NexentaStor 5.3 High Performance Replication (HPR) User Guide

Date: March, 2020
Part Number: ns-5.3-HPRuserguide-RevA
## Contents

1 Introduction
   - About this Document
   - Definitions
   - Introducing High Performance Replication (HPR)
     - Replication Service Types
     - Supported Deployment Topologies
     - Supported Logical Configurations
     - Multi-Destination Configurations
     - Features and Target Use Cases
   - HPR Key Concepts
     - HPR and Kernel Remote Replication Protocol
     - Network Prerequisites
     - Replication Algorithm
   - About Service Manager and Service Agent on the Primary and Secondary Site
     - Schedules, Snaplists and Retention Policies
     - Aging-out Datasets/User Snapshot
     - HPR Transfer Auto-Resume
     - Operational Considerations for Scheduled Replication Services
     - Operational Consideration for Continuous Replication Services
     - Limitations
   - Comparing NexentaStor 4.0 Auto-Sync and NexentaStor 5 HPR

2 Preparing Networks and Appliances for HPR
   - Prerequisites for Configuring Replication Services
   - Licensing Requirements for the Replication Service
   - Setting and Updating Replication Password
   - Network Considerations and Configuration
     - Firewall Port Rules
   - Setting HPR Address to Send or Receive Replication Traffic
   - Prerequisites
Configure the Interface

3 Managing Replication Services

- About Scheduled Replication (SR)
- Local Snapshot
- Scheduled Snapshot
- Cloned Snapshot at Destination
- On-demand Snapshot
- Managing Scheduled Replication Services
  - Create New Replication Service
  - About CRON
  - Enable Service
  - Get List of Existing HPR Services
  - View Service Properties and its State
  - Verify Service Schedule and Service Snapshots
  - How to Stop a Scheduled Replication
  - Disable Replication Service
  - On-demand Snapshots and Replication
  - Destroy Replication Service
  - Modifying an Existing Schedule
- Replicating Clones and User Snapshots
- Enabling Auto-Rollback On a HPR Service to Seamlessly Recover from Child Snapshot Rollback on Source
  - Impact of Auto-Rollback on Promote Operation on the Source
- Managing Snapshots - Snaplists
  - List Snapshots of a Given Replication Service
  - List Snapshots after Deleting a Service
  - Delete List of snapshots
  - Claim Snapshots Belonging to an Existing Schedule
- About Continuous Replication
- Managing Continuous Replication Services
  - Create Replication Services
  - Enable Continuous Replication Service
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Get List of Existing HPR Services</td>
<td>41</td>
</tr>
<tr>
<td>View Service Properties and Service State</td>
<td>41</td>
</tr>
<tr>
<td>Disable Replication Service</td>
<td>42</td>
</tr>
<tr>
<td>Destroy Replication Service</td>
<td>42</td>
</tr>
<tr>
<td>Validating Running Replication</td>
<td>42</td>
</tr>
<tr>
<td>For a Continuous Replication Service</td>
<td>42</td>
</tr>
<tr>
<td>For a Scheduled Replication Service</td>
<td>43</td>
</tr>
<tr>
<td>Activating a Destination Dataset</td>
<td>43</td>
</tr>
<tr>
<td>Recovering Broken Replication Service</td>
<td>44</td>
</tr>
<tr>
<td>Recreating Lost Service Instance</td>
<td>44</td>
</tr>
<tr>
<td>Scenario 1: Lost Primary/Secondary node and Source/Destination Pool</td>
<td>44</td>
</tr>
<tr>
<td>Scenario 2: Lost Primary/Secondary node and Pool Intact</td>
<td>45</td>
</tr>
<tr>
<td>Managing Multi-Destination Replication Services</td>
<td>46</td>
</tr>
<tr>
<td>Running NDMP on the Destination of an HPR Service</td>
<td>48</td>
</tr>
<tr>
<td>How NDMP Works on the Destination Datasets</td>
<td>48</td>
</tr>
<tr>
<td>4 Disaster Recovery Use Cases</td>
<td>49</td>
</tr>
<tr>
<td>Disaster Recovery Test on Secondary Site</td>
<td>50</td>
</tr>
<tr>
<td>User Scenario</td>
<td>50</td>
</tr>
<tr>
<td>Action Plan for NexentaStor and NexentaFusion</td>
<td>50</td>
</tr>
<tr>
<td>Disaster Recovery by Creating Clones at the Destination Appliance</td>
<td>51</td>
</tr>
<tr>
<td>Different Possibilities of Managing the Snapshots</td>
<td>52</td>
</tr>
<tr>
<td>For Scheduled Replication Service</td>
<td>53</td>
</tr>
<tr>
<td>For Continuous Replication Service</td>
<td>57</td>
</tr>
<tr>
<td>Graceful Failover and Flip Direction</td>
<td>61</td>
</tr>
<tr>
<td>User Scenario</td>
<td>61</td>
</tr>
<tr>
<td>Action Plan for NexentaStor and NexentaFusion</td>
<td>61</td>
</tr>
<tr>
<td>Disaster Recovery with Failover and Failback</td>
<td>65</td>
</tr>
<tr>
<td>User Scenario</td>
<td>65</td>
</tr>
<tr>
<td>Action Plan for NexentaStor and NexentaFusion</td>
<td>65</td>
</tr>
<tr>
<td>Node Maintenance</td>
<td>69</td>
</tr>
<tr>
<td>Disaster Recovery from Total Loss of Primary Site</td>
<td>70</td>
</tr>
<tr>
<td>User Scenario</td>
<td>70</td>
</tr>
</tbody>
</table>
5 Advanced Configuration ........................................... 82
  Replication Service Options .................................. 82
    View History of Last Replications .............................. 82
    List all HPR Service Properties ............................... 83
    Modify HPR Service Properties .............................. 84
    List of Dataset Properties that can be Ignored or Replaced 85
    Configure Destination Dataset of an active HPR SR service as Read-only 86
    Disable HPR Service ........................................ 87
    List Services .............................................. 87
    Other HPR Schedule and Snaplist Management Commands 89

6 Troubleshooting .................................................. 91
  Recovering Broken Replication Service ....................... 91
  Automatic Handling Network Errors .......................... 91
  Handling Destination Space Quota Errors .................... 92
    For Scheduled Replication ................................ 92
    For Continuous Replication .................................. 92
  Handling Alerts .............................................. 92
  List of All Default HPR Standard Configurations .......... 93
  Adjusting Memory Available for Replication Services ...... 94

7 Appendix ................................................................ 95
  About Moving Data using Standard OpenZFS Send/Receive 95
  Terms Used in this Appendix ................................ 96
  Use Cases and Direction of Replication Using ZFS Send/Receive 96
<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managing sendrecv Job</td>
<td>96</td>
</tr>
<tr>
<td>Create Sendrecv Job</td>
<td>97</td>
</tr>
<tr>
<td>List Sendrecv Jobs</td>
<td>99</td>
</tr>
<tr>
<td>Modify Sendrecv Job</td>
<td>100</td>
</tr>
<tr>
<td>Create Manual Snapshots to Send</td>
<td>101</td>
</tr>
<tr>
<td>Run Sendrecv Job</td>
<td>101</td>
</tr>
<tr>
<td>Implications of Using non-root Credentials</td>
<td>103</td>
</tr>
<tr>
<td>View Status of Running Job</td>
<td>103</td>
</tr>
<tr>
<td>Stop a Sendrecv Job</td>
<td>104</td>
</tr>
<tr>
<td>Delete a Sendrecv Job</td>
<td>104</td>
</tr>
<tr>
<td>List of Options Available</td>
<td>105</td>
</tr>
<tr>
<td>Sendrecv Create Command</td>
<td>105</td>
</tr>
<tr>
<td>Sendrecv list Command</td>
<td>105</td>
</tr>
<tr>
<td>Sendrecv get Command</td>
<td>106</td>
</tr>
<tr>
<td>Sendrecv run Command</td>
<td>106</td>
</tr>
</tbody>
</table>
Preface

This documentation presents information specific to Nexenta products. The information is for reference purposes and is subject to change.

Intended Audience

This documentation is intended for Network Storage Administrators and assumes that you have experience with data storage concepts, such as NAS, SAN, NFS, and ZFS.

Documentation History

The following table lists the released revisions of this documentation.

Product Versions Applicable to this Documentation:

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ns-5.3-HPRuserguide-RevA</td>
<td>March, 2020</td>
<td>5.3 GA version</td>
</tr>
</tbody>
</table>

Contacting Support

Send your support questions and requests to support@nexenta.com.

Comments

Your comments and suggestions to improve this documentation are greatly appreciated. Send any feedback to doc.comments@nexenta.com and include the documentation title, number, and revision. Refer to specific pages, sections, and paragraphs whenever possible.
Introduction

This section includes the following topics:

- Definitions
- Introducing High Performance Replication (HPR)
- HPR Key Concepts
- About Service Manager and Service Agent on the Primary and Secondary Site
- Comparing NexentaStor 4.0 Auto-Sync and NexentaStor 5 HPR

About this Document

This document demonstrates how to configure High Performance Replication (HPR) to replicate datasets using the NexentaStor Command Line Interface (CLI). The NexentaStor High-Performance Replication (HPR) feature is an extension of the snapshot-based data protection.

NexentaStor supports virtually unlimited numbers of high-performance, space-efficient, instant snapshots and clones. It is possible to take individual snapshots of filesystems and volumes to get crash-consistent, point-in-time copies of their content.

We recommend that you use this document in conjunction with the suggested Additional Resources. This document is organized as follows:

- **Chapter 1: Introduction**
  This chapter introduces the basics of the HPR that you will use to protect datasets.

- **Chapter 2: Preparing Networks and Appliances for HPR**
  This chapter covers the details on preparing the networks and the NexentaStor appliances before configuring a HPR service.

- **Chapter 3: Managing Replication Services**
  This chapter demonstrates how to configure the two different types of HPR services - Scheduled and Continuous replication services and also describes how to manage the replication services you create.

- **Chapter 4: Disaster Recovery Use Cases**
  This chapter describes the advanced configuration options using NexentaStor CLI for HPR service that are to be used only as disaster recovery solutions.

- **Chapter 5: Advanced Configuration**
  This chapter lists all the options associated with the HPR service and how to manage them. It also describes the CLI commands to manage the service schedules and the snapshots.
Chapter 6: Appendix

This chapter covers the details of sendrecv facility NexentaStor offers to send and receive datasets to and from any OpenZFS based systems using the standard ZFS send/receive facilities over ssh.

Definitions

This document uses the following terms.

<table>
<thead>
<tr>
<th>Terms</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Replication group</td>
<td>Replication group is a set of NexentaStor appliances that share the same Replication Password.</td>
</tr>
<tr>
<td>Scheduled replication (SR)</td>
<td>Scheduled replication is the default type of replication service that comes with the Enterprise Edition license.</td>
</tr>
<tr>
<td></td>
<td>SR generates snapshots of the datasets at the source NexentaStor appliance on a set schedule and replicates them to destination datasets, located either on the same appliance, or on a remote NexentaStor appliance.</td>
</tr>
<tr>
<td>Continuous replication (CR)</td>
<td>Continuous replication asynchronously sends every write transaction to the destination dataset and delivers as close to zero Recovery Point Objective as possible over any distance, without affecting application performance.</td>
</tr>
<tr>
<td>Source</td>
<td>Source represents the node where the data is replicated from.</td>
</tr>
<tr>
<td>Destination</td>
<td>Destination represents the site where the data is replicated to and can be either a local dataset on the same appliance or a remote NexentaStor appliance.</td>
</tr>
<tr>
<td>Primary</td>
<td>Primary represents the node where the replication service manager is running. See below for more details about the Service Manager. So the primary appliance can be either a source or destination.</td>
</tr>
<tr>
<td></td>
<td>See About Service Manager and Service Agent on the Primary and Secondary Site for more details on the Manager and Agent.</td>
</tr>
<tr>
<td>Secondary</td>
<td>Secondary represents where the replication service agent is running. See below for more details about the Service agent. The Secondary appliance can be either a source or destination.</td>
</tr>
<tr>
<td>Local to Remote replication</td>
<td>Replication from Manager to Agent node.</td>
</tr>
<tr>
<td>Remote to Local replication</td>
<td>Replication from Agent to Manager node.</td>
</tr>
</tbody>
</table>

Primary appliance (replication manager) Secondary appliance (replication agent)
Source dataset --------------------------> destination dataset
Destination dataset <------------------------ source dataset
Introducing High Performance Replication (HPR)

HPR stands for High Performance Replication, the new long distance replication service in NexentaStor 5. HPR fully replaces and is not compatible with the AutoSync replication functionality of previous versions of NexentaStor. It effectively is a full rewrite of that solution, at both the management level and the data transfer protocol level.

You can configure HPR to replicate datasets as follows:

• Between two datasets in the same pool or different pools on the same NexentaStor appliance.

• From a dataset on one NexentaStor appliance to another dataset on another NexentaStor appliance. In this case, the local appliance is known as the Primary site where the replication service manager runs and the remote is known as the Secondary site where the replication service agent runs. See About Service Manager and Service Agent on the Primary and Secondary Site for more details on the Manager and Agent.

Replication Service Types

HPR supports two different replication services:

• Scheduled Replication (SR) - enabled by default with Enterprise Edition license, with a snapshot schedule of “every 15 minutes” or larger. This replication service replicates snapshots taken on pre-defined schedules on the source dataset. With the CR license option, the snapshot schedule can be as tight as “every minute”.

• Continuous Replication (CR) - requires the CR license option. This replication service delivers close to zero RPO over any distance without affecting application performance. It works by asynchronously replicating every write transaction on the source dataset.

To avail of the CR option, contact sales@nexenta.com.

To verify if the CR service is activated, run the following command:

```
CLI@nexenta> license show
```

<table>
<thead>
<tr>
<th>Note:</th>
</tr>
</thead>
<tbody>
<tr>
<td>You can configure only one CR service per dataset, whereas you can configure more than one SR service.</td>
</tr>
<tr>
<td>NexentaStor HPR also supports multi-destination configurations by allowing multiple SR and single CR services to be configured on the same Primary site dataset.</td>
</tr>
</tbody>
</table>

To see the limitations on the operations for CR or SR service, see Operational Consideration for Continuous Replication Services and Operational Considerations for Scheduled Replication Services.
Supported Deployment Topologies

HPR can be configured in the following topologies. All appliances used with HPR must be running NexentaStor 5.

- Local replication within single node
- Single node to Single node
- Single node to HA cluster
- HA cluster to HA cluster
- HA cluster to Single node.

Figure 1-1: Most Supported HPR Deployment Topologies

<table>
<thead>
<tr>
<th>Single node to Single node</th>
<th>HA cluster to HA cluster</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single node to HA cluster</td>
<td>HA cluster to Single node</td>
</tr>
</tbody>
</table>

**Note:** Replication from/to a cluster built with nodes with different NexentaStor 5.x versions is not supported.

Supported Logical Configurations

HPR can be configured on:

- A file system- Recursive or Non-recursive
  
  Recursive on a parent file system with nested children file systems will replicate the parent and all the underlying file systems, with transaction level consistency.

- A volume group
  
  Recursive on volume groups (i.e. replicate all the underlying volumes, with transaction level consistency).

- A volume
HPR – Supported Logical Configurations

File System to File System

PoolA/fS -> PoolB/fS

Volume to Volume

PoolA/vg/vol -> PoolB/vg/vol

Recursive file systems to Recursive file systems

PoolA/fS
PoolA/fS/child_1
... PoolA/fS/child_n

PoolB/fS
PoolB/fS/child_1
... PoolB/fS/child_n

Volume group to Volume group

PoolA/vg
PoolA/vg/vol_1
... PoolA/vg/vol n

PoolB/vg
PoolB/vg/vol_1
... PoolB/vg/vol n
Multi-Destination Configurations

NexentaStor HPR also supports multi-destination configurations by allowing multiple SR and single CR services to be configured on the same Primary site dataset. HPR does not support cascaded replication, however, the state of the various SR services that share a common Primary dataset must be consistent with that limitation. Destination datasets can be located either on the local or remote appliance. Each service replicates only the snapshots created by it. When creating multiple services from the same source dataset, you can set unique retention policies for each destination site. Each replication service manages its own schedules, snaplists independently. You can also create multiple schedules per service and each schedule can have its own retention policy.

Figure 1-2: Example of Multiple HPR services to Multi-Destinations
If one of the destination appliances goes down, the corresponding HPR service fails. However, HPR services to other destination appliances will continue to run unaffected. And when the broken destination appliance comes back, the failed HPR service has to be restarted to continue to replicate to all the destination appliances.

Note: In multi-destination configurations, special care is required during failback scenarios. For example, when replicating back data to the primary site, only the service that is transferring data back to primary can be enabled and all other services should be disabled.

In case of multi-destination replication services, snapshots created by one of the services become the intermediate snapshots for the other service(s). For example, in multi-destination replication with two services, service-ab and service-ac, snapshots created by service-ab are intermediate snapshots for service-ac and vice versa.

Features and Target Use Cases

High Performance Replication allows simple transfer of data from any dataset or nested set of datasets to a remote NexentaStor pool. The functionality can be used for three broad categories of use cases:

- **Data migration:** in this use case, the content of datasets can be moved to a different pool on the same appliance, or to a different NexentaStor appliance. This can be accomplished in 2 steps. A long running first step that can operate without disrupting applications on the source datasets using SR. A second step where the applications at source are stopped, the last source dataset snapshot is quickly transferred and the applications are restarted on the destination dataset.

- **Remote data archive:** in this use case, snapshots of data on the primary site are replicated to a remote site for long term archiving. This can be accomplished using SR with a combination of schedules and retention policies that accomplish the desired archival policy. For example, using just 3 schedules with their respective retention policies in a SR service, a user can simply archive the last 12 hourly snapshots, the last 7 daily snapshots and the last 52 weekly snapshots of a given primary dataset.

- **Disaster recovery of applications:** in this use case, a primary dataset can be replicated with a tight Recovery Point Objective to a secondary site. With a standard Enterprise Edition license, RPO can be as low as 15 minutes. With the CR license option, SR services can be run with RPO as low as 1 minute. Or using continuous asynchronous replication services, it is possible to provide close to zero RPO without affecting the performance of the application on the primary site.

For applications that require maintaining transaction level consistency across multiple datasets (like complex databases and enterprise applications), HPR supports recursively replicating nested file systems, or all the volumes in a volume group, while ensuring write transaction consistency for all the underlying datasets.
When replicating nested file systems and volume groups, HPR maintains consistency as shown below:

For scenarios that require more complex data retention policies, a given HPR service can be configured with multiple schedules, each with its own set of local and remote retention policies.

Creating clones out of HPR snapshots on the destination dataset is supported and can be used as part of Disaster Recovery Test operations. While a clone exists on an HPR service snapshot, that snapshot is implicitly protected from all retention policy operations and will not be deleted by the HPR service.

See Disaster Recovery Test on Secondary Site for more information on creating clones on the destination dataset.
HPR Key Concepts

HPR and Kernel Remote Replication Protocol

HPR is only supported between NexentaStor 5 appliances. High Performance Replication consists of two high level components:

- a user space component (hpr worker) that provides the end-user service logic, schedule and snapshot management; and

- a kernel module that implements a Kernel based Remote Replication Protocol (krrp) that handles all data transfers between source and target datasets. KRRP enables support of continuous asynchronous replication services and near zero RPO. It also includes a number of reliability enhancements, such as the ability to automatically resume data transfers following an unexpected interruption.

Network Prerequisites

A standard HPR configuration relies on two separate networks, one supporting the service management plane where HPR control messages are issued, and the other supporting the service data plane where actual data transfers between source and destination take place.

The service data plane uses the interfaces configured in `hpr.dataAddresses` on each node supporting the HPR service.

The service management plane on the other hand relies on each node's management interfaces. As such, a HPR service is created by specifying a remote endpoint using its management address, resolvable hostname or Fully Qualified Domain Name (FQDN). When a HPR service is configured between two NexentaStor HA clusters (e.g. nodes A1-A2 and B1-B2), HPR automatically discovers all possible endpoints for the service using each node’s configured `system.managementAddress`. A HPR service uses that information to automatically re-establish its control sessions following cluster failover events. For example, an HPR service created between A1 and B1 will automatically discover the management endpoints for nodes A2 and B2 by pulling from their configured `system.managementAddress`. Best practice is to explicitly set the management address on all nodes supporting HPR services and avoid running HPR between nodes with `system.managementAddress` set to 0.0.0.0.

Note1: Setting `system.managementAddress` to 0.0.0.0 has historically been a common way to enable management access over all IP interfaces of a NexentaStor node. With NexentaStor 5.2.1 and above, the preferred method to accomplish this is to set `system.managementAddress` to a specific IP address and configure additional management API endpoints in `rest.managementAddress` (See NexentaStor 5.x CLI Configuration Guide for more details on Configuring Additional REST Management IP Addresses).

Note2: If in spite of the above recommendation, you have to create an HPR service between HA clusters where nodes have `system.managementAddress` set to 0.0.0.0, you must use resolvable hostnames or FQDNs to specify the HPR remote endpoint. The HPR service will then automatically discover the configured `system.hostName` and `system.domainName` of all nodes in the HA clusters and use those to re-establish sessions following cluster failover events. An HPR service relying on hostnames or FQDNs depends on all these names being reliably resolvable in order to run.
Replication Algorithm

You can configure HPR to replicate datasets as follows:

- Between two different datasets in the same pool or different pools on the same NexentaStor appliance.
- From a dataset on one NexentaStor appliance to another dataset on another NexentaStor appliance. In this case, the local appliance is known as the Primary site and the remote is known as the Secondary site. See About Service Manager and Service Agent on the Primary and Secondary Site for more details.

An HPR service is defined by:

- A type: Scheduled or Continuous
- A primary dataset
- A secondary dataset
- If the service is a SR service, one or more schedules each with their local and remote retention policy.

At service creation, the primary dataset is generally considered the source of replication, while the secondary dataset is the destination.

For any HPR service to work properly, it requires that datasets on both ends of the service “match”. In the case of a SR service, this simply means that both ends of the service must have the same “last transferred” snapshot. If HPR detects that a destination dataset was changed (by the user or otherwise) since data was last transferred to it, the service will go to a faulted state and require user intervention.

HPR does not implement hard limitations on the type of dataset operations (share configurations, snapshot operations, etc.) that can be performed on either source or destination datasets. This flexibility can be useful in disaster recovery situations. This flexibility also creates risks for the user to inadvertently make changes that will cause an HPR service to fail.
The replication service replication algorithm includes the following tasks:

For SR service:

1. Creates a snapshot at Primary appliance.
   The replication service executes according to the schedule. On every run, it creates a snapshot of the selected dataset.

2. Starts replication only if previous run is finished.

3. If a scheduled replication is already running, HPR created snapshots will be replicated on the next scheduled or manual or on-demand run.

4. Before the run, HPR determines the latest identical snapshots at the Primary appliance and Secondary appliance.
   The replication service compares the lists of snapshots at the Primary appliance and at Secondary appliance and locates a pair of the latest identical snapshots. If snapshots do not match in case of a manual (on-demand) run, service will fail to start and in case of scheduled run, service will go the faulted state.

5. Sends the incremental stream or initial stream to the destination.
   By comparing the set of snapshots on the source and the destination site, the replication service identifies if it needs to send a snapshot from the source to destination. Instead of copying the new snapshot from the source site, the service generates and transfers only the changes from the source site snapshot to the destination site snapshot.
   For the initial replication, a full snapshot is replicated to the destination.

For CR Service:

For a CR service, replication creates snapshots called autosnapshots every time a source change is detected. These autosnapshots are created by the kernel subsystem for each ZFS transaction group (txg). They are invisible to the user in the list of dataset snapshots and are continuously replicated to the destination. For the initial replication, a full snapshot is replicated to the destination.
About Service Manager and Service Agent on the Primary and Secondary Site

The remote replication service consists of two instances called manager service and agent service. In the case of local-to-remote replication, the primary appliance acts as the service manager node and will have the property “Manager” set to “Yes” and the secondary appliance acts as the service agent and will have the property “Manager” set to “No”. In case of remote-to-local replication the Manager and Agent roles are switched.

The agent service supports only service disable or destroy operations.

You cannot manage a replication service on an agent. However, you can destroy or disable the service using the Force flag.

- To list the existing replication services:
  ```
  CLI@nexenta> hpr list
  ```

- To identify if the current node is a manager or an agent, use the following command:
  ```
  CLI@nexenta> hpr get isManager <hpr service name>
  ```

The node that has the property “isManager” set to Yes has the HPR service running on it and is addressed as the Primary appliance.

```
CLI@nexenta> hpr get isManager service1
NAME   PROPERTY      VALUE
service1 isManager  yes
```
Schedules, Snaplists and Retention Policies

Once a SR service is created, you can add one or more schedules to it. For example, an SR could have 2 schedules, one to replicate hourly snapshots and keep 3 on the primary and 24 on the secondary, and another to replicate daily snapshots and keep 1 on the primary and 7 on the secondary, as shown in the diagram below.

Every snapshot created and replicated by a schedule belongs to schedule specific snaplists, one at each end of a replication service. When creating a schedule, you must define retention policies that control the maximum of numbers of snapshots that each snaplist can contain. For example, an hourly schedule with a local retention policy of “keep 6” and a remote retention policy of “keep 24” will keep 6 hours worth of snapshot on the primary site and a day’s worth of hourly snapshots on the secondary site.

Snaplists and the retention policies that apply to them are specific to a site, and independent of the direction of replication. For example, a SR configured to replicate data from A to B can be configured with a daily schedule, keep 2 on A and keep 7 on B. If the direction of data transfer is flipped to “B to A”, the service will continue to keep at most 2 snapshots on A and 7 snapshots on B.

If a user creates a clone from a snapshot in an HPR snaplist, that snapshot is automatically excluded from deletion operations by the schedule retention policies until the clone is deleted.

When an HPR service or a replication schedule is destroyed, snaplists are by default left behind. This provides the user full control on the appropriate handling of the data in the snapshots. More importantly, an existing snaplist can be re-attached to another schedule, allowing the user to reconfigure HPR services or recreate HPR services from scratch while retaining pre-existing snapshots.
When a snaplist is re-attached to a given schedule, the retention policy of that schedule will apply. For example, if a snaplist that contains 30 snapshots is attached to a schedule with a local “keep 10” policy, that snaplist will get trimmed to its latest 10 snapshots.

**Aging-out Datasets/User Snapshot**

If a source child dataset or a user snapshot is destroyed, it will be renamed at the destination. This aged out dataset or a user snapshot will be kept at the destination based on the time period configured using `hpr.agingOutKeepTime`. The default value to retain the dataset/user snapshot at the destination is 10 days.

1. View the default value.

```plaintext
CLI@nexenta> config get value hpr.agingOutKeepTime
NAME                  PROPERTY  VALUE
hpr.agingOutKeepTime  value     14400
```

2. As per your requirement, update the time period to retain the dataset/user snapshot at the destination that is deleted on the source.

```plaintext
CLI@nexenta> config set hpr.agingOutKeepTime=28800
NAME                  PROPERTY  VALUE
hpr.agingOutKeepTime  value     28800
```

---

**Note:**

To retain the user snapshot using the above described steps, you must have enabled the "sendUserSnapshots" option.

1. View the `sendUserSnapshots` for the service you created.

```plaintext
CLI@node> hpr get sendUserSnapshots my_repl1
NAME     PROPERTY           VALUE
fssrtsr  sendUserSnapshots  no
```

2. Enable the `sendUserSnapshots` for the service `my_repl1`.

```plaintext
CLI@node> hpr set sendUserSnapshots=true my_repl1
```

---

**Note:**

To reverse replication direction, all aging out datasets must either be deleted or renamed to be without aging-out postfix.

1. **Ensure no clones exist for aging-out datasets.**
For Filesystems:

CLI@nexenta> fs list -ro path,origin -O basic pool/fs|grep -v 'aging-out-[0-9]*[@[:space:]]' | grep -e 'aging-out-[0-9]*@'

For Volumes:

CLI@nexenta> volume list -o path,origin -O basic pool/volumegroup|grep -v 'aging-out-[0-9]*[@[:space:]]' | grep -e 'aging-out-[0-9]*@'

If there are no clones for these datasets the above commands will return empty output.

2. Identify aging-out datasets.

For Filesystems:

CLI@nexenta> filesystem list -r path/to/destination/dataset|grep aging-out

test/data1_backup/fs1-aging-out1561671767  23K   5.58G  23K    no   no

For Volumes:

CLI@nexenta> volume list path/to/destination/dataset|grep aging-out

pool1/vg1/vol1-aging-out-1561671999  100M     103M  5.68G  12K    32K

3. If you have enabled the `sensUserSnapshots` option, identify the aging-out snapshots.

CLI@nexenta> snapshot list -r test/aa | grep aging-out

pool1/data_backup@mysnap1-aging-out-1561670013  filesystem  0K    23K

4. Destroy all aging-out datasets.

for i in $(fs list -r -o path -O basic pool/fs|grep aging-out); fs destroy -Ry $i

for i in $(volume list -r -o path -O basic pool/vg|grep aging-out); volume destroy -Ry $i

for i in $(snapshot list -r -o path -O basic path/fs|grep aging-out); snapshot destroy -Ry $i

This will destroy all aging-out datasets with all their clones without warning.

**HPR Transfer Auto-Resume**

When replicating large amounts of data, it is not unusual to see transfers interrupted by intermittent network outages. HPR supports “auto-resume” functionality, allowing it to seamlessly handle such interruptions and resume a data transfer from the point where it was stopped without having to re-transfer the entire snapshot from scratch. For example, when replicating a 10 Terabyte snapshot, if the connection dropped after 8 Terabytes were transferred, HPR will automatically resume data transfer where it left off and only have to transfer the remaining 2 Terabytes to complete the operation.
Operational Considerations for Scheduled Replication Services

For multi-destination configuration, it is possible to configure multiple SR services on a given source dataset, each with its own target dataset. In this case, each SR service manages its schedules, snaplists and retention policies independently. Care must be taken when considering flip direction on an SR service in a multi-destination setup: flip direction can only be done on one service, and requires that all other services be disabled during the entire time the service is running in the backward direction.

Within context of a SR service, the user may create an on-demand snapshot and have it immediately replicated to the other site, independent of replication service schedules. These snapshots are not managed by the service schedules and their associated retention policies. It is up to the user to manually delete these on-demand snapshots.

File System / Volume - Scheduled Replication

When SR is configured from a simple source dataset (file system or volume) DS_A to a destination dataset DS_B consider the following factors.

- Do not access or modify the destination dataset DS_B.
- No local snapshot service may be configured on DS_B.
- Creating manual snapshots on the destination dataset is not supported while service is running.
- If the content of DS_B is modified, the SR service will fail. You may recover the SR service using the recover operation. This will cause the SR service to recover from the last common snapshot between DS_A and DS_B.
- Multiple schedules with their own retention policies can be created as part of the SR service on DS_A.
- Local snapshot service and local snapshot schedules can be configured on DS_A. These local snapshots are not replicated to DS_B. Only snapshots created within the context of an SR service on DS_A will get replicated.
- Deleting and renaming snapshots from SR snaplists is possible on either side of the service as long as no modification is done on the “last common snapshot” / “most recent common snapshot”.
- While the SR service is running, it is possible to create a clone from any snapshot in a scheduled snaplist on DS_B. That clone can subsequently be shared via file or block protocols (as appropriate) and used for Disaster Recovery Testing for example. The parent snapshot for this clone is protected from deletion by the SR schedule retention policy on DS_B as long as the clone exists. Once the clone is destroyed, the snapshot is automatically put back in its original SR schedule snaplist and will eventually be deleted per the SR schedule retention policy.
- Rollback of a local snapshot on DS_A is not supported while SR is enabled. If you need to rollback a local snapshot on DS_A, you must first disable SR, rollback the local snapshot, and then use the CLI command hpr recover to restart SR. This will cause deletion of snapshots on DS_B that are newer than the content of DS_A. Another option is to enable auto-Rollback option. Once it's enabled, manual recover operation will not be required on the destination since HPR will automatically rollback destination snapshots before starting a replication as they were rolled back on the source.
- DS_A may not be renamed.
Nested File System / Volume Group - Recursive Scheduled Replication

When recursive SR is configured from a volume group or nested file systems DS_A to a datasets DS_B consider the following factors.

- The above limitations apply.
- Newly created child file systems in DS_A (volumes if DS_A is a volume group) will be automatically replicated as child file systems (volumes, respectively) of DS_B.

Renaming of child file systems in DS_A (or volumes if DS_A is a volume group) is not supported while SR is running. To rename, the user must first disable SR, rename source child dataset, and enable the service again.

Deleting of child file systems in DS_A (or volumes if DS_A is a volume group) is not supported while SR is running. To delete, the user must first disable SR, then delete child dataset, and its corresponding child dataset on DS_B.

---

**Note:** Child datasets on the destination should not be renamed. If a child dataset is renamed, the service will fail and can only be recovered using the `hpr recover` command, causing the full child dataset to be replicated from scratch.

---

Operational Consideration for Continuous Replication Services

At most one CR service can be configured on a source dataset. For multi-destination configurations, it is possible to configure one CR service and one or more SR services on the same source dataset.

File System/Volume - Continuous Replication

When CR is configured from a source dataset (file system or volume) DS_A to a destination dataset DS_B consider the following factors listed below:

- Do not access or modify the destination dataset DS_B
- No local snapshot service may be created on DS_B
- Creating manual snapshots on the destination dataset is not supported while service is enabled
- If the content of DS_B is modified, the CR service will fail. You may recover the CR service using the recover operation. This will cause the CR service to recover replication from the last common autosnapshot between DS_A and DS_B
- DS_A on the other hand is available for sharing and application access
- Local snapshots and local snapshot schedules can be configured on DS_A, independent of the CR service
- Rollback of a local snapshot on DS_A is not supported while CR is enabled. If rollback of a local snapshot on DS_A is performed while CR is disabled, it is most likely that replication will have to be restarted from scratch, with a full transfer of the DS_A to site B
- Replication of user snapshots and clones is not supported.
- DS_A may not be renamed
Nested File system/Volume Group- Recursive Continuous Replication

When recursive CR is configured from a volume group or nested file systems DS_A to a datasets DS_B consider the following factors listed below.

- The above limitations apply
- Newly created child file systems in DS_A (volumes if DS_A is a volume group) will be automatically replicated as child file systems (volumes, respectively) of DS_B
- Renaming of child file systems in DS_A (or volumes if DS_A is a volume group) is not supported while CR is running. To rename, the user must first disable SR, rename source child dataset, and enable the service again.
- Deleting of child file systems in DS_A (or volumes if DS_A is a volume group) is not supported while CR is running. To delete, the user must first disable SR, then delete child dataset, and its corresponding child dataset on DS_B.

Limitations

- HPR can be configured only between NexentaStor 5 appliances.
- HPR cannot be configured on a pool. It can be configured recursive on a parent file system and recursive on volume groups.
- HPR does not replicate sharing properties such as LUN mapping and smb share permissions. Replication of NFS share permissions is disabled by default, use sendShareNfs option to replicate NFS share permissions to the destination filesystem.
- HPR does not support ZFS Send/Receive.
- KRRP does not support configuring an IPv6 as hpr.dataAddress to send or receive replication data.
- HPR does not support replication over multiple network interfaces.
- HPR does not support replication of snapshots created by another replication or snapping service.
- HPR supports creating remote to local replication only using the CLI and API. However, you can create local to local and local to remote replication using NexentaFusion.
- Promote operation for clones of any snapshot in an HPR snaplist is not supported. For example, a promote operation of a clone of the latest replicated snapshot parent/src@latest would move that snapshot from the HPR source parent/src to parent/src-promoted. The next time replication runs, it will not find the latest replicated snapshot on parent/src and replication will fail. To recover from such a failure, you would need to:
  - Promote parent/src to implicitly re-attach parent/src@latest to it.
  - Run the “hpr recover” command to recover the replication service.
- HPR does not support creating manual snapshots on the destination dataset while service is enabled.
- For any HPR service (Scheduled or Continuous), all file systems must be mounted with their default mount points. Mounting file systems with custom mount points will cause HPR services to go to a faulted state and isn’t supported.
- The maximum number of supported SR services on a NexentaStor 5 appliance (single node or HA cluster) is 50.
- The maximum number of supported CR services on a NexentaStor 5 appliance (single node or HA cluster) is 15.
- HPR does not support Cascade replication configurations. A dataset can either be the source or the destination of an HPR service. It cannot be both at the same time.
Comparing NexentaStor 4.0 Auto-Sync and NexentaStor 5 HPR

NexentaStor 5 HPR is a complete redesign of the NexentaStor remote data replication facility. It is not compatible with NexentaStor 4.0 Auto-Sync and requires that all nodes involved in a service run NexentaStor 5.

From a functional perspective, special attention was paid to simplifying management of typical replication configurations. For example, configuring HPR services between NexentaStor HA clusters does not require much more effort than configuring HPR between single node appliances.

HPR also adds a number of highly desirable new features:

- Simple, straightforward management of services, schedules and retention policies
- Simple management of snapshots in replication schedule snaplists
- Clean separation and independent management of snapshots from replication services and snapshots from local schedules on source datasets
- Auto-resume facility, allowing replication to simply pick-up where it left off following network connection interruptions
- Support for volumes / LUN consistency groups to provide cross volume transaction consistency of replicated volume groups
- HPR supports recursive replication of nested file systems and volume group level to simplify protection of large number of datasets
- More aggressive SR schedules supporting replication as often as “every minute”
- New continuous asynchronous replication facility providing close to zero RPO without affecting application performance
- Simplified disaster recovery failover & failback procedures

The main limitation of HPR compared to NexentaStor 4.0 Auto-Sync is the lack of support for pool level replication. Instead, HPR supports recursive replication of nested file systems and volume group level replication to simplify protection of large numbers of datasets.
Preparing Networks and Appliances for HPR

This section includes the following topics:

- Licensing Requirements for the Replication Service
- Setting and Updating Replication Password
- Network Considerations and Configuration
- Setting HPR Address to Send or Receive Replication Traffic

Prerequisites for Configuring Replication Services

To configure replication services on the NexentaStor storage appliances, you must have done the following as prerequisites:

Using the NexentaStor CLI

- Activate a license key with the optional CR feature.
- Set replication password for all nodes in the replication group.
- Configure the replication data network interface on each node with a static IP interface, one 10 Gigabit Ethernet with Jumbo Frames. It is generally recommended to also leverage IPMP or LACP link aggregation.
- If needed, add the HPR port as exceptions when configuring the firewall so that firewall does not block all connections to a NexentaStor appliance that uses the HPR port.
- Configure hpr.dataAddress to point to the IP address of network replication interface on each node in source and destination appliance. Depending on the direction of the replication, the interface that is configured as data address will begin to send or receive replication data.
- Ensure that the system management address is set to an explicit IP address. If you need management and REST API access over more than one interface, you should set the primary management address using system.managementAddress and configure additional addresses using rest.managementAddress.

See NexentaStor 5.2 CLI Configuration Guide for more details on Configuring Additional REST Management IP Addresses.

Note: If you set system.managementAddress to 0.0.0.0, you should make sure that system.hostName and system.domainName are configured on all nodes supporting HPR services and you create your HPR services using either hostnames or FQDN when specifying the remote endpoint.
Using the NexentaFusion UI

- Register the NexentaStor appliances to which the protection services must be applied.
- Verify if the CR license is activated if you need to configure CR service.
- If you have not configured the data address using the CLI, configure it using the NexentaFusion UI.

See NexentaFusion 1.2 User Guide to accomplish the tasks listed under Using the NexentaFusion UI

Licensing Requirements for the Replication Service

A NexentaStor Enterprise Edition license is required to create High Performance Replication services on a NexentaStor appliance. With the Enterprise Edition license, you can create Scheduled Replication (SR) services and schedules as tight as “every 15 minutes”. To further reduce the snapshot interval and achieve tighter Recovery Point Objectives (RPO), you can acquire the optional continuous replication (CR) option.

The CR feature license allows SR services with snapshot every minute, as well as continuous asynchronous replication services, allowing close to zero Recovery Point Objective (RPO) over any distance, without affecting application performance.

To verify if the CR service is activated, run the following command. Look for the `continuousReplication` feature in the “features” property.

```
CLI@nexenta> license show
PROPERTY     VALUE
guid         44454c4c-3600-104b-804c-b9c04f4e3232
valid        yes
status       ok
type         ENTERPRISE-TRIAL(Nexenta Internal)
product      NexentaStor
version      5.2
licensee     Nexenta-xxxxx@nexenta.com
serial       SR-DEV-NS-201617669
features     allFlash, fibrechannel, highAvailability, continuousReplication, scheduledReplication
issued       Thu Sep 29 17:00:00 2016
expires      Sun Nov 13 16:00:00 2017
capacity     no limit
maintenance  Thu Sep 29 17:00:00 2016 - Sun Nov 13 16:00:00 2017 (valid)
```

To activate a license key with the optional CR feature:

```
CLI@nexenta> license activate <key>
```
Setting and Updating Replication Password

Before configuring replication services between NexentaStor appliances, you must ensure that they are part of the same replication group. A replication group is defined as a set of NexentaStor appliances that share the same Replication Password. The password must be the same for all nodes in the replication group for the data protection feature to work.

The replication password provides a simple, non intrusive level of authentication that protects NexentaStor appliances from being configured as the replication target of systems that are not in the same replication group. During the NexentaStor installation, you were prompted and then entered a replication password. Should you want to change this after installation, use the following command:

```bash
CLI@nexenta> hpr password-set [--password=<str>]
```
Network Considerations and Configuration

To run HPR on a NexentaStor appliance, the Replication Data Network Interface on each node must be configured with a static IP interface, one 10 Gigabit Ethernet with Jumbo Frames. It is generally recommended to also leverage IPMP or LACP link aggregation.
Firewall Port Rules

NexentaStor 5 appliance uses the following ports. In case your firewall blocks any of the services offered by NexentaStor, use this table as a guidance to add the following ports as exceptions when configuring your firewall. Adding the follow ports will allow connections to the corresponding NexentaStor services.

HPR uses TCP 6000.

- **To change HPR port from the default value 6000:**
  
  CLI@nexenta> config set hpr.dataPort = <value>

---

### Table 2-1: List of Network Ports for Firewall Exception Rules

<table>
<thead>
<tr>
<th>Interface</th>
<th>Required and Recommended</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incoming to NexentaStor Management Interface</td>
<td>TCP ports 22 (SSH), 6557, 5557, 8443 (REST API)</td>
</tr>
<tr>
<td>Outgoing from NexentaStor appliance</td>
<td>SNMP (161, 162), DNS (53 UDP/TCP), NTP (UDP 123), ICAP (port 1344)</td>
</tr>
<tr>
<td>Outgoing to Internet</td>
<td><a href="https://licensingservice.nexenta.com/">https://licensingservice.nexenta.com/</a> (443) (ssl)</td>
</tr>
<tr>
<td></td>
<td><a href="https://logcollector.nexenta.com">https://logcollector.nexenta.com</a> (443/20/21/7000-8000) (ssl/ftp)</td>
</tr>
<tr>
<td></td>
<td><a href="https://prodpkg.nexenta.com">https://prodpkg.nexenta.com</a> (443) (ssl)</td>
</tr>
<tr>
<td>Support Bundles</td>
<td>TCP port 21 (FTP)</td>
</tr>
<tr>
<td>Outgoing to the mail server</td>
<td>TCP port 25 or 587 for relay only (SMTP)</td>
</tr>
<tr>
<td>Between the interfaces with the hpr.dataAddress for Replication Data Protocol</td>
<td>TCP port 6000 by default but can be changed</td>
</tr>
<tr>
<td>Between NexentaStor Management Interfaces of the clustered nodes</td>
<td>UDP and TCP port 1195</td>
</tr>
<tr>
<td>For data traffic on any data interfaces:</td>
<td></td>
</tr>
<tr>
<td>iSCSI Target</td>
<td>TCP port 3260</td>
</tr>
<tr>
<td>SMB</td>
<td>UDP ports: 137, 138</td>
</tr>
<tr>
<td></td>
<td>TCP ports 137, 139</td>
</tr>
<tr>
<td>CIFS</td>
<td>UDP and TCP port 445</td>
</tr>
<tr>
<td>NTP</td>
<td>UDP port 123</td>
</tr>
<tr>
<td>NFSv3</td>
<td>UDP ports: 111, 2049, 4045, 32768:65535</td>
</tr>
<tr>
<td></td>
<td>TCP ports: 111, 2049, 4045, 32768:65535</td>
</tr>
<tr>
<td>NFSv4</td>
<td>TCP port 2049</td>
</tr>
</tbody>
</table>
Setting HPR Address to Send or Receive Replication Traffic

Prerequisites

Configure hpr.dataAddress to point to the IP address of network replication interface. Depending on the direction of the replication, the interface that is configured as the data address will begin to send or receive replication data. Please refer to the NexentaStor CLI Config Guide on configuring IP address for an interface.

Configure the Interface

To configure the interface to send or receive all replication traffic on your replication node, see this example:

1. View the existing values for HPR system properties:

   CLI@nexenta> config get value hpr
   NAME                        FLAGS   VALUE
   hpr.connectTimeout          --       3000
   hpr.dataAddress             --       -
   hpr.dataPort                --       6000
   hpr.heartbeatFaultTolerance --       3
   hpr.heartbeatInterval       --      10000
   hpr.requestTimeout          --      30000
   hpr.syncMaxAttempts         --       12
   hpr.syncRetryInterval       --      15000
   hpr.totalMemoryLimit        --       25

2. Set the HPR service data address:

   CLI@nexenta> config set hpr.dataAddress=<Local IP address that will send or receive the replication data>

   Example:

   CLI@nexenta> config set hpr.dataAddress=10.3.10.38

3. Verify the HPR service data address:

   CLI@nexenta> config get value hpr.dataAddress
   NAME  PROPERTY   VALUE
   hpr.dataAddress  value  192.168.0.1

   CLI@nexenta> config get value hpr.dataPort
   NAME  PROPERTY   VALUE
   hpr.dataPort  value  6000

   Depending on the direction of the replication, the interface that is configured as the data address will begin to send or receive replication data.
Managing Replication Services

This section includes the following topics:

- About Scheduled Replication (SR)
- Managing Scheduled Replication Services
- Managing Snapshots - Snaplists
- About Continuous Replication
- Managing Continuous Replication Services
- Activating a Destination Dataset
- Recovering Broken Replication Service
- Recreating Lost Service Instance
- Managing Multi-Destination Replication Services
About Scheduled Replication (SR)

Replication services can be local or remote. A local replication service replicates data from one dataset to another within one pool, or between pools on the same NexentaStor node. A remote replication service replicates data from one NexentaStor node to another. A remote service consists of two instances called manager service (local instance) and agent service (remote instance).

This section lists the various snapshot configurations supported with SR.

Figure 3-1: Example of Different Snapshot Configurations

This section uses the following naming conventions.

- Primary/Source dataset          DS_A
- Secondary/Destination dataset   DS_B
Local Snapshot

The above diagram illustrates the local snapshots and local snapshot schedules that can be configured on the source datasets DS_A. These local snapshots are not replicated to the remote appliance. Only snapshots created within the context of an SR service on the source dataset will get replicated.

Scheduled Snapshot

The diagram also illustrates a typical replication between two NexentaStor appliances. Here we have a SR Service that replicates from the source dataset DS_A to the destination dataset DS_B. When configuring a SR service you can also define the retention policies to control the maximum of number of snapshots that each snaplist can contain. This diagram shows an hourly schedule with a local retention policy of “keep 3” and a remote retention policy of “keep 6”.

Cloned Snapshot at Destination

This diagram also illustrates the ability to test the destination dataset by creating a clone from any snapshot in a schedule snaplist on DS_B. You can subsequently share the clone you create on DS_B using file or block protocols (as appropriate) and use it for DR testing. The parent snapshot for this clone is protected from deletion by the SR schedule retention policy on DS_B as long as the clone exists. Once the clone is destroyed, the snapshot is released from its hold and will eventually be deleted per the SR schedule retention policy.

On-demand Snapshot

Within context of a SR service, the user may create an on-demand snapshot and have it immediately replicated to the destination, outside of the constraints of replication service schedules. These snapshots are not managed by the service schedules and their associated retention policies. You may eventually delete these user created snapshots.

Note:

- Deleting and renaming snapshots from SR snaplists is possible on either side of the service as long as no modification is done on the “last common snapshot” / “most recent common snapshot”.
- Immediately after an on-demand replication completes, the replicated on-demand snapshot is the most recent common snapshot for the service. It should not be deleted on either source or destination until after another snapshot has been transferred.
Managing Scheduled Replication Services

In order to create a new replication service, you will need:

- Path to pre-existing source dataset which may include remote appliance address or hostname or F.Q.D.N in case of remote-to-local replication.
- Path to destination dataset which may include remote appliance IP address or hostname or F.Q.D.N in case of local-to-remote replication. Note that the name provided must be for a ‘new’ dataset. The destination dataset will be created by the replication service when it runs for the first time.

Figure 3-2: Example of Multiple Schedules per Service
Create New Replication Service

To successfully create a replication service do the following in the order listed.

1. **Create a new replication service.**

   The example below creates a recursive “service A” from pool1/src to pool2/dst.
   
   CLI@host> hpr create <type> <source> <destination> <service-name>

   Example:
   CLI@host> hpr create --recursive scheduled pool1/src https://<IP Address>/pool2/dst service1

   | Note: | When a replication service is created, the remote node address should point to a management network interface, not to `hpr.dataAddress`. To specify remote node, HPR supports IP addresses, hostnames or Fully Qualified Domain Names. |

2. **Add schedules to the service you created.**

   Replication Schedules should be added on the Manager Node. Before adding a new schedule, you must disable the replication service. Each replication service can have multiple schedules, for example, daily and monthly. Specify the replication schedules in cron format.
   
   CLI@host> hpr schedule-add [-nv] <service-name> <cron> <keep-source> <keep-destination> [<schedule-name>]

   There are few shortcuts available: "hourly", "daily", or "monthly" and you can use it as a value instead of a cron expression. The first snapping service is scheduled as “now + specified period”.

   Example:
   - Every 15 minutes keeping 1 snapshot locally and 3 remote
     CLI@host1> hpr schedule-add service1 "0,15,30,45 * * * *" 1 3 schedule1
   - Hourly - keeping 2 snapshots locally and 6 remote
     CLI@host1> hpr schedule-add <service-name> hourly 2 6 <schedule-name>
   - Monthly (beginning of the month) - keeping 12 snapshots locally and 12 remote
     CLI@host1> hpr schedule-add service1 "0 0 1 * * *" 12 12 schedule1

3. **Enable the replication service.**

   CLI@host1> hpr enable <service-name>

   | Note: | In case of initial replication, the destination dataset should not exist. Otherwise replication will fail to start. |
   |       | By default, if the source dataset is a filesystem, the destination filesystem gets unmounted. |
About CRON

The cron (UNIX) utility uses an expression syntax to define job schedules on a periodic basis. Use the cron expression syntax to specify replication service schedules. A cron expression is a string of five fields (separated by white space) that represents a set of times for a schedule. The following graphic illustrates cron expression structure.

```
# ──────────── min (0 - 59)
# │ ─────────── hour (0 - 23)
# │ │ ──────── day of month (1 - 31)
# │ │ │ ───── month (1 - 12)
# │ │ │ │ day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use names;)
# │ │ │ │ 7 is also Sunday)
# │ │ │
# │ │
# │ *
```

Field Values:
- Minutes: 0-59
- Hours: 0-23
- Day of month: 1-31
- Month: 1-12
- Day of week: 0-6

Ensure that you follow the rules discussed below for the cron syntax:
- Each field can contain special characters * or -
- Commas, in a cron expression, are used to separate items of a list. For example, using "MON,WED,FRI" in the 5th field for 'day of the week'.
- Hyphens, in a cron expression, define ranges. For example, 1-12 in the 4th field indicates every month between January (1) and December (12), inclusive.

Enable Service

Replication services of type continuous start immediately once the service is enabled. For scheduled services, replication starts according to the defined schedule.

Get List of Existing HPR Services

To validate that the service has been created/enabled or to list the existing hpr services use the hpr list command. RUNNING state refers if it's currently replicating or waiting for the next SR.
CLI@host> hpr list

<table>
<thead>
<tr>
<th>NAME</th>
<th>TYPE</th>
<th>RECURSIVE</th>
<th>SOURCE</th>
<th>DESTINATION</th>
<th>STATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>scheduled</td>
<td>yes</td>
<td>pool1/src</td>
<td><a href="https://host2:8443/pool2/dst">https://host2:8443/pool2/dst</a></td>
<td>enabled</td>
</tr>
</tbody>
</table>

**View Service Properties and its State**

The same service exists on both source and destination and reports the same state.

CLI@host> hpr get all service1

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>service1</td>
<td>rollback</td>
<td>no</td>
</tr>
<tr>
<td>service1</td>
<td>destination</td>
<td><a href="https://host2:8443/pool2/dst">https://host2:8443/pool2/dst</a></td>
</tr>
<tr>
<td>service1</td>
<td>heartbeat</td>
<td>yes</td>
</tr>
<tr>
<td>service1</td>
<td>ignoreProperties</td>
<td>-</td>
</tr>
<tr>
<td>service1</td>
<td>isManager</td>
<td>yes</td>
</tr>
<tr>
<td>service1</td>
<td>running</td>
<td>no</td>
</tr>
<tr>
<td>service1</td>
<td>isSyncing</td>
<td>no</td>
</tr>
<tr>
<td>service1</td>
<td>maxBufferSize</td>
<td>100</td>
</tr>
<tr>
<td>service1</td>
<td>name</td>
<td>service</td>
</tr>
<tr>
<td>service1</td>
<td>recursive</td>
<td>yes</td>
</tr>
<tr>
<td>service1</td>
<td>replaceProperties</td>
<td></td>
</tr>
<tr>
<td>service1</td>
<td>sendShareNfs</td>
<td>no</td>
</tr>
<tr>
<td>service1</td>
<td>sendUserSnapshots</td>
<td>no</td>
</tr>
<tr>
<td>service1</td>
<td>source</td>
<td>pool1/src</td>
</tr>
<tr>
<td>service1</td>
<td>state</td>
<td>enabled</td>
</tr>
<tr>
<td>service1</td>
<td>type</td>
<td>scheduled</td>
</tr>
</tbody>
</table>

When you execute the **HPR get all** command, the output includes the state of the service which can be either “enabled/disabled/faulted/recovering”. If the state of the service is “Faulted”, the **HPR get all** command returns an additional service property **LastError** as shown below.

The following system response is an example of a Faulted HPR service.

춘나 To view the details of the last error:

CLI@host> hpr get state,lastError service1

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>service1</td>
<td>lastError snapshot</td>
<td>Destination has been modified since most recent</td>
</tr>
<tr>
<td>service1</td>
<td>state</td>
<td>faulted</td>
</tr>
</tbody>
</table>
Clear the Service State

Once the underlying issue that triggered the service to go into the Faulted state is identified and fixed, you can clear the state of the service from the Faulted state to the Disabled state using the following command.

```
CLI@host> hpr clear <service-name>
```

Verify Service Schedule and Service Snapshots

Each schedule can be disabled or enabled, regardless of service state. Schedule state is persistent through the service enable/disable cycle.

To verify that the retention policy is met, use the `hpr snaplist-find` command to compare that the number of snapshots per schedule is equal to "keep" value for source or destination respectively.

- **To validate that the retention policy is met, run the following command on the Manager node:**

  ```
  CLI@host1> hpr snaplist-find <service-name>
  ```

  **Example:**

  ```
  CLI@host1> hpr snaplist-find service1
  SNAPLISTID  SERVICENAME  SCHEDULE   CRON        SOURCESNAPSHOTS   DESTINATIONSNAPSHOTS
  668a9e50    service1     hourly     19 * * * *  1/1              24/24
  668a9e50    service1     monthly    41 14 9 * * 12/12            12/12
  ```

  See [Managing Snapshots - Snaplists](#) for more details.

- **To verify service snapshots:**

  ```
  CLI@host> hpr snapshots service1
  CREATION         PATH                                                   SENT  SCHEDULE
  Dec 13 22:47:04  pool1/vg1@hpr-ondemand-2016-12-13-22-47-04-092       yes
  Dec 13 22:47:04  pool1/vg1/vol1@hpr-ondemand-2016-12-13-22-47-04-092 yes
  Dec 13 22:47:25  pool1/vg1@hpr-ondemand-2016-12-13-22-47-25-430       yes
  Dec 13 22:47:25  pool1/vg1/vol1@hpr-ondemand-2016-12-13-22-47-25-430 yes
  Dec 13 22:48:56  pool1/vg1@hpr-ondemand-2016-12-13-22-48-56-627       yes
  Dec 13 22:48:56  pool1/vg1/vol1@hpr-ondemand-2016-12-13-22-48-56-627 yes
  Dec 13 22:52:34  pool1/vg1@hpr-ondemand-2016-12-13-22-52-34-349       yes
  ```
Dec 13 22:52:34 pool1/vg1/voll@hpr-ondemand-2016-12-13-22-52-34-349
yes  -

How to Stop a Scheduled Replication

To stop an active (running) scheduling replication session, you can use the `hpr stop` command. When the stop command is executed on a running SR session, it interrupts the running SR (including “running once”) without changing the service state.

```
CLI@host> hpr stop <service-name>
```

**Difference between `hpr stop` and `hpr disable`**

The difference between `hpr stop` and `hpr disable` is that `stop` does not change the service state. Another difference between `hpr disable --force` and `hpr stop` is that the stop operation does not ignore errors. For example, if the agent is not reachable this command will fail with the related error.

**Note:**
- When you run service once from a disabled state, it temporary switches to enabled and then, when replication completes, the service switches back automatically to disabled state.
- Continuous Replication (CR) service cannot be stopped, you must disable it instead.

Disable Replication Service

This operation disables automatic scheduling of the service according to service schedules.

```
CLI@host> hpr disable <service-name>
```

When disabling an existing service the following happens:

- In case of CR, when you disable it, the service goes to the stopping state, completes replicating the current snapshot and goes to the disabled state.
  
  If the service is a recursive replication, when you disable the service, all the nested snapshots will be replicated before the service goes to the disabled state.

- In case of non-running scheduled service you can disable without force. The service switches to disabled state.

- For a service that is running, if you need to disable it, you must force disable using the --force option. When you force disable a running service, it interrupts immediately the running replication without completing the replication of the current snapshot, and disables the service by ignoring the errors.
  
  When a running service is disabled with the --force flag, the incompletely received snapshots will remain on the destination dataset and replication will resume from that point where it was interrupted on the next run, i.e. snapshots will not be sent from the beginning.

  When you force disable a replication service and if the agent is unreachable, the manager service state gets set to disabled. When the connection to the agent is restored, the disabled state is propagated to the agent.
On-demand Snapshots and Replication

Within context of a SR service, you may create an on-demand snapshot and have it immediately replicated to the other site, outside of the constraints of replication service schedules. These snapshots are not managed by the service schedules and their associated retention policies. You may eventually delete these user created snapshots. Immediately after an on-demand replication completes, the replicated on-demand snapshot is the most recent common snapshot for the service. It should not be deleted on either source or destination until after another snapshot has been transferred.

Run Service Once

This command would create an on-demand snapshots and replicate immediately without waiting for the next SR. SR Service must not be running when you execute this command.

```
CLI@host> hpr run-once <service-name>
```

Destroy Replication Service

Only a disabled service can be destroyed. In order to destroy a service in any other state, --force option should be used. During the service destroy, you have an option to destroy the source and destination snapshots and the destination datasets.

```
CLI@host> hpr destroy <service-name>
CLI@host> hpr destroy --help
```

Options:

- `--source-snapshots` Destroy snapshots created by this service on the source dataset.
- `--destination-snapshots` Destroy snapshots created by this service on the destination dataset.
- `--destination` Destroy the destination dataset.
- `-f, --force` Destroys service regardless if it is running or agent node’s availability.

Modifying an Existing Schedule

NexentaStor 5.x allows you to modify an existing schedule in any service state. The name, cron and retention policy can be modified.

Note: The new retention policy gets applied immediately. In this example, changing the retention policy on the destination from 10 to 5, will remove the oldest 5 snapshots.
Replicating Clones and User Snapshots

Replication of clones and user snapshots is only available for scheduled replication and disabled by default. To enable it, please use `--user-snapshots` when you create a replication service.

```
CLI@host> hpr create S --recursive --user-snapshots pool1/data https://example.com/pool1/backup databackup
```

Once this option is enabled, HPR will replicate all user snapshots and their clones to the destination.

**Note:**
- clones of HPR scheduled, on-demand and recovery snapshots will never be replicated.
- when clone/promote/delete operation are expected to be done (e.g. when using for openstack), autoRollback option should also be enabled, otherwise replication may be faulted because a destination dataset may have a newer snapshot which belonged to the deleted source dataset and this snapshot must be rolled back to resume incremental replication.

Enabling Auto-Rollback On a HPR Service to Seamlessly Recover from Child Snapshot Rollback on Source

You can enable the auto-rollback option on a HPR service while creating it or later so that when a user rolls back sources to an earlier snapshot then the HPR service will automatically rollback the corresponding destination snapshot.

The following example shows how to enable the auto-rollback feature when creating a HPR service or after a service is created.

1. Create a scheduled recursive HPR service with auto-rollback feature enabled.
   ```
   CLI@node> hpr create S --recursive --rollback pool1/data https://example.com/pool_backup/data_backup my_repl1
   ```

2. Or enable the rollback feature once the service is created.
   ```
   CLI@node> hpr set rollback=true my_repl1
   ```

You can also rollback user snapshots by enabling `sendUserSnapshots` feature as shown here.

1. View the `sendUserSnapshots` for the service you created.
   ```
   CLI@node> hpr get sendUserSnapshots my_repl1
   NAME     PROPERTY           VALUE
   fssrtsr  sendUserSnapshots  no
   ```

2. Enable the `sendUserSnapshots` for the service `my_repl1`.

Note:
- clones of HPR scheduled, on-demand and recovery snapshots will never be replicated.
- when clone/promote/delete operation are expected to be done (e.g. when using for openstack), autoRollback option should also be enabled, otherwise replication may be faulted because a destination dataset may have a newer snapshot which belonged to the deleted source dataset and this snapshot must be rolled back to resume incremental replication.
Impact of Auto-Rollback on Promote Operation on the Source

To seamlessly recover the replication service, auto-rollback must be enabled when sendUserSnapshots option is ON and clone/promote/delete operation is expected. Failing to enable auto-rollback feature may cause the replication to go to a faulted state because the destination dataset might end up with a newer snapshot belonging to the deleted source dataset and to resume the incremental replication you must manually rollback the snapshot.

Managing Snapshots - Snaplists

An HPR snaplist refers to the list of snapshots that are created by a specific schedule in a SR service. A schedule in a SR service results in the creation of 2 snaplists, one at the source and another at the destination, each with its own maximum number of snapshots, as per the site specific retention policies for that schedule.

Snaplists can be particularly useful to manage snapshots once a schedule, or a service, is destroyed. By default, when a schedule or a service is destroyed, its snaplists and all their snapshots are kept on the source and destination system. You have full control on the remaining snaplists and can manually delete the snaplists, or some of the snapshots they contain. Or you may re-attach a snaplist to another schedule in an HPR service on the same dataset. This allows HPR services to be destroyed and fully reconfigured, without losing the set of snapshots that were created.

Examples:

DS_A represents datasets on Node A;

DS_B represents datasets on Node B.

• Schedule Replication service SR_hourly with an hourly schedule keeping 6 snapshots on DS_A and 12 snapshots on DS_B

• You can destroy SR_hourly. This leaves a snaplist “DS_A_6” on DS_A and a snaplist “DS_B_12” on DS_B

• You can create a new service SR_hourly with a new hourly schedule, keeping 12 snapshots on DS_A and 24 snapshots on DS_B.

• You can re-attach snaplist DS_A_6 to the hourly schedule on DS_A and snaplist DS_B_12 to the hourly schedule on DS_B

• These snapshots are automatically handled by the new schedule retention policies.

Note: Schedule retention policies work on “maximum number of snapshots”. If a schedule has a retention policy of “Keep 12”, the policy will automatically keep the “newest 12 snapshots”. If a snaplist with 24 snapshots is attached to such a schedule, the 13 oldest snapshots will automatically be deleted when the new retention policy gets applied.

List Snapshots of a Given Replication Service

CLI@host> hpr snapshots SR-hourly
List Snapshots after Deleting a Service

You can list the snapshots that may remain after deleting a replication schedule or a service using the following command. This command also displays snaplists that belong to an existing schedule and may be used to verify that the retention policy is met.

Run this command on the manager node. If the command is executed on the agent node, it displays only the agent snapshots.

```
CLI@host> hpr snaplist-find [-s <field>]
```

Example:
```
CLI@host> hpr snaplist-find SR-hourly
```

Existing snapshot lists can be claimed to a schedule using the `hpr snaplist-claim` subcommand.

Delete List of snapshots

You can delete the list of snapshots associated with a replication service.

```
CLI@host> hpr snaplist-delete [-v] <service-name> <snaplist-id>
```

Claim Snapshots Belonging to an Existing Schedule

To allow a schedule to claim a list of snapshots that belonged to a deleted schedule, run the following command:

```
CLI@host> hpr snaplist-claim [-v] <service-name> <schedule-name> <snaplist-id>
```

Where the `<schedule-name>` represents the replication schedule that will claim the snapshots.
About Continuous Replication

Continuous Replication (CR) is a new service type in NexentaStor 5. It is licensed as an additional option to the Enterprise Edition license. CR delivers continuous asynchronous replication and enables close to zero RPO (Recovery Point Objective) over any distance without affecting application performance on the source dataset. Internally, CR works in the kernel and sends every write transaction group to the destination dataset, in real time.

See the Operational Consideration for Continuous Replication Services section for more details.

Managing Continuous Replication Services

In order to create a new replication service, you will need:

- A pre-existing source dataset (appliance, pool and dataset path and name)
- Path to a pre-existing source dataset which may include the remote appliance address or hostname or F.Q.D.N in case of remote-to-local replication.
- Path to a remote destination dataset which may include remote appliance IP address or hostname or F.Q.D.N in case of local-to-remote replication.

Create Replication Services

Only a single CR can be created for a given dataset. Before creating a new HPR service, identify the dataset to be replicated. If the dataset is a consistency group (volume group or file system parent) then use the --recursive parameter.

```
CLI@host1> hpr create --recursive continuous pool1/src [https://host2] pool2/dst <service-name>
```

Note:

- In case of initial replication, the destination dataset should not exist. Otherwise replication will fail to start.
- By default, if the source dataset is a filesystem, the destination filesystem gets unmounted.

Enable Continuous Replication Service

Enabling a CR service starts replication immediately.

```
CLI@host1> hpr enable <service-name>
```
Get List of Existing HPR Services

To validate that the service has been created/enabled or to list existing hpr services use the `hpr list` command. In the case of an enabled CR service, the RUNNING state returns a value YES when the service is replicating data and may return a value NO when HPR restarts the replication after a network failure.

```plaintext
CLI@host> hpr list
NAME     TYPE       RECURSIVE  SOURCE     DESTINATION                   STATE
RUNNING
service  continuous yes  pool1/src  https://host2:8443/pool2/dst  enabled yes
```

View Service Properties and Service State

The same service exists on both the source and the destination and reports the same state.

```plaintext
CLI@host> hpr get all service1
NAME      PROPERTY           VALUE
service1  rollback           no
service1  destination        https://host2:8443/pool2/dst
service1  heartbeat          yes
service1  ignoreProperties   -
service1  isManager          yes
service1  running            no
service1  isSyncing          no
service1  maxBufferSize      100
service1  name               service
service1  recursive          yes
service1  replaceProperties
service1  source             pool1/src
service1  state              enabled
service1  sendShareNfs       no
service1  sendUserSnapshots  no
service1  type               scheduled
```

When you execute the `HPR get all` command, the output also includes the State of the service which can be either “enabled/disabled/faulted/stopping/recovering”. If the state of the service is “Faulted”, the `HPR get all` command returns an additional service property `lastError`. When `lastError` is shown for enabled or recovering service it indicates HPR retries to restart the replication after this error.

Once replication is successfully restarted, `lastError` will be cleared.

The following system response is an example of a Faulted HPR service.
CLI@host> hpr get state, lastError service1

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>service1</td>
<td>lastError</td>
<td>Destination has been modified since most recent snapshot</td>
</tr>
<tr>
<td>service1</td>
<td>state</td>
<td>faulted</td>
</tr>
</tbody>
</table>

Clear the Service State

Once the underlying issue that triggered the service to go into the Faulted state is identified and fixed, you can clear the state of the service from the Faulted state using the following command.

CLI@host> hpr clear <service-name>

Disable Replication Service

In case of continuous service, disabling switches the service to the stopping state. Once the currently replicating snapshot is completely sent and written to the destination, service will be switched to the disabled state. In order to disable the currently running continuous service use the --force option, without switching to the stopping state, to stop replication immediately.

CLI@host> hpr disable [--force] <service-name>

When a continuous service is disabled with the --force flag, incompletely received snapshot will remain on the destination dataset and replication will resume from the point it was interrupted on the next run, i.e. snapshot will not be sent from the beginning.

Destroy Replication Service

Using the hpr destroy command you can only destroy a service that is disabled. In order to destroy a service in any other state you must use the --force option. During the service destroy, the source and destination autosnapshots are automatically destroyed. The destination dataset, created by this service can also be optionally destroyed.

CLI@host> hpr destroy <service-name>

Validating Running Replication

To view the Current Throughput, Bytes Sent, and Remaining Bytes of an active replication service use the following command. This command can be applied to view the details of both Continuous replication and Scheduled replication.

For a Continuous Replication Service

- To display statistics of running replication:
  
  CLI@host> hpr statistics cr
To display statistics of running replication 5 times with one second interval:

<table>
<thead>
<tr>
<th>SENT</th>
<th>THROUGHPUT</th>
<th>RPO</th>
<th>MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>300.5M</td>
<td>19M/s</td>
<td>1s</td>
<td>134.4M</td>
</tr>
<tr>
<td>330.5M</td>
<td>20M/s</td>
<td>832ms</td>
<td>134.4M</td>
</tr>
<tr>
<td>380.5M</td>
<td>21M/s</td>
<td>1s</td>
<td>138.1M</td>
</tr>
<tr>
<td>380.5M</td>
<td>20M/s</td>
<td>1s</td>
<td>159.4M</td>
</tr>
<tr>
<td>430.4M</td>
<td>43M/s</td>
<td>1s</td>
<td>154.4M</td>
</tr>
</tbody>
</table>

For a Scheduled Replication Service

To display statistics of running replication:

CLI@host> hpr statistics sr

<table>
<thead>
<tr>
<th>SENT</th>
<th>REMAINING</th>
<th>PROGRESS</th>
<th>THROUGHPUT</th>
<th>MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>236.0M</td>
<td>1.55G</td>
<td>12%</td>
<td>25.0M/s</td>
<td>99.5M</td>
</tr>
</tbody>
</table>

To display statistics of running replication 5 times with one second interval:

CLI@host> hpr statistics sr 1 5

<table>
<thead>
<tr>
<th>SENT</th>
<th>REMAINING</th>
<th>PROGRESS</th>
<th>THROUGHPUT</th>
<th>MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>236.0M</td>
<td>1.55G</td>
<td>12%</td>
<td>25.0M/s</td>
<td>99.5M</td>
</tr>
<tr>
<td>261.5M</td>
<td>1.53G</td>
<td>14%</td>
<td>25.0M/s</td>
<td>99.5M</td>
</tr>
<tr>
<td>286.8M</td>
<td>1.50G</td>
<td>15%</td>
<td>25.0M/s</td>
<td>99.5M</td>
</tr>
<tr>
<td>312.0M</td>
<td>1.48G</td>
<td>17%</td>
<td>25.0M/s</td>
<td>99.5M</td>
</tr>
<tr>
<td>337.3M</td>
<td>1.45G</td>
<td>18%</td>
<td>25.0M/s</td>
<td>99.5M</td>
</tr>
</tbody>
</table>

Activating a Destination Dataset

In disaster recovery scenarios, you will need to explicitly activate the destination dataset (file system or volume) of a replication service before it can be shared to applications:

CLI@node-b> hpr activate <dataset>

Activating the destination dataset ensures that the dataset is ready to be shared and used by applications on the secondary site:

- if the destination is a file system, it ensures it is mounted to its default mount point. If file system is replicated with sendShareNfs option, file systems will be shared with the same settings as they were shared on the source.
- sets readonly of any datasets to OFF if it is already ON,
• destroys incompletely received snapshot of any datasets that remains after a replication gets interrupted and may prevent creating new snapshots.

The mount point property is inherited from the parent pool or filesystem. The `hpr activate` command can be used recursively (--recursive option).

**Recovering Broken Replication Service**

Modifying, deleting or creating snapshots of the destination dataset may cause the replication service to fail with one of the following errors:

• “Destination has been modified since most recent snapshots”
• “Destination dataset already exists”
• “Most recent snapshot does not match the incremental source”
• “Failed to run replication session: Common snapshot does not exist”
• “Source and destination snapshots do not match, replication service should be recovered”
• “Source snapshot not found, replication service should be recovered”

In order to recover the broken service, you should run the `hpr recover` command:

```
CLI@node-a> hpr recover <service-name>
```

This command proposes operations related to recovering the broken replication service. You can either execute or refuse the proposed operations. When the `hpr recover` command is executed, it will create a new recovery snapshot; try to find the common snapshot for each replicated dataset separately; and replicate all snapshots created after the common snapshot to the destination. Any changes on the destination or snapshots created after the common snapshot will be destroyed on the destination dataset. In case the common snapshot does not exist, dataset will be replicated from the scratch.

**Recreating Lost Service Instance**

This section discusses the following two scenarios:

• Source appliance disaster with total loss of node A and its pools.
• Source appliance disaster with total loss of node A, but no impact on the data pools.

**Scenario 1: Lost Primary/Secondary node and Source/Destination Pool**

If a source or a destination pool is completely lost, replication is not recovered automatically, you can simply recreate the lost service instance from the surviving node using the `hpr recreate` command. The command will automatically rebuild the service instance on the node that was rebuilt, with all its schedules and retention policies and reattached snaplists, as appropriate. This operation is supported for both scheduled and continuous replication services.
• If you rebuilt the lost node A with the exact same configuration information as before (same hostname, same IP configurations, same System Management Address, same pool names), you can recreate the replication service with few details using just the service name as shown in the example below. The hpr recreate command must be run on the surviving node (node B in this example).

  CLI@node-B> hpr recreate <service-name>

• If the new node A is built with different host name, IP address or pool names, you can recreate the service instance with additional information as shown in the following example. The hpr recreate command must be run on the surviving node (node B in this example).

  CLI@node-B> hpr recreate [nv] <name> [<endpoint>]

  endpoint == full URL

  Example:

  CLI@node-B> hpr recreate r https://10.3.35.163/tank2/dst

  When the service is recreated, it will be recreated in the disabled state. You will need to manually enable the service (and possibly flip direction) to start replicating data between sites.

**Scenario 2: Lost Primary/Secondary node and Pool Intact**

When the primary node is totally lost and if the pool still exists, and the node was rebuilt with the same configuration as before, you can simply import the pool on the newly built node with all its previous datasets. You do not have to recreate the service, the new node should recover all replication service configuration from the pool.
Managing Multi-Destination Replication Services

When creating a multi-destination replication service, only one CR service is allowed on a Primary dataset. However, HPR supports having one CR and one or more SR services configured on that Primary dataset. For example, dataset A is continuously replicated to site B, and replicated every day to site C via a separate SR service.

The following example shows two services:

- One CR service with host-a/dataset-a as the primary to replicate to the secondary host-b/dataset-b.
- And another SR with host-a/dataset-a as the primary to replicate to the secondary host-c/dataset-c.

Figure 3-3: Example of Multiple HPR services to Multi-Destinations

- To create and enable CR to continuously replicate host-a/dataset-a to host-b/dataset-b:
  
  CLI@host-a> hpr create -r C dataset-a https://host-b/dataset-b service-ab
  
  CLI@host-a> hpr enable service-ab

- To create and enable SR and service schedules to replicate host-a/dataset-a to host-c/dataset-c on a regular basis: hourly and daily:
  
  CLI@host-a> hpr create -r S dataset-a https://host-c/dataset-c service-ac
  
  CLI@host-a> hpr schedule-add service-ac hourly 1 24 schedule-hourly
  
  CLI@host-a> hpr schedule-add service-ac daily 1 30 schedule-daily
  
  CLI@host-a> hpr enable service-ac

To flip the direction of the SR/CR service in a multi-destination setup, disable all the services that replicate the source dataset except the one which needs to be flipped.
For additional information on the commands used in this section, type:

- For the man page of the HPR command and its subcommands
  CLI@host> man hpr

- To get the list of HPR subcommands
  CLI@host> hpr -h

- For subcommand usage syntax and options
  CLI@host> hpr <subcommand> --help
Running NDMP on the Destination of an HPR Service

If you plan on performing NDMP backup of the replicated destination dataset, use the following properties – `--ignore-properties=mountpoint --replace-properties="readOnly=yes"` when creating a HPR service.

- Mount the replication destination dataset in read-only mode when creating an HPR service:
  ```
  CLI@host> hpr create scheduled test/src test/dst2 --ignore-properties=mountpoint --replace-properties="readOnly=yes" testsvc
  ```

If you already created an HPR service without setting the above mentioned properties, you may modify the existing HPR service to include these two properties.

- Modify an existing HPR service to enable running NDMP on the destination dataset:
  ```
  CLI@host> hpr set ignoreProperties=mountpoint,replaceProperties="readOnly=yes" testsvc
  ```

How NDMP Works on the Destination Datasets

When you perform NDMP backup on a replicated destination dataset, the following actions are performed by the NDMP service:

1. Locates the parent dataset,
2. Finds the latest HPR snapshot,
3. Creates new temporary snapshot if there are no existing HPR snapshots,
4. Creates and mount temporary clone from snapshot,
5. Performs backup of clone,
6. Destroys clone,
7. Destroys snapshot (if it is the NDMP snapshot).

For information on Configuring NDMP for Backups, refer NexentaStor 5.x CLI Configuration Guide.
Disaster Recovery Use Cases

This section includes the following topics:

- Disaster Recovery Test on Secondary Site
- Graceful Failover and Flip Direction
- Disaster Recovery with Failover and Failback
- Node Maintenance
- Disaster Recovery from Total Loss of Primary Site
- Disaster Recovery using Multi-Destination Services
- Multi-Destination Sync-Back Scenario

Warning: This chapter describes the advanced configuration options that can be configured using the NexentaStor CLI for HPR service. Nexenta recommends that you use these advanced functionalities discussed in this chapter only as disaster recovery solutions. Before using NexentaFusion GUI for HPR service, ensure that all the prerequisites are met.
Disaster Recovery Test on Secondary Site

A normal HPR service operation is executed as listed here:

- Primary Node A, dataset A is replicated to Secondary Node B, dataset B

**User Scenario**

This is a test done to validate that in case of a disaster, network clients can run from the backup (destination/secondary dataset). The following use case is an example to show how to manage the snapshots on the destination appliance.

**Action Plan for NexentaStor and NexentaFusion**

- Clone from an existing snapshot on dataset B
- Share the cloned filesystems (if dataset B is filesystem), or create lun mappings for cloned volumes (if dataset B is volume)
- Mount/Connect it on the client
- Validate that the clients can read/write user data
- Unmount on client side
- Unshare cloned filesystem or remove lun mappings on cloned volume
- Delete clone

| Note: | In case of continuous replication (CR), live DR test is not supported, disable the service first, then take the snapshots manually and clone. |

See NexentaFusion 1.2 User Guide to ensure that all the prerequisites are met to configure HPR service using the NexentaFusion GUI and for details on the tasks listed in the Action Plan.
Disaster Recovery by Creating Clones at the Destination Appliance

You can separately manage the snapshots copied from the source appliance to the destination appliance. To manage these snapshots, clone them on the destination appliance. These cloned datasets can be read/written and renamed, shared via file or block protocols, and then destroyed. The parent snapshot for this clone is protected from deletion by the schedule retention policy set on the destination as long as the clone exists. Once the clone is destroyed, the parent snapshot is released from the hold and will eventually be deleted per the SR schedule retention policy.

Outdated snapshot are destroyed every time when replication is finished and a new snapshot has been delivered to the destination. If a snapshot cannot be destroyed (it’s held or cloned), HPR will attempt to destroy it every 10 minutes and every time when another snapshot has been replicated.

This section uses the following names as examples:

- Dataset on the destination node (node-b) pool/destination, pool/volumegroup/destination
- Clone on the destination node (node-b) pool/clone, pool/volumegroup/clone

![DR Test Using Clones at DR Site](image)

1. **Steady State**
   - 2 snaps at Primary, 3 at Secondary
   - Application running at Source

2. **User creates a clone from snap in service snaplist**
   - HPR will hold snapshot in snaplist as long as clone exists

3. **User performs DR test on clone**
   - HPR continues to run and manages the snaplist at Secondary while keeping the snapshot

4. **User deletes the clone**
   - Snapshot is released from the hold and will eventually be deleted as per the SR schedule retention policy
Different Possibilities of Managing the Snapshots

You can manage the snapshots copied on the destination appliance by cloning them using one of the different ways listed here. Choose the option based on your needs.

For scheduled service:

- Creating writable non-recursive clone,
  See Table 4-1, Steps to manage the cloned snapshots at the destination site by creating writable non-recursive clone (using the CLI).
- Creating writable recursive clone,
  See Table 4-2, Steps to manage the cloned snapshots at the destination site by creating writable recursive clone (using the CLI).

For continuous replication service:

- Creating writable non-recursive clone,
  See Table 4-3, Steps to manage the cloned snapshots at the destination site by creating writable non-recursive clone (using the CLI).
- Creating writable recursive clone.
  See Table 4-4, Managing cloned snapshots at the destination by creating writable recursive clone (using the CLI).

For SR, you can clone using the existing snapshots whereas for a CR service, disable the service first, create a destination snapshot and then enable the service. You can clone the snapshot either before or after enabling the service.

In all the examples below, the following naming conventions are used to identify node-a and node-b:

- The primary/source/manager host is node-a.
- The secondary/destination/agent host is node-b.
For Scheduled Replication Service

Table 4-1: Steps to manage the cloned snapshots at the destination site by creating writable non-recursive clone (using the CLI).

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Creating writable non-recursive clone:</td>
</tr>
<tr>
<td></td>
<td>1. Find latest replicated snapshot.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; hpr get commonSnapshot sr</td>
</tr>
<tr>
<td></td>
<td>NAME PROPERTY        VALUE</td>
</tr>
<tr>
<td></td>
<td>sr  commonSnapshot     hpr-ondemand-2017-06-09-23-26-45-643</td>
</tr>
<tr>
<td></td>
<td>2. Create non-recursive clone for existing and latest hpr snapshot.</td>
</tr>
<tr>
<td></td>
<td>For filesystem:</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; snapshot clone pool/destination@hpr-ondemand-2017-06-09-23-26-45-643 pool/clone</td>
</tr>
<tr>
<td></td>
<td>For volume:</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; snapshot clone pool/volumegroup/destination@hpr-ondemand-2017-06-09-23-26-45-643 pool/volumegroup/clone</td>
</tr>
<tr>
<td></td>
<td>3. Share the cloned filesystem.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; nfs share pool/clone</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; smb share pool/clone</td>
</tr>
<tr>
<td></td>
<td>4. Or create lun mappings for cloned volume.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; lunmapping create pool/volumegroup/clone target-group host-group</td>
</tr>
<tr>
<td>#</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>2</td>
<td><strong>Manage the cloned snapshots</strong></td>
</tr>
<tr>
<td></td>
<td>1. Mount it on clients (if destination dataset is filesystem) or map it to clients (if destination dataset is volume or volume group).</td>
</tr>
<tr>
<td></td>
<td>2. Validate that the clients can read/write user data.</td>
</tr>
<tr>
<td></td>
<td>3. Unmount on the client side.</td>
</tr>
<tr>
<td></td>
<td>4. If destination dataset is filesystem, unshare the cloned filesystem.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; smb unshare pool/clone</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; nfs unshare pool/clone</td>
</tr>
<tr>
<td></td>
<td>For volumes:</td>
</tr>
<tr>
<td></td>
<td>Remove lun mappings on volume (if destination dataset is volume or volume group):</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; lunmapping destroy pool/volumegroup/clone target-group host-group</td>
</tr>
<tr>
<td></td>
<td>5. Delete clone.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; filesystem destroy -v pool/clone</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; volume destroy -v pool/volumegroup/clone</td>
</tr>
</tbody>
</table>
Table 4-2: Steps to manage the cloned snapshots at the destination site by creating writable recursive clone (using the CLI).

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Creating writable recursive clone:</strong></td>
</tr>
<tr>
<td></td>
<td>1. Find latest replicated snapshot.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; hpr get commonSnapshot sr</td>
</tr>
<tr>
<td></td>
<td>NAME PROPERTY VALUE</td>
</tr>
<tr>
<td></td>
<td>sr commonSnapshot hpr-ondemand-2017-06-09-23-26-45-643</td>
</tr>
<tr>
<td></td>
<td>2. Create recursive clones for existing and latest hpr snapshots.</td>
</tr>
<tr>
<td></td>
<td><strong>For filesystem:</strong></td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; dataset=pool/destination</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; clone=pool/clone</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; snapshot=hpr-ondemand-2017-06-09-23-26-45-643</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; for i in $(filesystem list -r -O basic -o path $dataset</td>
</tr>
<tr>
<td></td>
<td><strong>For volume groups:</strong></td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; dataset=pool/destination</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; clone=pool/clone</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; snapshot=hpr-ondemand-2017-06-09-23-26-45-643</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; for i in $(volume list -O basic -o path $dataset</td>
</tr>
<tr>
<td></td>
<td>3. Share the cloned filesystem.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; nfs share pool/clone</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; smb share pool/clone</td>
</tr>
<tr>
<td></td>
<td>4. Or create lun mappings for cloned volumes in the volume group.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; for i in $(volume list -O basic -o path pool/clone</td>
</tr>
<tr>
<td>#</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>2</td>
<td>Manage the cloned snapshots</td>
</tr>
<tr>
<td></td>
<td>1. Mount it on clients (if destination is filesystem) or map it to clients (if destination is volume group).</td>
</tr>
<tr>
<td></td>
<td>2. Validate that the clients can read/write user data.</td>
</tr>
<tr>
<td></td>
<td>3. Unmount on the client side.</td>
</tr>
<tr>
<td></td>
<td>4. If destination dataset is filesystem, unshare the cloned filesystem.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; for i in $(filesystem list -r -O basic -o path pool/clone</td>
</tr>
<tr>
<td></td>
<td>For volume groups:</td>
</tr>
<tr>
<td></td>
<td>Remove lun mappings on volume (if destination dataset is volume group):</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; for i in $(volume list -O basic -o path pool/clone); do lunmapping destroy -a $i; done</td>
</tr>
<tr>
<td></td>
<td>5. Delete clone recursively.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; filesystem destroy -rv pool/clone</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; volumegroup destroy -rv pool/clone</td>
</tr>
</tbody>
</table>
### For Continuous Replication Service

Table 4-3: Steps to manage the cloned snapshots at the destination site by creating writable non-recursive clone (using the CLI).

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1  | Creating writable non-recursive clone:  
For SR, you can clone using the existing snapshots whereas for a CR service, disable the service first, create a destination snapshot and then enable the service. You can clone the snapshot either before or after enabling the service. Note: When disabling the service do not use the --force flag. Disabling a CR replication with --force flag will likely leave incompletely received snapshot which will prevent a creation a new snapshot.  
1. Disable hpr service at the primary appliance.  
   CLI@node-a> hpr disable cr  
2. Take a non-recursive snapshot for the destination dataset.  
   CLI@node-b> snapshot create pool/destination@snapshot  
3. Enable HPR service at the primary appliance.  
   CLI@node-a> hpr enable cr  
4. Create non-recursive clone of the snapshot.  
   For filesystem:  
   CLI@node-b> snapshot clone pool/destination@snapshot pool/clone  
   For volume:  
   CLI@node-b> snapshot clone pool/volumegroup/destination@snapshot pool/volumegroup/clone  
5. Share the cloned filesystem.  
   CLI@node-b> nfs share pool/clone  
   CLI@node-b> smb share pool/clone  
6. Or create lun mappings for cloned volume.  
   CLI@node-b> lunmapping create pool/volumegroup/clone target-group host-group |
## Manage the cloned snapshots

1. Mount it on clients (if destination dataset is filesystem) or map it to clients (if destination dataset is volume or volume group).
2. Validate that the clients can read/write user data.
3. Unmount on the client side.
4. If destination dataset is filesystem, unshare the cloned filesystem.
   ```
   CLI@node-b> smb unshare pool/clone
   CLI@node-b> nfs unshare pool/clone
   ```
   For volumes:
   ```
   Remove lun mappings on volume (if destination dataset is volume or volume group):
   CLI@node-b> lunmapping destroy pool/volumegroup/clone
target-group host-group
   ```
5. Delete clone.
   ```
   CLI@node-b> filesystem destroy -v pool/clone
   CLI@node-b> volumegroup destroy -v pool/clone
   CLI@node-b> volume destroy -v pool/volumegroup/clone
   ```
Table 4-4: Managing cloned snapshots at the destination by creating writable recursive clone (using the CLI).

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1 | **Creating writable recursive clone.**  
For SR, you can clone using the existing snapshots whereas for a CR service, disable the service first, create a destination snapshot and then enable the service. You can clone the snapshot either before or after enabling the service. Note: When disabling the service do not use the --force flag. Disabling a CR replication with --force flag will likely leave incompletely received snapshot which will prevent a creation a new snapshot. |
| 1. | Disable hpr service at the primary appliance.  
CLI@node-a> hpr disable cr |
| 2. | Take recursive snapshot for the destination dataset.  
CLI@node-b> snapshot create -r pool/destination@snapshot |
| 3. | Enable HPR service at the primary appliance.  
CLI@node-b> hpr enable cr |
| 4. | Create recursive clone of the snapshot.  
For filesystem:  
CLI@node-b> dataset=pool/destination  
CLI@node-b> clone=pool/clone  
CLI@node-b> snapshot=snapshot  
CLI@node-b> for i in $(filesystem list -r -O basic -o path $dataset | sort); do snapshot clone $i@$snapshot ${i/$dataset/$clone}; done  
For volume groups:  
CLI@node-b> dataset=pool/destination  
CLI@node-b> clone=pool/clone  
CLI@node-b> snapshot=snapshot  
CLI@node-b> for i in $(volume list -O basic -o path $dataset | sort); do snapshot clone $i@$snapshot ${i/$dataset/$clone}; done |
| 5. | Share the cloned filesystem.  
CLI@node-b> nfs share pool/clone  
CLI@node-b> smb share pool/clone |
| 6. | Or create lun mappings for cloned volumes in the volume group.  
CLI@node-b> lunmapping create pool/volumegroup/clone target-group host-group |
Manage the cloned snapshots

1. Mount it on clients (if destination is filesystem) or map it to clients (if destination is volume group).
2. Validate that the clients can read/write user data.
3. Unmount on the client side.
4. If destination dataset is filesystem, unshare the cloned filesystem.
   
   CLI@node-b> for i in $(filesystem list -r -O basic -o path pool/clone | sort -r); do smb list $i && smb unshare $i; nfs list $i && nfs unshare $i; done

   For volume groups:

   Remove lun mappings on volume (if destination is volume group):

   CLI@node-b> for i in $(volume list -O basic -o path pool/clone); do lunmapping destroy -a $i; done

5. Delete clone recursively.

   CLI@node-b> filesystem destroy -rv pool/clone
   CLI@node-b> volumegroup destroy -rv pool/clone
Graceful Failover and Flip Direction

User Scenario

The typical scenario would be Primary source node maintenance. Node A is the Primary node and initially runs the application, with data replicated from Node A to Node B. Replication is disabled, the application is set to run on Node B while maintenance is performed on Node A. Once Node A is available again, the replication service is flipped to replicate from Node B to Node A.

Action Plan for NexentaStor and NexentaFusion

The example described in the CLI table below covers the more complex scenario of recursive replication.

1. Stop the application running off on the source dataset.
2. For source dataset, unshare and unmount the filesystem or remove the lun mappings on the volume.
3. Forcibly disable HPR service on Node A.
4. Run replication one more time for consistency, to replicate deltas to destination dataset.
5. Ensure the following checks: Validate that the replication has completed by making sure:
   1. the HPR service is not running;
   2. the state of the HPR service is not running but not faulted.
   2. the latest snapshots match on the source and destination.
6. Activate the destination dataset.
7. Share destination filesystem or create lun mappings for destination volume(s).
8. Destroy or rename aging-out datasets (if it exists) on the destination and perform flip operation (on node A) to reverse the replication direction.
9. Start the client application on the destination dataset.
10. Enable the HPR service on node A.
11. To Restore original replication direction, perform this scenario again.

See NexentaFusion 1.2 User Guide to ensure that all the prerequisites are met to configure HPR service using the NexentaFusion GUI and for details on the tasks listed in the Action Plan.
Graceful Flip & Sync-Back

1. Stop Application on Primary
   Run-once replication from P to S to get latest data changes

2. Flip Direction and Replication from S to P

3. Start Application on Secondary
Table 4-5: Steps to Reverse the Replication using Flip-Direction (using the CLI)

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1  | In the following examples, source is represented as dataset A and destination is represented as dataset B.  
To reverse the replication direction, do the following:  
1. If the source dataset is filesystem, recursively unshare the filesystem A (removes SMB and NFS shares).  
   CLI@node-a> for i in $(filesystem list -r -O basic -o path pool/dataset-a | sort -r); do smb list $i && smb unshare -v $i; nfs list $i && nfs unshare -v $i; filesystem unmount $i; done  
2. If the source dataset is volume or volume group, recursively remove lun mappings on volume A.  
   CLI@node-a> for i in $(volume list -O basic -o path pool/dataset-a); do lunmapping destroy -a $i; done  
3. If HPR service is enabled, forcibly disable the service on the primary host.  
   CLI@node-a> hpr disable -f service-ab  
4. Run replication one more time for consistency, to replicate deltas to dataset B.  
   CLI@node-a> hpr run-once service-ab  
5. Validate that the replication has completed, ensure that the HPR service is not running, and the latest snapshots match on source and destination.  
   CLI@node-a> while true; do  
   state=$(hpr get running -O basic service-ab | awk '{print $NF}')  
   if [[ "$state" != "yes" ]]; then  
   echo "service: last time replication done."  
   break  
   fi  
   echo "service: waiting for last time replication..."  
   sleep 10  
   done
<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
</table>
| 2  | 6. Activate the destination dataset B. In case of recursive replication, use --recursive option to activate it recursively.  
   CLI@node-b> hpr activate -rv pool/dataset-b  

7. Share the destination filesystem or create lun mappings for volumes.  
   For nfs shares:  
   CLI@node-b> nfs share pool/dataset-b  
   For smb shares:  
   CLI@node-b> smb share pool/dataset-b  
   For volumes:  
   CLI@node-b> lunmapping create pool/dataset-b/volume target-group host-group  

8. To reverse the replication direction, first destroy or rename aging-out datasets (refer to [Destroy Aging-out Datasets](#))  
   Once there are no aging-out datasets, perform flip operation to reverse the replication direction. Source dataset A becomes the destination and destination dataset B becomes the source.  
   CLI@node-a> hpr flip <service-name>  
   Example:  
   CLI@node-a> hpr flip service-ab  

9. Verify the flip.  
   CLI@node-a> hpr get source,destination service-ab  
   NAME PROPERTY VALUE  
   service-ab source https://node-b:8443/pool/dataset-b  
   service-ab destination pool/dataset-a  

10. The source dataset becomes the destination dataset and vice versa.  

11. Enable the replication service on the Primary.  
    CLI@node-a> hpr enable service-ab  

12. To Restore original replication direction, this scenario should be performed again.  

Disaster Recovery with Failover and Failback

A normal HPR service operation is executed as listed here:

- Primary Node A, dataset A is replicated to Secondary Node B, dataset B

**User Scenario**

In the event of a source appliance failure (Primary node A is down) and if it is temporarily unavailable, follow the steps described in this section, so that the network clients can access the data at the destination appliance (node B) and can use backup node B as datastore. After you recover the source appliance, you can switch the service roles back to normal operation.

**Action Plan for NexentaStor and NexentaFusion**

The following lists the action items to be executed when the source node goes down:

1. Forcibly disable HPR service on node B.
2. Activate the destination dataset B.
3. Share filesystem B or create lun mappings for volumes or volume group B.

Node A and dataset A comes back up.

HPR service changes to disabled state, because it was disabled on secondary node B.

4. Unshare filesystem A (if dataset A is a filesystem) or remove lun mappings on volumes (if dataset A is volume or volume group).
5. Destroy or rename aging-out datasets (if they exist) on the destination and perform flip operation (on node A) to reverse the replication direction.
6. Recover the service using the `hpr recover` command.
7. Enable HPR service on node A.
8. When ready, switch back to the original state using the flow from [Graceful Failover and Flip Direction](#).

See NexentaFusion 1.2 User Guide to ensure that all the prerequisites are met to configure HPR service using the NexentaFusion GUI and for details on the tasks listed in the Action Plan.
### Table 4-6: Steps to Restore HPR operation using Flip-Direction as a Disaster Recovery Process

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
</table>
| 1  | **The source/primary NexentaStor appliance (node A) becomes unavailable**  
In the event of a disaster and when the source appliance becomes unavailable, the network clients can access their data temporarily on the destination node B. To set the destination as the primary appliance, do the following:  
1. Forcibly disable the HPR service on the secondary node B.  
   CLI@node-b> hpr disable -f service-ab  
   This is done to release the datasets if the replication is still waiting for data from the source node.  
2. Activate the destination dataset. In case of recursive replication, use the --recursive option to activate it recursively.  
   CLI@node-b> hpr activate -rv <dataset at destination>  
   CLI@node-b> hpr activate -rv pool/dataset-b  
3. Share the filesystems or create lun mappings for volumes.  
   **For nfs shares:**  
   CLI@node-b> nfs share pool/dataset-b  
   **For smb shares:**  
   CLI@node-b> smb share pool/dataset-b  
   **For volumes:**  
   CLI@node-b> lunmapping create pool/dataset-b/volume target-group host-group |
| 2  | **The source NexentaStor appliance (node A) is back to operation**  
After you recover the source appliance, you can re-enable the replication in the backward direction A -> B by following the steps listed here.  
1. If the source dataset is filesystem, unshare it.  
   CLI@node-a> for i in $(filesystem list -r -O basic -o path pool/dataset-a | sort -r); do smb list $i && smb unshare -v $i; nfs list $i && nfs unshare -v $i; done  
2. If the source dataset is a volume or volume group, remove the lun mapping.  
   CLI@node-a> for i in $(volume list -O basic -o path pool/dataset-a); do lunmapping destroy -a $i; done |

---

**The source/primary NexentaStor appliance (node A) becomes unavailable**

In the event of a disaster and when the source appliance becomes unavailable, the network clients can access their data temporarily on the destination node B. To set the destination as the primary appliance, do the following:

1. Forcibly disable the HPR service on the secondary node B.
   ```
   CLI@node-b> hpr disable -f service-ab
   ```
   This is done to release the datasets if the replication is still waiting for data from the source node.

2. Activate the destination dataset. In case of recursive replication, use the --recursive option to activate it recursively.
   ```
   CLI@node-b> hpr activate -rv <dataset at destination>
   CLI@node-b> hpr activate -rv pool/dataset-b
   ```

3. Share the filesystems or create lun mappings for volumes.
   - **For nfs shares:**
     ```
     CLI@node-b> nfs share pool/dataset-b
     ```
   - **For smb shares:**
     ```
     CLI@node-b> smb share pool/dataset-b
     ```
   - **For volumes:**
     ```
     CLI@node-b> lunmapping create pool/dataset-b/volume target-group host-group
     ```

**The source NexentaStor appliance (node A) is back to operation**

After you recover the source appliance, you can re-enable the replication in the backward direction A -> B by following the steps listed here.

1. If the source dataset is filesystem, unshare it.
   ```
   CLI@node-a> for i in $(filesystem list -r -O basic -o path pool/dataset-a | sort -r); do smb list $i && smb unshare -v $i; nfs list $i && nfs unshare -v $i; done
   ```

2. If the source dataset is a volume or volume group, remove the lun mapping.
   ```
   CLI@node-a> for i in $(volume list -O basic -o path pool/dataset-a); do lunmapping destroy -a $i; done
   ```
### Description

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>On the node A, flip the direction to reverse the replication direction from node A/dataset A to node B/dataset B.</td>
</tr>
<tr>
<td></td>
<td>1. Destroy or rename aging-out datasets (refer to <a href="#">Destroy Aging-out Datasets</a>).</td>
</tr>
<tr>
<td></td>
<td>2. Execute the flip service command on the Node A to reverse the replication direction.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-a&gt; hpr flip service-ab</td>
</tr>
<tr>
<td></td>
<td>3. On node A run the <code>hpr recover</code> command.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-a&gt; hpr recover service-ab</td>
</tr>
<tr>
<td></td>
<td>4. Switch back to the original state using the steps from <a href="#">Graceful Failover and Flip Direction</a>.</td>
</tr>
</tbody>
</table>
Node Maintenance

For source node maintenance follow the graceful failover and flip direction scenario, see Graceful Failover and Flip Direction. For destination node maintenance do the following:

• From the manager node, disable all the replication services that replicate to this node.
  • Take a note of the services that will be disabled to simplify re-enabling.
  • See List Services, Filtered by remote node; Filtered by destination dataset: to list services, filtered by host or destination dataset.

• Perform maintenance.
  • Note: Snapshots will not be taken during the downtime.

• Enable the replication services.
Disaster Recovery from Total Loss of Primary Site

A normal HPR service operation is executed as listed here:

- Primary Node A, dataset A is replicated to Secondary Node B, dataset B

**User Scenario**

Use this solution as a disaster recovery action in the event of Primary/Source failure, if all the source data is lost, and the node A or the pool A on node A is not recoverable. In this scenario, you can use the backup node B as the data source for the client applications and replace the broken node or pool with a new node C. Now copy all the data from B to C and reinstate C as the primary storage and as source for replication.

**Action Plan for NexentaStor and NexentaFusion**

The following section summarizes the steps to be executed as the disaster recovery solution in case of a total loss of the primary site:

1. Activate the destination dataset B. In case of recursive replication, use the `--recursive` option to activate it recursively.
2. Share filesystem B or create lun mappings.
3. Replace the broken node A or pool with a new node C.
4. Recreate service on node B.
5. Destroy or rename aging-out datasets (refer to [Destroy Aging-out Datasets](#)) and flip service on Node C to replicate from the existing node B/dataset B => new node C/dataset C.
6. Optionally start HPR service to start immediate replication to send data to Node C.
7. Enable the new HPR service or follow "Graceful Failover and Flip Direction" to restore original replication direction.

See NexentaFusion User Guide to ensure that all the prerequisites are met to configure HPR service using the NexentaFusion GUI and for details on the tasks listed in the Action Plan.
Table 4-7: Action Plan for Total Loss of Primary Site (using the CLI)

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Activate the destination dataset B. In case of recursive replication, use the --recursive option.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; hpr activate -rv pool/dataset-b</td>
</tr>
<tr>
<td>2</td>
<td>Share the filesystem B or create lun mappings for volume or volume group B.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; nfs share pool/dataset-b</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; smb share pool/dataset-b</td>
</tr>
<tr>
<td></td>
<td>For volumes:</td>
</tr>
<tr>
<td></td>
<td>Create lun mappings for cloned volume (if destination dataset is volume or volume group):</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; lunmapping create pool/dataset-b/volume target-group host-group</td>
</tr>
<tr>
<td>3</td>
<td>Replace Node A with a new Node C and recreate HPR service on node B.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; hpr recreate service-ab</td>
</tr>
<tr>
<td>4</td>
<td>Destroy or rename aging-out datasets (refer to <a href="#">Destroy Aging-out Datasets</a>) and flip service on Node C to replicate the existing host B/dataset B =&gt; new node C/dataset C.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-c&gt; hpr flip service-ab</td>
</tr>
<tr>
<td>5</td>
<td>Optionally, start the HPR service to start immediate replication.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-c&gt; hpr run-once service-ab</td>
</tr>
<tr>
<td>#</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>-------------</td>
</tr>
<tr>
<td>6</td>
<td>If required, enable the new HPR service.</td>
</tr>
</tbody>
</table>

```
CLI@node-b> hpr enable service-bc
```
Disaster Recovery using Multi-Destination Services

A multiple HPR operation using multi-destination services is set up as listed here in this example:

- The primary/source/manager here is node A/dataset A.
- CR service-ab between node A/dataset A and node B/dataset B (remote appliance). This is the actual DR setup.
- SR service-ac between node A/dataset A and node C/dataset C (remote appliance). This is used for remote backups and not for the actual DR procedure.

User Scenario

Source node A goes down. All data is lost.

Action Plan for NexentaStor

The following section summarizes the steps to be executed as the disaster recovery solution:

1. Select the latest destination dataset for recovery of the source dataset (usually the dataset with CR - node B/dataset B).
2. Activate the dataset B.
3. Share filesystem B or create lun mappings for volume B. Node B/dataset B will be used as the primary for user applications until the primary node A is not available.
   Node A comes up (without user data) and the user want to restore original replication flows.
4. Recreate a service-ab on node B.
5. Start and wait for on-demand initial replication to complete successfully on node A (downtime is not required and the system can be used for normal productive operations). This step may consume lot of time.

6. Plan the downtime required for scheduled maintenance and data migration during which the system cannot be used for normal productive operations.

7. Unshare and unmount filesystem B (if dataset B is filesystem) or remove lun mappings on volume or volume group B (if dataset B is a volume or volume group).

8. Run replication one more time on node A for consistency, to replicate the delta to dataset A and validate that the replication has completed, HPR service should not be running and faulted.

9. Destroy or rename aging-out datasets (if they exist) on the destination and perform flip operation for service-ab (on node A) to reverse the replication direction.

10. Share filesystem A or create lun mappings for volume A on node A.

11. Destroy the existing dataset C on node C.

12. Recreate service-ac on node C.

13. Enable the SR service. Full initial resynchronization will be started, because common snapshots for datasets A and C do not exist.

See NexentaFusion 1.2 User Guide to ensure that all the prerequisites are met to configure HPR service using the NexentaFusion GUI and for details on the tasks listed in the Action Plan.

Table 4-8: Steps for Creating Multiple Replication Services to Multi-Destinations (using the CLI)

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select the latest destination dataset for recovery of the source dataset (usually the dataset with CR - node B/dataset B).</td>
</tr>
<tr>
<td>2</td>
<td>Activate the dataset B.</td>
</tr>
</tbody>
</table>

  CLI@node-b> hpr activate -rv pool/dataset-b
### Description

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
</table>
| 3  | Share the filesystem B or create lun mappings for volumes or volume group B. Node B/dataset B will be used as the primary for the network client applications until the primary node A is not available:  
  ```  
  CLI@node-b> filesystem list -r -O basic -o path pool/source | sort -r  
  For smb shares:  
  CLI@node-b> smb list  
  CLI@node-b> smb share -v pool/dataset-b  
  For nfs shares:  
  CLI@node-b> nfs list  
  CLI@node-b> nfs share -v pool/dataset-b  
  If the source dataset is volume or volume group, remove the lun mapping.  
  CLI@node-b> volume list -O basic -o path pool/source  
  CLI@node-b> lunmapping create pool/dataset-b/volume target-group host-group  
  | Node A comes up (without user data) and you may want to restore original replication flows. |
| 4  | Recreate a service -ab on node B  
  ```CLI@node-b> hpr recreate service-ab```  
| 5  | Start and wait for an on-demand initial replication to complete on node A (downtime is not required and the system can be used for normal productive operations). This step may consume a lot of time:  
  ```  
  CLI@node-a> hpr run-once -v service-ab  
  CLI@node-a> while true; do  
  
  state=$(hpr get running -O basic service-ab | nawk '{print $NF}')  
  if [[ "$state" != "yes" ]]; then  
  
  echo "service-ab: initial replication done."  
  break  
  fi  
  
  echo "service-ab: waiting for initial replication..."  
  sleep 10  
  done  
  done  
  ``` |
<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>•</td>
<td>Validate that the replication has completed, ensure that the HPR service is not running and not faulted, and the latest snapshots match on source and destination.</td>
</tr>
<tr>
<td>•</td>
<td>Plan the downtime for a scheduled maintenance and data migration during which the system cannot be used for normal productive operations.</td>
</tr>
<tr>
<td>6</td>
<td>If dataset B is a filesystem, unshare and unmount the filesystem B or remove lun mappings on volume B (if dataset B is a volume or volume group)</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; for i in $(filesystem list -r -O basic -o path pool/dataset-b</td>
</tr>
<tr>
<td></td>
<td>If dataset B is volume or volume group, remove lun mappings on volume or volume group B:</td>
</tr>
<tr>
<td></td>
<td>CLI@node-b&gt; for i in $(volume list -O basic -o path pool/dataset-b); do lunmapping destroy -a $i; done</td>
</tr>
<tr>
<td>7</td>
<td>Run replication one more time on node A for consistency and to replicate the deltas to dataset A and validate that the replication has completed. Note: The HPR service should not be running when executing the following steps.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-a&gt; hpr run-once -v service-ab</td>
</tr>
<tr>
<td></td>
<td>CLI@node-a&gt; while true; do</td>
</tr>
<tr>
<td></td>
<td>state=$(hpr get running -O basic service-ab</td>
</tr>
<tr>
<td></td>
<td>if [[ &quot;$state&quot; != &quot;yes&quot; ]]; then</td>
</tr>
<tr>
<td></td>
<td>echo &quot;service-ab: last time replication done.&quot;</td>
</tr>
<tr>
<td></td>
<td>break</td>
</tr>
<tr>
<td></td>
<td>fi</td>
</tr>
<tr>
<td></td>
<td>echo &quot;service-ab: waiting for last time replication...&quot;</td>
</tr>
<tr>
<td></td>
<td>sleep 10</td>
</tr>
<tr>
<td></td>
<td>done</td>
</tr>
<tr>
<td>8</td>
<td>Destroy or rename aging-out datasets (refer to <a href="#">Destroy Aging-out Datasets</a>); Perform flip direction for the service-ab on node A to reverse the replication direction. This reverses replication direction from the node B/dataset B to node A/dataset A</td>
</tr>
<tr>
<td></td>
<td>CLI@node-a&gt; hpr flip service-ab</td>
</tr>
<tr>
<td>9</td>
<td>Share the file system A or create lun mappings for the volume A on node A:</td>
</tr>
<tr>
<td></td>
<td>CLI@node-a&gt; nfs share pool/dataset-a</td>
</tr>
<tr>
<td></td>
<td>CLI@node-a&gt; smb share pool/dataset-a</td>
</tr>
<tr>
<td></td>
<td>CLI@node-a&gt; lunmapping create pool/dataset-a/volume target-group host-group</td>
</tr>
<tr>
<td>#</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>-------------------------------------------------------</td>
</tr>
<tr>
<td>10</td>
<td>Destroy the existing dataset C on node C</td>
</tr>
<tr>
<td>11</td>
<td>Recreate SR service on node C:</td>
</tr>
<tr>
<td></td>
<td>CLI@node-c&gt; hpr recreate service-ac</td>
</tr>
<tr>
<td>12</td>
<td>Enable a new SR service-ac on node A</td>
</tr>
<tr>
<td></td>
<td>CLI@node-a&gt; hpr enable service-ac</td>
</tr>
</tbody>
</table>
Multi-Destination Sync-Back Scenario

The Sync-back in a multi-destination HPR service is set up as listed here:

- The Primary/source/manager here is node A/dataset A
- CR service-ab is configured between node A/dataset A and node B/dataset B. This is the actual sync-back setup.
- SR service-ac is configured between node A/dataset A and node C/dataset C. This is used for remote backups and not for the sync-back procedure.

Replication Flow

The source is node A/dataset A
- node B/dataset B <= CR <= node A/dataset A <= SR => node C/dataset C

User Scenario

In the event of a primary/source node A failure where it becomes temporarily unavailable, you can use node B/dataset B as the primary for network clients. When node A comes back, you can restore the original replication flows.

Action Plan for NexentaStor and NexentaFusion

The following section summarizes the steps to be executed as the sync-back action plan:

1. Select the latest destination dataset for the recovery source dataset (usually the dataset with CR - node B/ dataset B).
2. Forcibly disable the HPR service which should be synced back (let's consider it's service-ab) on node B.
3. Activate the dataset B.
4. Share the file system B or create lun mappings for volume B. Now Node B/dataset B can be used as the primary for user applications while the primary node A is not available.
5. Node A comes up and you can restore the original replication flows.
6. Unshare the file system A (if dataset A is file system) or remove lun mappings on volume or volume group A if dataset A is a volume or volume group.
7. Disable HPR service-ac on node A.
8. Destroy or rename aging-out datasets (if they exist) on the destination and perform flip operation for service-ab (on node A) to reverse the replication direction.
9. Recover HPR service-ab on node A. It will destroy possible changes on the dataset A made after the most-recent snapshot.
10. Plan the downtime required for scheduled maintenance and data migration during which the system cannot be used for normal productive operations.

11. If dataset B is a file system, unshare and unmount the file system B or remove lun mappings on volume or volume group B, if dataset B is a volume or volume group.

12. Run service-ab once on Node A and wait for on-demand initial replication to complete successfully on node A.

13. Share the file system A or create lun mappings for volumes or volume group A. Now Node A/dataset A can be used as the primary for user applications.

14. Perform flip direction for service-ab to reverse the replication direction. This restores the original replication direction from node A/dataset A to the node B/dataset B.

15. Enable the HPR service-ab on node A.

16. Enable the HPR service-ac on node A.

See NexentaFusion 1.2 User Guide to ensure that all the prerequisites are met to configure HPR service using the NexentaFusion GUI and for details on the tasks listed in the Action Plan.

Table 4-9: Steps to Sync-back in Multi-Destination Services (using the CLI)

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Select the latest destination dataset for the recovery source dataset (usually the dataset with CR - node B/ dataset B).</td>
</tr>
</tbody>
</table>
| 2  | Forcibly disable the HPR service-ab on node B:  
   CLI@node-b> hpr disable -f service-ab |
| 3  | Activate the dataset B.  
   CLI@node-b> hpr activate -rv pool/dataset-b |
| 4  | Share the file system B or create lun mappings for volumes or volume group B. Node B/ dataset B will be used as the primary for user applications when the primary node A is not available:  
   CLI@node-b> nfs share pool/dataset-b  
   CLI@node-b> smb share pool/dataset-b  
   CLI@node-b> lunmapping create pool/dataset-b/volume  
   target-group host-group |

Note:  
- Sync back from the node B to the node A may destroy the most recent snapshots of the service-ac.  
- In multi-destination replication, it is recommended to retain at least 2 snapshots on the source and on the destination.
5. Node A comes up and the user wants to restore the original replication flows

Unshare the file system A (if dataset A is a file system):

```
CLI@node-a> for i in $(filesystem list -r -O basic -o path pool/dataset-a | sort -r); do smb list $i && smb unshare -v $i; nfs list $i && nfs unshare -v $i; filesystem unmount $i; done
```

or remove lun mappings on volume or volume group A (if dataset A is a volume or a volume group):

```
CLI@node-a> for i in $(volume list -O basic -o path pool/dataset-a); do lunmapping destroy -a $i; done
```

6. Disable the HPR service-ac on node A. If replication is running, you can use the --force flag to disable it immediately:

```
CLI@node-a> hpr disable [-f] service-ac
```

7. Perform flip direction for the service-ab to reverse the replication direction. This creates a new replication direction from the node B/dataset B to the node A/dataset A:

```
CLI@node-a> hpr flip service-ab
```

8. Recover HPR service-ab on node A:

```
CLI@node-a> hpr recover service-ab
```

9. Plan the downtime required for scheduled maintenance and data migration during which the system cannot be used for normal productive operations.

10. Unshare the file system B (if dataset B is a file system):

```
CLI@node-b> for i in $(filesystem list -r -O basic -o path pool/dataset-b | sort -r); do smb list $i && smb unshare -v $i; nfs list $i && nfs unshare -v $i; done
```

or remove lun mappings on volume or volume group B (if dataset B is a volume or a volume group):

```
CLI@node-b> for i in $(volume list -O basic -o path pool/dataset-b); do lunmapping destroy -a $i; done
```

11. Run service-ab once on Node A and wait for on-demand initial replication to complete successfully on node A:

```
CLI@node-a> hpr run-once service-ab
```

12. Share the file system A or create lun mappings for volumes or volume group A:

```
CLI@node-a> nfs share pool/dataset-a
CLI@node-a> smb share pool/dataset-a
CLI@node-a> lunmapping create pool/dataset-a/volume target-group host-group
```

13. Destroy or rename aging-out datasets (refer to [Destroy Aging-out Datasets](#));

Perform flip direction for the service-ab on node A to reverse the replication direction. This reverses replication direction from the node B/dataset B to node A/dataset A

```
CLI@node-a> hpr flip service-ab
```
<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>Enable the HPR service-ab on node A:</td>
</tr>
<tr>
<td></td>
<td>CLI@node-a&gt; hpr enable service-ab</td>
</tr>
<tr>
<td>15</td>
<td>Enable the HPR service-ac on node A.</td>
</tr>
<tr>
<td></td>
<td>CLI@node-a&gt; hpr enable service-ac</td>
</tr>
</tbody>
</table>
Advanced Configuration

This section includes the following topics:

- View History of Last Replications
- List all HPR Service Properties
- Modify HPR Service Properties
- List of Dataset Properties that can be Ignored or Replaced
- Disable HPR Service
- List Services
- Other HPR Schedule and Snaplist Management Commands

Replication Service Options

View History of Last Replications

HPR stores result of last 10 runs including run-once, scheduled and recovery run.

```bash
CLI@node-a> hpr history service-ab
RUNNUMBER STARTTIME        ENDTIME          SOURCESNAPSHOT
          SUCCEED SENDSIZE LASTERROR
 1          Jul 13 04:10:01  Jul 13 04:11:11  hpr-2017-07-13-04-10-01-024  yes  200M -
 3          Jul 13 04:30:01  Jul 13 04:31:15  hpr-2017-07-13-04-30-01-017  no   400M Destination space quota exceeded
 4          Jul 13 04:35:13  Jul 13 04:36:17  hpr-ondemand-2017-07-13-04-37-02-387  yes  200M -
 5          Jul 13 04:40:00  Jul 13 04:41:03  hpr-2017-07-13-04-40-00-720  no   300M Destination has been modified since most recent snapshot
 6          Jul 13 05:05:01  Jul 13 05:06:16  hpr-recovery-2017-07-13-05-05-01-986  yes  300M -
```
List all HPR Service Properties

CLI@node-a> hpr get all my_repl1

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>my_repl1</td>
<td>rollback</td>
<td>no</td>
</tr>
<tr>
<td>my_repl1</td>
<td>commonSnapshot</td>
<td>hpr-2019-07-19-06-58-00-043</td>
</tr>
<tr>
<td>my_repl1</td>
<td>creation</td>
<td>Jul 25 09:25:14</td>
</tr>
<tr>
<td>my_repl1</td>
<td>destination</td>
<td>tank/fssr/fsdessr</td>
</tr>
<tr>
<td>my_repl1</td>
<td>ignoreProperties</td>
<td>-</td>
</tr>
<tr>
<td>my_repl1</td>
<td>isManager</td>
<td>yes</td>
</tr>
<tr>
<td>my_repl1</td>
<td>running</td>
<td>no</td>
</tr>
<tr>
<td>my_repl1</td>
<td>isSyncing</td>
<td>no</td>
</tr>
<tr>
<td>my_repl1</td>
<td>lastError</td>
<td>Replication disabled due to the expired maintenance period</td>
</tr>
<tr>
<td>my_repl1</td>
<td>maxBufferSize</td>
<td>100</td>
</tr>
<tr>
<td>my_repl1</td>
<td>name</td>
<td>fssrtsr</td>
</tr>
<tr>
<td>my_repl1</td>
<td>recursive</td>
<td>no</td>
</tr>
<tr>
<td>my_repl1</td>
<td>replaceProperties</td>
<td></td>
</tr>
<tr>
<td>my_repl1</td>
<td>runNumber</td>
<td>44323</td>
</tr>
<tr>
<td>my_repl1</td>
<td>sendShareNfs</td>
<td>no</td>
</tr>
<tr>
<td>my_repl1</td>
<td>sendUserSnapshots</td>
<td>no</td>
</tr>
<tr>
<td>my_repl1</td>
<td>source</td>
<td>tank/fssr</td>
</tr>
<tr>
<td>my_repl1</td>
<td>state</td>
<td>faulted</td>
</tr>
<tr>
<td>my_repl1</td>
<td>type</td>
<td>scheduled</td>
</tr>
</tbody>
</table>

Recommendation

Nexenta does not recommend changing the following two properties unless absolutely needed. If you need to modify these properties, do it ideally with an approval from Nexenta support team.

- **recursive** - replicate dataset recursively. Changing this property may break incremental replication.
- **remoteNode** - Allow to set new IP or F.Q.D.N for the remote node.
  
  Example: `hpr set remoteNode=https://new.destination.com`

Replicate User Snapshots and Their Clones to the Destination

For the HPR service to be able to replicate user snapshots and their clones to the destination you must set this HPR service option `sendUserSnapshots` to `true`.

1. View the `sendUserSnapshots` for the service you created.

   CLI@node> hpr get sendUserSnapshots my_repl1
Modify HPR Service Properties

You can modify the service properties only on a disabled service. This table lists the properties that can be modified:

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
</table>
| description | service description  
Example to set the service description:  
```  
CLI@node-a> hpr set description="Daily replication" service1  
``` |
| throttle | throttle connection speed (bytes per second)  
Example to set service throttle connection speed:  
```  
CLI@node-a> hpr set throttle=10M service1  
``` |
| ignoreProperties | exclude the specified properties from the receive stream  
Example to exclude the compression property from the receive stream:  
```  
CLI@node-a> hpr set ignoreProperties=compressionMode service1  
``` |
| replaceProperties | set the specified properties during receive  
```  
CLI@node> hpr set replaceProperties="compression=off,quota=0" service1  
``` |
| rollback | rollback destination snapshots that were rolled back at the source  
```  
CLI@node> hpr set rollback=yes service1  
``` |
| maxBufferSize | maximum replication buffer size (Mbytes)  
```  
CLI@node> hpr set maxBufferSize=500 service1  
``` |
| sendShareNfs | If this property is enabled, HPR will send NFS share settings to the destination filesystem. When destination filesystem is activated or mounted, it will be shared with the same NFS settings as the source filesystem.  
```  
CLI@node> hpr set sendShareNfs=yes service1  
``` |
| sendUserSnapshots | replicate user snapshots and their clones  
```  
CLI@node> hpr set sendUserSnapshots=yes service1  
``` |
List of Dataset Properties that can be Ignored or Replaced

<table>
<thead>
<tr>
<th>#</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>aclInherit</td>
<td>Controls how ACL entries are inherited when files and directories are created</td>
</tr>
<tr>
<td>aclMode</td>
<td>Controls ACL behavior when a file is initially created or whenever a file or directory's mode is modified by the chmod command</td>
</tr>
<tr>
<td>allowExtendedAttributes</td>
<td>Indicates whether extended attributes are enabled or disabled for this file system</td>
</tr>
<tr>
<td>checksumMode</td>
<td>Controls the checksum used to verify data integrity</td>
</tr>
<tr>
<td>compressionMode</td>
<td>Enables or disables compression for a dataset</td>
</tr>
<tr>
<td>dataCopies</td>
<td>Sets the number of copies of user data per file system</td>
</tr>
<tr>
<td>dedupMode</td>
<td>Controls whether duplicate data is removed from the file system</td>
</tr>
<tr>
<td>logBiasMode</td>
<td>Use this property to provide a hint to ZFS about handling synchronous requests for a specific dataset</td>
</tr>
<tr>
<td>mountpoint</td>
<td>When this property is ignored, HPR will mount destination filesystems to the default mountpoint. When this option is combined with replaceProperties=readonly=yes, HPR will mount destination filesystem in read-only mode. Mounted in read-only mode destination filesystem is required for NDMP-based backup environment. Nexenta does not recommend to ignore mountpoint without readonly since this will break replication.</td>
</tr>
<tr>
<td>nonBlockingMandatoryMode</td>
<td>Controls if non-blocking mandatory locks are enabled or disabled. The filesystem needs to be remounted after changing this property in order to have any effect</td>
</tr>
<tr>
<td>primaryCacheMode</td>
<td>Controls what is cached in the primary cache (ARC)</td>
</tr>
<tr>
<td>quotaSize</td>
<td>Limits the amount of disk space a dataset and its descendents can consume. Value zero means no quota</td>
</tr>
<tr>
<td>rateLimit</td>
<td>Limits QoS transfer rate for the file system</td>
</tr>
<tr>
<td>readOnly</td>
<td>Controls whether a dataset can be modified</td>
</tr>
<tr>
<td>recordSize</td>
<td>Specifies a suggested block size for files in a file system</td>
</tr>
<tr>
<td>redundantMetadata</td>
<td>Controls what types of metadata are stored redundantly</td>
</tr>
<tr>
<td>reservationSize</td>
<td>Sets the minimum amount of disk space guaranteed to a dataset and its descendents. Value zero means no reservation.</td>
</tr>
<tr>
<td>secondaryCache</td>
<td>Controls what is cached in the secondary cache (L2ARC)</td>
</tr>
</tbody>
</table>
Configure Destination Dataset of an active HPR SR service as Read-only

You can share the destination filesystem of a SR service as read-only while the SR is enabled and running. This allows a user to access the dataset as read-only, including all of its snapshots in the .zfs directory for NFS file systems. You can leverage this capability to easily recover data from replicated datasets or past replicated snapshots.

1. **Create a new Scheduled replication service.**

   The example below creates a recursive “service1” from data1/src to data1/dst1 with the property “readOnly=yes” during receive and also excludes the specified property “mountPoint” from the receive stream.

   CLI@host> hpr create <type> <source> <destination> <service-name>

   Example:

   CLI@host> hpr create --recursive scheduled --ignore-properties=mountPoint --replace-properties=readOnly=yes data1/src data1/dst1 service1

   Nexenta does not recommend to ignore mountpoint without readonly since this will break replication.

2. **Verify the service you created.**

   CLI@host> hpr get all service1

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>service1</td>
<td>rollback</td>
<td>no</td>
</tr>
<tr>
<td>service1</td>
<td>commonSnapshot</td>
<td>hpr-ondemand-2016-12-13-22-51-26-105</td>
</tr>
<tr>
<td>service1</td>
<td>destination</td>
<td><a href="https://node-b:8443/data1/dst1">https://node-b:8443/data1/dst1</a></td>
</tr>
<tr>
<td>service1</td>
<td>heartbeat</td>
<td>no</td>
</tr>
<tr>
<td>service1</td>
<td>ignoreProperties</td>
<td>mountpoint</td>
</tr>
<tr>
<td>service1</td>
<td>isManager</td>
<td>yes</td>
</tr>
<tr>
<td>service1</td>
<td>running</td>
<td>no</td>
</tr>
<tr>
<td>service1</td>
<td>isSyncing</td>
<td>no</td>
</tr>
</tbody>
</table>
service1  maxBufferSize      100
service1  name               service
service1  recursive          yes
service1  replaceProperties  readOnly: yes
service1  source             data1/src
service1  state              disabled
service1  type               scheduled
service1  sendShareNfs       no
service1  sendUserSnapshots  no

3. Enable the Service
   CLI@host> hpr enable service1

4. Verify the destination filesystem
   CLI@host> filesystem get mounted,mountPoint,readOnly data1/dst1
   PATH        PROPERTY    VALUE        SOURCE
   data1/dst1  mounted     yes          -
   data1/dst1  mountPoint  /data1/dst1  local
   data1/dst1  readOnly    yes          received

Disable HPR Service

On Primary Node:

   CLI@node-a> hpr disable <service-name>

On Secondary Node

To disable HPR service on the secondary node, use the force -f option:

   CLI@node-b> hpr disable -f <service-name>

List Services

Filtered by remote node:

   CLI@node-a> hpr list -O basic -o name,source,destination | grep -w 'node-b'
   service-fs1  pool1/fs1       https://node-b:8443/pool2/fs1
   service-fs2  pool1/fs2       https://node-b:8443/pool2/fs2
   service-vg1  pool1/vg1       https://node-b:8443/pool2/vg1
   service-vg2  pool1/vg2       https://node-b:8443/pool2/vg2
Filtered by destination dataset:

CLI@node-a> hpr list -O basic -o name,source,destination | grep '/pool2/fs1$'

| service-fs1 | pool1/fs1 | https://node-b:8443/pool2/fs1 |
## Other HPR Schedule and Snaplist Management Commands

This table enumerates some of the CLI commands to manage the service schedules and the snapshots.

<table>
<thead>
<tr>
<th>Task</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Disable schedule</td>
<td><code>CLI@node&gt; hpr schedule-disable &lt;service-name&gt; &lt;schedule-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>CLI@node&gt; hpr schedule-disable --help</code></td>
</tr>
<tr>
<td>Remove existing schedule</td>
<td><code>CLI@node&gt; hpr schedule-remove &lt;service-name&gt; &lt;schedule-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>CLI@node&gt; hpr schedule-remove --help</code></td>
</tr>
<tr>
<td>Rename existing schedule</td>
<td><code>CLI@node&gt; hpr schedule-rename &lt;service-name&gt; &lt;schedule-name&gt; &lt;new-schedule-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>CLI@node&gt; hpr schedule-rename --help</code></td>
</tr>
<tr>
<td>Change schedule properties</td>
<td><code>CLI@node&gt; hpr schedule-set &lt;property=value&gt; &lt;service-name&gt; &lt;schedule-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>CLI@node&gt; hpr schedule-set --help</code></td>
</tr>
<tr>
<td>View schedules on the appliance</td>
<td><code>CLI@node&gt; hpr schedules</code></td>
</tr>
<tr>
<td></td>
<td>This will list all schedules on the appliance along with the service name.</td>
</tr>
<tr>
<td>List service snapshots list</td>
<td><code>CLI@node&gt; hpr snapshots &lt;service-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>CLI@node&gt; hpr snapshots --help</code></td>
</tr>
<tr>
<td>Claim snapshots list</td>
<td><code>CLI@node&gt; hpr snaplist-claim &lt;service-name&gt; &lt;schedule-name&gt; &lt;snaplist-id&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>CLI@node&gt; hpr snaplist-claim --help</code></td>
</tr>
<tr>
<td>Delete snapshots list</td>
<td><code>CLI@node&gt; hpr snaplist-delete &lt;service-name&gt; &lt;snaplist-id&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>CLI@node&gt; hpr snaplist-delete --help</code></td>
</tr>
<tr>
<td>Find snapshots list</td>
<td><code>CLI@node&gt; hpr snaplist-find &lt;service-name&gt;</code></td>
</tr>
<tr>
<td></td>
<td><code>CLI@node&gt; hpr snaplist-find --help</code></td>
</tr>
</tbody>
</table>
Troubleshooting

This chapter covers the following topics:

- Recovering Broken Replication Service
- Automatic Handling Network Errors
- Handling Destination Space Quota Errors
- Handling Alerts
- List of All Default HPR Standard Configurations
- Adjusting Memory Available for Replication Services

Recovering Broken Replication Service

Modifying, deleting or creating snapshots of the destination dataset may cause the replication service to fail with one of the following errors:

- “Destination has been modified since most recent snapshots”
- “Destination dataset already exists”
- “Most recent snapshot does not match the incremental source”
- “Failed to run replication session: Common snapshot does not exist”
- “Source and destination snapshots do not match, replication service should be recovered”
- “Source snapshot not found, replication service should be recovered”

In order to recover the broken service, you should run the `hpr recover` command:

```
CLI@node-a> hpr recover <service-name>
```

This command proposes operations related to recovering the broken replication service. You can either execute or refuse the proposed operations. When the `hpr recover` command is executed, it will create a new recovery snapshot; try to find the common snapshot for each replicated dataset separately; and replicate all snapshots created after the common snapshot to the destination. Any changes on the destination or snapshots created after the common snapshot will be destroyed on the destination dataset. In case the common snapshot does not exist, dataset will be replicated from the scratch.

Automatic Handling Network Errors

With NexentaStor version 5.2.1 and above, HPR services are not faulted if a scheduled transfer fails because of network errors (e.g. connection failure, network request timeouts, non responsive remote node, or other similar scenarios). This minimizes operational overhead and simply improves service
resilience. When a transfer fails because of a network issue, an alert will be generated and logged, but the service will keep retrying every \texttt{hpr.syncRetryInterval}. If the network issue persists, the HPR service will continue to take snapshots at the source, according to its defined schedules. It will continue to delete old snapshots per the configured retention policies on source. If the network issue persists further and the oldest snapshot for a schedule ends up being the last common snapshot, the HPR service will stop taking new snapshots on source for that schedule to comply with the set retention policy. Throughout all this, the HPR service will continue to retry transferring snapshots.

Once the network issue is resolved, HPR services will seamlessly restart replicating snapshots and catch-up with their set schedules.

If for some reason you would rather have HPR services fault after a set number of retries (which is the behavior in versions prior to 5.2.1), that can be set using the \texttt{hpr.syncMaxAttempts} tunable.

See \texttt{List of All Default HPR Standard Configurations} and \texttt{Handling Alerts} sections for more details.

### Handling Destination Space Quota Errors

#### For Scheduled Replication

If a "Destination space quota exceeded" error occurs for SR, replication service will not be faulted, only alert will be emitted. Replication will start during the next schedule start without any manual intervention. Note that when configuring file system quotas on replicated file systems, you should use "refquota" so the capacity consumed by HPR snapshots isn’t counted against the set value.

#### For Continuous Replication

If a "Destination space quota exceeded" error occurs for CR, replication service will be faulted. Once you fix the destination quota space, you must change the state of the service from the Faulted state to the Disabled state in order to clear the Faulted state.

```
CLI@host> hpr clear <service-name>
```

### Handling Alerts

HPR alerts are generated in case of network or replication failures, when replication fails to start or scheduled snapshots fail. Use the set of "Alert" commands such as alert cases, alert reports, and alert acquit commands to troubleshoot replication problems. These set of alert commands provide a detailed list of the system events. Use alert cases and alert reports commands to list existing cases and display report with detailed information about the failure.

For more information on how to use the alert command to troubleshoot problems, see NexentaStor CLI Config Guide.

- To view the alert cases:

```
CLI@nexenta> alert cases [-av] [-u <uuid>] [-c <code>] [-t <time-spec>]
```
Example:

CLI@nexenta> alert cases --all

Each replication problem is represented by a code and reason that triggered the failure. Once the problem is fixed, alert case is repaired automatically, but it can also be acquitted manually, using the ‘alert acquit’ command.

The following table lists some of the HPR alerts and describes WHEN the cases are cleared automatically.

<table>
<thead>
<tr>
<th>Code</th>
<th>Reason for the alert</th>
<th>Cases cleared when</th>
</tr>
</thead>
<tbody>
<tr>
<td>NEX-8800-ED</td>
<td>Emitted when a service goes to faulted state.</td>
<td>Repaired when service is disabled (cleared, disabled with --force, recovered).</td>
</tr>
<tr>
<td>NEX-8801-ED</td>
<td>Emitted when a scheduled start failed because of a temporary error. In this case hpr service is not switched to faulted but alert is emitted.</td>
<td>Repaired when service is disabled (cleared, disabled with --force, recovered) or when a replication is finished successfully.</td>
</tr>
<tr>
<td>NEX-8802-WD</td>
<td>Emitted when a scheduled start is missed because another snapshot is already being replicated.</td>
<td>Repaired when missed snapshot is delivered to the destination.</td>
</tr>
<tr>
<td>NEX-8803-WD</td>
<td>Emitted when a scheduled start cannot create snapshot because number of non-replicated snapshots hits the source retention policy limit.</td>
<td>Repaired when next successful replication is finished.</td>
</tr>
<tr>
<td>NEX-8804-WD</td>
<td>Emitted when heartbeat to the agent node is lost.</td>
<td>Repaired when heartbeat is restored or service disabled.</td>
</tr>
<tr>
<td>NEX-8805-WD</td>
<td>Emitted when a network failure happens while replication was running. HPR emits this alert and tries to restart replication.</td>
<td>Repaired when a replication has successfully been run.</td>
</tr>
<tr>
<td>NEX-8807-ED</td>
<td>Emitted when the replication fails because the HPR snapshots on destination would run into quota limits on the destination file system.</td>
<td>Repaired when a replication has successfully been run.</td>
</tr>
</tbody>
</table>

**Note:** In a clustered environment, alerts are stored locally and they do not migrate with pools. When a pool is exported, alerts of exported hpr services remain in the node they were generated.

List of All Default HPR Standard Configurations

Nexenta does not recommend to change the default settings unless purely needed.
To view the list of all default HPR settings:

```
CLI@nexenta> config get all hpr
hpr.agingOutKeepTime - Period of time to keep deleted source datasets at
the destination (minutes). Default value is 10 days
hpr.syncMaxAttempts - number of retries that HPR will do before switching
the replication service to the faulted state. Default value is 0 which
means there is no limit of retry attempts
hpr.syncRetryInterval - time period between attempts to restart (in
milliseconds), the default value is 60000 which means 1 minute
hpr.connectTimeout - timeout to establishing connection to remote host,
ms. default value is 10 sec
hpr.dataAddress - IP address to receive replication data
hpr.dataPort - TCP/IP port to receive replication data, default value 6000
hpr.heartbeatFaultTolerance - Number of failed heartbeats before
reporting an error
hpr.heartbeatInterval - Interval of heartbeat requests (ms)
hpr.requestTimeout - Timeout of request to remote HPR node, default
value is 5 minutes
hpr.totalMemoryLimit - % of system memory HPR is allowed to use for
replication
```

Adjusting Memory Available for Replication Services

If a replication service fails to start or goes to a faulted state with a memory exceeded error, you should
increase the HPR memory limit.

Each replication service uses system memory to send and receive data. The size of the replication buffer can
be configured per service using maxBufferSize property which is, by default, 100MB for SR and 1GB for CR.
You can also configure the memory that is to be used for all replications.

```
CLI@nexenta> config get value hpr.totalMemoryLimit
NAME                  PROPERTY  VALUE
hpr.totalMemoryLimit  value     25
```

Where the “VALUE” defines the % of system memory HPR is allowed for the replication buffers.
Appendix

This chapter covers the following topics:

- About Moving Data using Standard OpenZFS Send/Receive
- Terms Used in this Appendix
- Use Cases and Direction of Replication Using ZFS Send/Receive
- Create Sendrecv Job
- List Sendrecv Jobs
- Modify Sendrecv Job
- Create Manual Snapshots to Send
- Run Sendrecv Job
- Implications of Using non-root Credentials
- View Status of Running Job
- Stop a Sendrecv Job
- Delete a Sendrecv Job

About Moving Data using Standard OpenZFS Send/Receive

NexentaStor 5.3 sendrecv provides you the flexibility to send and receive datasets to and from any OpenZFS based systems using the standard ZFS send/receive facilities over ssh. NexentaStor 5.3 sendrecv can be used to migrate data between NexentaStor 5.3 appliances and NexentaStor 4 appliances, or between NexentaStor 5 appliances and OpenZFS pools on Linux hosts. NexentaStor 5.3 sendrecv does not provide a scheduler, you must create snapshots manually (or use existing one) and then send them with a sendrecv job.

For use cases requiring movement of data between two NexentaStor 5 appliances, you should take advantage of the High Performance Replication (HPR) feature that is included with the NexentaStor Enterprise license, allowing you to configure automated replication schedules, site specific retention policies, etc.

See Chapter 3, Data Protection for more information on Scheduled and Continuous HPR services.

**Note:** Sendrecv feature is available with NexentaStor Enterprise Edition 5.3.
Terms Used in this Appendix

OpenZFS system
NexentaStor 4.x and Linux hosts running OpenZFS file systems that support standard ZFS send and receive data streams.

source
Source represents the dataset the data is sent from.

destination
Destination represents the dataset the data is received on.

Use Cases and Direction of Replication Using ZFS Send/Receive

The 2 primary use cases are:

• Data Migration from NexentaStor 4.x appliance to NexentaStor 5.3 Appliance.
• Data backup from NexentaStor 5.3 to Linux based OpenZFS Appliance.

To address the use cases above NexentaStor 5.3 sendrecv supports:

• Sending data to an OpenZFS system.
• Receiving data from an OpenZFS system.

Managing sendrecv Job

To run a sendrecv job, you must do the following:

• Identify source and destination hosts and datasets.
• Preferably, have root level ssh access on the OpenZFS system. Alternatively, you must have ssh access to the OpenZFS system with appropriate zfs permissions. Refer to the section “Implications of Using non-root Credentials”.
• Create a sendrecv job for the dataset on the NexentaStor 5.3 appliance.
• Create a manual snapshot of the dataset on the Source.
• Run the job you created to transfer the snapshot to/from the OpenZFS system.
Create Sendrecv Job

Sendrecv create command listed in this section creates a OpenZFS send-receive job. Each job is referenced with a UUID (universally unique identifier) which is generated when a job is created. The created job is stored on NexentaStor pool that will either send/receive data to/from the remote OpenZFS system.

Note: When the pool on which the send-receive job is stored is exported or destroyed, all jobs stored on this pool are also exported or destroyed. When a pool with jobs is imported, the jobs that are stored on this pool are also imported with the pool. If sendrecv jobs were actively running when a pool was exported, the replication streams will be interrupted and will have to be manually restarted once the pool is imported again.

Sendrecv uses ssh for its transport layer and every replication runs zfs send or zfs receive processes on local and remote host respectively to the replication direction. So when you create a sendrecv job, you also need to specify user name which will be used to login over ssh to the remote host.

When a job is created, sendrecv create command returns the UUID for that job. This UUID cannot be changed and is used to identify the job in get, list, set, destroy, stop commands.

❖ To create a sendrecv job to replicate from NexentaStor (local) to a remote OpenZFS system:

```
CLI@nexenta> sendrecv create <source> <destination>
```

Example:
```
CLI@nexenta> sendrecv create tank1/local_src root@dst.example.com:tank2/remote_dst
```

❖ To create a job to replicate data from a remote OpenZFS system to a local NexentaStor:

```
CLI@nexenta> sendrecv create root@src.example.com:tank2/remote_src tank1/local_dst
```

❖ To create a sendrecv job if ssh is running on a non-standard TCP/IP port:

```
In this case, specify the port explicitly in the sendrecv create command.
```

```
CLI@nexenta> sendrecv create root@src.example.com:2022:tank2/remote_src tank1/local_dst
```

You can also specify the remote hosts using hostname or IPV4 or IPv6 address enclosed in square brackets.

```
CLI@nexenta> sendrecv create root@dsthost:tank2/remote_src tank1/local_dst
CLI@nexenta> sendrecv create tank1/local_src root@192.168.0.1:tank2/remote_dst
CLI@nexenta> sendrecv create root@[fe80::250:56ff:feb8:d8f1]:tank2/remote_src tank1/local_dst
```

Note: In case of initial replication, the destination dataset should not exist. Otherwise replication will fail to start. The sendrecv forceRollback option can be used to automatically destroy and replace existing destination dataset.

❖ Replication of entire pool is not supported.
Sendrecv supports the following options that work exactly the same as they do for standard Openzfs send/receive commands:

- **-D, --deduplicated** Sends a deduplicated stream. Blocks which would have been sent multiple times in the send stream will only be sent once.

- **-F, --force** Forces a rollback of the destination dataset to the most recent snapshot before performing the receive operation. If receiving an incremental replication stream, destroy snapshots and datasets that do not exist on the source.

- **-I, --intermediate** Replicates all intermediate snapshots.

- **-L, --large-blocks** Sends a stream which may contain blocks larger than 128K.

- **-R, --recursive** Sends all descendent datasets. When received, all properties, snapshots, descendent file systems, and clones are preserved.

- **-e, --embedded** Sends a more compact stream by using WRITE_EMBEDDED records for blocks which are stored more compactly on disk by the embedded_data pool feature.

- **-p, --properties** Includes the dataset's properties in the stream. This flag is implicit when -R is specified.

- **-u, --unmountDestination** Does not mount destination filesystem(s).

- **-t, --throttle=N** Throttles connection speed (in bytes per second).

To create a job with additional options, either short or long names can be used:

CLI@nexenta> sendrecv create -LIR root@dsthost.com:tank2/remote_src tank1/local_dst

CLI@nexenta> sendrecv create --large-blocks --intermediate --recursive root@dsthost.com:tank2/remote_src tank1/local_dst
List Sendrecv Jobs

You may use the following commands to view the job you created and also to view the source and the destination of the sendrecv job.

- **To list all the sendrecv jobs:**

  CLI@nexenta> sendrecv list [-s <field>]... [-S <field>]...[-o <properties>] [-O <flags>]

  Example:

  CLI@nexenta> sendrecv list

  JOBID             SOURCE                    DESTINATION                                  SOURCESNAPSHOT RECURSIVE
  RUNNING  SUCCEED
  c0610240-----  tank1/local_src     root@10.3.199.135:tank2/remote_dst  -                          no         no
  no
  d436fb30-----  root@10.3.199.135:tank2/remote_src tank1/local_dst    -                           no         no
  no

  Additionally the following options can be used.

  -s, --sort=<field>    Sort the given fields in ascending order.
  -S, --sort-descending=<field> Sort the given fields in descending order.
  -o, --properties=<properties> A comma-separated list of properties to be displayed.
  -O, --output=<flags> A comma-separated list of output flags (for details please see help for sendrecv list command).

- **Alternatively, use the following command to verify:**

  CLI@nexenta> sendrecv get source, destination <job-id>

  Example:

  CLI@nexenta> sendrecv get source, destination 04ea0210-359a-11e7-b2d6-f303ba51a18e

  NAME                                  PROPERTY      VALUE
  04ea0210-359a-11e7-b2d6-f303ba51a18e  destination   root@dst.example.com:tank2/dst
  04ea0210-359a-11e7-b2d6-f303ba51a18e  source                    tank1/src

View Properties of the Created Sendrecv Job

- **To view the properties of the job:**

  CLI@nexenta> sendrecv get (all | <properties>) [-s <field>]... [-S <field>]...[-o <flags>] <job-id>
Example:

```bash
CLI@nexenta> sendrecv get all 04ea0210-359a-11e7-b2d6-f303ba51a18e
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>destination</td>
<td><a href="mailto:root@dst.example.com">root@dst.example.com</a>:tank2/dst</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>force</td>
<td>no</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>jobId</td>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>progress</td>
<td>0 %</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>runNumber</td>
<td>0</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>running</td>
<td>no</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>deduplicated</td>
<td>no</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>embedded</td>
<td>no</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>intermediate</td>
<td>no</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>largeBlocks</td>
<td>no</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>properties</td>
<td>no</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>recursive</td>
<td>no</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>sendSize</td>
<td>0.0K</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>source</td>
<td>tank1/src</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>succeed</td>
<td>no</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
<td>throttle</td>
<td>0.0K</td>
</tr>
</tbody>
</table>

**Note:** Some optional properties are not shown here (e.g. lastError, startTime, endTime). To see the list of all available properties, use `sendrecv get --help` command.

## Modify Sendrecv Job

You can modify any job that is not actively running using `sendrecv set` command. For example, to modify the properties on an existing job:

- **Use the `sendrecv set` command to modify:**

  ```bash
  CLI@nexenta> sendrecv set intermediate=no,recursive=no c0610240-3ee7-11e7-9ba5-4731930f747e
  
  Use the `sendrecv set --help` to obtain the full list of modifiable properties.
  ```

When the jobs are stored on local pools, you can change the path to remote dataset and path to the local dataset, however, you cannot change the local pool. Specifically, if a job was created to replicate data from/to local pool tank1, you can change source/destination dataset only within pool tank1.
If a pool, on which a job is stored is imported with another name, sendrecv will automatically update the path of the local dataset:

```
CLI@nexenta> pool export -y tank1
CLI@nexenta> pool import tank1 tankZ
CLI@nexenta> sendrecv list
```

```
JOBID               SOURCE                                                                 DESTINATION  SOURCESNAPSHOT  RECURSIVE  RUNNING  SUCCEED
SOURCESNAPSHOT  RECURSIVE  RUNNING  SUCCEED
```

```
d436fb30-        root@10.3.199.140:tank2/remote_src tankZ/local_dst                                  -
 no              no             no
```

```
c0610240-        tankZ/local_src                                  root@10.3.199.135:tank2/remote_dst   -
 no              no             no
```

Create Manual Snapshots to Send

- For NexentaStor 5.x:
  CLI@nexenta> snapshot create [-o <properties>] <snapshot>
  Example:
  CLI@nexenta> snapshot create [-r] pool/dataset@snap1

- For NexentaStor 4.x:
  nmc@nexenta:/$ create snapshot pool/dataset@snap1[-r]

- For OpenZFS System, use standard zfs command or refer to the appropriate document:
  root@openzfs: zfs snapshot [-r] pool/dataset@snap1
  where pool/dataset is the filesystem and snap1 is the name of the snapshot created.
  Now that you have created the snapshots of the filesystem manually, you can migrate the filesystem to the destination OS using the steps listed in the next section.

Run Sendrecv Job

To send the created snapshot to the destination dataset use the sendrecv run command with the specific snapshot name. If both source and destination dataset have the same common snapshot, source snapshot will be sent incrementally. If you do not specify the source snapshot, sendrecv run command will use the source snapshot from the previous run which allows to resume from the interrupted recursive replication.

```
CLI@nexenta> sendrecv run <job-id> <source-snapshot>
```

Example:

```
CLI@nexenta> sendrecv run 04ea0210-359a-11e7-b2d6-f303ba51a18e snap1
```
System response:

Please enter password: [enter password to login over ssh]

You may also specify the password along with the command. But when you include the password in the command line itself, the command history will record it. So including the password in the command line is generally not recommended.

```
CLI@nexenta> sendrecv run 04ea0210-359a-11e7-b2d6-f303ba51a18e snap1-p P@SsWord
```

When a job is in progress, you can see the following other properties in the sendrecv get all output.

- **startTime**: date and time when replication was started
- **sourceSnapshot**: name of snapshot that is being sent
- **progress**: replication progress in terms of percentage
- **running**: indicates that the job is currently in progress
- **sendSize**: number of bytes to send with this run

To check the result of replication, observe the following properties of the sendrecv job:

- **endTime**: date and time when replication completes
- **succeed**: if the replication completes successfully, this property will return true
- **lastError**: if the replication failed, the error message will be displayed in the lastError property

```
CLI@nexenta> sendrecv get succeed,lastError c0610240-3ee7-11e7-9ba5-4731930f747e
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>c0610240-3ee7-11e7-9ba5-4731930f747e</td>
<td>succeed</td>
<td>no</td>
</tr>
<tr>
<td>c0610240-3ee7-11e7-9ba5-4731930f747e</td>
<td>lastError</td>
<td>receive command failed with error: cannot receive incremental stream: destination tank2/parent_dst/remote_dst/1 has been modified since most recent snapshot.</td>
</tr>
</tbody>
</table>

Incremental Replication

When you run a sendrecv job, incremental replication is automatically applied if you specify a source snapshot and if an incremental common snapshot exists between source and destination. For example, if both source and destination have the common snapshot snap1, data will be replicated incrementally by sending only the deltas between snap1 and snap2.

```
CLI@nexenta> sendrecv run c0610240-3ee7-11e7-9ba5-4731930f747e snap2
```
Resuming Interrupted Recursive Job

If a recursive replication was interrupted or destination snapshot was partially destroyed, you may want to resume replication without re-sending snapshots that were already sent.

For example, if the source filesystem src has nested filesystems sub1 and sub2 and the replication was interrupted, due to network failure, when it was replicating src/sub1@snap1, you would have transferred snap1 on destination for the parent file system, but will be missing snap1 for nested filesystems sub1 and sub2.

In order to resume replication just run replication again, without specifying source snapshot:

```
CLI@nexenta> sendrecv run c0610240-3ee7-11e7-9ba5-4731930f747e
```

In this case replication will be resumed and only src/sub1@snap1 and src/sub2@snap1 will be sent.

Implications of Using non-root Credentials

When using non-root credentials (for example, user “admin” below) on the OpenZFS system, the appropriate set of zfs operations must be explicitly allowed for that user.

When replicating to NexentaStor, you should allow send and hold snapshots on the source dataset:

```
root@openzfs:~# zfs allow -u admin send,hold tank2/remote_src
```

When replicating from NexentaStor you should allow to receive and create dataset on the destination parent:

```
root@openzfs:~# zfs allow -u admin create,receive tank2
```

If the job is created with --sendProperties options, you should also allow reception of the transferred properties as well, e.g:

```
root@openzfs:~# zfs allow -u admin create,receive,compression,readonly,userprop tank2
```

Note: userprop allows to replicate user properties that contain ":" in names.

When recursively replicating filesystems, unmountDestination option should be enabled or alternatively you must use mountpoint=none on the parent destination filesystem to ensure OpenZFS will not automatically mount any received nested filesystems.

View Status of Running Job

```
CLI@nexenta> sendrecv get progress 04ea0210-359a-11e7-b2d6-f303ba51a18e
```

<table>
<thead>
<tr>
<th>NAME</th>
<th>PROPERTY</th>
<th>VALUE</th>
</tr>
</thead>
<tbody>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e jobld</td>
<td></td>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e</td>
</tr>
<tr>
<td>04ea0210-359a-11e7-b2d6-f303ba51a18e progress</td>
<td></td>
<td>10 %</td>
</tr>
</tbody>
</table>

Stop a Sendrecv Job

To interrupt currently running replication use the sendrecv stop command:
Delete a Sendrecv Job

To destroy replication job, use the sendrecv destroy command:

```
CLI@nexenta> sendrecv destroy 04ea0210-359a-11e7-b2d6-f303ba51a18e
```

In order to destroy a currently running job, --force option can be used:

```
CLI@nexenta> sendrecv destroy -f c0610240-3ee7-11e7-9ba5-4731930f747e
```
List of Options Available

The following table lists some of the options that are available to be used with the sendrecv commands.

**Sendrecv Create Command**

- **-D, --deduplicated** Sends a deduplicated stream. Blocks which would have been sent multiple times in the send stream will only be sent once.

- **-F, --force** Forces a rollback of the destination dataset to the most recent snapshot before performing the receive operation. When receiving an incremental replication stream, destroy snapshots and datasets that do not exist on the source.

- **-I, --intermediate** Replicates all intermediate snapshots.

- **-L, --large-blocks** Sends a stream which may contain blocks larger than 128KB.

- **-R, --recursive** Sends all descendent datasets. Use this option to preserve all the properties, snapshots, descendent file systems and clones.

- **-e, --embedded** Sends a more compact stream by using WRITE_EMBEDDED records for blocks which are stored more compactly on disk by the embedded_data pool feature.

- **-p, --properties** Includes the dataset's properties in the stream. This flag is implicit when -R is specified.

- **-t, throttle** Throttles connection speed (in bytes per second).

- **-n, --dry-run** Validates command line input without making any changes.

- **-v, --verbose** Prints detailed information.

- **-h, --help** Prints short or detailed usage information.

**Sendrecv list Command**

The following table lists some of the options that are available to be used with the sendrecv list command.

- **-s, --sort** Sorts the given fields in ascending order.

- **-S, --sort-descending** Sorts the given fields in descending order.

- **-O, --output** A comma separated list of output flags.
Sendrecv get Command

The following table lists some of the options that are available to be used with the sendrecv get command.

- **-s, --sort**  
  Sorts the given fields in ascending order.

- **-S, --sort-**  
  Sorts the given fields in descending order.

- **-O, --output**  
  A comma separated list of output flags.

Sendrecv run Command

The following table lists some of the options that are available to be used with the sendrecv run command.

- **-p, --password**  
  Password to use when you connect to a remote host. If you do not provide the password along with the command, you will be prompted for one.

Example:

Following is an example of remote-to local replication where the source is a non-NexentaStor appliance.

```
CLI@Nexenta: sendrecv run 8fbc1300-359b-11e7-b2d6-f303ba51a18e snap1
```

Please enter password: