NexentaStor 5.0 High Performance Replication (HPR) User Guide

Date: January, 2017

Subject: HPR

Software: NexentaStor

Software Version: 5.0.2

Part Number: 3000-HPR-5.0-000058-A

Product	Versions supported
NexentaStor™	5.0.2

Contents

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:

Revision	Date	Description
3000-HPR-5.0-000058-A	January, 2017	GA

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.

1

Introduction

This section includes the following topics:

Definitions

This document uses the following terms.

Terms	Definition
Replication group	Replication group is a set of NexentaStor appliances that share the same Replication Password.
Scheduled replication (SR)	Scheduled replication is the default type of replication service that comes with the Enterprise Edition license. 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 to a remote NexentaStor appliance.
Continuous replication (CR)	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 performance application.
Source	Source represents the node where the data is replicated from.
Destination	Destination represents the site where the data is replicated to and can be either local dataset on the same appliance or a remote NexentaStor appliance. Note: When the replication service is running and if the destination dataset is a filesystem, the filesystem gets unmounted.
Primary	Primary represents the node where the replication service manager is running. See below for more details about Service Manager. So primary appliance can be either source or destination. See About Service Manager and Service Agent for more details on the Manager and Agent.
Secondary	Secondary represents where the replication service agent is running. See below for more details about Service agent. Secondary appliance can be either source or destination.

Terms	Definition
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 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 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
```

Note:

- You can configure only one CR service per dataset, whereas you can configure more than one SR service.
- NexentaStor HPR also supports multi-destination configurations by allowing multiple SR and single CR services to be configured on the same Primary site dataset.

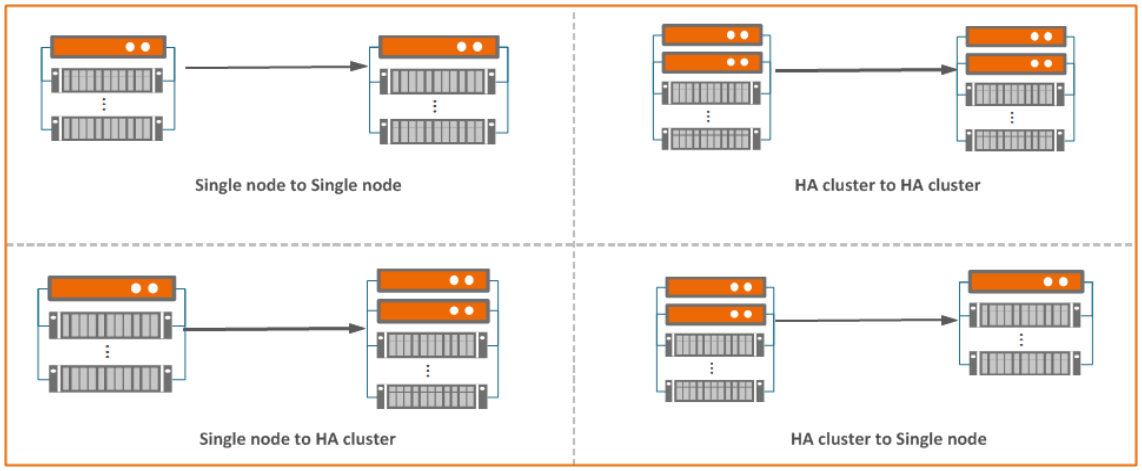
To see the limitations on the operations for CR or SR service, see Operational Consideration for Continuous Replication Services and Operational Considerations for Scheduled Replication Services.

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



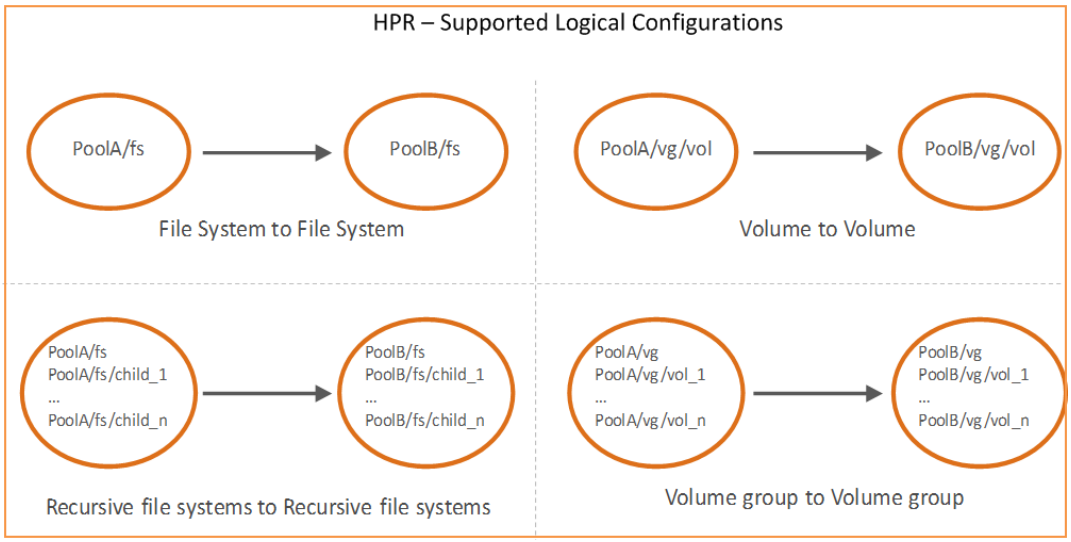
Supported Logical Configurations

HPR can be configured on:

- A file system

  Recursive on a parent file system with nested children file systems (i.e. replicate the parent and all the underlying file systems, with transaction level consistency).
- A volume group
- A volume

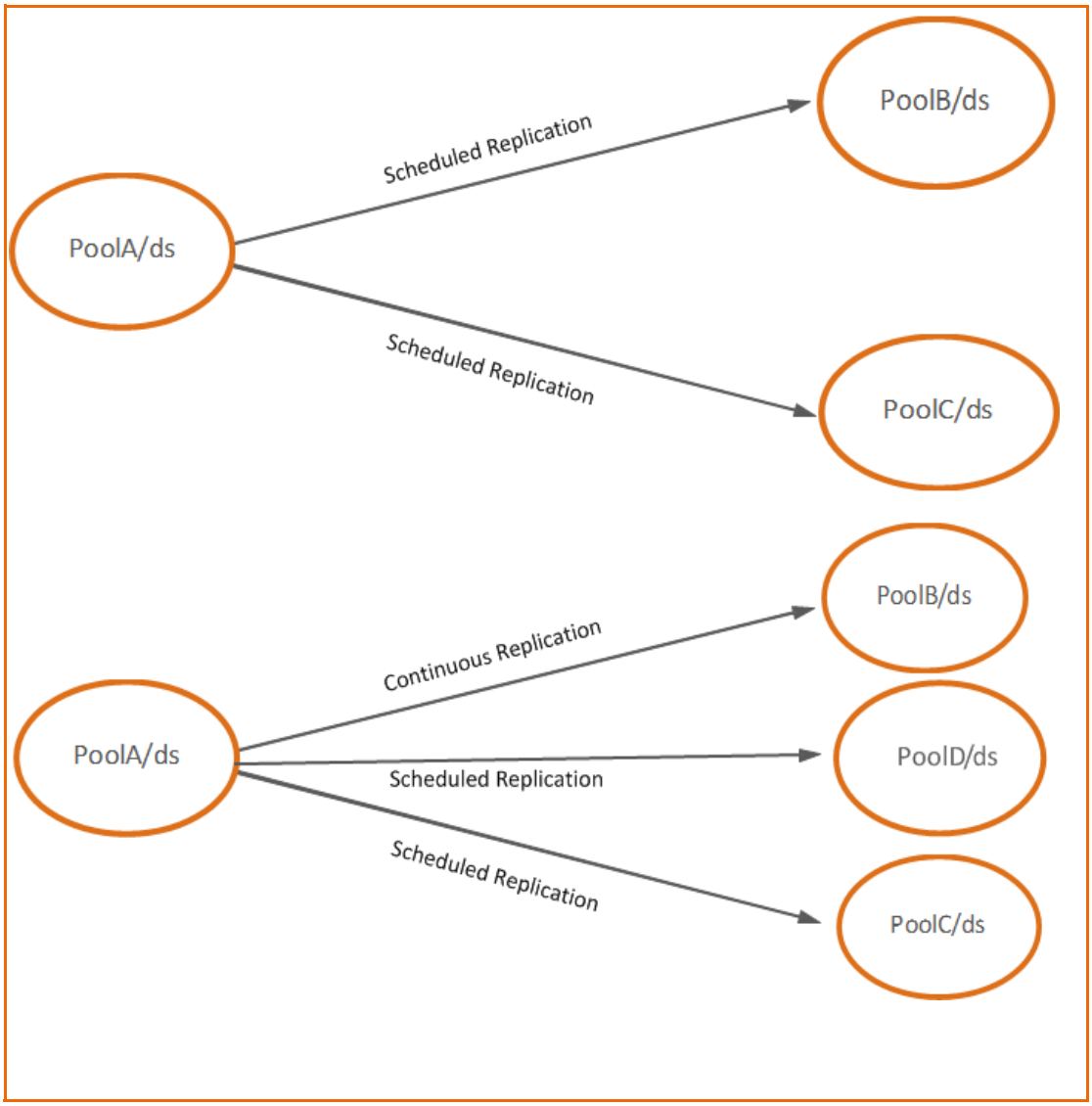
Recursive on volume groups (i.e. replicate all the underlying volumes, with transaction level consistency).



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 deleted.

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.

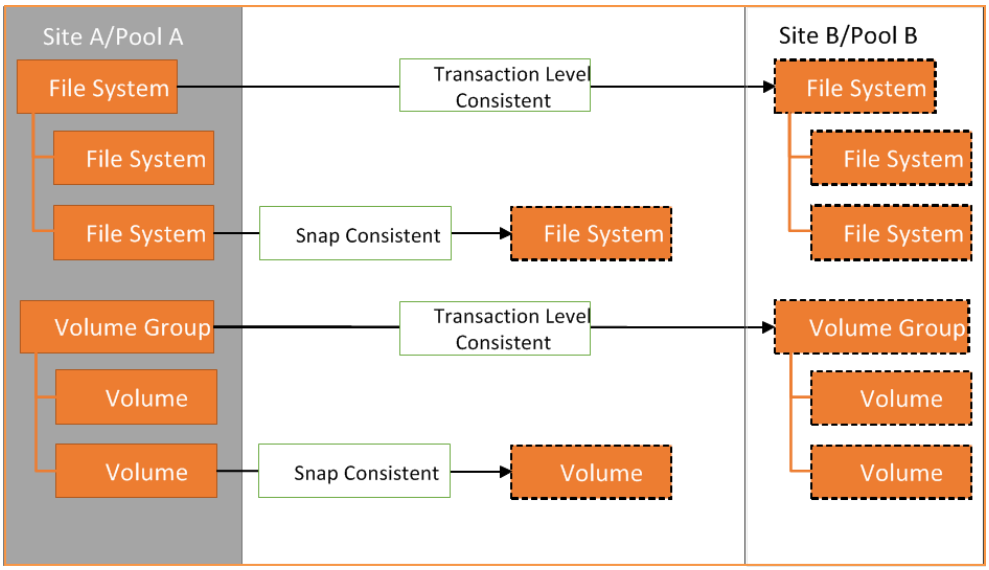
Features and Target Use Cases

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 two 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 archival of data: in this use case, snapshots of data on the primary site are replicated to a remote site for long term archival. 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 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 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 that 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 exist 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.

HPR 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 for more details.

An HPR service is defined by:

- A type: Scheduled or Continuous
- A primary dataset or datasets
- A secondary dataset or datasets
- 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 information. If HPR detects that a destination dataset was changed (by the user or otherwises) 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 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.

   Replication service executes according to schedule. On every run, it creates a snapshot of the selected dataset.

2. Starts replication only if previous run is finished.

   If previous replication is still ongoing, current replication service will terminate after creating the snapshot.

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

4. On the next scheduled run, all unreplicated snapshots including newly created one will be sent to the destination.

5. Before the run, HPR determines the latest identical snapshots at Primary appliance and Secondary appliance.

   Replication service compares the lists of snapshots at Primary appliance and at Secondary appliance and locates a pair of latest identical snapshots.

6. Sends the incremental stream from the common snapshot (the incremental source) to the current snapshot (the incremental Destination) if a change is detected.

   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 CR Service:

For CR service, replication creates snapshots called autosnapshots every time when 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 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

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.

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:*

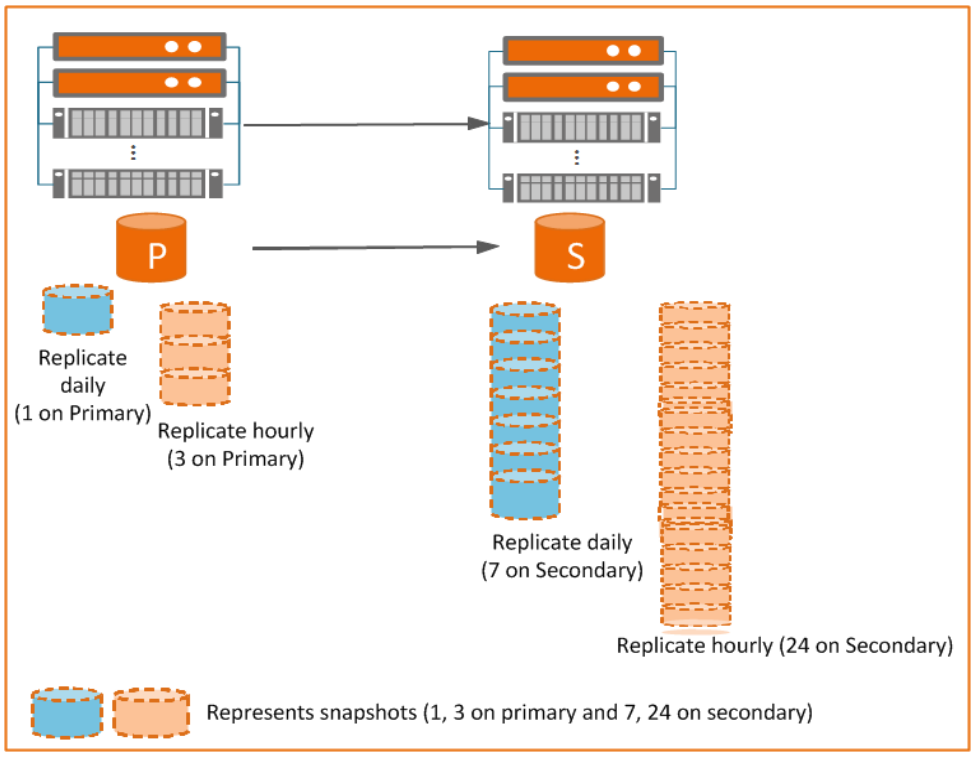
  ```
  CLI@nexenta> hpr get all <hpr service name>
  ```

  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 all test
  NAME     PROPERTY               VALUE
  test     ignoreProperties       -
  test     includeAllSnapshots    no
  test     isManager              no
  ```

Schedules, Snaplists and Retention Policies

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



As a schedule runs, it creates snapshots and adds to schedule specific snaplists, one at each end of a replication service. When creating a schedule, you can also 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 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, is 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 the next time the schedule runs.

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 deleted during the entire time the service is flipped.

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 user created 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 dataset DS_B at the destination
- No manual snapshot may be configured on DS_B
- If the content of DS_B is modified, the SR service will fail. You may restart the SR service using the CLI “hpr run-once --force” command. This will cause the SR service to restart replication from the last common snapshot between DS_A and DS_B
- Local snapshots and local snapshot schedules can be configured on DS_A, independent of replication services
- Multiple schedules with their own retention policies can be created as part of the SR service on DS_A.
- Local snapshots 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.
- DS_A may not be renamed while SR is enabled and service is running.

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 or 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 rename or delete a child dataset, the user must first disable SR, then manually rename or delete the child dataset on DS_A and its corresponding child dataset on DS_B and enable service again.

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 manual snapshot may be created on DS_B
- If the content of DS_B is modified, the CR service will fail and change to faulted state. The user may restart the CR service using the CLI “hpr run-once --force” command, causing CR to recover from its last common checkpoint
- 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.

- DS_A may not be renamed while CR is enabled.

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 or deleting child file systems in DS_A (or volumes if DS_A is a volume group) is not supported while CR is enabled. To rename or delete a child dataset, the user must first disable CR, then manually rename or delete the child dataset on DS_A and its corresponding child dataset on DS_B and enable service again.

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 in NexentaStor 5 focuses on replicating the content of datasets. Specifically, file system sharing configuration and LUN mapping information must be managed separately and are not handled as part of an HPR service.
- HPR does not replicate sharing properties such as LUN mappings, sharesmb, sharenfs and mountpoint properties.
- 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 intermediate snapshots.

  Intermediate snapshots are snapshots created by a user or 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.
- While NexentaStor supports snapshots and clone operations on datasets that are part of an HPR service, the clone promote operation is not supported when an HPR service is enabled.
- HPR does not support creating manual snapshots on the destination dataset of an HPR service.
- 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.
- HPR does not support replication in a clustered environment (HA) if you configured the NEF management IP as 0.0.0.0. If you configured the NEF management address as 0.0.0.0, HPR fails to detect the active node management address.

- 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 full ground-up 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
- 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.

Finally, as a result of tighter management of snapshots and replication specific snaplists, HPR does not provide the ability to replicate “intermediate snapshots” that are created on the source dataset outside of the replication service.

2

Preparing Networks and Appliances for HPR

This section includes the following topics:

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 `coninuousReplication` 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      5.0.0
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 2016
capacity     no limit
```

```
maintenance  Thu Sep 29 17:00:00 2016 - Sun Nov 13 16:00:00 2016 (valid)
```

To activate a license key with the optional CR feature:

```
CLI@nexenta> license activate <key>
```

Setting and Updating Replication Password

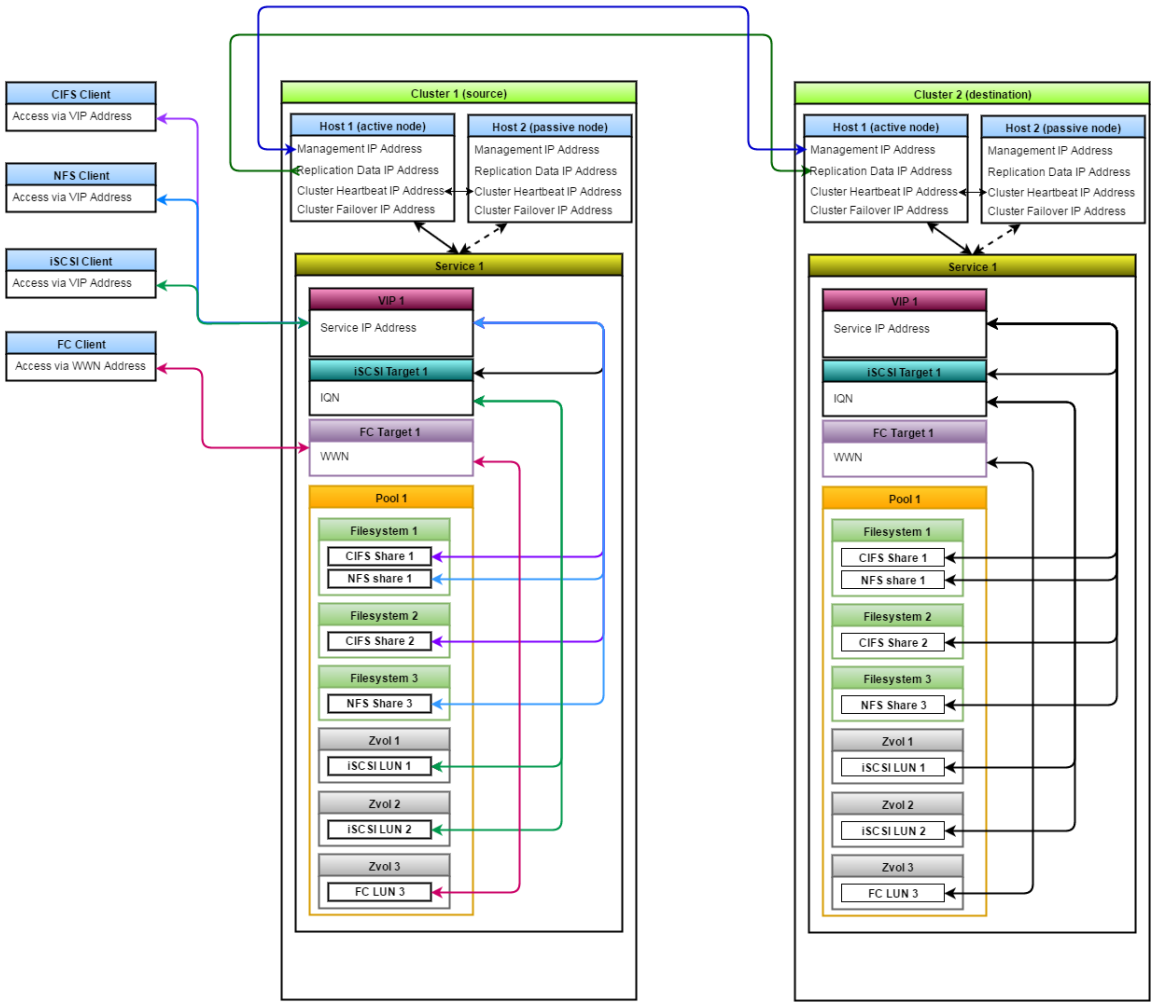
Before configuring replication services between NexentaStor appliances, you must ensure that they are part of the same replication group. A replication group is defined as a set of NexentaStor appliances that share the same Replication Password. Password must be the same for all nodes in the replication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

```
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. You must add these ports as exceptions when configuring the firewall so that firewall does not block all connections to a NexentaStor appliance that uses these ports.

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

Interface	Required and Recommended
Incoming to NexentaStor Management Interface	TCP ports 22 (SSH), 6557, 5557, 8443 (REST API)
Outgoing from NexentaStor appliance	SNMP (161, 162), DNS (53 UDP/TCP) , NTP (UDP 123), ICAP (port 1344)
Outgoing to Internet	usftp.nexenta.com for uploading (20,21,28000-30000) (ftp, ssl) https://licensingservice.nexenta.com/ (443) (ssl) https://logcollector.nexenta.com (443) (ssl) https://prodpkg.nexenta.com (443) (ssl)
Support Bundles	TCP port 21 (FTP)
Outgoing to the mail server	TCP port 25 or 587 for relay only (SMTP)
Between the interfaces with the hpr.dataAddress for Replication Data Protocol	TCP port 6000 by default but can be changed
Between NexentaStor Management Interfaces of the clustered nodes	UDP and TCP port 1195
For data traffic on any data interfaces:	
iSCSI Target	TCP port 3260
SMB	UDP ports: 137, 138 TCP ports 137, 139
CIFS	UDP and TCP port 445
NTP	UDP port 123
NFSv3	UDP ports: 111, 2049, 4045, 32768:65535 TCP ports: 111, 2049, 4045, 32768:65535
NFSv4	TCP port 2049

Setting HPR Address to Send or Receive Replication Traffic

Prerequisites

An interface must be configured with a static IP address 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. Please refer to 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. To view existing values for HPR system properties:

```
CLI@nexenta> config get value hpr.dataAddress
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. To set HPR service data address:

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

Example:

```
CLI@nexenta> config set hpr.dataAddress=10.3.10.38
```

3. To verify the HPR service data address:

```
CLI@nexenta> config get value hpr.dataAddress
PROPERTY  VALUE
path      hpr.dataAddress
type      variable
value     10.3.10.38
```

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

3

Managing Replication Services

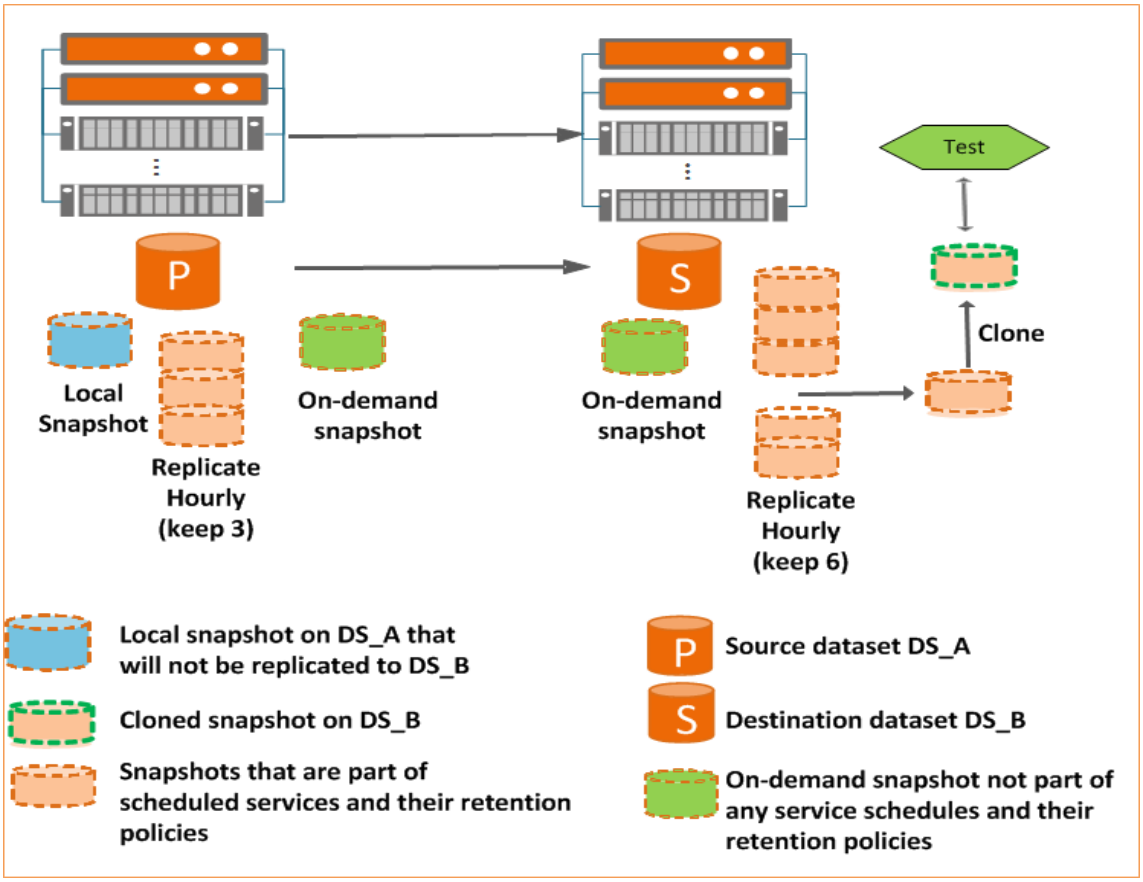
This section includes the following topics:

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 remote appliance. Only snapshots created within the context of an SR service on 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 automatically put back in its original SR schedule snaplist 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.

- 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”.

Note:

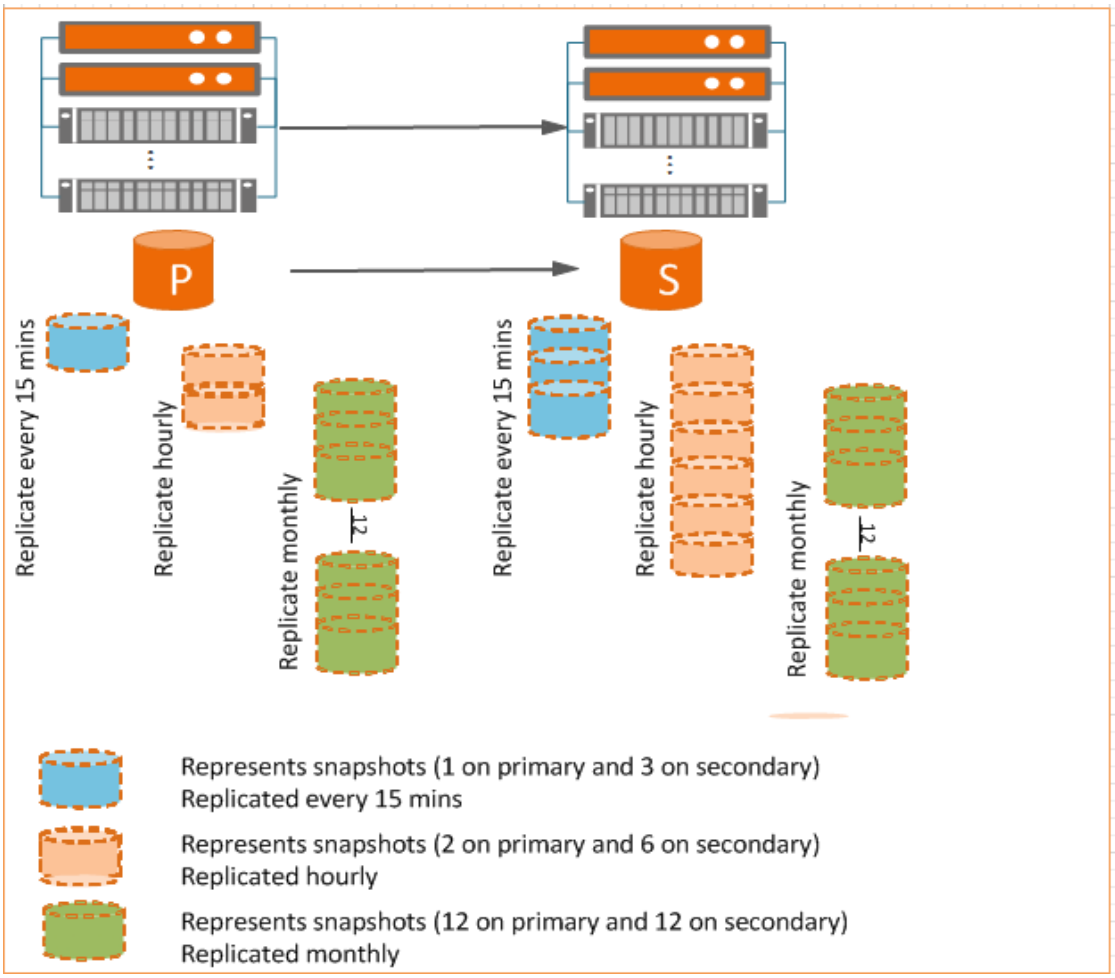
- Immediately after an on-demand replication completes, the replicated on-demand snapshot is the most recent common snapshot for the service. It should not be deleted on either source or destination until after another snapshot has been transferred.

Managing Scheduled Replication Services

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

- A pre-existing source dataset (appliance, pool and dataset path and name)
- A path to a remote destination dataset (appliance, pool and full dataset path and name). 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 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://host2/
   pool2/dst <service-name>
   ```

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

   ```
   CLI@host> hpr schedule-add [-nv] <service-name> <cron> <keep-source>
   <keep-destination> [<schedule-name>]
   ```

   Example:

- Every 15 minutes keeping 1 locally and 3 remote

  ```
  CLI@host1> hpr schedule-add <service-name> “0,15,30,45 * * * *” 1 3
  <schedule-name>
  ```

- 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 locally and 12 remote

  ```
  CLI@host1> hpr schedule-add <service-name> “0 0 1 * *” 12 12 <schedule-
  name>
  ```

3. **Enable the replication service**.

   ```
   CLI@host1> hpr enable <service-name>
   ```

Add Schedules to a Replication Service

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.

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”.

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 3rd field indicates every month between January (1) and December (12), inclusive.

Enable Service

Replication services of type `continuous` starts 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 `hpr list` command. RUNNING state refers if it's currently replicating or waiting for the next SR.

```
CLI@host> hpr list
NAME     TYPE      RECURSIVE  SOURCE    DESTINATION              STATE
RUNNING
service  scheduled yes        pool1/src https://host2:8443/pool2/dst
enabled  no
```

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 <service-name>
NAME     PROPERTY           VALUE
service  destination        https://host2:8443/pool2/dst
service  heartbeat          yes
service  ignoreProperties   -
service  isManager          yes
service  running            no
service  isSyncing          no
service  maxBufferSize      100
service  name               service
service  recursive          yes
service  replaceProperties
service  source             pool1/src
service  state              enabled
service  type               scheduled
```

When you execute the `HPR get all` command, it displays 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.

```
CLI@host> hpr get all <service-name>
NAME     PROPERTY           VALUE
service  destination        https://host2:8443/pool2/dst
service  heartbeat          yes
service  ignoreProperties   -
service  isManager          yes
```

```
service  running            no
service  isSyncing           no
service  lastError          Destination has been modified since most
recent snapshot
service  maxBufferSize      100
service  name               service
service  recursive          yes
service  replaceProperties
service  source             pool1/src
service  state              faulted
service  type               scheduled
```

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>
```

Verify Service Schedule and Service Snapshots

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

To verify that the retention policy is met, compare that the number of snapshots per schedule is equal to "keep" value for source or destination respectively.

- *To verify replicated snapshots and retention policy on the source node:*

```
CLI@host1> hpr schedules <service-name>
NAME              CRON         KEEPSOURCE  KEEPDESTINATION  DISABLED
schedule-hourly   19 * * * *   1           24               no
schedule-monthly  41 14 9 * *  12          12               no


CLI@host1> hpr snapshots <service-name>
CREATION         PATH                                      SENT  SCHEDULE
Dec 8 14:19:00 pool1/src@hpr-2016-12-08-14-19-00-785 yes  schedule-
hourly
Dec  9 14:41:00  pool1/src@hpr-2016-12-09-14-41-00-945  yes   schedule-
monthly
```

- *To verify replicated snapshots and retention policy on the destination node:*

```
CLI@host2> hpr snapshots <service-name>
CREATION         PATH                          SENT  SCHEDULE
```

```
Dec  8 14:19:00  pool2/dst@hpr-2016-12-08-14-19-00-785           yes
schedule
Dec  8 14:37:56  pool2/dst@hpr-ondemand-2016-12-08-14-37-56-303  yes   -
Dec  8 14:37:58  pool2/dst@hpr-ondemand-2016-12-08-14-37-58-394  yes   -
Dec  8 14:38:01  pool2/dst@hpr-ondemand-2016-12-08-14-38-01-033  yes   -
Dec  8 14:38:43  pool2/dst@hpr-ondemand-2016-12-08-14-38-43-337  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 agent is not reachable this command will fail with the related error.

Note:

- When you run service once from disabled state, it temporary switches to enabled and then, when replication completed it goes to disabled state automatically.
- 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 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 immediately.
- 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 the running replication immediately, disables the service by ignoring the errors.

  When you force disable a running replication services 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 on-demand snapshots and replicate immediately without waiting for the next SR. In this case you have two options to `run-once` or `run-once with force` flag to overwrite data on destination. SR Service must not be running when you execute this command.

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

Destroy Replication Service

Only disabled service can be destroyed. In order to destroy service in any other state, `--force` option should be used. During the service destroy, source and destination snapshots and destination datasets, created by this service also can be destroyed.

```
CLI@host> hpr destroy <service-name>
CLI@host> hpr destroy --help
Options:
  --source-snapshots      Destroy snapshots created by this service on
                          source dataset.
  --destination-snapshots Destroy snapshots created by this service on
                          destination dataset.
  --destination           Destroy destination dataset.
 -f, --force              Destroys service regardless if it is running or
                          agent node's availability.
```

For additional information on the commands used in this section, type:

- For the man page of the HPR command and its subcommands pre-pended with service

  ```
  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
  ```

Modifying an Existing Schedule

You can modify an existing schedule if the HPR service is disabled. The name, cron and retention policy can be modified.

Note: Modification of cron or retention policy might result in loss of snapshots. For example, changing the retention policy on destination from 10 to 5, will remove oldest 5 snapshots after next replication.

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 snaphots 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_everyhour with an hourly schedule keeping 6 snapshots on DS_A and 12 snapshots on DS_B
- You can destroy SR_every hour. 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_everyhour 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 can be used as “most recent common” snapshots. They also 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 by the retention policy the next time the schedule runs.

List Snapshots of a Given Replication Service

The following sections enumerate the CLI commands involved in deleting, and managing snapshots associated with a replication service

```
CLI@host> hpr snaplists <service-name>
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/vol1@hpr-ondemand-2016-12-13-22-52-34-349
yes    -
```

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.

```
CLI@host> hpr snaplist-find [-s <field>]... [-S <field>]...[-O <flags>]
<service-name>

SNAPLISTID                                          SERVICE  SCHEDULE
CRON         SOURCESNAPSHOTS  DESTINATIONSNAPSHOTS

39996710-a127-11e6-be66-3f0c3222b645 -     -      14 * * * *  24
0

4c61e7a0-a127-11e6-be66-3f0c3222b645 -     -      17 18 * * *  31
0
```

Existing snapshot lists can be claimed to a schedule using the `hpr 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 remote 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)
- A path to a remote destination dataset (appliance, pool and full dataset path and name). 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.

Then do the following in the order listed.

- Create new replication service
- Enable the replication service

Create Replication Services

Only single CR can be created for a given dataset. Before creating a new HPR service identify the dataset to be replicated, if dataset is a consistency group (volume group or file system parent) then use the --recursive parameter. In the case of a CR service, you cannot share the destination between the multiple replicated services.

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

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 always returns a value YES.

```
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 source and destination and reports the same state. Note: in the case of CR service, RUNNING state will always be YES when the service is enabled.

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

When you execute the `HPR get all` command, it displays 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`.

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

```
CLI@host> hpr get all <service-name>
NAME     PROPERTY           VALUE
service  destination        https://host2:8443/pool2/dst
service  heartbeat          yes
service  ignoreProperties   -
service  isManager          yes
service  running            no
service  isSyncing          no
service  lastError          Destination has been modified since most
recent snapshot
service  maxBufferSize      100
```

```
service  name                service
service  recursive           yes
service  replaceProperties
service  source              pool1/src
service  state               faulted
service  type                scheduled
```

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 causes switching 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 currently running continuous service use --force option, without swiching to the stopping state, to stop replication immediately.

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

Destroy Replication Service

Only disabled service can be destroyed. In order to destroy service in any other state you must use the --force option. During the service destroy, source and destination snapshots are automatically destroyed. The destination dataset, created by this service also can also be destroyed.

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

Activating Filesystem

The `hpr activate` command is used in disaster recovery scenarios. While replication is running the destination file system is unmounted. Following a failure at the source site, once replication is disabled, the destination file system must first be “activated” before it can shared via NFS or SMB, as appropriate.

From a technical perspective, the `hpr activate` command will mount an unmounted file system to its default mount point and makes it available for sharing configuration.

```
CLI@node-a> hpr activate pool/filesystem
```

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

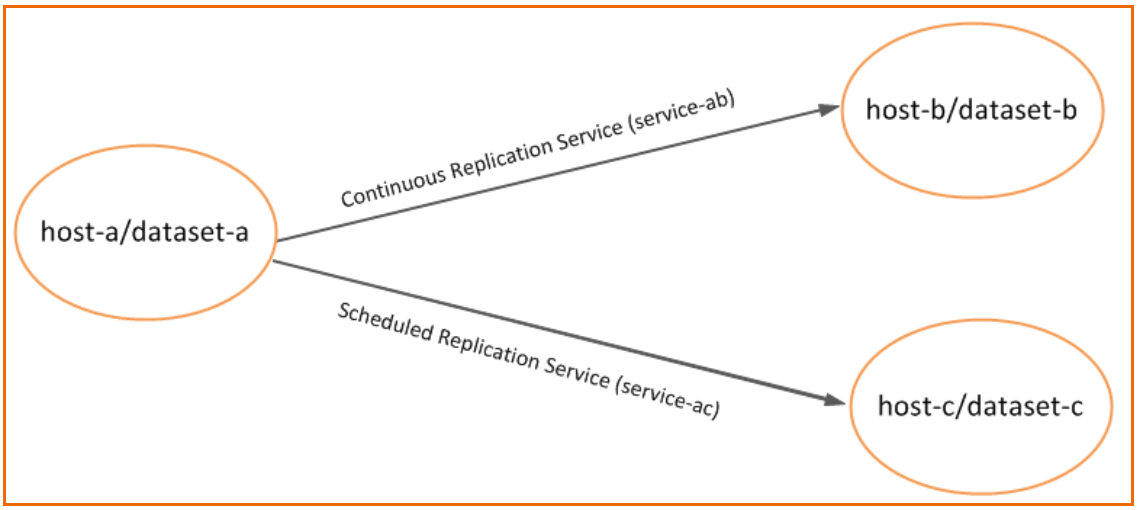
Managing Multi-Destination Replication Services

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, destroy all the services that replicates the source dataset except that one which needs to be flipped and after replicating the changes back on the source, recreate the deleted services.

4

Disaster Recovery Use Cases

This section includes the following topics:

Warning:	This chapter describes the advanced configuration options for HPR service. Nexenta recommends that you use this functionality only as disaster recovery solutions.

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

- 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 and clone. Alternatively a parallel SR can be set up to a different dataset on the destination B.

Disaster Recovery by Creating Clones at the Destination Appliance

You can separately manage the snapshots copied from the source appliance on the destination appliance. To manage these snapshots, clone them on the destination appliance. These cloned datasets you can read-write, rename or destroy as needed and subsequently be shared via file or block protocols. 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 snapshot is automatically put back in its original SR schedule snaplist and will eventually be deleted per the SR schedule retention policy.

Note: Clone feature is supported both for scheduled replication (SR) and CR.

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

This section uses the following names as examples:

- Datset on the destination node (node B) pool/dataset-b
- clone on the destination node (node B) pool/clone

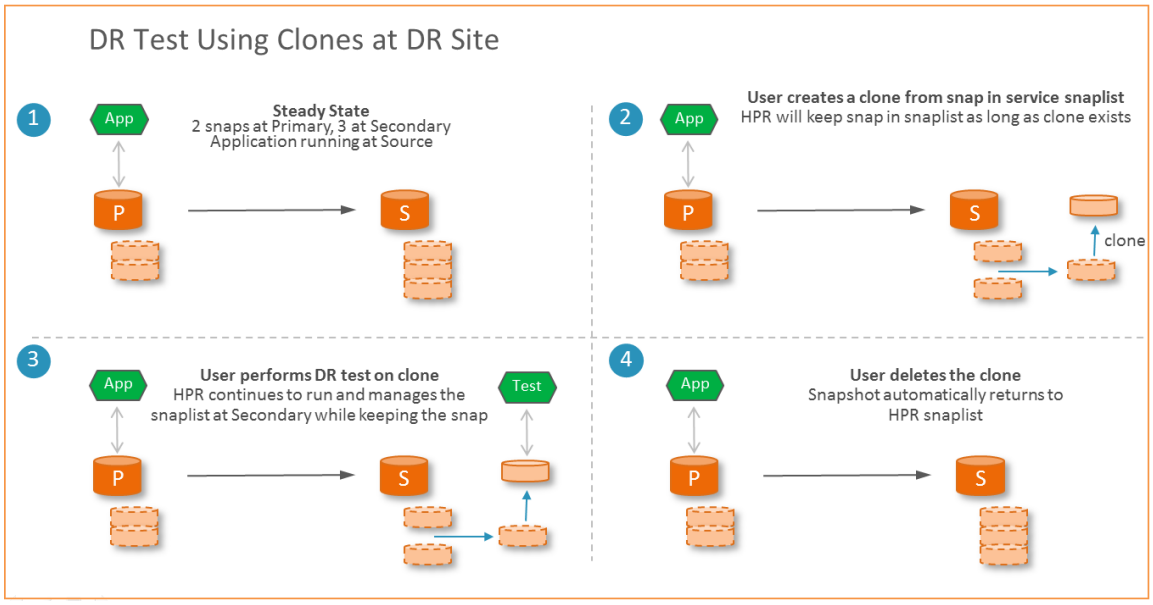


Table 4-1: Steps to Manage the Cloned Snapshots at the Destination Site

#	Description

1	**Create writable clones of the snapshot**
	For SR, you can clone using the existing snapshots whereas for CR service, create a destination snapshot and then enable service. You can clone the snapshot either before or after enabling the service.
	1. For CR service, disable HPR service at the primary appliance.
	`CLI@node-a> hpr disable <service>`
	2. In case of CR service, take recursive snapshot for the destination dataset. For SR use the existing snapshots to clone.
	`CLI@node-b> snapshot create -r pool/destination@snapshot`
	3. Clone the snapshots owned by the HPR service on the destination appliance.
	`For filesystem:`
	`CLI@node-b> for i in $(filesystem list -r -O basic -o path $dataset | sort); do snapshot clone -v $i@snapshot ${i/$dataset/$clone}; done`
	For volume:
	`CLI@node-b> dataset=pool/destination`
	`CLI@node-b> clone=pool/clone`
	`CLI@node-b> for i in $(volume list -O basic -o path $dataset | sort); do snapshot clone -v $i@snapshot ${i/$dataset/$clone}; done`
	4. Share the cloned filesystem.
	`CLI@node-b> nfs share pool/clone`
	`CLI@node-b> smb share pool/clone`
	5. Or create lun mappings for cloned volume.
	`CLI@node-b> lunmapping create pool/clone/volume target-group host-group`

#	Description

2	**Manage the cloned snapshots**
	1. Mount it on clients (if destination dataset is filesystem) or connect it on 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> for i in $(filesystem list -r -O basic -o path pool/clone | sort -r); do smb list $i && smb unshare -v $i; nfs list $i && nfs unshare -v $i; done`
	For volumes:
	Remove lun mappings on volume group (if destination dataset is volume or 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`
	`CLI@node-b> volume destroy -rv pool/clone`

Graceful Failover, Flip Direction and Sync-back

User Scenario

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

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

1. Stop the application running off of dataset A
2. For dataset A, unshare filesystem or remove lun mappings on volume
3. Forcibly disable HPR service on Node A
4. Run replication one more time for consistency, to replicate delta to dataset B
5. Ensure the following checks: Validate that the replication has completed; the HPR service is not running; and the latest snapshots match on the source and destination
6. Perform flip operation to reverse replication direction.

   Source dataset A becomes destination and destination dataset B becomes source
7. If dataset B is a filesystem, activate filesystem B. Activate feature resets mount point of file system to default value and mounts that file system
8. Share file system B or create lun mappings for volume or volume group B
9. Start the application on dataset B
10. Enable HPR service on node A

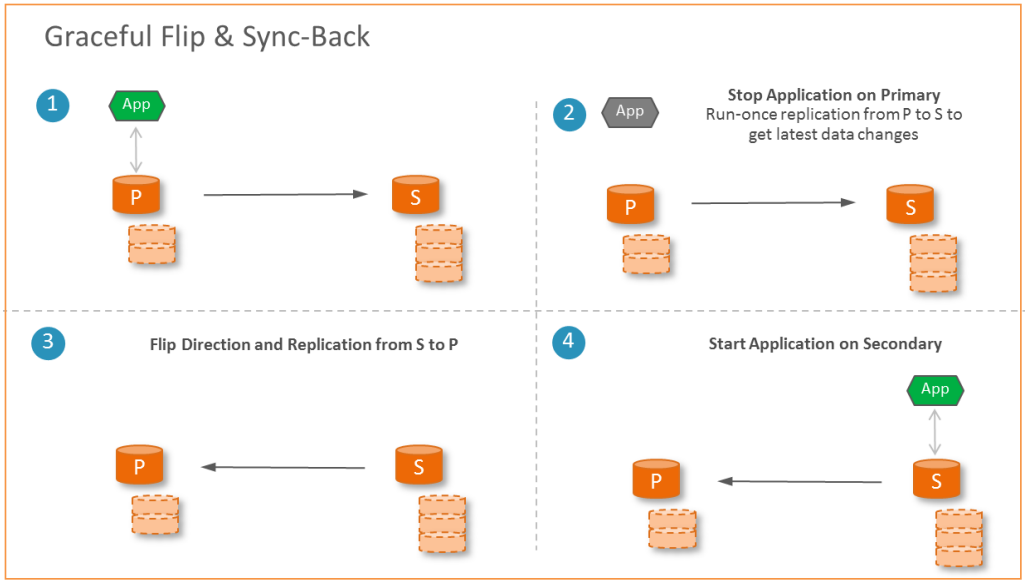


Table 4-2: Steps to Reverse the Replication using Flip-Direction

#	Description

1	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; 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 $i; done`
	3. Forcibly disable the HPR service on the primary host.
	`CLI@node-a> hpr disable -f <service-name>`
	4. Run replication one more time for consistency, to replicate delta to dataset B.
	`CLI@node-a> hpr run-once <service-name>`
	5. Validate that the replication has completed, ensure that the HPR service is not running, and the latest snapshots match on source and destination.
	`CLI@node-a> while true; do`
	`state=$(hpr get running -O basic service | nawk '{print $NF}')`
	`if [[ "$state" != "yes" ]]; then`
	`echo "service: last time replication done."`
	`break`
	`fi`
	`echo "service: waiting for last time replication..."`
	`sleep 10`
	`done`

#	Description

	6. Perform flip operation to reverse replication direction.
	Source dataset A becomes destination and destination dataset B becomes source.
	`CLI@node-a> hpr flip <service-name>`
	7. Verify the flip.
	`CLI@node-a> hpr get source,destination service`
	`NAME PROPERTY VALUE`
	`service source https://node-b:8443/pool/dataset-b`
	`service destination pool/dataset-a`
	Source dataset becomes destination dataset and vice versa.

2	**To switch the service back to normal operation.**
	1. Recursively activate file system B. Activate feature resets mount point of file system to default value and mounts that file system. This step is required if dataset B is filesystem.
	`CLI@node-b> hpr activate -rv pool/dataset-b`
	2. 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`
	3. Enable the replication service on the source.
	`CLI@node-a> hpr enable <service-name>`

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

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

1. Forcibly disable HPR service on node B.
2. Recursively activate file system B. Activate feature resets mount point of file system to default value and mounts that file system. This step is required if dataset B is filesystem.
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 filesystem) or remove lun mappings on volume or volume group A (if dataset A is volume or volume group).
5. Perform flip direction to reverse replication direction node A/dataset A to node B/dataset B on node A.
6. Execute HPR service on node A with enabled force receive property. If the force receive property is not enabled, replication will fail with an error stating that the destination has been modified (because of possible delta on node A/dataset A).
7. Enable HPR service on node A.
8. Optional step: when ready, switch back to original state using flow “Graceful flip and sync-back”

To restore the snapshots that was taken right before the system failure to the source appliance, the destination appliance must have the latest replication of the snapshots.

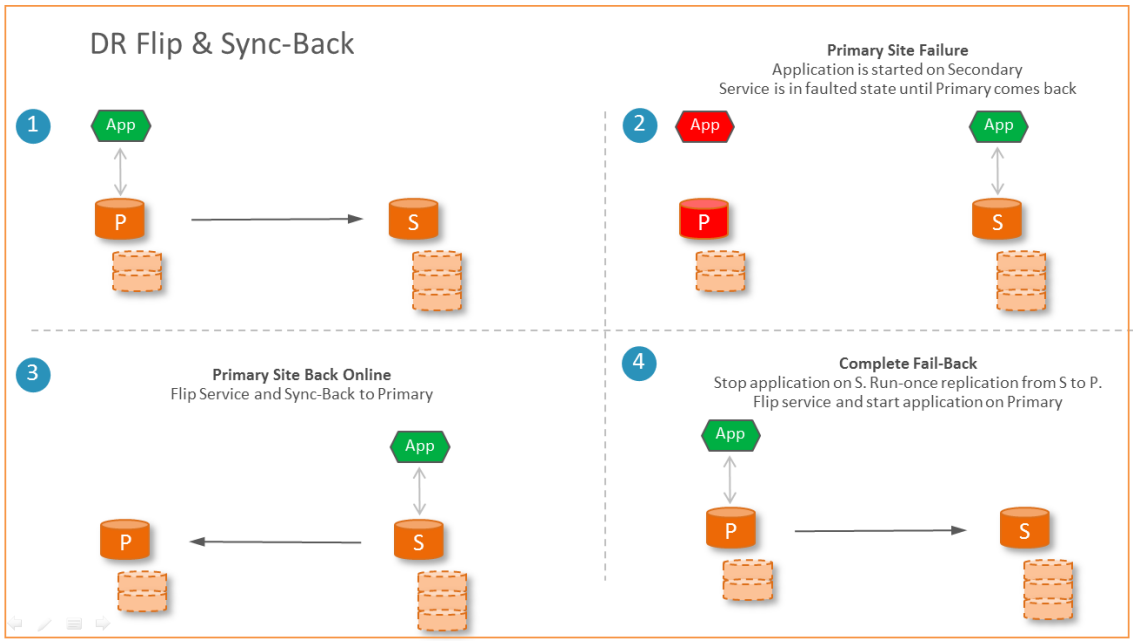


Table 4-3: Steps to Restore HPR operation using Flip-Direction as a Disaster Recovery Process

#	Description

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-name>`
	This is done to ensure that the datasets in the destination appliance do not get overwritten in case the source appliance comes back.
	2. If the destination dataset is a filesystem, you must recursively activate filesystem on the destination. Activate feature resets mount point of file system to default value and mounts that file system.
	`CLI@node-b> hpr activate -rv <dataset at destination>`
	`CLI@node-b> hpr activate -rv pool/dataset-b`
	3. Share 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`
	This allows the network clients to modify the datasets on the destination appliance.

2	**The source NexentaStor appliance (node A) is back to operation**
	After you recover the source appliance, you can switch the service roles back to normal operation 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 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

3	**On the node A, flip the direction to reverse the replication direction from node A/ dataset A to node B/dataset B.**
	1. Execute the flip service command on the primary host to reverse the replication direction.
	`CLI@node-a> hpr flip <service-name>`
	2. Execute HPR service on node A with enabled force receive property. Use the forceReceive flag to force the sync even if the destination snapshot is different from the source snapshot.
	`CLI@node-a> hpr run-once -f <service-name>`
	3. Enable the replication service on the source.
	`CLI@node-a> hpr enable <service-name>`
	Data starts replicating from the destination appliance to the source appliance. Wait until data is replicated.
	4. Let the network clients access the data from the source appliance.

Node Maintenance

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

- From the manager node, disable all the replication services on the node under maintenance. The manager node can be either the destination node or remote source node.
  - Take a note of the services that will be disabled to simplify re-enabling.
  - Alternatively the services can be disabled on the agent with force flag.
  - 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 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 such 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 re-instate C as the primary storage and as source for replication.

Action Plan

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

1. Destroy HPR service (retain all service snapshots and destination dataset) on node B.
2. If dataset B is a filesystem, activate filesystem B.

   Activate feature resets mount point of file system to default value and mounts that file system
3. Share filesystem B or create lun mappings.
4. Replace the broken node A or pool with a new node C.
5. Create new HPR service on node B to replicate existing node B/dataset B => new node C/dataset C.
6. In case of scheduled service, identify the list of existing snapshots of dataset B.
7. In case of scheduled service, create new schedule using values from hpr snaplist-find commands output.
8. In case of scheduled service, claim list of existing snapshots of dataset B. All existing snapshots of dataset B will be owned by new HPR schedule.
9. Optionally start HPR service to start immediate replication.
10. Enable new HPR service.

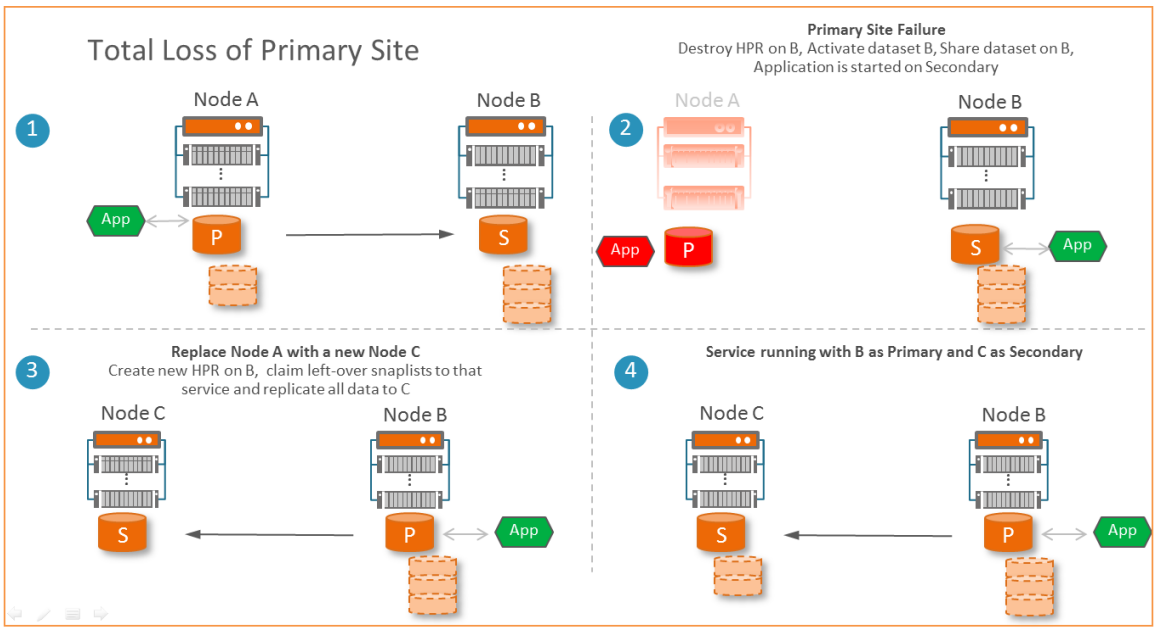


Table 4-4: Action Plan for Total Loss of Primary Site

#	Description

1	**The primary/source node A is down:**
	Destroy HPR service (retain all service snapshots and destination dataset) on node B.
	`CLI@node-b> hpr destroy -f <service-name>`
	`CLI@node-b> snapshot list`

2	Recursively activate file system B. Activate feature resets mount point of file system to default value and mounts that file system. This step is required if the dataset B is a filesystem. `CLI@node-b> hpr activate -rv pool/dataset-b`

3	Share filesystem B or create lun mappings for volume or volume group B.
	`CLI@node-b> nfs share pool/dataset-b`
	`CLI@node-b> smb share pool/dataset-b`
	For volumes:
	Create lun mappings for cloned volume (if destination dataset is volume or volume group):
	`CLI@node-b> lunmapping create pool/dataset-b/volume target-group host-group`

#	Description
4	Create new HPR service on node B to replicate existing host B/dataset B => new node C/ dataset C. For scheduled service: `CLI@node-b> hpr create -r S pool/dataset-b https://node-c/pool/dataset-c <service-name>`

5	Find the list of existing snapshots of dataset B (this step is required in case of scheduled service).
	`CLI@node-b> hpr snaplist-find <service-name>`
	`SNAPLISTID SERVICE SCHEDULE CRON SOURCESNAPSHOTS DESTINATIONSNAPSHOTS`
	`39996710-a127-11e6-be66-3f0c3222b645 - - 14 * * * * 24 0`
	`4c61e7a0-a127-11e6-be66-3f0c3222b645 - - 17 18 * * * 31 0`

6	Create new schedule using values from hpr snaplist-find commands output (this step is required in case of scheduled service).
	`CLI@node-b> hpr schedule-add <service-name> "14 * * * *" 24 24 schedule1`
	`CLI@node-b> hpr schedule-add <service-name> "17 18 * * *" 31 31schedule2`

7	Claim list of existing snapshots of dataset B. All existing snapshots of dataset B will be owned by new HPR schedule (this step is required in case of scheduled service).
	`CLI@node-b> hpr snaplist-claim -v <service-name> schedule1 39996710-a127-11e6-be66-3f0c3222b645`
	`CLI@node-b> hpr snaplist-claim -v <service-name> schedule2 4c61e7a0-a127-11e6-be66-3f0c3222b645`

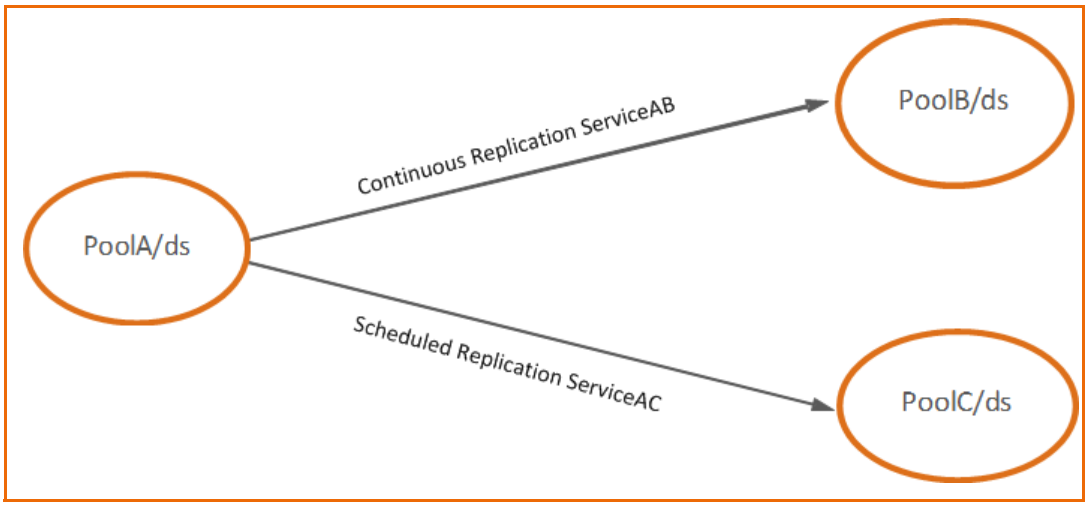
8	Optionally, start HPR service to start immediate replication. `CLI@node-b> hpr run-once <service-name>`
9	Enable new HPR service. `CLI@node-b> hpr enable <service-name>`

Disaster Recovery using Multi-Destination Services

A multiple HPR operation using multi-destination service 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 actual for DR procedure.

Figure 4-1: Example of Multiple HPR services to Multi-Destinations



User Scenario

Source node A goes down. All data is lost.

Action Plan

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

1. Destroy HPR services service-ab on node B and service-ac on node C (all HPR services with failed with node A/dataset A).
2. Select latest destination dataset for recovery source dataset (usually dataset with CR - node B/ dataset B).
3. If dataset B is filesystem, activate file system B. Activate feature resets mount point of file system to default value and mount that file system.
4. Share filesystem B or create lun mappings for volume B. Node B/dataset B will be used as primary for user applications until the primary node A is not available.

5. Node A comes up (without user data) and user want to restore original replication flows.
6. Create new CR service-ab to replicate data from node B/dataset B to node A/dataset A on node A.
7. Start and wait for on-demand initial replication to complete successfully on node A (downtime is not required and system can be used for normal productive operations). This step may consume lot of time.
8. Plan the downtime required for scheduled maintenance and data migration during which system cannot be used for normal productive operations.
9. Unshare filesystem B (if dataset B is filesystem) or remove lun mappings on volume or volume group B (if dataset B is volume or volume group).
10. Run replication one more time on node A for consistency, to replicate delta to dataset A and validate that the replication has completed, HPR service should not be running.
11. Perform flip direction for service-ab to reverse replication direction on node A. This creates a new replication direction from node B/dataset B to node A/dataset A.
12. Share filesystem A or create lun mappings for volume A on node A.
13. Destroy existing dataset C on node C.
14. Create new SR service (and schedules) to replicate data from node A/dataset A to node C/ dataset C on node A. This step requires full initial resynchronization, because common snapshots for datasets A and C does not exist.
15. Enable SR service.

Table 4-5: Steps for Creating Multiple Replication Service to Multi-Destinations

#	Description

1	Destroy HPR services, service AB on node B and service AC on node C (all HPR services with failed node A/dataset A).
	`CLI@node-b> hpr destroy -f <service name AB>`
	`CLI@node-c> hpr destroy -f <service name AC>`

2	Select latest destination dataset for recovery source dataset (usually dataset with CR - node B/dataset B).
3	Activate file system B. Activate feature resets mount point of file system to default value and mount that file system. This step is required if dataset B is filesystem: `CLI@node-b> hpr activate -rv pool/dataset-b`

#	Description

4	Share filesystem B or create lun mappings for volumes or volume group B. Node B/dataset B will be used as 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.	
5	Create and enable new CR service-ab to replicate data from node B/dataset B to node A/dataset A on node A. This step requires full initial resynchronization, because node A was replaced after node/pool failure: `CLI@host C> hpr create -r C https://node-b/pool/dataset-b pool/dataset-a service-ab`

6	Start and wait for on-demand initial replication to complete on node A (downtime is not required and system can be used for normal productive operations). This step may consume 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`

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

7	If dataset B is a filesystem, unshare the filesystem B or remove lun mappings on volume or volume group B (if dataset B is volume or volume group)
	`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`
	If dataset B is volume or volume group, remove lun mappings on volume or volume group B :
	`CLI@node-b> for i in $(volume list -O basic -o path pool/dataset-b); do lunmapping destroy -a $i; done`

8	Run replication one more time on node A for consistency and to replicate delta to dataset A and validate that the replication has completed. Note: HPR service should not be running when executing the following steps.
	`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: last time replication done."`
	`break`
	`fi`
	`echo "service-ab: waiting for last time replication..."`
	`sleep 10`
	`done`
	`done`

9	Perform flip direction for service-ab to reverse replication direction on node A. This creates a new replication direction from node B/dataset B to node A/dataset A `CLI@node-a> hpr flip service-ab`

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

11	Destroy existing dataset C on node C

#	Description

12	Create new SR service (and schedules) to replicate data from node A/dataset A to node C/dataset C on node A. This step requires full initial resynchronization, because common snapshots for datasets A and C do not exist:
	`CLI@node-a> hpr create -r S pool/dataset-a https://node-c/pool/dataset-c service-ac`
	`CLI@node-a> hpr schedule-add service-ac daily 1 30 schedule-ac`

13	Enable new SR service-ac on node A `CLI@node-a> hpr enable service-ac`

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 actual for 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 and becomes temporary unavailable, you can use node B/dataset B as primary for network clients. When node A comes back, you can restore original replication flows.

Action Plan

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

1. Select latest destination dataset for recovery source dataset (usually dataset with CR - node B/dataset B).
2. Forcibly disable HPR service which should be synced back (let's consider it's service-ab) on node B.
3. Activate file system B. Activate feature resets mount point of file system to default value and mount that file system. This step is required if dataset B is filesystem.
4. Share filesystem B or create lun mappings for volume B. Node B/dataset B will be used as primary for user applications until the primary node A is not available.
5. Node A comes up and you can restore original replication flows.
6. Unshare filesystem A (if dataset A is filesystem) or remove lun mappings on volume or volume group A If dataset A is volume or volume group.
7. **Forcibly destroy HPR service-ac on node A.**

   When data is replicated back to the primary site for a multi-destination configuration, all the other HPR services must be destroyed. This is required because a dataset cannot concurrently be both source and destination of different HPR services.

8. Perform flip direction for service-ab to reverse replication direction. This creates a new replication direction node B/dataset B to node A/dataset A.
9. If dataset B is filesystem, unshare filesystem B or remove lun mappings on volume or volume group B if dataset B is volume or volume group.
10. Run HPR service once on node A with --force option. Otherwise replication may fail with error if destination has been modified (since we might have had delta on node A/dataset A).
11. Perform flip direction for service-ab to reverse replication direction. This restores original replication direction from node A/dataset A to node B/dataset B.
12. Share filesystem A or create lun mappings for volumes or volume group A.
13. Enable HPR service-ab on node A:
14. Create new SR service (and schedules) to replicate data from node B/dataset B to node C/ dataset C on node A.
15. Claim snapshots which belonged to service-ac
16. Enable HPR service-ac on node A

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.

Table 4-6: Steps to Sync-back in Multi-Destination Services

#	Description
1	Select latest destination dataset for recovery source dataset (usually dataset with CR - node B/ dataset B).
2	Forcibly disable HPR service-ab on node B: `CLI@node-b> hpr disable -f service-ab`
3	Activate file system B. Activate feature resets mount point of file system to default value and mounts that file system. This step is required if dataset B is filesystem: `CLI@node-b> hpr activate -rv pool/dataset-b`

4	Share filesystem B or create lun mappings for volumes or volume group B. Node B/dataset B will be used as primary for user applications until 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`

#	Description

5	Node A comes up and user want to restore original replication flows
	Unshare filesystem A (if dataset A is filesystem):
	`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`
	or remove lun mappings on volume or volume group A (if dataset A is volume or volume group):
	`CLI@node-a> for i in $(volume list -O basic -o path pool/dataset-a); do lunmapping destroy -a $i; done`

6	Forcibly destroy HPR service-ac on node A: `CLI@node-a> hpr destroy --force service-ac`
7	Perform flip direction for service-ab to reverse replication direction. This creates a new replication direction from node B/dataset B to node A/dataset A: `CLI@node-a> hpr flip service-ab`

8	Unshare filesystem B (if dataset B is filesystem):
	`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 volume or volume group):
	`CLI@node-b> for i in $(volume list -O basic -o path pool/dataset-b); do lunmapping destroy -a $i; done`

9	Run service on node A once with --force flag, --force is required to overwrite possible delta on dataset A. Otherwise replication will fail with error if there is a change in the dataset A: `CLI@honode-ast-a> hpr run-once -f service-ab`
10	Perform flip direction for service-ab to reverse replication direction. This restores original replication direction from node A/dataset A to node B/dataset B: `CLI@node-a> hpr flip service-ab`

11	Share filesystem 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`

12	Enable HPR service-ab on node A: `CLI@node-a> hpr enable serice-ab`

#	Description

13	Create new HPR service on node A to replicate host A/dataset A => new node C/dataset C.
	For scheduled service:
	`CLI@node-a> hpr create -r S pool/dataset-a https://node-c/pool/dataset-c service-ac`
	In case of scheduled service, find the list of existing snapshots for service-ac on node A.
	`CLI@node-a> hpr snaplist-find service-ac`
	`SNAPLISTID SERVICE SCHEDULE SOURCESNAPSHOTS DESTINATIONSNAPSHOTS`
	`39996710-a127-11e6-be66-3f0c3222b645 - 0 0 * * * 10 16`
	`4c61e7a0-a127-11e6-be66-3f0c3222b645 - 0 0 0 * * 5 10`
	In case of scheduled service, create new schedule using values from `hpr snaplist-find` commands output.
	`CLI@node-a> hpr schedule-add service-ac "0 0 * * *" 10 16 schedule1`
	`CLI@node-a> hpr schedule-add service-ac "0 0 0 * *" 5 10 schedule2`

14	In case of scheduled service, claim list of existing snapshots of dataset B. All existing snapshots of dataset B will be owned by new HPR schedule.
	`CLI@node-a> hpr snaplist-claim -v service-ac schedule1 39996710-a127-11e6-be66-3f0c3222b645`
	`CLI@node-a> hpr snaplist-claim -v service-ac schedule2 4c61e7a0-a127-11e6-be66-3f0c3222b645`

15	Enable HPR service-ac on node A. `CLI@node-a> hpr enable service-ac`

Recovering Broken Replication Service

Modifying, deleting or creating snapshots of destination dataset may cause 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 incremental source”
- “Failed to run replication session: Common snapshot does not exist”

The first thing to try and recover the replication service is hpr run-once --force. This overwrites all destination changes and destroy any snapshots created after the common snapshot.

If the hpr run-once command is not able to fix the replication service, you should run the hpr recover command:

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

This command will create a new on-demand snapshot, try to find common snapshot for each replicated dataset separately and replicate all snapshots created after the common snapshot to destination. Any changes or snapshots created after the common snapshot will be destroyed. In case common snapshot does not exist, dataset will be replicated from the beginning.

5

Advanced Configuration

This section includes the following topics:

Replication Service Options

List all HPR Service Properties

```
CLI@node-a> hpr get all <service-name>
NAME     PROPERTY           VALUE
service  commonSnapshot     hpr-ondemand-2016-12-13-22-51-26-105
service  destination        https://node-b:8443/pool2/dst
service  heartbeat          no
service  ignoreProperties   -
service  isManager          yes
service  running            no
service  isSyncing          no
service  maxBufferSize      100
service  name               service
service  recursive          yes
service  replaceProperties
service  source             pool1/src
service  state              enabled
service  type               scheduled
```

Modify HPR Service Properties

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

#	Description
description	service description Example to set the service description: `CLI@node-a> hpr set description="Daily replication" <service-name>`
throttle	throttle connection speed (bytes per second) Example to set service throttle connection speed: `CLI@node-a> hpr set throttle=2000000 <service-name>`
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 <service-name>`
replaceProperties	set the specified properties during receive
maxBufferSize	maximum replication buffer size (Mbytes)

List of Dataset Properties that can be Ignored or Replaced

#	Description
aclInherit	Controls how ACL entries are inherited when files and directories are created
aclMode	Controls ACL behavior when a file is initially created or whenever a file or directory's mode is modified by the chmod command
allowExtendedAttributes	Indicates whether extended attributes are enabled or disabled for this file system
checksumMode	Controls the checksum used to verify data integrity
compressionMode	Enables or disables compression for a dataset
dataCopies	Sets the number of copies of user data per file system
dedupMode	Controls whether duplicate data is removed from the file system
logBiasMode	Use this property to provide a hint to ZFS about handling synchronous requests for a specific dataset
nonBlockingMandatoryMode	Controls if non-blocking mandatory locks are enabled or disabled. Filesystem needs to be remounted after changing this property in order to have any effect
primaryCacheMode	Controls what is cached in the primary cache (ARC)
quotaSize	Limitstheamountofdiskspaceadatasetanditsdescendentscanconsume. Value zero means no quota.
readOnly	Controls whether a dataset can be modified
recordSize	Specifies a suggested block size for files in a file system
redundantMetadata	Controls what types of metadata are stored redundantly
reservationSize	Sets the minimum amount of disk space guaranteed to a dataset and its descendents. Value zero means no quota.
secondaryCache	Controls what is cached in the secondary cache (L2ARC)
smartCompression	Smart compression dynamically tracks per-file compression ratios to determine if a file is compressible or not. When the compression ratio being achieved is too low, smart compression progressively backs off attempting to compress the file.
snapshotDirectory	Controls whether the zfs directory is hidden or visible in the root of the file system
syncMode	Control synchronous behavior
updateAccessTime	Controls whether the access time for files is updated when they are read
vscan	Enables virus scanning on a file system

Note:	Nexenta does not recommend to change the following two properties unless purely needed. If you need to modify these properties, do it ideally with an approval from Nexenta support team. • `recursive - replicate dataset recursively` • `remoteNode - remote node URL`

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 force 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.

Task	Command
Disable schedule	`CLI@node> hpr schedule-disable <service-name> <schedule-name>` `CLI@node> hpr schedule-disable --help`
Remove existing schedule	`CLI@node> hpr schedule-remove <service-name> <schedule-name>` `CLI@node> hpr schedule-remove --help`
Rename existing schedule	`CLI@node> hpr schedule-rename <service-name> <schedule-name> <new-schedule-name>` `CLI@node> hpr schedule-rename --help`
Change schedule properties	`CLI@node> hpr schedule-set <property=value> <service-name> <schedule-name>` `CLI@node> hpr schedule-set --help`
List service snapshots list	`CLI@node> hpr snapshots <service-name>` `CLI@node> hpr snapshots --help`
Claim snapshots list	`CLI@node> hpr snaplist-claim <service-name> <schedule-name> <snaplist-id>` `CLI@node> hpr snaplist-claim --help`
Delete snapshots list	`CLI@node> hpr snaplist-delete <service-name> <snaplist-id>` `CLI@node> hpr snaplist-delete --help`
Find snapshots list	`CLI@node> hpr snaplist-find <service-name>` `CLI@node> hpr snaplist-find --help`

Global Headquarters
451 El Camino Real, Suite 201
Santa Clara, CA 95050
USA

3000-HPR-5.0-000058-A