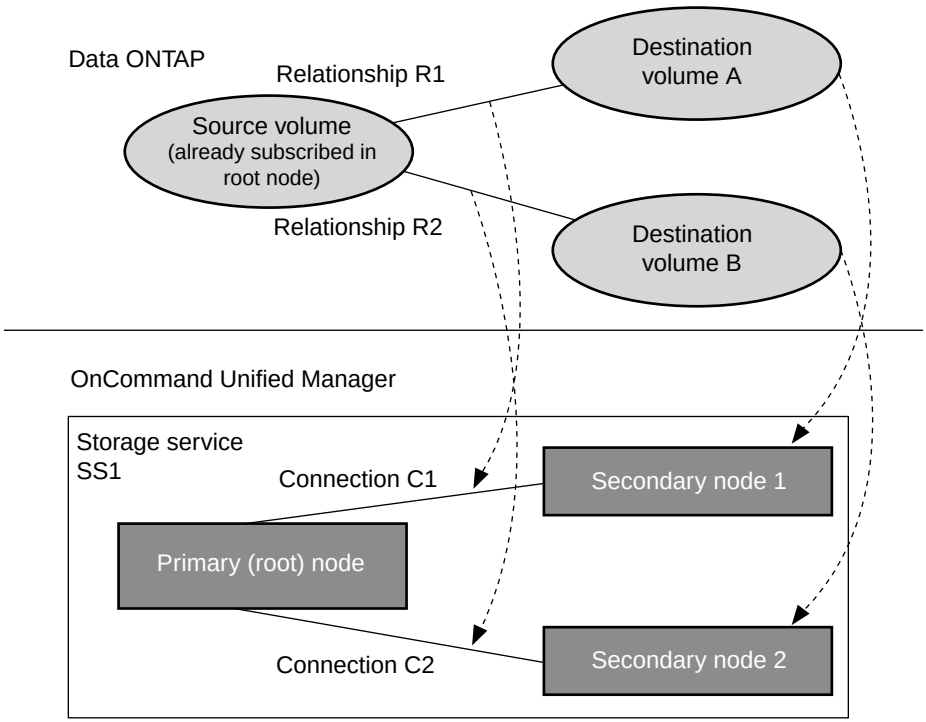
Two relationships, R1 and R2, are being imported into storage service SS1, which has a two-leg fan-out topology. Both relationships share a source volume that is already a member of the storage service. (This could happen in some cases if, for example, the volume was subscribed, but conformance failed, and the relationships were then created outside of Unified Manager.)

Since the source volume is already a member of the storage service, subscription context is not required. Each relationship and connection pair is specified in its own storage-service-import-info element. During the import job, relationship R1 is imported to connection C1, and R2 to C2. When the conformance check is performed after the import job, no fixes are required, since the imported relationships perfectly match the storage service topology.

The following sample XML shows the required inputs for this example:

```
<storage-service-import>
     <storage-service-resource-key>SS1 resource key</storage-service-
resource-key>
```

```
     <import-info>
          <storage-service-import-info>
               <connection-resource-key>C1 resource key</connection-
resource-key>
               <relationship-resource-key>R1 resource key</relationship-
resource-key>
          </storage-service-import-info>
          <storage-service-import-info>
               <connection-resource-key>C2 resource key</connection-
resource-key>
               <relationship-resource-key>R2 resource key</relationship-
resource-key>
          </storage-service-import-info>
     </import-info>
</storage-service-import>
```

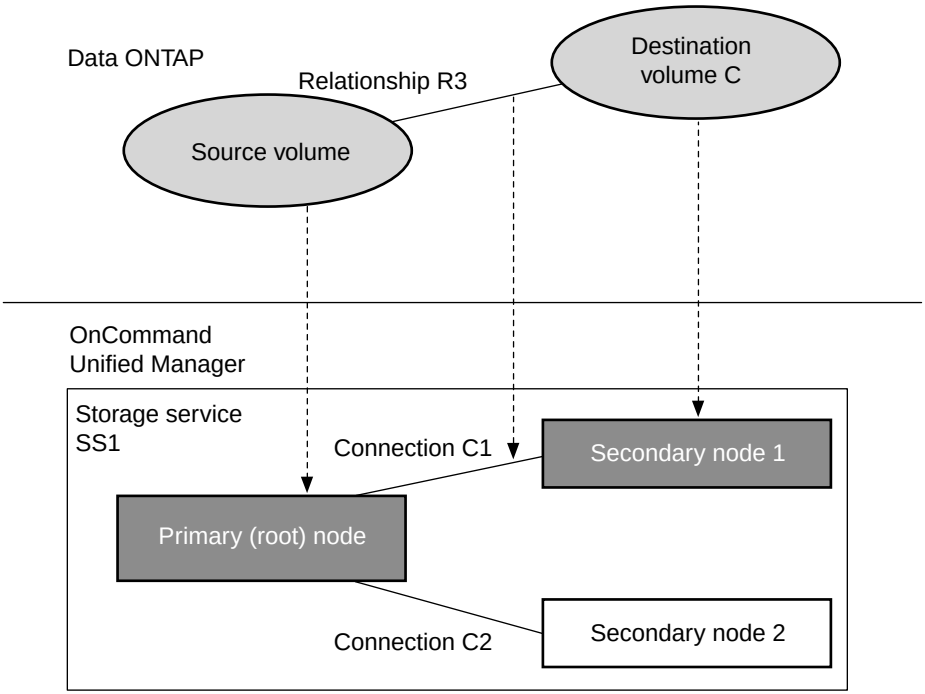


Example: Import one relationship into a fan-out when the source volume is not a storage service member

Relationship R3 is being imported into connection C1 of storage service SS1, which has a two-leg fan-out topology. The source volume of relationship R3 is not a member of the storage service, and a subscription context is required. When the conformance check is performed after the import job, a new relationship is created for connection C2 to complete the storage service topology.

The following sample XML shows the required inputs for this example:

```
<storage-service-import>
     <storage-service-resource-key>SS1 resource key</storage-service-
resource-key>
     <import-info>
          <storage-service-import-info>
               <connection-resource-key>C1 resource key</connection-
resource-key>
               <relationship-resource-key>R3 resource key</relationship-
resource-key>
          </storage-service-import-info>
     </import-info>
     <subscription-context>12345</subscription-context>
</storage-service-import>
```



Unsubscription of a root member and destruction of protection artifacts

A client application can use the `storage-service-unsubscribe` API to unsubscribe root members from an existing storage service. Unsubscribing stops any further protection updates for a member object but does not remove it. After unsubscribing objects, the client application can use the

`storage-service-cleanup` API to destroy all associated protection artifacts, such as data protection relationships and all relevant provisioned data objects.

Requirements

In Unified Manager, the backup administrator must have the following credentials for the source cluster and, if applicable, the destination cluster:

- IP address
- Administrator rights

Scenario

The following scenario describes the existing environment before the client application calls the `storage-service-unsubscribe` API.

The client application requested the Unified Manager `storage-service-create` API to create a mirror storage service called “Mirror to single destination”. Unified Manager provisioned three destination volumes and created three SnapMirror relationships. Several SnapMirror updates were initiated on the three volumes as the client application created local Snapshot copies on the primary volumes and transferred them to SnapMirror destinations using the Unified Manager `storage-service-protection-update-start` API.

Task flow

The unsubscription of a root member proceeds as follows:

1. The client application determines that one of the volumes no longer requires protection and requests that Unified Manager unsubscribe the volume from the storage service by calling the following API:

   ```
   storage-service-unsubscribe
   ```

2. The client application requests the removal of the destination volume and the associated SnapMirror relationships, signaling Unified Manager to start the cleanup job by calling the following API:

   ```
   storage-service-cleanup
   ```

   When calling the API, the client application specifies the primary volume name and `keep-storage-artifacts=false` inputs.

3. Unified Manager begins the storage service cleanup job and returns a job identifier using the following element:

   ```
   job-id
   ```

4. The client application performs the following process loop to monitor the progress of the job:

a. The application starts monitoring for completion of the job associated with the job ID by calling the following API:

```
job-wait-for-state
```

When issuing this call, the client application specifies a timeout value.

b. If the `job-wait-for-state` call times out before the job finishes, the client application optionally tracks the progress of the job by calling the following API:

```
job-task-iter
```

c. This loop repeats until the job finishes.

d. When the job has completed, the `job-wait-for-state` API returns the following element:

```
job-ID
```

5. The client application checks the success or failure of the completed job and retrieves any error messages by performing the task flow *Detection and troubleshooting of a failed job* on page 37.

The job runs in the background. The SnapMirror relationship and the destination are destroyed, and both volumes are removed from the storage service; however, the primary volume is not destroyed.

Unsubscription of a root member and relinquishment of its protection artifacts

The client application can use the `storage-service-unsubscribe` API to unsubscribe root member objects from an existing storage service. Unsubscribing stops any further protection updates for an object but does not remove it. After unsubscribing objects, the client application can use the `storage-service-cleanup` API to relinquish the associated protection artifacts of the unsubscribed object without destroying them.

Requirements

In Unified Manager, the backup administrator must have the following credentials for the source cluster and, if applicable, the destination cluster:

- IP address
- Administrator rights

Scenario

- The client application requests the Unified Manager `storage-service-create` API to create a mirror storage service called “Mirror to single destination.”
- The client application subscribes three volumes to the “Mirror to single destination” storage service.

- Unified Manager provisions three destination volumes and creates three SnapMirror relationships.
- Several SnapMirror updates are initiated on the three volumes as the client application creates local Snapshot copies on the primary volumes and transfers them to SnapMirror destinations using the Unified Manager `storage-service-protection-update-start` API.

Task flow

1. The client application requests that Unified Manager unsubscribe a root member volume from its storage service by calling the following API:

   ```
   storage-service-unsubscribe
   ```

2. The client application requests the removal of the destination volume and the associated SnapMirror relationships, signaling Unified Manager to start the cleanup job by calling the following API:

   ```
   storage-service-cleanup
   ```

   When calling the API, the client application specifies the primary volume name and `keep-storage-artifacts=true` inputs. In this case, the client application requests that the artifacts not be destroyed.

3. Unified Manager begins the storage service cleanup job and returns a job identifier using the following element:

   ```
   job-id
   ```

4. The client application performs the following process loop to monitor the progress of the job:

   a. The application starts monitoring for completion of the job associated with the job ID by calling the following API:

      ```
      job-wait-for-state
      ```

      When issuing this call, the client application specifies a timeout value.

   b. If the `job-wait-for-state` call times out before the job finishes, the client application optionally tracks the progress of the job by calling the following API:

      ```
      job-task-iter
      ```

   c. This loop repeats until the job finishes.

   d. When the job has completed, the `job-wait-for-state` API returns the following element:

      ```
      job-ID
      ```

5. The client application checks the success or failure of the completed job and retrieves any error messages by performing the task flow *Detection and troubleshooting of a failed job* on page 37.

The job runs in the background. The protection artifacts that are associated with the unsubscribe root member volume (the SnapMirror relationship and the and destination volumes) are removed from the storage service, but they are not destroyed. Keeping the artifacts enables the storage administrator to either import the relationship into a new storage service or into the same storage service at a later time.

Detection and troubleshooting of a failed job

The client application can call the `job-iter` and `job-task-iter` APIs to detect when a completed job has failed and to retrieve error messages if so.

Scenario

In this task flow, a job has completed on the Unified Manager server, and the client application detects that the job failed. The application then retrieves error messages that might indicate the reason for failure, and provides an HTTP link to additional information about the failed job.

Task flow

1. The client application checks the success or failure of the completed job by calling the following API:

   ```
   job-iter
   ```

2. Detecting that the job has failed, the client application retrieves error messages by calling the following API:

   ```
   job-task-iter
   ```

   When making these API calls, the client application specifies the job ID.

3. The Unified Manager server returns task message information in the following element:

   ```
   records
   ```

   This element includes information such as timestamps, task type, reason for failure, and any messages generated during execution of the job.

4. To facilitate further troubleshooting, the client application provides an HTTP link to information about the failed job on the job's task details page in the Unified Manager GUI.
   The link should be constructed with the following syntax, using standard URL encoding for strings (replacing colons with “%253A”):

   ```
   https://<name_address_um_srvr>:8443
   /#job-details:details-place-object-id=
   <job-ID>
   ```

For example, if the IP address of the Unified Manager server is 10.229.155.49:8443, and the job ID of the failed job is bb9a5ff37d10cb4e:e2c16a8:13a8da5bf0d:5671, then the client application generates the following link:

```
https://10.229.155.49:8443/
#job-details:details-place-object-id=
bb9a5ff37d10cb4e%253Ae2c16a8%253A13a8da5bf0d%253A5671
```

Recovery from unexpected deletion of a non-root storage service member

A client application can call the `storage-service-member-iter` API, the `storage-service-cleanup` API, and the `storage-service-conform` API to recover from the unexpected deletion of non-root storage service member volumes.

An unexpected deletion of a non-root storage service member volume is the deletion of that volume by any means other than a `storage-service-cleanup` API call. Deletion of non-root storage service member volumes by any other means, might cause disruption of operations related to the storage service in question.

Scenario

If a non-root storage service member volume undergoes an unexpected deletion (for example, by a Data ONTAP `volume delete` command), a transfer fail error results the next time a mirror or vault update operation on the storage service members is attempted.

Task flow

To rectify the error condition, Unified Manager must clean up the affected protection relationship in the affectedstorage service by first destroying or relinquishing all other protection artifacts associated with that protection relationship, then re-creating those protection artifacts for the original root member source volume.

The following task flow pin points and rectifies an unexpected deletion of a non-root storage service member volume:

1. To detect which member volume is missing in the storage service, the client application calls the following API:

   ```
   storage-service-member-iter
   ```

2. For each member volume listed in the storage service record, the Unified Manager server returns the following item:

   ```
   storage-service-member-info
   ```

Included in the `storage-service-member-info` item is the following information important to this task flow:

- The resource key for the storage service node on which each member volume in the storage service record is located
- The `is-member-deleted` flag set to true for the member volume that was deleted by non-Unified Manager means

3. To clean up the storage service, the client application issues the following API:

```
storage-service-cleanup
```

Included in the API call are the following important inputs:

- The `storage-service-nodes` item, which specifies the resource key of the storage service node on which the deleted volume is located
- The `keep-storage-artifacts` flag set either to false (to destroy the associated protection artifacts) or to true (to relinquish the associated protection artifacts)

As a result of the cleanup operation, the original source root member volume remains subscribed to the storage service but all protection artifacts associated with the root member volume are destroyed or relinquished.

4. To restore the protection relationship that was just cleaned up, the client application calls the following API:

```
storage-service-conform
```

When issuing this API call, the client application specifies the context string and object ID of the volume whose protection relationship failed.

5. The Unified Manager server starts the storage service conform job and returns a job identifier in the following element:

```
job-ID
```

6. The storage service conform job begins to check conformance for the specified root member source volume.

7. The client application starts a process loop to monitor the progress of the storage service conform job, checks the success or failure status of the completed job, and retrieves error messages if the job has failed.

8. If the conformance job finishes successfully, the deleted non-root volume and its protection relationship with the original root member volume are re-created.

Cleanup of unexpected deletion of a root storage service member

If a periodic check issued by the Unified Manager server discovers an unexpected deletion of a storage service root member volume, the Unified Manager server issues a `storageservice.unexpected.volume.deletion` event, which the client application can respond to with a `storage-service-cleanup` API call to destroy or relinquish the protection artifacts of the deleted storage service root member volume.

An unexpected deletion of a storage service root member volume is any deletion of that volume while it is still subscribed to a storage service. Any such deletion might disrupt operations related to that storage service.

Scenario

In this scenario, a root storage service member volume is accidentally deleted (for example, a Data ONTAP `volume delete` command).

Task flow

In the following task flow, the Unified Manager server discovers the occurrence of the unexpected root member volume deletion while updating cluster status and issues an event for which the client application can issue an operator-invoked storage service cleanup API request.

1. The Unified Manager server discovers the unexpected root member volume deletion, flags the root member volume as deleted, and issues the following event:

   ```
   storageservice.unexpected.volume.deletion
   ```

   The event source is the storage service resource key of the affected storage service.
   Included in the event is the resource key of the deleted volume and the resource key of the node on which the missing volume is located.

2. In response, the client application displays the event information to the storage administrator and presents an option to clean up the leftover protection artifacts.
   If the storage administrator selects that option, the client application issues the following API:

   ```
   storage-service-cleanup
   ```

   Included in the API call are the following important inputs:

   - The `keep-storage-artifacts` flag, set either to `true` or `false`:
     - `true` relinquishes the storage service membership of the protection artifacts associated with the deleted root member volume, but preserves the existence of those artifacts for possible restore of the root member volume in a future operation.
     - `false` destroys all protection artifacts associated with the deleted root member volume.

- The `members` item, which specifies the resource key of the missing root member volume to clean up
- The `storage-service-nodes` item, which specifies the resource key of the storage service node on which the protection artifacts of the missing root member volume are located

3. Unified Manager begins the storage service cleanup job and returns a job identifier using the following element:

```
job-id
```

4. The client application performs the following process loop to monitor the progress of the job:
   a. The application starts monitoring for completion of the job associated with the job ID by calling the following API:

```
job-wait-for-state
```

      When issuing this call, the client application specifies a timeout value.

   b. If the `job-wait-for-state` call times out before the job finishes, the client application optionally tracks the progress of the job by calling the following API:

```
job-task-iter
```

   c. This loop repeats until the job finishes.
   d. When the job has completed, the `job-wait-for-state` API returns the following element:

```
job-ID
```

5. The client application checks the success or failure of the completed job and retrieves any error messages by performing the task flow *Detection and troubleshooting of a failed job* on page 37.

As a result of the cleanup operation, all protection artifacts associated with the root member volume are either destroyed or removed as members from the affected storage service.

API guidelines

API guidelines are known issues or practices recommended to address known issues associated with the use of the Unified Manager APIs to implement storage service protection.

Storage service compatibility requirements

The storage service protection capabilities implemented by the Unified Manager API calls require that Data ONTAP 8.2 or later be installed on the cluster nodes that are to host the storage service member volumes.

Restriction on mirror-to-vault cascade protection

Unified Manager does not support a single volume being subscribed to more than one instance of a mirror-to-vault cascade topology.

The `storage-service-create` API prevents specifying more than one mirror-to-vault cascade in a single storage service. However, the API does not prevent a client application user from improperly subscribing the same volume to two different storage services that both use a mirror to vault cascade topology.

To address this issue, the best practice for the client application designer is to design the client application user interface in a way that forestalls the user from subscribing a volume to more than one storage service configured to provide mirror to vault protection.

Currency of storage service conformance checks

When the Unified Manager server executes a storage service conformance check, it checks for, and then attempts to fix, any conformance issues apparent in the most-recently refreshed cache of cluster configuration information.

When a storage service conformance check returns a SUCCESS status, it means only that the most-recently refreshed cache of cluster conformance information is conformant with storage service requirements. Changes to the cluster configuration since the last refresh are not accounted for in the conformance check.

By default, automatic refreshes of cluster configuration information occur every 15 minutes.

If the cache currency of the cluster information is in doubt, the client application can issue a call (`datasource-object-refresh`) to refresh the cached cluster information immediately before issuing the `storage-service-conform` API call.

Link insertion to failed job detail information

To enable a client application to link to information about a failed Unified Manager job, you can construct an HTTP link to the Unified Manager task details page for that job.

The syntax of the link is as follows:

```
https://<name_address_um_srvr>:8443
/#job-details:details-place-object-id=
<job-id_with_colons_replaced_by_%253A>
```

For example, if the IP address of the Unified Manager server is 10.229.155.49:8443, and the job-ID is bb9a5ff37d10cb4e:e2c16a8:13a8da5bf0d:5671, then the inserted link is as follows:

```
https://10.229.155.49:8443/
#job-details:details-place-object-id=
bb9a5ff37d10cb4e%253Ae2c16a8%253A13a8da5bf0d%253A5671
```

Storage service destination volume naming format

When Unified Manager provisions storage service destination volumes, it names those volumes according to a predefined formula.

The naming format of a destination volume that is provisioned through a storage service is *<StorageServiceName>_<SourceSvmName>_<SourceVolName>_<DestStorageServiceNodeName>*: for example, storageserviceA_svmA_volumeA_storageserviceAnodeB.

When the name of a provisioned destination volume matches the name of an existing volume in the destination , a suffix consisting of two underscores and four integers (__*<nnnn>*) is appended to the generated destination volume name to avoid a name collision condition: for example, storageserviceA_svmA_volumeA_storageserviceAnodeB__0002.

If the name of the matching generated destination volume is greater than 203 characters, that name is truncated to 197 characters and appended with the same suffix: __*<nnnn>*.

Glossary

The Unified Manager API glossary contains terms with specific meanings in the context of OnCommand Unified Manager documentation.

Glossary terms

It is important for you to understand terminology that is specific to Unified Manager protection APIs.

client application	An application that calls Unified Manager APIs to enable its operator to configure, monitor, and initiate data management operations to be executed on the Unified Manager server.
container object	An object, such as an aggregate or a Vserver, in which data objects reside.
data object	A container of data, such as a file, directory, volume, or LUN, that can be discovered, monitored, protected, created, or restored by the Unified Manager server.
destination	The storage to which source data is backed up, mirrored, or migrated.
protection artifact	An object, such as a destination data object, or a protection relationship that the Unified Manager server creates to support protection jobs when a data object is subscribed to a storage service.
protection relationship	The SnapMirror or SnapVault relationship that exists between a source data object and a destination data object.
resource pool	A collection of unused physical storage (such as storage systems or aggregates) from which new volumes or LUNs can be provisioned to contain data.
storage service	• In clustered systems, a preconfigured set of automated data management services, such as protection, that can be supplied to subscribed members. • In 7-Mode systems, a user-defined combination of a protection policy, provisioning policies, resource pools, and vFiler templates that you can assign as a package to a dataset or a class of datasets with common needs for protection, provisioning, and vFiler unit attachment.
member	Any data object that subscribes to or is created by a storage service.
root member	A data object that subscribes to a storage service.
volume	A file system.

Copyright information

Trademark information

How to send your comments

You can help us to improve the quality of our documentation by sending us your feedback.

Your feedback is important in helping us to provide the most accurate and high-quality information. If you have suggestions for improving this document, send us your comments by email to *doccomments@netapp.com*. To help us direct your comments to the correct division, include in the subject line the product name, version, and operating system.

You can also contact us in the following ways:

- NetApp, Inc., 495 East Java Drive, Sunnyvale, CA 94089 U.S.
- Telephone: +1 (408) 822-6000
- Fax: +1 (408) 822-4501
- Support telephone: +1 (888) 463-8277

Index

U

V

W