Multi-Tenant File Services NexentaStor 5.2
CLI User Guide

Date: April, 2019
Part Number: ns-5.2-mtfsuserguide-RevB
Copyright © 2019 Nexenta Systems™, ALL RIGHTS RESERVED

Notice: No part of this publication may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, or stored in a database or retrieval system for any purpose, without the express written permission of Nexenta Systems (hereinafter referred to as “Nexenta”).

Nexenta reserves the right to make changes to this document at any time without notice and assumes no responsibility for its use. Nexenta products and services only can be ordered under the terms and conditions of Nexenta Systems’ applicable agreements. All of the features described in this document may not be available currently. Refer to the latest product announcement or contact your local Nexenta Systems sales office for information on feature and product availability. This document includes the latest information available at the time of publication.

Nexenta, NexentaStor, NexentaEdge, NexentaFusion, and NexentaCloud are registered trademarks of Nexenta Systems in the United States and other countries. All other trademarks, service marks, and company names in this document are properties of their respective owners.
## Contents

**Preface** .............................................................................................................. vi

**1 Intro to Multi-Tenants** ................................................................. 1

- About this Document ................................................................. 1
  - About Host and Tenant Admin ...................................................... 1
  - About Host Management Interface and Tenant Specific Management Interface .................................................. 2

- Multi-Tenant File Services Overview ................................................ 2
  - Introduction ................................................................................. 2
  - Benefits and Target Use Cases .................................................... 2
  - Feature Design Overview ......................................................... 3

- Data Protection Services in Multi-Tenant Configurations ....................... 6
- Management of Software Versions in MTFS Configurations ..................... 7

- Feature Limitations in MTFS Configurations ......................................... 7

**2 Creating and Managing Tenants** ............................................................... 9

- Prerequisites ................................................................. 9
  - Verify Multi-Tenant Feature ...................................................... 9
  - Verify Existing HA Services ...................................................... 10

- System Sizing Recommendations ................................................... 10

- Networking Guidelines .......................................................... 11

- Initialize the Tenant Subsystem .................................................... 11

- Creating Tenants ................................................................. 15
  - Create Tenant ........................................................................ 15
  - Verify Tenant Creation .......................................................... 17
  - Start Tenant ........................................................................... 18

- Verify Boot Environment of the Tenant .............................................. 19

- Managing Tenant Resources from Host ................................................. 19
  - Manage VNICS ...................................................................... 20

- Manage Storage ...................................................................... 22

- Monitor CPU and Memory Utilization ............................................. 24

- Managing Tenants ..................................................................... 24
Start a Tenant ..................................................25
View Tenant Status ...........................................25
Stop a Tenant ...................................................26
Reboot a Tenant ................................................26
Destroy a Tenant ..............................................26
Managing Multi-Tenant Configurations via REST API .............................................27

3 Managing In-Tenant Services ..............................................29
Logging in to a Tenant .........................................29
Commands Available from a Tenant .................................29
Post Tenant-Creation Checks ......................................30
Basic In-Tenant Configurations ....................................31
  Time Date Configuration ...................................31
  Enable Swagger API Documentation .........................31
  Set Management Address ..................................32
  Change Admin User Password .................................32
  Create a New User/New User Group .........................32
  Configure SMTP Email Service ...............................33
  Upload the SSL Certificate ..................................33
Managing Network .................................................34
  Verify the Available Network Interface .......................34
  Assign or Update the IP Address to Network Interface ....34
  Create a Network Route ......................................34
Managing Datasets ..................................................34
  Enable System Service ......................................34
  Configure Filesystems ......................................35
  Configure NFS & SMB Services ..............................36
  Configure AD/LDAP ...........................................37
  Configure ACLs ...............................................37
  Configure Manual Snapshots ..................................37
Managing Boot Environment Checkpoints in a Tenant .......................37
  Activate Boot Environment for a Tenant ......................37
# Multi-Tenant File Services User Guide

## NexentaStor Upgrades with Multi-Tenant Configuration

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Upgrade Tenants in a Clustered Environment</td>
<td>39</td>
</tr>
<tr>
<td>Supported Upgrade Configuration</td>
<td>39</td>
</tr>
<tr>
<td>Upgrade Steps</td>
<td>39</td>
</tr>
<tr>
<td>Rollback Clustered Nodes with Multi-Tenants</td>
<td>45</td>
</tr>
<tr>
<td>Rollback Steps with Examples</td>
<td>45</td>
</tr>
</tbody>
</table>

## Backup and Restore of Boot Environment

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Backup and Restore of NexentaStor Boot Environment in a Multi-Tenant Clustered Environment</td>
<td>48</td>
</tr>
<tr>
<td>Backup and Restore Steps</td>
<td>48</td>
</tr>
</tbody>
</table>

## Troubleshooting

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Verifying Tenant Init Completed Successfully</td>
<td>54</td>
</tr>
<tr>
<td>Troubleshooting Tenant Initialization</td>
<td>54</td>
</tr>
<tr>
<td>Handling Tenant Creation Failure</td>
<td>56</td>
</tr>
<tr>
<td>Reinitialize Master Tenant</td>
<td>56</td>
</tr>
<tr>
<td>Verify Master Tenant is Reinitialized Using the API</td>
<td>56</td>
</tr>
<tr>
<td>Retrieving Information about a Tenant(s)</td>
<td>57</td>
</tr>
<tr>
<td>Handling Tenant Login Failure</td>
<td>57</td>
</tr>
<tr>
<td>Verifying if a Dataset is Delegated to a Tenant</td>
<td>58</td>
</tr>
<tr>
<td>Monitoring Storage Performance with FSSTAT</td>
<td>58</td>
</tr>
<tr>
<td>Troubleshooting Upgrade</td>
<td>59</td>
</tr>
<tr>
<td>Verify the Appliance Version in the Local Tenant Repository</td>
<td>59</td>
</tr>
<tr>
<td>Clean the Local Repo</td>
<td>60</td>
</tr>
<tr>
<td>Bundle Services for Tenants</td>
<td>60</td>
</tr>
<tr>
<td>Create a Support Bundle</td>
<td>60</td>
</tr>
<tr>
<td>Identify the Core Files</td>
<td>61</td>
</tr>
<tr>
<td>Include Core Files to SB</td>
<td>61</td>
</tr>
<tr>
<td>Verify the Bundle</td>
<td>61</td>
</tr>
<tr>
<td>Download the Tenant Bundle Locally</td>
<td>61</td>
</tr>
<tr>
<td>Uploading Tenant Bundle</td>
<td>63</td>
</tr>
<tr>
<td>Default Support Bundle Protocol and Settings</td>
<td>65</td>
</tr>
<tr>
<td>Other Bundle Management Commands</td>
<td>65</td>
</tr>
</tbody>
</table>
Preface

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

Intended Audience

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

Documentation History

The following table lists the released revisions of this documentation.

<table>
<thead>
<tr>
<th>Product Versions Applicable to this Documentation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Revision</td>
</tr>
<tr>
<td>-----------------------------------</td>
</tr>
<tr>
<td>ns-5.2-mtfsuserguide-RevB</td>
</tr>
</tbody>
</table>

Contacting Support

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

Comments

Your comments and suggestions to improve this documentation are greatly appreciated. Send any feedback to doc.comments@nexenta.com and include the documentation title, number, and revision. Refer to specific pages, sections, and paragraphs whenever possible.
This page intentionally left blank
Intro to Multi-Tenants

This chapter covers the following topics:

- About this Document
- Multi-Tenant File Services Overview
- Feature Limitations in MTFS Configurations
- Benefits and Target Use Cases
- Feature Design Overview
- Data Protection Services in Multi-Tenant Configurations
- Management of Software Versions in MTFS Configurations

About this Document

This document is intended for host and tenant administrators. This guide demonstrates the basic steps and includes tenant specific commands to create a tenant and manage its resources from a host storage system as a host admin. Tenant specific commands are supported only on the host system. To view all the tenant specific CLI commands available from the host, type:

```
CLI@host> tenant --help
```

Once a tenant is set up, you can perform tenant level functions by logging into the tenant as tenant admin through the host. These management functions within the tenants can be done using standard NexentaStor CLI commands. To learn about the commands associated with tenant-specific management, we recommend that you use this document in conjunction with the following guides:

- NexentaStor 5.2 CLI Reference Guide
- NexentaStor 5.2 CLI Configuration Guide
- NexentaStor 5.2 HA CLI Configuration Guide
- NexentaStor 5.2 HPR User Guide

About Host and Tenant Admin

Managing tenant services can be done either by logging into the tenant as host admin or by logging into the tenant as tenant admin through tenant specific management interfaces.

As a host admin you have the privileges to manage the host and all the tenants hosted on the host storage system. As a host admin, you can log into any tenant on the host system through the host management interface. However, when you log into a tenant, you will be switched to the context of that specific tenant
and will be allowed to manage only that specific tenant. To learn about the management functions available within a tenant, run `CLI@tenant> help`. This will list all the commands that can be executed on the tenants.

About Host Management Interface and Tenant Specific Management Interface

Managing tenant services can be done either by logging into the tenant as host admin through the main NexentaStor management interface or by logging into the tenant as tenant admin through tenant specific management interfaces. When you login to the tenant, as tenant admin, through the tenant management interface, you will be prompted for the password. This password is setup by the host admin when creating the tenants on the host storage system. This password can be changed once you login to the tenant.

See Creating Tenants for information on how the password for the tenant administrators are setup when creating the tenant.

See Change Admin User Password for information on changing the tenant admin passwords.

Multi-Tenant File Services Overview

Introduction

The advanced features, performance and scalability of NexentaStor make it a great platform for consolidating diverse storage workloads and clients on a single hardware cluster. NexentaStor administrators have historically achieved this with a combination of layer 2 vlans, layer 3 strong host model routing, careful share and ACL configurations and, in more complex cases, NexentaStor Virtual Storage Appliances on ESXi.

The NexentaStor Multi-Tenant File Services (MTFS) feature goes a step further and delivers extremely flexible, strong and secure segregation of storage and network resources between multiple tenants, departments, or applications.

A NexentaStor tenant runs as a lightweight resource container on a NexentaStor node running its own dedicated IP routing engine, filesystems, NFS and SMB servers, Active directory and Kerberos authentication services, and Nexenta management engine. Similar to a Docker container, CPU and memory resources for a given NexentaStor tenant are dynamically allocated on an as needed basis, minimizing the overall footprint of each tenant on the underlying hardware. The only resources that are explicitly delegated to a NexentaStor tenant are storage datasets and virtual network interfaces (vnic), generally based on layer 2 virtual LAN links.

NexentaStor Multi-Tenant File Services is a separately licensed optional feature of NexentaStor Enterprise Edition.

Benefits and Target Use Cases

MTFS allows a single NexentaStor hardware appliance to provide secure and segregated file services to a number of different tenants, thereby eliminating the complexity and management overhead of deploying a dedicated full appliance per tenant. Each tenant can be configured with its own IP networks, its own Active Directory and user authentication services, its own NFS and SMB server settings, totally independent of other
tenants on the same NexentaStor appliance. The solution can even handle multiple tenants with overlapping IP subnets without issues. MTFS is only supported on NexentaStor High-Availability clusters and provides HA for all tenant services. Managing tenant services can be done either through the main NexentaStor management interface or through tenant specific management interfaces.

This solution is ideally suited to hosting providers leveraging NexentaStor clusters for backend storage and providing fully customizable file services to their end-customers, including tenant managed NAS services. In Enterprise environments, the solution can also be used to provide a higher level of segregation between different departments consuming file services, or between different applications.

The architecture and design of NexentaStor MTFS aims at providing a great deal of flexibility to ease consolidation of diverse tenants on a single NexentaStor clusters, with simple management, security, isolation and ultimately mobility of tenants with all their associated data and file services.

Feature Design Overview

NexentaStor Multi-Tenant File Services (MTFS) builds on the flexible storage and management framework designs introduced with the original release of NexentaStor 5. The picture below lays out key architectural concepts, focusing on a single NexentaStor node.

Figure 1-1: Multi-Tenant Configuration, Focusing on One NexentaStor Host

In the context of MTFS, functions are either Host level functions running in the main NexentaStor operating system, or Tenant level functions running within a tenant resource container. For example, the picture above shows a single NexentaStor host with its own host level IP, SMB and AD/LDAP configuration (in orange); and two tenants, tenant_1 and tenant_2, each with its own specific set of file services, IP and AD/LDAP configuration (in blue and purple).
The host level management control point (Host_Mgmt) provides the management interface to the overall system. This management interface also provides access to each tenant management control point (Tenant_Mgmt). This allows a host administrator to manage all features at the Host level, create and destroy tenants, login to any tenant and manage all features within any tenant.

Access to a Tenant_Mgmt module can also be enabled over an IP interface on the tenant networks. This allows a tenant administrator to login to his tenant and manage all in-tenant features and services. Such a tenant administrator is by definition limited to managing his particular tenant and does not have access to the Host or any other tenant on the system.

Within a tenant, IP interfaces are supported by a set of virtual network interfaces (vnic) that are assigned to the tenant from the underlying host. Typically, a vnic will map to a layer 2 virtual lan on an underlying physical interface or aggregate. A vnic must be assigned to a tenant at the time of its creation. Once a tenant exists, it is possible to add additional vnics to it. An individual vnic cannot be shared between tenants, however multiple vnics can share the same physical link(s) or vlan.

From a storage perspective, all tenant datasets are stored in standard NexentaStor data pools, allowing tenants to simply inherit all high-availability configurations of the underlying data pool. As such, a pool failover event will automatically and seamlessly failover all the tenants it supports.

When creating a tenant, the administrator must specify a parent dataset. This implicitly specifies the data pool that will store all tenant datasets, and provides flexibility to control location of a tenant and its data within the overall pool namespace. A top level <tenant_name> dataset is then created as a child of the specified parent. <tenant_name> contains a root child dataset that will contain the tenant specific boot environment.<tenant_name> also contains at least one <storage> child dataset (named “storage” by default) that acts as the top level dataset for tenant filesystems. For example, looking at the case of tenant_1 in the figure above, its two filesystems fsa1 and fsa2 are actually nested under parent/tenant_1/storage.

This dataset layout provides a good foundation for simple configuration of tenant level attributes and data protection services. For example, the <storage> dataset is originally set with a quota that controls how much capacity is available for filesystems within the tenant. Taking snapshots of “all the filesystems” in a tenant can also be done with a single snapshot schedule service on the <storage> dataset for that tenant. Similarly, replicating all the data of a given tenant can be done with a single HPR service on <storage>. Looking ahead, the ability to move a full tenant (data and boot environment) from one NexentaStor cluster to another will be enabled in a future release with a combination of HPR on the <tenant_name> dataset and propagation of a limited set of control information at the NexentaStor Host level.

NexentaStor MTFS is only supported on NexentaStor HA clusters.

In a cluster configuration, special attention to the network interface configuration on both nodes such that the configuration of the vnic in terms of naming and vlan configuration must be identical between the nodes.
For example, for a tenant with vnic10 on vlan tag 120 to be able to properly failover in a NexentaStor HA cluster, both NexentaStor hosts must have a vnic10 (same name) configured on vlan tag 120 (same vlan).

In NexentaStor 5.2, ensuring that vnic configuration are consistent across both NexentaStor hosts of a HA cluster is the responsibility of the NexentaStor administrator.
Data Protection Services in Multi-Tenant Configurations

The only data protection mechanisms that are managed directly from within a tenant are on demand snapshots and clones. More advanced data protection services for tenant filesystems such as scheduled snapshots and replication must be managed from the Host, by the Host administrator.

As was shown in Figure 1-1, Multi-Tenant Configuration, Focusing on One NexentaStor Host, all tenant filesystems reside in a standard data pool imported on the NexentaStor Host. The tenant filesystems are visible from the Host, allowing the administrator to configure snapshot schedules, scheduled replication and continuous replication services directly on them.

Figure 1-3: Configure Snapshot and HPR Services on the Host

The hierarchical layout of storage for a particular tenant, paired with the ability to configure recursive snapshot schedules and replication services provides a lot of flexibility for a service provider to offer different storage protection SLAs to its tenants.

Data protection services are only supported on the filesystem and parent/tenant/storage top level dataset delegated to a tenant. While it may be tempting to want to configure a recursive replication service on the top level /parent/tenant dataset itself, this is not a supported configuration as of NexentaStor 5.2.

For example, a service provider could configure a tenant with 2 top level storage datasets each with a particular data protection service that would automatically apply to filesystems created by the tenant in them:

- `/parent/tenant_1/storage_hourlysnap/` could be configured with a recursive hourly snapshot service that would apply to all child filesystems the tenant would create in that folder.
- `/parent/tenant_2/storage_dailySr/` could be configured with a recursive daily scheduled replication service that would apply to all child filesystems the tenant would create in that folder.

Providing disaster recovery services for a tenant should be approached the same way disaster recovery is planned between NexentaStor clusters: it requires 2 tenants, one on primary site and one on the secondary site and replication services configured on the tenant/storage filesystems rather than on the tenant itself.
Management of Software Versions in MTFS Configurations

As was depicted in Figure 1-1, Multi-Tenant Configuration, Focusing on One NexentaStor Host, each tenant in a multi-tenant configuration has its own root dataset configured as a child of the parent/tenant dataset. This root dataset in turn contains boot environment datasets that contain the subset of system packages and configuration details required to run a given tenant’s services. While NexentaStor automates management of tenant software versions, this section provides background information on two concepts critical to tenant software management: the tenant software repository provisioned on the Host rpool, and the master tenant instance.

Before an administrator can create tenants on a NexentaStor cluster, the system must be initialized for multi-tenant services. This step creates a local software repository on the Host rpool that contains the software packages that will be used to provision tenant boot environments. It then uses the software packages in this local repository to create a master tenant instance that will serve as the gold image for all new tenants provisioned on the cluster.

The purpose of the tenant initialization step is twofold:

1. It minimizes the amount of data transferred over the network by creating a local repository that can be used by all tenants on the NexentaStor cluster. During a software upgrade, the packages in the local repository get updated first, generally from public repositories over the network. Each tenant then gets to access the local repository to perform its software upgrade instead of going over the internet to access public repositories.

2. It minimizes the time required to create new tenants. Instead of processing a new tenant creation as a fresh software install in a new resource container, new tenants get created as full clone images of the master tenant.

Tenant initialization effectively takes some effort (and time) upfront to streamline future tenant creation and tenant software upgrade operations.

Once a multi-tenant system has been initialized, tenants can very quickly be created on it. Each tenant runs as a resource container on the NexentaStor host OS image, creating a dependency between the main NexentaStor software running on the host and the limited software that supports a particular tenant. The general rule of thumb that is implemented by the upgrade procedure described later in this document is to ensure that Host and Tenants are kept running the same version of NexentaStor software at all times. This is accomplished by the NexentaStor software carefully orchestrating update of the Host software, update of the software in the rpool local repository and update of each tenant’s software based on the local repository.

Feature Limitations in MTFS Configurations

As the name implies, MTFS is limited to SMB and NFS file services. Block services are not available inside a tenant.

Production deployments of MTFS are only supported in High-Availability cluster configurations.

- As of NexentaStor release 5.2, tenants and in-tenant services can only be managed using API and CLI. NexentaFusion specifically does not provide management of tenant services.
- A maximum of 32 tenants can be run on a NexentaStor HA cluster with 256GB of DRAM per host. A maximum of 16 tenants should be configured in a given pool / HA service.
To support tenant specific IP network configurations, virtual network interfaces (vnic) delegated to tenants should be created on layer 2 VLANs.

- While scheduled data protection services (snapshot schedules, scheduled replication and continuous replication) can be configured on tenant filesystems, these services must be managed from the host. The only data protection services available from within a tenant are on demand snapshots and clones.
- vsan and ICAP based virus scanning are not supported on NexentaStor hosts with tenant services.
- NexentaStor Boot Environment backup and restore functionality require special handling in MTFS configurations. Please see Chapter 5, Backup and Restore of Boot Environment for more details.
Creating and Managing Tenants

This chapter covers the following topics:

- Prerequisites
- Creating Tenants
- Managing Tenant Resources from Host
- Managing Tenants
- Managing Boot Environment Checkpoints in a Tenant
- Managing Tenants

Prerequisites

Verify Multi-Tenant Feature

The NexentaStor High Availability (HA) cluster is an enterprise-proven add-on feature to NexentaStor. Before you proceed with configuring tenants review your baseline settings, after installation, from the CLI.

To use the multi-tenant feature, license tokens for this must be activated and listed in the `features` row as shown in the following sample output. If this feature is not enabled, contact sales@nexenta.com to obtain the feature license.

```
CLI@host> license show
PROPERTY     VALUE
guid         534d4349-0002-6190-2500-6190250015b0
valid        yes
status       ok
type         ENTERPRISE-TRIAL(Nexenta Internal)
product      NexentaStor
version      5.x
licensee     Nexenta-abcd@nexenta.com
serial       SR-DEV-NS-201616578
features     allFlash, continuousReplication, highAvailability, scheduledReplication, fibrechannel, multiTenancy
issued       Sun May 21 17:00:00 2017
expires      Tue August 29 16:00:00 2018
capacity     no limit
```
If the multi-tenant feature is lost as part of a license update, any existing tenant will continue to run. But the ability to login to tenants, create and manage new tenants will be lost. In other words, a system without the multiTenant feature will not be able to start or restart tenants as part of a pool failover.

**Verify Existing HA Services**

For tenants to be under HA control, they must be created on a pool that is already part of an HA service. If an HA service is configured after tenants have already been provisioned, the tenants do not inherit the new HA protection: they would have to be entirely re-provisioned in order to be protected by the HA service. See NexentaStor 5.x HA CLI Configuration Guide for information related to HA services.

- **To verify existing HA services:**
  
  CLI@host> haservice list

**System Sizing Recommendations**

The following values are applicable for any Host supporting multi-tenant configurations.

MTFS is supported in NexentaStor HA cluster configurations where each NexentaStor node should have a minimum of:

- Minimum DRAM: 256GB
- Minimum rpool size: 256 GB, preferably 1TB

Such a NexentaStor HA cluster can support up to 32 tenants, with a maximum of 16 tenants per HA service / pool.

For each tenant, you should plan for:

**Storage quota for a tenant**

- About 4GB of DRAM
- About 1GB of pool space used for the tenant boot environments.

  See [Manage Storage](#) for more information on managing storage capacity for tenants.
Networking Guidelines

Tenant networking is configured on top of Virtual Network Interface Interfaces (VNIC). To simplify network management in multi-tenant environments, it is recommended to configure VNICs on Layer 2 Virtual LANs, associating a VNIC to a specific VLAN tag, also called VLAN identifier (VID) in the NexentaStor CLI. A VNIC must be assigned to a tenant at the time of its creation. Since VNICs are not automatically coordinated between nodes in a NexentaStor HA cluster, it is up to the administrator to manually create VNICs on both nodes with identical names in a clustered environment. This is required for tenant failover to function accurately in the event of a node failure.

For example, for a tenant with vnic20 on vlan tag 120 to be able to properly failover in a NexentaStor HA cluster, both NexentaStor hosts must have a vnic20 configured on vlan tag 120.

Provision VNIC

1. Create the VNIC on the appropriate physical link with the appropriate VLAN identifier (vid).
   
   CLI@host> link create vnic <name> <link> <vid>
   
   Example:
   
   CLI@host> link create vnic vnic20 e1000g1 120

2. Verify the VNIC you created.
   
   CLI@host> link show-vnic vnic20
   
   NAME   OVER     SPEED  MAC             MACTYPE  VID TENANT
   vnic20 e1000g1  1000   2:8:20:5a:96:5  random   120 -

Initialize the Tenant Subsystem

When you are creating a tenant for the first time, you must first initialize the tenant subsystem using the tenant init command. To initialize the tenant subsystem, valid license is required and you must have activated the license.

   CLI@host> tenant init
This init command creates a local package repository on the rpool. This repository is then used to create a master tenant that acts as a gold image for new tenant creation (see Management of Software Versions in MTFS Configurations for more information). Creation and population of the local repository, as well as creation of the master tenant image are done as part of the tenant init command. The local repository acts as a proxy for all tenant software related operations, instead of having each tenant go individually download packages from remote online repositories, the local repository is updated once over the wide area network and then used locally by all tenants. Since the initialization procedure pulls all packages from Nexenta repositories, the host needs access to Nexenta repositories. This initialization can be accomplished With or Without internet connection (as shown in the following subsections With Internet Connection and Without Internet Connection).

**With Internet Connection**

The initialization process grabs the images from the following locations, bypassing the need to download images.

1. Verify the publisher list. Ensure that you can reach the location specified here.

   CLI\@host> publisher list

<p>|</p>
<table>
<thead>
<tr>
<th>PUBLISHER</th>
<th>STATUS</th>
<th>LOCATION</th>
</tr>
</thead>
<tbody>
<tr>
<td>nexenta</td>
<td>online</td>
<td><a href="https://prodpkg.nexenta.com/nstor/pkg5/">https://prodpkg.nexenta.com/nstor/pkg5/</a></td>
</tr>
</tbody>
</table>

   **Note:**

   When creating tenant on a clustered node, you must run the `tenant init` command on both nodes in the HA cluster before any tenant can be created.

   `tenant init` command has the following options that can be used for various troubleshooting scenarios.

   `-- master`

   `-- repo`

   `-- all`

   For more details on the use cases of these options refer to the Troubleshooting section.

   - `tenant init -- master` is used in the following cases:
     - if tenant create fails due to some inconsistency between the Master tenant and the host.
     - used to clean up any cruft from the master tenant in case of a failed upgrade attempt.

     This re initializes master tenant image. Once this command is executed, the Master tenant will reset to the software version currently running on the host node.

   - `tenant init -- repo` is used in the following cases:
     - to update the local package repository on the rpool. Use this if you made any changes to the host configuration at any point.
     - used when the Software Upgrade fails.

     This re initializes local tenant repository to the software version currently running on the host node.

   **With Internet Connection**

The initialization process grabs the images from the following locations, bypassing the need to download images.

1. Verify the publisher list. Ensure that you can reach the location specified here.
2. Initialize the tenant.

   CLI@host> tenant init

   When prompted, type “y” to initialize the tenant subsystem.

   Initialize tenant subsystem? [y/N] y

Without Internet Connection

Initializing a tenant without an internet connection requires the use of a special dark site upgrade ISO, available from support@nexenta.com.

Once you have the dark site upgrade ISO, you can either:

- mount the ISO through IPMI management console, or
- create a loadable DVD to be used for the upgrade.

Using Loadable Image into the DVD Slot

1. Log in to the NexentaStor node that does not have internet connection.
2. Load the DVD you created from the loadable image into the DVD Slot.
   The ISO gets mounted automatically.

Using IPMI Management Console

1. Log in to the NexentaStor node that does not have internet connection.
2. Log in to the IPMI management console and mount the ISO file through virtual media.

Verify Availability of Media

   CLI@host> publisher discover

   MEDIA          PUBLISHER         LOCATION
   NS_UpgradeCD   nexenta           /media/NS_UpgradeCD/nexenta
   NS_UpgradeCD   HighAvailability  /media/NS_UpgradeCD/rsf

Initialize the Tenant

1. From the CLI, list the current publisher.

   CLI@host> publisher list

   PUBLISHER          STATUS  LOCATION
   nexenta           online  http://nefpkg.nexenta.com/ns520/
   HighAvailability  online  http://nefpkg.nexenta.com/rsf520/

2. Destroy the current publisher first.

   CLI@host> publisher destroy <publisher of the repository to remove>

   Example:

   CLI@host> publisher destroy nexenta
CLI@host> publisher destroy HighAvailability

3. Discover publishers on CD.
   CLI@host> publisher discover
   MEDIA      PUBLISHER        LOCATION
   NS_UpgradeCD  nexenta   /media/NS_UpgradeCD/nexenta
   NS_UpgradeCD  HighAvailability  /media/NS_UpgradeCD/rsf

4. Configure newly discovered publishers.
   CLI@host> publisher create nexenta /media/NS_UpgradeCD/nexenta
   CLI@host> publisher create HighAvailability /media/NS_UpgradeCD/rsf

5. Verify the publisher list.
   CLI@host> publisher list
   PUBLISHER      STATUS      LOCATION
   nexenta        online      file:///media/NS_UpgradeCD/nexenta/
   HighAvailability  online    file:///media/NS_UpgradeCD/rsf/

6. Now initialize the tenant.
   CLI@host> tenant init
   When prompted, type “y” to initialize the tenant subsystem.
   Initialize tenant subsystem? [y/N] y
   98% Finalizing master tenant configuration transferred. Tenant subsystem successfully initialized
Creating Tenants

This section includes the steps and commands to set up tenants on the host system.

The following name conventions are used in the examples shown below:

CLI@host> Commands following the prefix CLI@host are management actions performed from the host.

CLI@tenant> Commands following the prefix CLI@tenant are management actions performed within a tenant.

Create Tenant

Prerequisites for a Clustered Environment

When you create tenants in a clustered environment, both nodes must be “active”. Also ensure that the License token for features HighAvailability and multiTenancy is activated on both nodes of the cluster. For this release, Nexenta only supports tenants that are configured only on pools with HA services so that in the event of either node failing, the surviving node takes over all tenants from the failed Nexentastor node.

Note:

• When creating tenants across a set of NexentaStor nodes, it is recommended to use unique tenant names across all NexentaStor nodes and specifically avoid reusing the same tenant name between clusters.

• While not required, a related best practice is to name VNICs that will be mapped to a particular tenant within the context of that tenant’s name. For example, if tenant “acme” is supposed to have 2 VNICs, you could use "acme.vnic1" and "acme.vnic2" as their VNIC names. This facilitates ensuring that VNIC names do not conflict across nodes (e.g. you would want to have 2 different tenants configured to use "vnic1") and that all the VNIC names that are required to support a particular tenant are properly configured across the set of nodes the tenant should be able to run on.

Steps to Create Tenant

1. Create tenant and associate the VNIC to the tenant you are creating.

   When creating a tenant provide the parent <parent> name along with the <quota> for that data storage space. Also provide the top level <dataset> name using the -d parameter. If you do not provide the top level <dataset> name, default name “storage” will be used. This dataset will be automatically delegated to the tenant you are creating.
Description of options you provide when you create a tenant:

- **p** `<parent>`
  All tenant datasets are stored in this data pool you specify. This allows tenants to simply inherit all high-availability configurations of the underlying data pool. As such, a pool failover event will automatically and seamlessly failover all the tenants it supports.

- **d** `<dataset>`
  This is the top level `<tenant_name>` dataset that is created as a child of the `<parent>` you specify. This forms the top level dataset for tenant file systems.

- `<tenant>`
  This tenant name contains a root dataset that will contain the tenant specific boot environment. `<tenant>` name also contains at least one `<storage>` dataset (named “storage” by default) that provides the top level dataset for tenant file systems, if not provided.

- **P** `<password>`
  This is the password you set for the tenant admin account that may be required to login to the tenant directly using a SSH client.

- `<vnic>`
  vnic that will be assigned to the tenant from the underlying host.

Note that vnics cannot be shared between tenants.

By default tenant create does not start the tenant once it is created unless you use the `-s` or `-start` flag to auto start tenant at creation. You can always start the tenant later, once it is created, using the **tenant start** command.

CLI@host> **tenant create** [-nvH] -p `<parent>` -V `<vnic>` -q `<quota>` [-P `<password>`][-d `<dataset>`] `<tenant>`

Example:

CLI@host> **tenant create** -p /tank -d tanktsr -V vnic20 -q 1GB -s tenanttest

Password for tenant admin:
Re-enter Password:
Name : tenanttest
Parent directory: /tank
VNIC : vnic20
Storage path: tantktsr
Storage size: 1GB
Admin password: *****

Note: tenant will be started automatically after creation.

Create a tenant using this configuration? [y/N] y

Note:
- Any tenant hosted on a pool (parent dataset) that has HA service will be HA enabled.
- Once a tenant exists, you can add additional storage and vnics to it from the host. See [Managing Tenant Resources from Host](#)
Verify Tenant Creation

1. From your host, list the tenant “tenanttest” you created.

   CLI@host> tenant list
   NAME        STATE    HOST        VERSION   CPU    MEM    QUOTA   USED AVAIL
   tenanttest  running  smc-53-109 5.2.1.3 0.05% 1.67% 1.0 GB 24.6 KB 1000.0 MB /tank/tenanttest

2. From your host view the dataset mount. Verify the dataset “tanktsr” is created under the parent directory “tank”.

   CLI@host> fs list
   PATH                    USED    AVAIL   REFER  NFS  SMB  MOUNTPOINT
   tank/tenanttest/tantktsr 24K     953.7M  24K    no   no   /tanktsr

   Later if you want to snapshot the whole tenant, you can run the snapshot on tank/tenant recursively.

3. From your host verify if the dataset is automatically delegated to the tenant.

   CLI@host> fs get delegated tank/tenanttest/tantktsr
   PATH                     PROPERTY   VALUE  SOURCE
   tank/tenanttest/tantktsr  delegated  yes

4. Log into the tenant you created and verify the dataset mount from your tenant.

   CLI@host> tenant login tenanttest
   [Connected to zone 'tenant' pts/2]
   Verify the dataset mount.

   CLI@tenant> fs list
   PATH      USED  AVAIL   REFER  NFS  SMB  MOUNTPOINT
   tantktsr  24K   953.7M  24K    no   no   /tantktsr

5. From the tenant you created, query for the VNIC you associated with the tenant.

   CLI@tenant> link show-vnic

6. Assign address to the newly added interface. The following example assigns a static IP address to the vnic20 NIC.

   CLI@tenant> ip create static <name> <address>

   Example:

   CLI@tenant> ip create static vnic20/v4 10.3.10.38/24

Note: The use of DHCP for network configurations is not supported for tenants. Once you create a tenant, assign static IP address to the interface you associate with the tenant. See Step 6 for information on assigning a static IP to the interface.
7. If you have already assigned an IP address and need to update the IP address, destroy the IP address and then assign a new IP as in the previous command.

   CLI@tenant> ip destroy vnic20/v4

8. View the NIC configuration to verify the newly added interface.

   CLI@tenant> ip list

   System response:

   NAME       TYPE    STATE  ADDRESS
   vnic20/v4  static  ok     10.3.10.53/24
   lo0/v4     static  ok     127.0.0.1/8
   lo0/v6     static  ok     ::1/128

9. Login to the tenant and create a Network Route.

   Create a static route manually using the CLI. When creating a route, a network or host should be specified as the destination, and a reachable network address should be specified as the gateway.

   To create the default route, specify 'default' for the destination.

   CLI@tenant> route create <destination> <gateway>
   Example:

   CLI@tenant> route create default 10.3.53.1

10. Verify the route you created.

   CLI@tenant> route list default

   DESTINATION  GATEWAY    PROTOCOL  INTERFACE  REFS  USE     STATIC
   default      10.3.53.1  ipv4      -          8     228397  yes

Start Tenant

If the auto-start option was not included during the tenant creation process, you may use the following command to start the tenant once it exists.

   CLI@host> tenant start <tenant>
   Example:

   CLI@host> tenant start tenanttest
   Start tenant 'tenanttest' ? [y/N] y

   Verify the state of the tenant.

   CLI@host> tenant list

   NAME      STATE    HOST       VERSION   CPU    MEM    QUOTA   USED     AVAIL
   ROOT
   tenanttest running host 5.2.1.3 8.92%  0.10%  1.0 MB  24.6 KB  975.4 KB /tank/tenanttest
If you must start all the tenants at the same time, use “all” option with the tenant start command.

CLI@host> tenant start all

Verify Boot Environment of the Tenant

Ensure that the tenant software version is same as the host’s. From the host, you can verify the BE of all the tenants using the `tenant list` command.

CLI@host> tenant list

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATE</th>
<th>HOST</th>
<th>VERSION</th>
<th>CPU</th>
<th>MEM</th>
<th>QUOTA</th>
<th>USED</th>
<th>AVAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT</td>
<td>running</td>
<td>host</td>
<td>5.2.1.3</td>
<td>0.04%</td>
<td>1.26%</td>
<td>2.1 GB</td>
<td>170.0 KB</td>
<td>2.1 GB</td>
</tr>
</tbody>
</table>

Managing Tenant Resources from Host

When a tenant is created with a top level `<dataset>`, the dataset is explicitly delegated to the tenant in the following hierarchy `<parent>/<tenant_name>/<dataset>`. You can manage the `<datasets>` under `<parent>/<tenant_name>` either from within the tenant as tenant admin or you can manage the delegated `<datasets>` from the host using the tenant add/remove storage commands. Similarly to manage the network resources use tenant add/remove vnic commands from the host or you can also perform the tenant resource management operations from within the tenant using the standard CLI commands available for the NexentaStor node itself.

Before managing the tenant resources from the host, you may want to view the current details of network configuration, storage details, and the resource utilization of a specific tenant.

CLI@host> tenant status

<table>
<thead>
<tr>
<th>STATE</th>
<th>HOST</th>
<th>UUID</th>
<th>VERSION</th>
<th>ROOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>running</td>
<td>smc-53-109</td>
<td>09ac1aba-6cd9-c7d3-d673-b1288256526d</td>
<td>5.2.1.3</td>
<td>/tank/tenanttest</td>
</tr>
</tbody>
</table>

Network configuration:

<table>
<thead>
<tr>
<th>VNIC</th>
<th>LINK</th>
<th>ADDRESS</th>
<th>ADDRTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td>vnic20</td>
<td>e1000g1</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Storage:

<table>
<thead>
<tr>
<th>NAME</th>
<th>QUOTA</th>
<th>USED</th>
<th>AVAILABLE</th>
<th>PATH</th>
</tr>
</thead>
<tbody>
<tr>
<td>tanktsr</td>
<td>1.0 MB</td>
<td>24.6 KB</td>
<td>975.4 KB</td>
<td>tank/tenanttest/tanktsr</td>
</tr>
</tbody>
</table>

Resource utilization:

<table>
<thead>
<tr>
<th>CPU</th>
<th>MEMORY</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.07%</td>
<td>1.30%</td>
</tr>
</tbody>
</table>
If a storage resource to be removed contains any active SMB/NFS shares, or open files and directories, you must remove the shares, or close files and directories that are open.

**Manage VNICs**

Any modifications to the vlinks should be done after stopping the tenant. vlinks cannot be added / removed while a tenant is running. Once you add a VNIC to a tenant, it is moved from the host storage system to the tenant, similarly when you remove a VNIC it becomes available to the host system.

**Add VNIC to a Tenant**

- **View the current VNIC Information of the Tenants.**
  
  CLI@host> tenant status

- **List the free VNICs available.**
  
  To see the VNICs that are not being used by any tenants, use the following command.
  
  CLI@host> link show-vnic

  Look for the TENANT column in the system response. Choose the vnic that is not used by any tenant.

<table>
<thead>
<tr>
<th>NAME</th>
<th>OVER</th>
<th>SPEED</th>
<th>MAC</th>
<th>MACTYPE</th>
<th>VID</th>
<th>TENANT</th>
</tr>
</thead>
<tbody>
<tr>
<td>vnic20</td>
<td>e1000g1</td>
<td>1000</td>
<td>2:8:20:e6:7:be</td>
<td>random</td>
<td>0</td>
<td>tenanttest</td>
</tr>
<tr>
<td>vnic2</td>
<td>e1000g1</td>
<td>1000</td>
<td>2:8:20:dc:79:38</td>
<td>random</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>

- **Add additional VNIC vnic2 to an already existing tenant, using the following command.**
  
  1. Stop the tenant if it is already running.
     
     CLI@host> tenant stop tenanttest

  2. Add the vnic.
     
     CLI@host> tenant add vnic [-nv] -t <tenant> <vnic_name>

     Example:
     
     CLI@host> tenant add vnic -t tenanttest vnic2

     In a clustered environment ensure that the vnic is also available on both nodes with identical names.

  3. Start the tenant.
     
     CLI@host> tenant start tenanttest

**Modify VNIC properties**

To make any necessary updates on the network, stop the tenant before the updates. Once the network updates are done, start the tenant again.

1. Stop the tenant if it is already running.
   
   CLI@host> tenant stop tenanttest
2. View the VNIC.
   
   CLI@host> link show-vnic vnic2

3. List the properties of the VNIC.

   CLI@host> link get all vnic2

4. Modify the VNIC properties of a specific tenant as needed.

   CLI@host> link set [-nvt] <properties> <link>

5. Start the tenant.

   CLI@host> tenant start tenanttest

---

**Note:**

- In the process of updating the existing network, if the IP address internal to the tenant changes, then any clients connected to the shares exported from this tenant would need to remount those shares.
- In a clustered environment, balance any VNIC/VLAN changes on both nodes. They are not automatically coordinated between nodes.

---

**Remove VNIC from a Tenant**

Once you remove a vnic from a tenant, you must reboot the host for the changes to be effective. Once a vnic is removed from the tenant, it becomes available to the host system.

1. Remove the network configuration in the tenant. Login to the tenant and destroy the IP address associated with the tenant.

   CLI@tenant> ip destroy vnic20/v4

2. Verify the IP is destroyed from the tenant by running the “ip list” command.

3. From the host, stop the tenant.

   CLI@host> tenant stop tenanttest

4. Remove the vnic from the tenant.

   CLI@host> tenant remove vnic -t tenanttest vnic20

   Remove vnic 'vnicg1' from tenant 'tenanttest'? [y/N] y

   Changes in tenant configuration require tenant reboot.

   Would you like to reboot tenant 'tenanttest' now? [y/N] y

   Rebooting tenant 'tenanttest'

   Tenant 'tenanttest' successfully rebooted.

5. Start the tenant.

   CLI@host> tenant start tenanttest
Manage Storage

Add Storage Capacity

Adding storage capacity to a tenant can be done using two ways:

- one is by increasing the quota on the existing parent dataset,
- and the other by adding a new storage dataset with its own quota.

 prow  Change Storage Quota for a Tenant prow
1. Verify the current quota status for a parent dataset of the tenants.
   CLI@host> tenant list <tenant>
   NAME        STATE    HOST        VERSION   CPU    MEM    QUOTA   USED
   AVAIL   ROOT
   tenanttest  running  smc-53-109  5.2.1.3  0.04%  1.23%  2.0 GB  24.6 KB
   2.0 GB  /tank

2. Log into the tenant you created and verify the dataset mount from your tenant.
   CLI@host> tenant login tenanttest
   [Connected to zone 'tenant' pts/2]
   Verify the dataset mount.

   CLI@tenant> fs list
   PATH      USED  AVAIL   REFER  NFS  SMB  MOUNTPOINT
tantktsr  24K   953.7M  24K    no   no   /tantktsr

3. From the host, add quota to the existing parent dataset or any datasets.
   CLI@host> fs set quota=2G tank/tenanttest/tantktsr

4. Verify the quota is added to the dataset.
   CLI@nexenta> fs get quota tank/tenanttest/tantktsr
   PATH                     PROPERTY  VALUE  SOURCE
   tank/tenanttest/tantktsr  quota     2GB    local

 prow  Add New Storage Pool to a Tenant prow
New storage capacity in the form of a top level file system can be delegated to a tenant.
However, the tenant will have to be restarted for that new storage to be visible and available
inside the tenant, the tenant must be restarted. The following “tenant add storage” command
creates a new top level file system under tenant’s parent path and delegates it to the tenant.

1. Add new dataset to an existing tenant.
   CLI@host> tenant add storage [-nv] -t <tenant> <storage_name> -q <quota>
   Example:
   CLI@host> tenant add storage [-nv] -t tenanttest newdataset -q 1MB
   Changes in tenant configuration require tenant reboot.
Would you like to reboot tenant 'tenanttest' now? [y/N]

2. From your host verify if the dataset is delegated to the tenant.

   CLI@host> fs get delegated tank/tenanttest/newdataset
   PATH                        PROPERTY   VALUE  SOURCE
   tank/tenanttest/newdataset  delegated  yes

3. Or login to your tenant and verify the storage is added.

   CLI@tenant> fs list
   PATH                       USED  AVAIL   REFER  NFS  SMB  MOUNTPOINT
   tank                        47K   929.6K  24K    no   no   /tank
   tank/tenanttest/newdataset  23K   929.6K  23K    no   no   /tank/

Remove Storage from a Tenant

In order to remove a top level dataset that is delegated to a tenant, the dataset should not contain any child filesystem. If the top level dataset to be removed has in-tenant filesystems then they must be first destroyed before the storage can be removed from the tenant.

To remove storage capacity from a tenant, run the following commands from the host:

1. Ensure that there are no filesystems under the dataset (tank/tenanttest/newdataset) you are about to remove.

   CLI@host> fs list
   PATH                       USED  AVAIL   REFER  NFS  SMB  MOUNTPOINT
   tank                        47K   929.6K  24K    no   no   /tank
   tank/tenanttest/newdataset  23K   929.6K  23K    no   no   /tank/

2. If there are any under the filesystem, tank/tenanttest/newdataset, login to the tenant and remove it from the tenant as shown in this example. If not, proceed to Step 3.

   CLI@tenant> fs destroy tank/tenanttest/newdataset/fsnew
   Destruction of a file system is irreversible.
   All data on the destroyed file system will be lost.
   Enter the pool name to proceed, or press Enter to cancel: tank

3. Now remove the top level dataset from the host.

   CLI@host> tenant remove storage [-nyv] -t <tenant> <storage_name>
   Example:

   CLI@host> tenant remove storage -t tenanttest newdataset
   Remove dataset 'storage' from tenant 'tenanttest' ? [y/N] y
   This command will undelegate the dataset and destroy it. If needed, reboot the tenant to undelegate the dataset.
Monitor CPU and Memory Utilization

Use the following "tenant" commands to monitor how much CPU and Memory are consumed by a tenant.

- To view the CPU and Memory resources being consumed by each tenant
  
  CLI@host> tenant status

  Example:
  
  Name: tenanttest
  
  STATE HOST UUID VERSION ROOT
  running smc-53-109 6b51514b-b316-4114-ee5b-f71ece3c245b 5.2.1.3 /
tank/tenanttest

  Network configuration:

  VNIC LINK ADDRESS ADDRTYPE
  vnic20 e1000g1 10.3.53.150/22 static

  Storage:

  NAME QUOTA USED AVAILABLE PATH
  storage 1.0 GB 49.2 KB 1000.0 MB tank/tenanttest/storage

  Resource utilization:

  CPU MEMORY
  8.28% 0.08%

Managing Tenants

This following table covers the management actions available on the tenants.

Table 2-1: Managing Tenants

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Starting a tenant</td>
<td>CLI@nexenta&gt; tenant start &lt;tenant&gt;</td>
</tr>
<tr>
<td></td>
<td>Starts a specific tenant.</td>
</tr>
<tr>
<td>Stopping a tenant</td>
<td>CLI@nexenta&gt; tenant stop &lt;tenant&gt;</td>
</tr>
<tr>
<td></td>
<td>Stops a specific tenant.</td>
</tr>
<tr>
<td>Rebooting a tenant</td>
<td>CLI@nexenta&gt; tenant reboot &lt;tenant&gt;</td>
</tr>
<tr>
<td></td>
<td>Reboots a specific tenant.</td>
</tr>
<tr>
<td>Destroying a tenant</td>
<td>CLI@nexenta&gt; tenant destroy &lt;tenant&gt;</td>
</tr>
<tr>
<td></td>
<td>Destroys a specific tenant and all its underlying datasets, freeing up all</td>
</tr>
<tr>
<td></td>
<td>associated resources&quot;.</td>
</tr>
</tbody>
</table>
**Start a Tenant**

To start a tenant that was in the stopped state, use the following command.

```
CLI@host> tenant start <tenant>
```

Example:

```
CLI@host> tenant start tenanttest
```

**View Tenant Status**

To view the detailed information about tenants, including all its resources such as general configuration, network resources, active shares, and so on, use the following command. You can also view the tenant’s VNIC’s status using this command.

```
CLI@host> tenant status <tenant>
```

Example:

```
CLI@host> tenant status tenanttest
```

Table 2-2: Different Possible States of Tenants

<table>
<thead>
<tr>
<th>State</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Configured</td>
<td>The tenant’s configuration is complete and saved on the host. However the tenant’s operating environment is not yet installed on this host. This is the state that will be shown while transitioning as part of a failover. Once a tenant has been booted on a host node the tenant list command will show the state as running.</td>
</tr>
<tr>
<td>• Incomplete</td>
<td>During a tenant create or destroy operation, the state of the target tenant will show as incomplete. Upon successful completion of the operation, the state is set to the correct state. A tenant left in this state is not usable and would need to be cleaned up with a tenant destroy.</td>
</tr>
<tr>
<td>• Installed</td>
<td>The tenant’s configuration is instantiated on the system and packages are installed under the tenant’s root path. This is a state shown for a tenant that is configured and installed, but is not running.</td>
</tr>
<tr>
<td>• Ready</td>
<td>The “virtual platform” for the tenant is established. This simply means that the kernel created the zsched process, network interfaces are set up and available to the tenant, the tenant’s file systems are mounted, the tenant’s devices are configured and a unique tenant ID is assigned by the system. At this stage, no processes associated with the tenant have been started. This is another transitional state that may be shown while a tenant is booting. Once the tenant is booted the state will change to “running”.</td>
</tr>
<tr>
<td>• Running</td>
<td>User processes associated with the tenant environment are running. The tenant enters the running state as soon as the first user process associated with it’s environment (init) is created.</td>
</tr>
</tbody>
</table>
Stop a Tenant

In a scenario where you need to destroy a tenant, you must first stop the tenant.

CLI@host> tenant stop <tenant>

Example:

CLI@host> tenant stop tenanttest
Stop tenant 'tenanttest' ? [y/N] y

CLI@host> tenant list

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATE</th>
<th>HOST</th>
<th>VERSION</th>
<th>CPU</th>
<th>MEM</th>
<th>QUOTA</th>
<th>USED</th>
<th>AVAIL</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROOT</td>
<td>installed</td>
<td>smc-53-109</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.0 MB</td>
<td>24.6 KB</td>
<td>975.4 KB</td>
</tr>
</tbody>
</table>

Once the tenant is stopped and the host storage system is rebooted for some reason, you must restart the stopped tenant. The stopped state of the tenant persists across reboots. Similarly, if you reboot the host and it had tenants in the running state those tenants will come back to running state when the host is rebooted.

Reboot a Tenant

For the new delegated dataset to be visible, reboot the tenant. Tenants must be in “running” state in order to reboot.

CLI@host> tenant reboot tenanttest

Destroy a Tenant

Using the command listed here you can destroy a specific tenant and all its underlying datasets and free up all associated resources.

1. Before you destroy a tenant, you must stop the tenant if it is running.

   CLI@host> tenant stop tenanttest

2. Destroy the tenant.

   CLI@host> tenant destroy tenanttest

   Destroy tenant 'tenanttest' ? [y/N] y

3. Verify the associated resources are freed up and the underlying datasets are deleted.
Managing MultiTenant Configurations via REST API

Managing tenant services can be done either by logging into the tenant as host admin through the main NexentaStor management interface or by logging into the tenant as tenant admin through tenant specific management interfaces. Both CLI and REST API are available to a tenant admin over the tenant specific management interfaces. To enable access to the detailed REST API documentation from the tenant, see Enable Swagger API Documentation.

The REST API on the main NexentaStor management interface also supports management of all in-tenant resources and services. To enable access to the detailed REST API documentation from the host, see NexentaStor 5.x REST API QuickStart Guide.

To learn about the details on the management operations that can be performed from the host REST API, explore the APIs by selecting the methods under “tenants”.

<table>
<thead>
<tr>
<th>Method</th>
<th>Path</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>GET</td>
<td>tenants</td>
<td>Show/Hide</td>
</tr>
<tr>
<td></td>
<td>Collection of tenants</td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>tenants</td>
<td>List Operations</td>
</tr>
<tr>
<td></td>
<td>Create a new tenant</td>
<td></td>
</tr>
<tr>
<td>GET</td>
<td>tenants/{tenantName}</td>
<td>List Operations</td>
</tr>
<tr>
<td></td>
<td>Get available tenants</td>
<td></td>
</tr>
<tr>
<td>DELETE</td>
<td>tenants/{tenantName}</td>
<td>List Operations</td>
</tr>
<tr>
<td></td>
<td>Destroy a tenant</td>
<td></td>
</tr>
<tr>
<td>PUT</td>
<td>tenants/{tenantName}</td>
<td>List Operations</td>
</tr>
<tr>
<td></td>
<td>Set the given tenant properties</td>
<td></td>
</tr>
<tr>
<td>POST</td>
<td>tenants/{tenantName}/checkTenantSubsystem</td>
<td>Check tenant subsystem</td>
</tr>
<tr>
<td>GET</td>
<td>tenants/{tenantName}/datasets</td>
<td>List of top-level datasets provisioned to a tenant</td>
</tr>
<tr>
<td>POST</td>
<td>tenants/{tenantName}/datasets</td>
<td>Add top-level dataset to given tenant</td>
</tr>
<tr>
<td>GET</td>
<td>tenants/{tenantName}/datasets/{path}</td>
<td>Get provisioned datasets</td>
</tr>
</tbody>
</table>
The following chapter, Chapter 3, Managing In-Tenant Services, provides some examples of the management operations that can be performed within a specific tenant using the CLI.
Managing In-Tenant Services

This chapter covers the following topics:

- Logging in to a Tenant
- Commands Available from a Tenant
- Post Tenant-Creation Checks
- Basic In-Tenant Configurations
- Managing Network
- Managing Datasets

Logging in to a Tenant

You can log in to any tenants as a host admin through the host management interface. A host admin can log in to any tenant without having to provide any password. However, when you log into a tenant, you will be switched to the context of that specific tenant and will be allowed to manage only that specific tenant on the host system.

You can also log in to a tenant from any SSH client application through the tenant management IP interface.

Commands Available from a Tenant

Once you set up the tenant(s), you can perform the management operations within a specific tenant by logging into the tenant. Each tenant can be configured with its own IP networks, its own Active Directory and user authentication services, its own NFS and SMB server settings, totally independent of other tenants on the same NexentaStor appliance.

The above operations can be done using the standard CLI commands available for the management of the NexentaStor node itself. These operations can be run only on storage resources owned by the tenant you log into. To learn about the standard CLI commands associated with tenant-specific management, use Nexentator CLI Config Guide in conjunction with this document. However, within a tenant you will not have all the commands that you can normally execute from a host storage system. To identify the commands that you can execute from a tenant, type “help” in the CLI prompt as shown here.

- To view the commands that can be executed from a tenant:
  1. From the host, log in to the tenant as tenant admin.
     
     CLI@host> tenant login tenanttest
  
  2. Type “help” at the cli prompt.
CLI@tenant> help

List of available shell utilities:

<table>
<thead>
<tr>
<th>acl</th>
<th>config</th>
<th>ip</th>
<th>net</th>
<th>snapshot</th>
</tr>
</thead>
<tbody>
<tr>
<td>alert</td>
<td>filesystem</td>
<td>journal</td>
<td>nfs</td>
<td>software</td>
</tr>
<tr>
<td>audit</td>
<td>group</td>
<td>idmap</td>
<td>ldapclient</td>
<td>route</td>
</tr>
<tr>
<td>auditreduce</td>
<td>idmap</td>
<td>link</td>
<td>smb</td>
<td>user</td>
</tr>
</tbody>
</table>

List of available management commands:

- **Verify software version installed**
  CLI@tenant> software version

- **Verify the installed software and the boot environment**
  CLI@tenant> software list

- **See your system’s baseline state**
  CLI@tenant> system status

- **Check that the system services of interest are enabled**
  CLI@tenant> svc list

- **Validate the system date and time, and time zone**
  CLI@tenant> config list system.date
  CLI@tenant> config list system.timeZone

  Ensure that the time, date, and time zone are consistent with the host.

For details on the commands to verify the baseline settings, use NexentaStor CLI Configuration Guide.

**Post Tenant-Creation Checks**

Before you proceed with creating and managing child datasets under the parent dataset from within the tenant review the following baseline settings.

- Verify software version installed
  CLI@tenant> software version

- Verify the installed software and the boot environment
  CLI@tenant> software list

- See your system’s baseline state
  CLI@tenant> system status

- Check that the system services of interest are enabled
  CLI@tenant> svc list

- Validate the system date and time, and time zone
  CLI@tenant> config list system.date
  CLI@tenant> config list system.timeZone

  Ensure that the time, date, and time zone are consistent with the host.
Basic In-Tenant Configurations

Now that you have verified the baseline settings you can complete the basic optional configurations. Some of the basic settings are accomplished by setting the management configuration variables. To view the complete list of supported tenant management configuration variables, type “config list” after logging into the tenant through a host.

```bash
CLI@host> tenant login tenanttest
[Connected to zone 'tenant' pts/2]
CLI@tenant> config list
```

This provides the full list of appliance variables that can be configured from within a tenant.

The following sections are examples of few tenant-specific basic configurations. For a complete list of supported operations on a tenant, type “help” after logging into the tenant through the host.

See [Commands Available from a Tenant](#) for information on how to utilize the standard NexentaStor CLI commands to accomplish the following configurations. And for details on the commands refer [NexentaStor CLI Configuration Guide](#).

### Time Date Configuration

By default a tenant inherits the date, time and time zone from the host once you create the tenant but if you changed the time and date or timeZone of the host at later point then you should login to the tenants individually and set the time, date, and timeZone to be consistent with the host.

Login to the tenant and run the following CLI commands:

- **To set the tenant date and time to be consistent with the host:**
  ```bash
  CLI@tenant> config set system.date=3/31/2018 9:20:00
  ```

- **To set the tenant timeZone to be consistent with the host:**
  ```bash
  CLI@tenant> config set system.timeZone=America/Los_Angeles
  ```

### Enable Swagger API Documentation

Enable access to the detailed REST API documentation from the tenant. See [NexentaStor 5.x REST API QuickStart Guide](#) for more details.

1. Login to the tenant and run this CLI command:
   ```bash
   CLI@tenant> config set rest.useSwagger=true (case-sensitive parameter)
   ```

2. Point your browser to [https://<system.managementAddress>:8443/docs](https://<system.managementAddress>:8443/docs) using the management address you added to the interface for the tenant.
   <staticIPaddress> should be the management IP address of the tenant.

3. Login to try out the APIs using the password you set up during the tenant creation.
4. Explore specific APIs by selecting a method, entering part of the URL string, and clicking Try It Out.

- Within a tenant you will have only the tenant specific URLs.
- Enabling access in a production environment is not recommended due to potential security risks. Regardless of the Swagger Documentation interface is enabled or not, the API is always available for use.

**Set Management Address**

Once the tenant is created successfully, the management address is set to open by default. However, you may set the management address to a VNIC address on the tenant.

1. Verify the default management address.
   ```
   CLI@tenant> config list | grep system.managementAddress
   system.managementAddress                 --     0.0.0.0
   ```

2. Set the management address to a VNIC IP other than the default value.
   ```
   CLI@tenant> config set system.managementAddress=<IP>
   ```

3. Verify the changes made to the management IP address.
   ```
   CLI@tenant> config get all system.managementAddress
   NAME                      PROPERTY     VALUE
   system.managementAddress  description  Management IP address
   system.managementAddress  flags        --
   system.managementAddress  name         system.managementAddress
   system.managementAddress  schema       <STRING>
   system.managementAddress  value        10.3.53.149
   ```

**Change Admin User Password**

Tenant admin can have a different password from the host admin. This password is used to login to the tenant over the tenant management IP interface.

After tenant creation you can login to the tenant and change the tenant admin password you set up during the tenant creation.

```shell
CLI@tenant> user passwd [-p <password>] <login>
```  

**Create a New User/New User Group**

You can create as many user accounts as needed. Use the `user`, `group` commands to create local tenant users and groups for NFS and SMB users. When creating a new user, you can specify the group the user belongs to.
To create a new user:

```
CLI@tenant> user create [options] [-p <password>] [-g <group>] <login>
```

By default a user will be created with “default” profile and will have no access to the tenant using the CLI and API. However, you can configure file share permissions for these users. Once you create a user, you may enable API access/admin privileges anytime using the `user set-profile` command.

To create a new group:

```
CLI@tenant> group create <group name>
```

You can limit the amount of space consumed by filesystem that are owned by a particular user or group. See NexentaStor CLI Configuration Guide for the commands to dynamically manage the disk quota for a user or a group. If your environment has a large number of users or groups, consider setting user and group quotas.

Restricting the space consumed by tenant-filesystem for a user or group must be done from the tenants.

**Configure SMTP Email Service**

- Query and modify the email address of your tenant admin user.
- Configure email notifications for alerts by querying and modifying the alert settings.
- Set up SMTP mail server for the tenant.
- Query and modify SMTP service attributes.

```
CLI@tenant> config set system.administratorEmail=<email address>
CLI@tenant> config set alert.email.address=abcd@nexenta.com
CLI@tenant> config set smtp.host=smtp.gmail.com
CLI@tenant> config set smtp.port=465
CLI@tenant> config set smtp.authMethods=PLAIN
CLI@tenant> config set smtp.user=abcd@gmail.com
CLI@tenant> config set smtp.password=nexenta1
CLI@tenant> config set smtp.security=ssl
CLI@tenant> config set smtp.senderEmail=test@gmail.com
```

**Upload the SSL Certificate**

Tenant uses a self-signed SSL certificate. You may update the SSL certificate if your company security policy requires to use a specific SSL certificate.

- To generate a new certificate:
  - Type
    ```
    CLI@tenant> config set rest.certificate.generate=true
    ```
    For details on uploading an SSL certificate, see NexentaStor CLI Configuration Guide.
Managing Network

The following sections are examples of few management tasks that you can perform on a tenant.

Verify the Available Network Interface

During the tenant creation, you may have set up a network interface card (NIC) for the tenant. Verify that the network interface is in place and configured.

Assign or Update the IP Address to Network Interface

You must have already assigned an IP address to the network interface. See Step 6 for information on assigning an IP address. If you have already assigned an IP address and needed to update the IP address, destroy the IP address and then assign a new IP. Also optionally, set a new DNS name server.

Create a Network Route

If you have not already set up the network route, you can also create a static route manually using the CLI. When creating a route, a network or host should be specified as the destination, and a reachable network address should be specified as the gateway.

Use the `route`, `net`, `ip` commands to configure tenant with its own IP networks totally independent of other tenants on the same NexentaStor appliance. However creating and managing a system link on a tenant must be handled from the host.

Managing Datasets

Enable System Service

By default the services offered within a tenant are disabled. However, the SMB and NFS services get enabled automatically when you create an SMB share and NFS share respectively.

View the list of services within a tenant by logging in to the tenant and executing the following command. Ensure that you enable the system services of interest.

```
CLI@tenant> svc list
NAME        DESCRIPTION    STATE
idmap       idmap service  disabled
ldapclient  LDAP client    disabled
nfs         NFS server     disabled
smb         SMB server     disabled
smbclient   SMB client     disabled
```
Configure Filesystems

You can create and manage tenant specific filesystems and file services from within the tenant by logging into the tenant through the host. You may share the datasets for anonymous access or authenticated access in workgroup mode or domain. These operations can be done using the same standard CLI commands that are also used to manage filesystems in NexentaStor.

From within the tenant or from the host you can manage quota on all the tenant filesystems, similar to how you would set and manage quotas on file system hierarchies in a standard pool. If additional storage capacity is needed for a tenant, the simplest solution is to increase the quota on the top level /tenant/storage dataset from the host. This can be done without disruption to the host or tenant services.

See Add Storage Capacity for information on adding quota to an existing parent dataset.
Configure NFS & SMB Services

This section uses the following name conventions as examples:

- parent dataset
- tenant_name
- storage dataset (the top level dataset for tenant filesystems)
- tenant filesystems

When creating a tenant, you must have specified a parent dataset which is the data pool that will store all tenant datasets. A top level <tenant_name> dataset is then created as a child of the specified parent. This <tenant_name> also contains at least one <storage> dataset (named “storage” by default) that provides the top level dataset for tenant filesystems. To allow access to these filesystems for NFS and SMB users, configure NFS and SMB shares on these filesystems that are created underneath the top level dataset <storage>.

```
The tenant filesystems when viewed from the host are displayed as shown here:

CLI@host> fs list
PATH                         USED  AVAIL  REFER  NFS  SMB MOUNTPOINT
parent/tenant_1/storage      75K   901.6K  25K   no   no  /storage
parent/tenant_1/storage/fsa1 13K   901.6K  24K    no   yes /storage/fsa1
parent/tenant_1/storage/fsa2  24K  901.6K   24K    yes  no   /storage/fsa2
```

```
The tenant filesystems when viewed from the tenant, tenant_1, are displayed as shown here:

CLI@tenant_1> fs list
PATH          USED  AVAIL   REFER  NFS  SMB  MOUNTPOINT
storage       75K   901.6K  25K    no   no   /storage
storage/fsa1  13K   901.6K  24K    no   yes /storage/fsa1
storage/fsa2  24K   901.6K  24K    yes  no   /storage/fsa2
```

In the example shown here, shares can be created only on <fsa1>,<fsa2>, and <fsa3> not on the top level dataset <storage>.
Configure AD/LDAP

NexentaStor supports concurrent sharing of any filesystem over NFS and SMB while maintaining tight control on end-user authentication and access permissions. This is managed through the NexentaStor identity mapping service which can be configured to Use a directory service, such as Active Directory (AD).

Configure ACLs

Configure ACL on the filesystem to allow the respective users to access the filesystems delegated to a tenant.

Configure Manual Snapshots

The only data protection mechanism that is managed directly from within a tenant is on demand manual snapshot. All other advanced data protection services for tenant filesystems are managed from the Host.

Managing Boot Environment Checkpoints in a Tenant

Activate Boot Environment for a Tenant

You can control what BE is used to boot a particular tenant by rolling back to a specific checkpoint.

A bootable snapshot of the tenant’s system configuration at a specific point in time is called a boot environment checkpoint. A checkpoint is automatically created when the host system is upgraded. However, it is recommended to create a new checkpoint before making significant configuration changes on the tenant so that if the configuration change doesn’t work or triggers something unexpected, you can roll back to an old configuration to restore the state of a system at a particular point in time. You can create a new checkpoint using the `software checkpoint` command.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Creating a checkpoint</td>
<td><code>CLI@tenant&gt; software checkpoint &lt;name of the checkpoint&gt;</code>&lt;br&gt;You can create a system bootable checkpoint of the system configuration at a specific point in time.&lt;br&gt;<strong>Note:</strong> Do not use the name <code>zbe-&lt;#&gt;</code> for the checkpoints you create because the automatic checkpoints created by the system follow this naming convention.</td>
</tr>
<tr>
<td>Rolling back to a checkpoint</td>
<td><code>CLI@tenant&gt; software activate &lt;name of the checkpoint&gt;</code>&lt;br&gt;You roll back to a NexentaStor checkpoint to restore the state of a system at a particular point in time.</td>
</tr>
<tr>
<td>Viewing existing checkpoints</td>
<td><code>CLI@tenant&gt; software list</code>&lt;br&gt;You can view the current list of checkpoints and information about each one.</td>
</tr>
<tr>
<td>Display specific checkpoints</td>
<td><code>CLI@tenant&gt; software list &lt;name of the checkpoint&gt;</code></td>
</tr>
</tbody>
</table>
Deleting a checkpoint

CLI@tenant> software destroy <name of the checkpoint>
You can delete a checkpoint, or multiple checkpoints, immediately to free space on the disk.

Note:

Implications of rolling back to a checkpoint:

When you roll back to a checkpoint, all the changes done after the creation of the checkpoint will be lost and you will be reverted back to the configuration state of the rolled-back checkpoint. The package version, log files, configuration files and so on will go back to the state as when the checkpoint was created. Also note that rolling back to a checkpoint involves a reboot of the appliance.
NexentaStor Upgrades with Multi-Tenant Configuration

This chapter covers the following topics:

- Upgrade Tenants in a Clustered Environment
- Supported Upgrade Configuration
- Upgrade Steps
- Rollback Clustered Nodes with Multi-Tenants
- Rollback Steps with Examples

Upgrade Tenants in a Clustered Environment

In a clustered environment the nodes are upgraded one at a time. Before the first node is upgraded, the tenants from the node to be upgraded are failed over to the other node before the first node is upgraded and rebooted. All the tenants on the second node, including the failed over tenants, are then upgraded on the second node along with that node. Before the second node is rebooted, the failed over tenants are then failed back to the first node. Once both nodes have the new software bits, move the pools back to its original configuration.

Supported Upgrade Configuration

For this release, Nexenta only supports tenants that are configured on pools with HA services. This example shows:

- two nodes (NodeA and NodeB),
- two pools (PoolA and PoolB) with PoolA on NodeA and PoolB on NodeB,
- a HA service (PoolA) running on NodeA and a HA service (PoolB) running on NodeB,
- one tenant (tenantA) running on NodeA and on HA pool 'PoolA'.

Upgrade Steps

This section covers the following steps:

1. List the tenants running on NodeA.
2. Failover tenants (tenantA) from NodeA to NodeB.
   All services (PoolA and PoolB) will be running on NodeB.
3. Upgrade NodeA.
4. Reboot NodeA.
   Now the first host node (NodeA) is running the new appliance version.

5. Upgrade NodeB that in turn upgrades all tenant(s) on NodeB, in this example tenantA.

6. Now failover all upgraded tenants (tenantA) from NodeB to NodeA.
   By now all tenants are ready to boot into the updated version.
   All services (PoolA and PoolB) will be running on NodeA.

7. Reboot NodeB.

8. Move PoolB to NodeB to bring back to the original cluster configuration.

   Steps to upgrade the tenants:

1. List the tenants running on NodeA.
   CLI@NodeA> tenant list
   NAME  STATE   HOST    VERSION   CPU    MEM    QUOTA   USED     AVAIL   ROOT
   tenantA running NodeA  5.2.0.15  0.21%  0.23%  5.0 GB  24.6 KB  5.0 GB /PoolA

2. Verify the existing pools on both nodes.
   CLI@NodeA> pool list
   NAME        SIZE     ALLOC    FREE     AVAIL  DEDUP  EXPANDSZ  FRAG  HEALTH
   rpool       899G     191.95G  707.05G  79%    1.00x  -         9%    ONLINE
   PoolA  269.31G  717.6M   268.61G  100%   1.00x  -         0%    ONLINE
   CLI@NodeB> pool list
   NAME   SIZE     ALLOC    FREE     AVAIL  DEDUP  EXPANDSZ  FRAG  HEALTH
   qqq    538.62G  100.3M   538.53G  100%   1.00x  -         0%    ONLINE
   PoolB  899G     167.79G  731.21G  81%    1.00x  -         34%   ONLINE

3. From any node, verify the HA service running on the nodes.
   CLI@NodeA> haservice list
   NAME        GUID                 DESCRIPTION           VIPs  NODES
   RUNNING   STOPPED   BROKEN
   PoolA  3961466805660122306  No description given        NodeA,NodeB  NodeA
   NodeB  -
   PoolB  3961466805660122306  No description given        NodeA,NodeB  NodeB
   NodeA  -

4. Failover all tenant(s) from NodeA to NodeB.
   The example here shows only one cluster service (PoolA) running on NodeA. If you have more
   than one service on the node, when failing over all the services will fail over to the other node
   in the cluster.
   CLI@NodeA> haservice failover <from-node> <to-node>
Example:

CLI@NodeA> haservice failover NodeA NodeB

The following services can be moved:

PoolA

Move 1 service(s) from node 'NodeA' to node 'NodeB'? [y/N] y

Moving service 'PoolA' ... OK

All running services have been successfully moved

Now the HA Service failed over to NodeB and PoolA imported to NodeB

5. From NodeA or NodeB, verify that the HA service PoolA from NodeA failed over to NodeB.

CLI@NodeA> haservice status

System response:

service: PoolA

<table>
<thead>
<tr>
<th>NODE</th>
<th>STATUS</th>
<th>MODE</th>
<th>UNBLOCKED</th>
<th>ERRORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NodeB</td>
<td>running</td>
<td>automatic</td>
<td>yes</td>
<td>_</td>
</tr>
<tr>
<td>NodeA</td>
<td>stopped</td>
<td>automatic</td>
<td>yes</td>
<td>_</td>
</tr>
</tbody>
</table>

service: PoolB

<table>
<thead>
<tr>
<th>NODE</th>
<th>STATUS</th>
<th>MODE</th>
<th>UNBLOCKED</th>
<th>ERRORS</th>
</tr>
</thead>
<tbody>
<tr>
<td>NodeB</td>
<td>running</td>
<td>automatic</td>
<td>yes</td>
<td>_</td>
</tr>
<tr>
<td>NodeA</td>
<td>stopped</td>
<td>automatic</td>
<td>yes</td>
<td>_</td>
</tr>
</tbody>
</table>

6. From NodeB, verify that the tenantA from NodeA imported to NodeB.

CLI@NodeB> tenant list

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATE</th>
<th>HOST</th>
<th>VERSION</th>
<th>CPU</th>
<th>MEM</th>
<th>QUOTA</th>
<th>USED</th>
<th>AVAIL</th>
<th>ROOT</th>
</tr>
</thead>
<tbody>
<tr>
<td>tenantA</td>
<td>configured</td>
<td>NodeB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0 Bytes</td>
<td>0</td>
<td>Bytes</td>
<td>/PoolA</td>
</tr>
</tbody>
</table>

The configured state indicates that the tenantA configuration is complete and saved on the host. However the tenant’s operating environment is not yet installed on this host.

This configured state is displayed while transitioning as part of a failover. Once a tenant has been booted on a host node the tenant list command will display the state as running.

7. From NodeB, verify that PoolA is imported to NodeB.

CLI@NodeB> pool list

<table>
<thead>
<tr>
<th>NAME</th>
<th>SIZE</th>
<th>ALLOC</th>
<th>FREE</th>
<th>AVAIL</th>
<th>DEDUP</th>
<th>EXPANDSZ</th>
<th>FRAG</th>
<th>HEALTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>rpool</td>
<td>899G</td>
<td>167.79G</td>
<td>731.21G</td>
<td>81%</td>
<td>1.00x</td>
<td>-</td>
<td>34%</td>
<td>ONLINE</td>
</tr>
<tr>
<td>PoolA</td>
<td>269.31G</td>
<td>725.0M</td>
<td>268.60G</td>
<td>100%</td>
<td>1.00x</td>
<td>-</td>
<td>0%</td>
<td>ONLINE</td>
</tr>
<tr>
<td>PoolB</td>
<td>269.31G</td>
<td>725.0M</td>
<td>268.60G</td>
<td>100%</td>
<td>1.00x</td>
<td>-</td>
<td>0%</td>
<td>ONLINE</td>
</tr>
</tbody>
</table>

8. Now upgrade NodeA. Before the actual upgrade, do a dry run to ensure that you are upgrading to the intended version.
CLI@NodeA> software upgrade -n
System response:
Would perform upgrade from version 5.2 to later versions.

9. **Now run the actual upgrade.**

All tenants on the host node must be running in order to upgrade. If any of the tenants are halted for some reason, upgrade of the host node will fail.

CLI@NodeA> software upgrade
By default, this allows you to upgrade to the latest 5.2 version.

10. **Reboot once the upgrade completes by typing y when prompted. Or to reboot automatically once the upgrade has successfully completed, use the –y option along with the upgrade command.**

CLI@NodeA> software upgrade -y

11. **Confirm that the upgrade version is activated and to see your boot environment list.**

CLI@NodeA> software list

<table>
<thead>
<tr>
<th>NAME</th>
<th>SPACE</th>
<th>ACTIVENOW</th>
<th>ACTIVEAFTERREBOOT</th>
<th>CREATIONTIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>nexenta</td>
<td>6.3M</td>
<td>no</td>
<td>no</td>
<td>Feb 1 16:29:04</td>
</tr>
<tr>
<td>nexentastor-1</td>
<td>2.77G</td>
<td>yes</td>
<td>yes</td>
<td>Feb 6 15:02:56</td>
</tr>
</tbody>
</table>

12. **Validate the software version.**

CLI@NodeA> software version

<table>
<thead>
<tr>
<th>PUBLISHER</th>
<th>VERSION</th>
<th>PACKAGINGDATE</th>
</tr>
</thead>
<tbody>
<tr>
<td>nexenta</td>
<td>5.2.1</td>
<td>Mon May 6 08:49:24 2017</td>
</tr>
</tbody>
</table>

13. **From NodeB verify that the NodeA is listed in the cluster after upgrading NodeA.**

CLI@NodeB> hacluster status

== Cluster status ==
NAME  STATUS  NODES  SERVICES  DESCRIPTION
nef   ok      2/2    1/1       No description given

== Cluster configuration ==
FC MONITORING  SERIAL HEARTBEATS ENABLED  NETWORK MONITORING
no             no                         yes

== Nodes ==
NODE   STATUS  SERVICES  ADDRESS      HostId    Release
NodeB  up      2/2       172.16.1.56  907658bd  5.2.0.15
NodeA  up      0/2       172.16.1.57  43c7b5c2  5.2.1.3

Now the first host node NodeA is running the new appliance version.

14. **Verify the HA service running on the nodes.**

CLI@NodeA> haservice list
15. Now Upgrade NodeB that in turn upgrades all tenant(s) on NodeB. All tenants on the host node must be running in order to upgrade. If any of the tenants are halted for some reason, upgrade of the host node will fail. Before the actual upgrade, do a dry run to ensure that you are upgrading to the intended version.

```
CLI@NodeB> software upgrade -n
```

System response:
```
Would perform upgrade from version 5.2.0 to 5.2.1.
```

16. Now run the actual upgrade. Make sure that you do not use the `-y` option.

```
CLI@NodeB> software upgrade
```

By default, this allows you to upgrade to the latest version.

17. Type “N” when you are prompted with the reboot question. You must not reboot the node at this point. Rebooting the node to the upgraded software version must be done after you failover all upgraded tenants.

18. Now failover all upgraded tenants from NodeB to NodeA so the HA service “PoolA” is back in its original configuration.

```
CLI@NodeB> haservice failover <from-node> <to-node>
```

Example:

```
CLI@NodeB> haservice failover NodeB NodeA
```

19. From NodeA or NodeB, verify that the HA Service failed over to NodeA.

```
CLI@NodeA> haservice status
```

System response:
```
service: PoolA

NODE     STATUS   MODE    UNBLOCKED
NodeB     stopped  manual  yes
NodeA     running  manual  yes
```

```
service: PoolB

NODE     STATUS   MODE    UNBLOCKED
NodeB     stopped  manual  yes
NodeA     running  manual  yes
```

20. From NodeA, verify that the tenantA moved from NodeB to NodeA and also verify the tenants are upgraded by checking the software version of the tenant.
CLI@NodeA> tenant list
NAME     STATE   HOST     VERSION  CPU     MEM     QUOTA     USED
        AVAIL    ROOT
tenantA  running NodeA  5.2.1.3  0.10%  0.11%  107.4 GB  24.6 KB
107.4 GB /PoolA/tenantA

21. Reboot NodeB.

22. From NodeA move PoolB back to NodeB so the cluster is back in its original configuration with PoolA on NodeA and PoolB on NodeB.
   CLI@NodeA> haservice move <service> NodeB
   Example:
   CLI@NodeB> haservice move PoolB NodeB

23. Now verify from both the nodes that the HA service is up and running from their original configuration. Run the following commands from both the nodes.
   CLI@NodeA> haservice status
   CLI@NodeB> haservice status
   CLI@NodeA> haservice list
   Name       GUID                     VIPs       NODES     RUNNING  STOP
   PoolA  14089758145006079395   Avip    NodeA,NodeB  NodeA    NodeB
   CLI@NodeB> haservice list
   Name       GUID                     VIPs       NODES     RUNNING  STOP
   PoolB  14089758145006079395   Avip    NodeA,NodeB  NodeB    NodeA
Rollback Clustered Nodes with Multi-Tenants

Follow the steps outlined here, with the assistance of Nexenta support, if you have booted your clustered nodes into the latest 5.2.x system and would like to restore your 5.2 appliance.

<table>
<thead>
<tr>
<th>Note:</th>
<th>If you rollback from 5.2 to earlier versions of 5.x, all the existing tenants will be lost.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>List the tenants running on NodeB.</td>
</tr>
<tr>
<td>2.</td>
<td>Failover the tenants from NodeB to NodeA.</td>
</tr>
<tr>
<td>3.</td>
<td>All services (PoolA and PoolB) will be running on NodeA.</td>
</tr>
<tr>
<td>4.</td>
<td>Rollback NodeB to previous BE (software activate).</td>
</tr>
<tr>
<td>5.</td>
<td>Reboot NodeB.</td>
</tr>
<tr>
<td></td>
<td>Now the second host node (NodeB) is running the old appliance version 5.2.</td>
</tr>
<tr>
<td>6.</td>
<td>Failover tenants to NodeB.</td>
</tr>
<tr>
<td>7.</td>
<td>Run tenant rollback command on each tenant from the host (tenant rollback).</td>
</tr>
<tr>
<td></td>
<td>This finds the correct tenant BE for this Host BE, mounts the tenant root and attaches the tenant.</td>
</tr>
<tr>
<td>8.</td>
<td>Boot the tenants.</td>
</tr>
<tr>
<td>9.</td>
<td>Rollback NodeA to previous BE (software activate).</td>
</tr>
<tr>
<td>10.</td>
<td>Reboot NodeA.</td>
</tr>
<tr>
<td>11.</td>
<td>Return the tenants to their original Host Nodes.</td>
</tr>
<tr>
<td>12.</td>
<td>Check the local tenant repo.</td>
</tr>
<tr>
<td>13.</td>
<td>If there is any discrepancy in the software version between the Master tenant/local repo and the</td>
</tr>
<tr>
<td></td>
<td>host, re initialize the master tenant image/local repo.</td>
</tr>
</tbody>
</table>

Rollback Steps with Examples

This example shows:

- two nodes (NodeA and NodeB),
- two pools (PoolA and PoolB) with PoolA on NodeA and PoolB on NodeB,
- a HA service (PoolA) running on NodeA and a HA service (PoolB) running on NodeB,
- one tenant (tenantA) running on NodeA and on HA pool 'PoolA'.

- **Steps to rollback NodeA and NodeB:**
  1. List the tenants running on NodeB.
     
     CLI@NodeB> tenant list
  2. Verify the existing pools on both nodes.
CLI@NodeA> pool list
CLI@NodeB> pool list

3. Failover all tenant(s) from NodeB to NodeA.
   CLI@NodeB> haservice failover NodeB NodeA

4. From NodeA or NodeB, verify that the HA service PoolB from NodeB failed over to NodeA.
   CLI@NodeA> haservice status

5. From NodeA, verify that the tenantB from NodeB imported to NodeA.
   CLI@NodeA> tenant list

   CLI@NodeB> software activate <version>

7. Reboot NodeB once the rollback completes
   CLI@NodeB> reboot
   Now the second host node (NodeB) is running the old appliance version 5.2.

8. Confirm that the rolled back version is activated and verify your boot environment list.
   CLI@NodeB> software list

9. Validate the software version.
   CLI@NodeB> software version

10. Failover all tenant(s) from NodeA to NodeB.
    CLI@NodeA> haservice failover NodeA NodeB
    Now you will notice that all tenants in configured state. In order to get the tenants to “Running”
        state you must execute the command listed in the next step to attach the tenants.

11. Run tenant rollback command on each tenant from the host.
    This finds the correct tenant BE for this Host BE, mounts the tenant root and attaches the tenant.
    CLI@NodeB> tenant rollback <tenant>

12. Boot all the tenants from the host.
    CLI@NodeB> tenant start all

13. Verify the local tenant repo.
    CLI@host> tenant init -n
    Would initialize tenant subsystem
    Current status of tenant subsystem:
    Tenant repository: 5.2.0.23
    Master tenant image: Initialized

Note:
- Tenants created after software upgrade will not be available to run after rollback
  to a previous checkpoint. Destroy the tenants that will not be booted back in the
  environment they were created.
14. If there is any discrepancy between the between the local repo and the host, clean the Local repo using the tenant init --repo command to update the local tenant repository on the rpool.

   CLI@host> tenant init --repo
   This re initializes the local tenant repository with the currently running software version on the host node.

15. Rollback NodeA to previous BE.

   CLI@NodeA> software activate <version>

16. Reboot NodeA.

   CLI@NodeA> reboot
   Now the second host node (NodeB) is also running the old appliance version 5.2.

17. Return the tenants to their original Host Nodes. From NodeB move PoolA back to NodeA so the cluster is back in its original configuration with PoolA on NodeA and PoolB on NodeB.

   CLI@NodeB> haservice move PoolA NodeA

18. Now verify from both the nodes that the HA service is up and running from their original configuration. Run the following commands from both nodes.

   CLI@NodeA> haservice status
   CLI@NodeB> haservice status
   CLI@NodeA> haservice list
   CLI@NodeB> haservice list
Backup and Restore of Boot Environment

This chapter covers the following topics:

- Steps to Backup on NodeA
- Steps to Restore BE on NodeA in the Case of Loss of All Boot Devices
- Steps to Restore on NodeA in the case of Total Loss of Server Hardware

Backup and Restore of NexentaStor Boot Environment in a Multi-Tenant Clustered Environment

Use this process to rebuild a NexentaStor node in case of loss of boot device, or loss of server hardware. In order to backup a NexentaStor node, the tenants must NOT be on the node to be backed up, they should be failed over to the other node.

Backup and Restore Steps

The following example shows a typical Active / Active 2-node (NodeA, NodeB) HA cluster configuration with the steps to backup the BE of NodeA. In this example, the partner node (NodeB) is used to store the BE backup of NodeA.

The steps below show creation of a backup on NodeA and then restoring the backup after the failed hardware is replaced.

Example scenarios of a failed hardware:

- Scenario 1 - Loss of boot device in NodeA
- Scenario 2 - Loss of server hardware in NodeA

The example here shows only one cluster service (PoolA) running on NodeA. If you have more than one service on the node, when failing over all the services will fail over to the other node in the cluster.

This section covers the following steps:

- **To create SW Backup on NodeA:**
  1. Failover all tenants from NodeA to NodeB.
  2. Create a config backup of NodeA.
  3. Rebalance the cluster by moving tenants back to NodeA.

- **To restore the backup on NodeA:**
  1. Assuming all tenants are running on NodeB, install NexentaStor on the fixed/replaced NodeA.
2. Configure the networking.
3. Restore the backup on NodeA.
4. Boot to restored checkpoint.
5. Create new checkpoint based on restored checkpoint.
   While this step may seem redundant, it is required for multi-tenant configurations and ensures that all lower level configurations are applied to the boot environment to allow it to run tenants currently on the other node in the cluster.
6. Boot to new checkpoint.
7. Since a new server will have a different system GUID, you will need an updated NexentaStor license. Work with Nexenta support services to get an updated license matching the new server GUID.
8. Once you are booted in the restored boot environment, you will need to apply the new license with the updated server GUID. Activate the new license on both nodes of a cluster because the license file for an HA cluster references the GUIDs of both nodes. You do not need to activate the license in case of loss of scenarios such as fixing the boot device where the GUID of the appliance does not change.
9. Once you are done restoring the backup on the newly installed appliance, verify the cluster is back on by running `hacluster status` command.
10. If the cluster has not rejoined, re-join the cluster by restarting the HA cluster service on the new node.
11. Return the tenants belonging to NodeA from NodeB to NodeA.
12. Destroy the now obsolete restored checkpoint.

Steps to Backup on NodeA

This section describes appliance’s functionality to back up and restore a NexentaStor appliance boot environment using software backup and software restore commands. A software backup creates a single backup file for a given NexentaStor boot environment and stores this backup file in the location specified in the `system.backupDirectory` configuration variable.

Software restore restores the configuration from the backup directory and makes it available as boot environment. This restored boot environment can then be activated to automatically re-apply IP / user / AD / Mappings / etc type information that was stored in the backup image.

1. List the tenants running on NodeA.
   ```
   CLINodeA> tenant list
   NAME      STATE    HOST    VERSION   CPU    MEM    QUOTA   USED     AVAIL   ROOT
   tenantA  running  NodeA  5.2.0.15  0.21%  0.23%  5.0 GB  24.6 KB  5.0 GB   /PoolA
   ```
2. Failover all tenant(s) from NodeA to NodeB.
The example here shows only one cluster service (PoolA) running on NodeA. If you have more than one service on the node, when failing over all the services will fail over to the other node in the cluster.

CLI@NodeA> haservice failover <from-node> <to-node>
Example:
CLI@NodeA> haservice failover NodeA NodeB

3. List existing boot environments on NodeA and locate boot environment to backup.

CLI@NodeA> software list

<table>
<thead>
<tr>
<th>NAME</th>
<th>SPACE</th>
<th>ACTIVENOW</th>
<th>ACTIVEAFTERREBOOT</th>
<th>CREATIONTIME</th>
</tr>
</thead>
<tbody>
<tr>
<td>NexentaStor-5.2.1</td>
<td>506.3M</td>
<td>no</td>
<td>no</td>
<td>Apr 18 12:58:19 2017</td>
</tr>
<tr>
<td>kstat_test</td>
<td>6.30G</td>
<td>no</td>
<td>no</td>
<td>May 22 14:04:53 2017</td>
</tr>
<tr>
<td>kstat_test-backup</td>
<td>174K</td>
<td>no</td>
<td>no</td>
<td>May 22 22:12:20 2017</td>
</tr>
<tr>
<td>kstat_test-backup</td>
<td>1.0M</td>
<td>no</td>
<td>no</td>
<td>Jun 7 20:20:06 2017</td>
</tr>
<tr>
<td>NexentaStor-5.2.0</td>
<td>58.75G</td>
<td>yes</td>
<td>yes</td>
<td>Dec 31 11:41:06 2017</td>
</tr>
</tbody>
</table>

4. Create a filesystem locally on NodeB to store the backup.

CLI@NodeB> filesystem create nfs/backup

5. Configure the ACL on the filesystem and add an Access Control Entry (ACE) with appropriate modify permissions on the filesystem to enable access from the other node (NodeA) that will access the configured backups.

CLI@NodeB> acl set A+user:nobody:modify_set:fd:allow nfs/backup

6. Enable NFS sharing on the filesystem for the share to be accessible from NodeA.

CLI@NodeB> nfs share -o anon=nobody nfs/backup

7. Configure the backup directory on NodeA to use the NFS share that was created on NodeB.

CLI@NodeA> config set system.backupDirectory=nfs://<IP address of NodeB>/nfs/backup

8. After adding the backup remote target location in ‘system.backupDirectory’ settings, run the backup command.

CLI@NodeA> software backup [-nyv] <version> [-b <backup>]

Example:

CLI@NodeA> software backup NexentaStor-5.2.0 -b NodeAbackupcurrentversion
Backup software version 'NexentaStor-5.2.0'? [y/N] y
Backing up "NexentaStor-5.2.0" as "NodeAbackupcurrentversion" ...
Backup created successfully.

9. List the backup to verify the size of the backup.

Note: When using an explicit backup name, you should prefix it with the Node name of the BE you are backing up.
CLI@NodeA> software show-backup

<table>
<thead>
<tr>
<th>NAME</th>
<th>CREATED</th>
<th>SIZE</th>
</tr>
</thead>
</table>

10. Now move the NodeA tenants from NodeB to NodeA.

   CLI@NodeB> haservice move poolA <from-node> <to-node>

   Example:
   CLI@NodeB> haservice move poolA NodeB NodeA

Steps to Restore BE on NodeA In the Case of Loss of All Boot Devices

1. Assuming all tenants from NodeA are in NodeB, from NodeB, verify that the tenantA from NodeA imported to NodeB.

   CLI@NodeB> tenant list

<table>
<thead>
<tr>
<th>NAME</th>
<th>STATE</th>
<th>HOST</th>
<th>VERSION</th>
<th>CPU</th>
<th>MEM</th>
<th>QUOTA</th>
<th>USED</th>
</tr>
</thead>
<tbody>
<tr>
<td>tenantA</td>
<td>configured</td>
<td>NodeB</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0 Bytes</td>
<td>0 Bytes</td>
</tr>
</tbody>
</table>

2. Install new boot devices on NodeA.

3. Install NexentaStor on NodeA and configure networking and apply the license.

4. On NodeA, list existing boot environments to make sure no collisions occur.

   CLI@NodeA> software list

5. Configure the remote backup NFS share in the 'system.backupDirectory' settings.

   CLI@NodeA> config set system.backupDirectory=nfs://<IP address>/nfs-configbackupfs

6. View existing Backups.

   CLI@NodeA> software show-backup

7. Restore the backup on NodeA.

   Apply the "software restore" command for booting up the new appliance with the saved configuration that would automatically re-apply IP / user / AD / Mappings / etc type information that was stored in the backup directory.

   CLI@NodeA> software restore [-nyv] <version> -f <backup>

   Example:

   CLI@NodeA> software restore newname -f NodeAbackupcurrentversion

   Restore software version 'newname' from backup 'NodeAbackupcurrentversion

   [y/N] y

   Software version restored successfully.

8. After the new version is restored, you will be prompted with a choice to activate the new restored BE version.
9. Type y to activate the new restored BE.

10. Now reboot to the restored checkpoint.
    CLI@NodeA> reboot

11. While this step may seem redundant, it is required for multi-tenant configurations and ensures that all lower level configurations are applied to the boot environment to allow it to run tenants currently on the other node in the cluster.
    CLI@NodeA> software checkpoint newcheckpoint

12. Activate and boot to the new checkpoint.
    CLI@NodeA> software activate newcheckpoint
    CLI@NodeA> reboot

13. Once you are done restoring the backup on the newly installed appliance, verify the cluster is back on by running hacluster status.
    CLI@nexenta> hacluster status

14. If the cluster has not rejoined, re-join the cluster by restarting the HA cluster service on the new node.
    CLI@nexenta> svc restart ha

15. Move tenants from NodeB to NodeA.

16. Destroy the now obsolete restored checkpoint.
    CLI@NodeA> software delete backupcurrentversion

Steps to Restore on NodeA in the case of Total Loss of Server Hardware

1. Install a brand new server if the node is completely lost.
   Set up a server with identical Hardware, same server model, same NIC, same HBA, and so on.

2. Configure the networking.

3. Assuming all tenants from NodeA failedover to NodeB, from NodeB, verify that the tenantA from NodeA imported to NodeB.
   CLI@NodeB> tenant list
   NAME      STATE       HOST      VERSION   CPU    MEM    QUOTA    USED
   AVAIL    ROOT
   tenantA  configured  NodeB  -         -      -      0 Bytes  0 Bytes  0
   Bytes /PoolA

4. Execute Steps from 5 to 16 from the previous scenario.

5. Display the system GUID using the following command.
   CLI@NodeA> config list | grep -irn guid

Since the system GUID changes after a fresh NexentaStor installation, you must activate the license as shown in the step below.
6. Once you are booted in the restored boot environment, you will need to reapply the new license with the updated server GUID. To use the new NexentaStor appliance, activate the new license on both nodes of a cluster because the license file for an HA cluster references the GUIDs of both nodes. NexentaStor supports either online or offline license activation. The online activation process uses the activation key you received from Nexenta Systems.

See NexentaStor 5.2 Installation Guide for more details on the steps to activate online and offline.
Troubleshooting

This chapter covers the following topics and includes pointers in troubleshooting and diagnosing problems you may encounter:

- Verifying Tenant Init Completed Successfully
- Troubleshooting Tenant Initialization
- Handling Tenant Creation Failure
- Retrieving Information about a Tenant(s)
- Handling Tenant Login Failure
- Verifying if a Dataset is Delegated to a Tenant
- Monitoring Storage Performance with FSSTAT
- Troubleshooting Upgrade
- Bundle Services for Tenants

Verifying Tenant Init Completed Successfully

To find out if the initialization of the tenant completed successfully, run tenant init again to see the status of the initialization.

- To view the status of tenant initialization:
  
  CLI@host> tenant init

Troubleshooting Tenant Initialization

Use the journal log NEF command to see the log about the initialization.

  CLI@host> journal tail -f nef
  tenant stdout: Retrieving target catalog 'nsdev' ...
  tenant stdout: Done
  tenant stdout: Retrieving catalog 'nsdev' ...
  tenant stdout: Done
  [ Sep 11 14:45:03 tenant: [info] Initializing master tenant ]
  [ Sep 11 14:45:03 tenant: [info] Initializing master tenant (55%) ]
  [ Sep 11 14:45:03 tenant: [info] Creating master tenant (56%) ]
  [ Sep 11 14:45:03 tenant: [info] Installing master tenant (58%) ]
[ Sep 11 14:45:33 tenant: [info] Populating local repository (18.4 MB transferred) (61%) ]

[ Sep 11 14:49:16 rest: [error] POST /tenants/global/init failed: EBUSY: Tenant initialization already in progress: Populating local repository (18.4 MB transferred) (61%) ]

[ Sep 11 14:49:54 tenant: [info] Populating local repository (74.4 MB transferred) (62%) ]

[ Sep 11 14:49:59 tenant: [info] Populating local repository (129.9 MB transferred) (64%) ]

[ Sep 11 14:50:04 tenant: [info] Populating local repository (183.7 MB transferred) (65%) ]

[ Sep 11 14:50:09 tenant: [info] Populating local repository (234.9 MB transferred) (67%) ]

[ Sep 11 14:50:14 tenant: [info] Populating local repository (280.2 MB transferred) (68%) ]

[ Sep 11 14:50:24 tenant: [info] Populating local repository (320.1 MB transferred) (69%) ]

[ Sep 11 14:50:29 tenant: [info] Populating local repository (395.0 MB transferred) (71%) ]

[ Sep 11 14:50:34 tenant: [info] Populating local repository (467.8 MB transferred) (73%) ]

[ Sep 11 14:50:39 tenant: [info] Populating local repository (568.9 MB transferred) (76%) ]

[ Sep 11 14:50:44 tenant: [info] Populating local repository (618.2 MB transferred) (77%) ]

[ Sep 11 14:50:50 tenant: [info] Booting master tenant (95%) ]

[ Sep 11 14:50:52 tenant: [info] Performing post-install configuration of master tenant (95%) ]

[ Sep 11 14:50:54 tenant: [info] Populating local repository (627.8 MB transferred) (78%) ]

[ Sep 11 14:50:57 tenant: [info] Finalizing master tenant configuration (98%) ]

[ Sep 11 14:50:59 tenant: [info] Populating local repository (617.9 MB transferred) (77%) ]

[ Sep 11 14:50:59 tenant: [info] Complete (100%) ]

[ Sep 11 14:50:59 tenant: [info] Tenant subsystem successfully initialized ]
Handling Tenant Creation Failure

Reinitialize Master Tenant

Tenant creation might fail, if there is some inconsistency between the Master tenant and the host. To resolve this, reinitialize the master tenant using `tenant init --master` command on the host.

1. Verify the publisher list. Ensure that the publishers are pointing to the correct repo and ensure that you can reach the location specified here.

```
CLI@host> publisher list
PUBLISHER         STATUS  LOCATION
nexenta           online  https://prodpkg.nexenta.com/nstor/pkg5/
HighAvailability  online  https://prodpkg.nexenta.com/thirdparty/HAC/rsf/pkg5/
```

2. Now reinitialize the tenant.

```
CLI@host> tenant init --master
```

This command resets the master tenant image to contain the software version currently running on the host node. This also provides a fresh master tenant free of any upgrade failure inconsistencies.

Verify Master Tenant is Reinitialized Using the API

1. Enable access to the detailed REST API documentation from the tenant and the host. Login to the tenant and the hosts and run these CLI commands respectively.

```
CLI@tenant> config set rest.useSwagger=true (case-sensitive parameter)
CLI@host> config list rest.useSwagger=true
```

2. Point your browser to `https://<system.managementAddress>:8443/docs` using the management address you added to the interface for the tenant.

   `<staticIPaddress>` should be the management IP address of the host.

3. Login as the host admin and select the method “Check tenant subsystems” under “tenants”.

4. Enter the tenant name in the tenantName parameter.

5. Click the “Try it out” button.

6. Look for the master tenant parameter in the “Response Body” as shown here.
Retrieving Information about a Tenant(s)

Use `tenant list` and `tenant status` commands to view the following information:

- the number of tenants running on a host,
- the resource (CPU and Memory) utilization,
- the VNICs associated with a tenant,
- used and available storage capacity.
- status of a tenant

- **To view the number of tenants running on a host:**
  
  CLI@host> tenant list

- **To monitor the CPU, resource utilization, software version running on a tenant:**
  
  CLI@host> tenant status

- **To view the list of VNICs associated within a Tenant:**
  
  CLI@host> tenant status

- **To view the status of a tenant:**
  
  CLI@host> tenant status

See Table 2-2, Different Possible States of Tenants for the description on different possible states of tenants.

Handling Tenant Login Failure

Tenants should be in “running” state in order to log in to a tenant. You must login from the host where the tenant is running.
Ensure that the tenant status to be “running” by executing the following command:

CLI@host> tenant list

If the tenant is not running, start the tenant:

CLI@host> tenant start <tenant>

Now login to the tenant:

CLI@host> tenant login <tenant>

Verifying if a Dataset is Delegated to a Tenant

CLI@host> fs get delegated <filesystem>

Example:

CLI@host> fs get delegated tank/tenanttest/tanktsr

<table>
<thead>
<tr>
<th>PATH</th>
<th>PROPERTY</th>
<th>VALUE</th>
<th>SOURCE</th>
</tr>
</thead>
<tbody>
<tr>
<td>tank/tenanttest/tanktsr</td>
<td>delegated</td>
<td>yes</td>
<td></td>
</tr>
</tbody>
</table>

The property “delegated=yes” indicates that the filesystem is delegated to a tenant.

Monitoring Storage Performance with FSSTAT

To view the IO stats on tenant filesystem:

Log into the tenant and run:

CLI@tenant> fsstat [-a|f|i|n|v] [-T d|u] {-F | {fstype | fspath}...} [interval [count]]

Example:

CLI@tenant> fsstat -F

<table>
<thead>
<tr>
<th>new name</th>
<th>name</th>
<th>attr</th>
<th>attr lookup</th>
<th>rddir</th>
<th>read</th>
<th>read</th>
<th>write</th>
<th>write</th>
</tr>
</thead>
<tbody>
<tr>
<td>file</td>
<td>remov</td>
<td>chng</td>
<td>get</td>
<td>set</td>
<td>ops</td>
<td>ops</td>
<td>bytes</td>
<td>bytes</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3.07K</td>
<td>379</td>
<td>1.12K</td>
<td>738K</td>
<td>310</td>
<td>3.63M</td>
<td>10.0K</td>
<td>313K</td>
<td>738M</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4.91K</td>
<td>186</td>
<td>37.9K</td>
<td>76</td>
<td>14.2K</td>
<td>4</td>
<td>73.3K</td>
<td>14.3K</td>
<td>87.5M</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Troubleshooting Upgrade

Upgrade might fail when the local tenant repository on the host does not contain the latest available version of NexentaStor. When there is a conflict in SW version between the local tenant repository and the host, you must clean the local tenant repository.

Verify the Appliance Version in the Local Tenant Repository

You can verify the appliance version in the local package repository using the REST API or the CLI.

Using API

1. Enable access to the detailed REST API documentation from the tenant and the host. Login to the tenant and the hosts and run these CLI commands respectively.

   CLI@tenant> config set rest.useSwagger=true (case-sensitive parameter)
   CLI@host> config list rest.useSwagger=true

2. Point your browser to https://<system.managementAddress>:8443/docs using the management address you added to the interface for the tenant.

   <staticIPaddress> should be the management IP address of the host.

3. Login as the host admin and select the method “Check tenant subsystems” under “tenants”.

4. Enter the tenant name in the tenantName parameter.

5. Click the “Try it out” button.

6. Look for the version number in the “Response Body” as shown here.

   ![Response Body Example]

Using CLI

CLI@host> tenant init -n
Would initialize tenant subsystem
Current status of tenant subsystem:
Tenant repository: 5.2.0.23
Master tenant image: Initialized

**Clean the Local Repo**

Use the `tenant init --repo` command to update the local tenant repository on the rpool.

```
CLI@host> tenant init --repo
```

This re initializes local tenant repository with the software version currently running on the host node. Basically, the "init --repo" cleans up the local repo to contain only the current version.

**Bundle Services for Tenants**

This section describes the management of tenant support bundles. A tenant support bundle (SB) is an archive containing limited system information for Nexenta support service. The tenant bundles are stored in the tenant’s root filesystem and can be accessed from the host when uploading or accessing bundles.

All the tenant bundle management operations must be done on the host using the standard bundle command available for the management operations of the NexentaStor node itself with an additional -t argument followed by the tenant name.

**Create a Support Bundle**

You can create a tenant bundle only when the tenant is running and on the node where the tenant resides.

**For a Specific Tenant**

You can create a tenant specific bundle as a compressed archive on your system. A generated universally unique identifier (GUID) along with the host name and tenant name is displayed to reference the created bundle.

```
CLI@host> bundle create [-Fnv] [-t tenant][-c cores]
```

Example:

```
CLI@host> bundle create -t tenant1
```

System response:

```
```

Where TENANTTEST is the tenant name and SMC-53-109 is the host name. This does not include the host bundle.
Identify the Core Files

With the improvised version of Core and crash dump management in NexentaStor, you have the following advantages:

- You can include the tenant core files in the support bundle (SB), only if you need to, which means the cores are not included in the support bundle by default.
- You can also identify and select the cores associated with a specific tenant to be included in the tenant support bundle. This filtering of cores will speed up the upload of the support bundle. This also implies that the cores will not be removed from the NexentaStor appliance by default.

```
CLI@host> core list -t tenantname
FILE              CREATED                 SIZE     ALERT
vmdump.0  May 31 13:01:35  463.9M  ccaf1761-f271-4436-b159-97493f9ebb26
```

Include Core Files to SB

```
CLI@host> bundle create -c | --cores = < Comma separated list of core files to include or "all" to include all of them> [-t tenant_name]
```

Verify the Bundle

Ensure that the bundling has completed. The following command lists all the supported bundles.

```
CLI@host> bundle list -t tenanttest
BUNDLEID                                               CREATED          SIZE    ACTION  DONE  UPLOADED  STATUS
SMC-53-109:TENANTTEST_SERIAL_2018-09-19T18-43-51-472Z  Sep 20 00:13:58  278.8K  create 100%  -         ok
```

Download the Tenant Bundle Locally

In the absence of internet connection from NexentaStor node to the Support server and if you are unable to upload bundle to the Support server, you can download the support bundles locally directly from NexentaStor appliance by following these steps.

Using CLI

1. Verify the bundle download settings.

   ```
   CLI@host> config list support.downloadEnabled
   NAME                        FLAGS  VALUE
   support.downloadEnabled     --     false
   ```

2. If the download option was limited by setting to “false”, enable the bundle download option.

   ```
   CLI@host> config set support.downloadEnabled=true
   ```
3. Obtain the bundle links.

   CLI@host> bundle get download SMC-53-109:TENANTTEST_SERIAL_2018-09-19T18-43-51-472Z -O basic -t tenanttest

   NAME                                                   PROPERTY  VALUE

4. Download the bundles by opening a browser and copying the bundle link in the URL following https://<IP>:8443

   https://10.3.53.109:8443/support/bundles/<bundleId>/download

   Example:

5. Save the file locally or in a location from where it can be uploaded to the support server.

Using REST API

You can download the tenant bundle from the host API by following the steps listed here.

1. Enable access to the detailed REST API documentation from the tenant and the host. See NexentaStor 5.x REST API QuickStart for more details. Login to the tenant and run this CLI command:

   CLI@tenant> config set rest.useSwagger=true (case-sensitive parameter)

   CLI@host> config list rest.useSwagger=true

2. Point your browser to https://<system.managementAddress>:8443/docs using the management address.

   <system.managementAddress> should be the management IP address of the host.

3. Login as the host admin and select the method “support/bundles/{bundleId}/download” under “Support”.

4. Enter the tenant bundle ID in the bundleId parameter.

5. Click “Try it Out”

   Request URL
6. Download the bundles by opening a browser and copying and pasting the URL from the Request URL field.

   Save the file locally or in a location from where it can be uploaded to the support server.

**Uploading Tenant Bundle**

**Using CLI**

1. You can upload the tenant bundle by using the following command:

   CLI@host> bundle upload -t <tenant> <id>

   Example:

   CLI@host> bundle upload -t tenanttest SMC-53-109:TENANTTEST_SR-DEV-NS-201945892_2019-02-22T00-10-48-812Z

2. Verify the tenant bundle upload.

   CLI@host> bundle list -t <tenant>

   Example:

   CLI@host> bundle list -t tenanttest

<table>
<thead>
<tr>
<th>BUNDLEID</th>
<th>CREATED</th>
<th>SIZE</th>
<th>ACTION</th>
<th>DONE</th>
<th>UPLOADED</th>
<th>STATUS</th>
</tr>
</thead>
</table>

**Using REST API**

You can upload the tenant bundle from the host API by following the steps listed here.

1. Enable access to the detailed REST API documentation from the tenant and the host. See NexentaStor 5.x REST API QuickStart for more details. Login to the tenant and run this CLI command:

   CLI@tenant> config set rest.useSwagger=true (case-sensitive parameter)

   CLI@host> config list rest.useSwagger=true

2. Point your browser to https://<system.managementAddress>:8443/docs using the management address.

   <system.managementAddress> should be the management IP address of the host.

3. Login as the host admin and select the method “Upload support bundle to a remote FTP server” under Support.
4. Enter the tenant bundle ID in the bundleId parameter.

5. Click Try It Out.
   This returns a 202 code that implies the call is in progress.

6. Click on the 202 button to get a response. The call responds when the request is complete. Or you can check the job status by copying and pasting the job ID in the jobId parameter of the "jobstatus" method and by reading the "done" parameter in the "Response Body".

7. You can also check the Upload status by running the following command from the CLI.
   
   CLI@host> bundle list -t tenanttest
Default Support Bundle Protocol and Settings

By default, support bundles are uploaded via S3 protocol to Nexenta servers. For information on modifying the default settings and uploading support bundles to Nexenta’s https/ftp server, see NexentaStor 5.x CLI Configuration Guide.

Other Bundle Management Commands

The following table contains the other management commands available for the tenant support bundles.

<table>
<thead>
<tr>
<th>Task</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ensure that the bundling has completed.</td>
<td><code>CLI@host&gt; bundle list [-Fv] [-o &lt;properties&gt;] [-t tenant_name]</code>&lt;br&gt;This command lists all the supported bundles for a specific tenant.&lt;br&gt;&lt;br&gt;<code>CLI@host&gt; bundle list [-Fv] [-o &lt;properties&gt;] -t all</code>&lt;br&gt;This command lists all the bundles created for all the tenants on the host.</td>
</tr>
</tbody>
</table>
| Display the properties of the specified support bundle. | `CLI@host> bundle get [-Fv] (all | <properties>) [-t tenant_name] [-s <field>]... [-S <field>]... [-O <flags>] [<id>...]`<br><br>`CLI@host> bundle get [-Fv] (all | <properties>) [-t tenant_name] [-s <field>]... [-S <field>]... [-O <flags>] [<id>...]`<br><br>`CLI@host> bundle get [-Fv] <id>`<br>`CLI@host> bundle get [-Fv] [-t tenant_name] <id>`<br>`CLI@host> bundle get [-Fv] [-t tenant_name] <id>`<br>`CLI@host> bundle get [-Fv] [-t tenant_name] <id>`
| To upload and to cancel the upload of a bundle. | `CLI@host> bundle upload [-Fnv] [-t tenant_name] <id>`<br>`CLI@host> bundle cancel [-nv] [-t tenant_name] <id>`<br>`CLI@host> bundle cancel [-nv] [-t tenant_name] <id>`<br>`CLI@host> bundle cancel [-nv] [-t tenant_name] <id>`<br>`CLI@host> bundle cancel [-nv] [-t tenant_name] <id>`<br>`CLI@host> bundle cancel [-nv] [-t tenant_name] <id>`