



DDN Infinia

Installation and Administration Guide

Version 1.1 | 96-00589-001 | Revision A0

Information in this document is subject to change without notice and does not represent a commitment on the part of DataDirect Networks, Inc. No part of this manual may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying and recording, for any purpose other than the purchaser's personal use without the written permission of DataDirect Networks, Inc.

© 2024 DataDirect Networks, Inc. All rights reserved.

DataDirect Networks, the DataDirect Networks logo, DDN, AI200, AI200X, AI400, AI400X, AI400X2, AI400X2T, AI7990, AI7990X, A3I, DataFlow, DirectMon, Enterprise Fusion Architecture, EFA, ES7K, ES12K, ES14KX, ES18K, ES18KX, ES200NV, ES200NVX, ES200NVX2, ES400NV, ES400NVX, ES400NVX2, ES400NVX2T, ES400X2, ES400X2T, ES7990, ES7990X, EXAScaler, GRIDScaler, GS7K, GS12K, GS14KX, GS18K, GS18KX, GS200NV, GS200NVX, GS400NV, GS400NVX, GS400NVX2, GS400NVX2T, GS400X2, GS400X2T, GS7990, GS7990X, IME, IME140, IME14K, IME240, Infinite Memory Engine, Information in Motion, In-Storage Processing, MEDIAScaler, NAS Scaler, NoFS, ObjectAssure, ReACT, SFA, SFA 10000 Storage Fusion Architecture, SFA10K, SFA12K, SFA12KX, SFA14K, SFA14KX, SFA18K, SFA18KX, SFA200NV, SFA200NVX, SFA200NVX2, SFA200NVX2E, SFA400NV, SFA400NVX2, SFA400NVX2T, SFA400NVX2E, SFA400NVX2TE, SFA400X2, SFA400X2TE, SFA7700, SFA7700X, SFA7990, SFA7990X, SFX, Storage Fusion Architecture, Storage Fusion Fabric, Storage Fusion Xcelerator, SwiftCluster, WOS, and the WOS logo are registered trademarks or trademarks of DataDirect Networks, Inc. All other brand and product names are trademarks of their respective holders.

DataDirect Networks makes no warranties, express or implied, including without limitation the implied warranties of merchantability and fitness for a particular purpose of any products or software. DataDirect Networks does not warrant, guarantee or make any representations regarding the use or the results of the use of any products or software in terms of correctness, accuracy, reliability, or otherwise. The entire risk as to the results and performance of the product and software are assumed by you. The exclusion of implied warranties is not permitted by some jurisdictions; this exclusion may not apply to you.

In no event will DataDirect Networks, their directors, officers, employees, or agents (collectively DataDirect Networks) be liable to you for any consequential, incidental, or indirect damages, including damages for loss of business profits, business interruption, loss of business information, and the like, arising out of the use or inability to use any DataDirect product or software even if DataDirect Networks has been advised of the possibility of such damages by you. Because some jurisdictions do not allow the exclusion or limitation of liability for consequential or incidental damages, these limitations may not apply to you. DataDirect Networks liability to you for actual damages from any cause whatsoever, and regardless of the form of the action (whether in contract, tort including negligence, product liability or otherwise), is limited to the sum you paid for the DataDirect product or software.

All Products cited in this document are subject to the DDN Limited Warranty Statement and, where applicable, the terms of the DDN End User Software License Agreement (EULA). The cited documents are available at: ddn.com/company/resource-library/. For archived versions of either document, please contact DataDirect Networks.

DataDirect Networks responsibly processes personal information ensuring compliance with the applicable data protection laws. For more details, see ddn.com/privacy-policy/.

December 2024

Preface

Audience

This guide is intended for field engineers and system administrators deploying, managing, and maintaining the DDN Infinia software. The level of content presented assumes the reader has expert knowledge of hardware, network configuration, and system administration in a Linux environment.

About this Guide

This guide introduces DDN Infinia including coverage of the following:

- [Overview](#) (Section 1 on page 9)
- [Installation and Setup](#) (Section 2 on page 25)
- [Cluster Management](#) (Section 3 on page 76)
- [User Management and Authorization](#) (Section 4 on page 107)
- [Data Service Management](#) (Section 5 on page 124)
- [Hardware Management](#) (Section 6 on page 137)
- [Cluster Administration](#) (Section 7 on page 148)
- [Monitoring and Failure Handling](#) (Section 8 on page 159)
- [Realm Configuration Input Parameters](#) (Appendix A on page 180)
- [DDN Infinia S3 DataService API Endpoints](#) (Appendix B on page 182)
- [Using the DDN Infinia REST API](#) (Appendix C on page 248)
- [nettest Utility](#) (Appendix D on page 257)
- [Using the DDN Infinia User Interface](#) (Appendix E on page 266)
- [Terminology](#) (Appendix F on page 286)
- [redsupport](#) (Appendix G on page 290)

Document Conventions

In general, the following definitions are used throughout this document:

| Convention | Definition | Example |
|------------|---|--|
| argument | A command consists of an array of arguments. | <code>redcli config update {config_name}</code> |
| flag | A flag is a type of argument used to enable or disable a specific feature or behavior of a command. Typically, flags are boolean. | <code>redcli cluster show -s</code> |
| option | An option is a type of argument modifying the behavior of a command. Typically, options require a value specified in its parameter. | <code>redcli user login realm_admin -p <password></code> |
| parameter | A parameter is an argument that provides information to an option. | <code>redcli user login realm_admin -p <password></code> |

The following conventions are used for command usage throughout this document.

| Convention | Description | Example |
|------------|--|--|
| { } | Curly-brackets around arguments, options, or flags denote that they are required. | <code>redcli config update {config_name}</code> |
| [] | Square-brackets around an argument, an option, or a flag denote that it is optional. | <code>redcli cluster create [cluster_name]</code> |
| < > | Angle brackets are used for the parameter of an option or to indicate placeholder content. | <code>redcli user login realm_admin -p <password></code> <code>redcli config init <your_new_cfg></code> |
| | Arguments separated by a pipe character are mutually exclusive. | <code>redcli events {admin cluster hardware system}</code> |
| [flags] | The term “flags” in square-brackets denote optional flags and options are available. | <code>redcli user info [flags]</code> |

Related Documentation

The following documents are additional sources of information for DDN Infinia:

- DDN Infinia Product Release Notes
- DDN Infinia CLI Reference
- DDN Infinia REST API Reference
- AI2000 Hardware Installation and Maintenance Guide

The latest version of the documentation is available on the DDN Customer Support Portal (support.ddn.io).

Table of Contents

| | | |
|---------|--|----|
| 1. | Overview | 9 |
| 1.1 | Features and Capabilities | 10 |
| 1.2 | Deployments..... | 11 |
| 1.3 | Instance | 12 |
| 1.4 | Realm and Cluster | 13 |
| 1.5 | Tenants, Subtenants, and Data Services | 14 |
| 1.6 | Core Affine Storage Targets..... | 16 |
| 1.7 | User Scope and Capability..... | 17 |
| 1.8 | Identity Providers | 20 |
| 1.9 | Quotas | 21 |
| 1.10 | Command Line Interface | 23 |
| 1.11 | REST API..... | 24 |
| 2. | Installation and Setup | 25 |
| 2.1 | Deploy DDN Infinia | 26 |
| 2.1.1 | Plan Your Configuration | 27 |
| 2.1.1.1 | Internet Access | 27 |
| 2.1.1.2 | Containers..... | 27 |
| 2.1.1.3 | Additional Software Utilities Required..... | 27 |
| 2.1.1.4 | Hardware and Networking Requirements..... | 27 |
| 2.1.1.5 | redsetup Options | 28 |
| 2.1.2 | Complete Configuration Prerequisites | 31 |
| 2.1.2.1 | Set a Static BMC IP Address | 31 |
| 2.1.2.2 | Setup Host Name | 32 |
| 2.1.2.3 | Specify IPv4 Address/Subnet for Management Network..... | 32 |
| 2.1.2.4 | Specify IPv4 Address/Subnet for HSN Active Link #1 and #2..... | 33 |
| 2.1.2.5 | Setup Time Zone and NTP | 34 |
| 2.1.2.6 | Enable RoCE on the High-Speed Network Switch (optional) | 35 |
| 2.1.3 | Run Setup..... | 36 |
| 2.2 | Deploy the Cluster..... | 38 |
| 2.2.1 | Initial Cluster Setup by using CLI | 40 |
| 2.2.1.1 | Generate and Deploy Realm Configuration | 40 |
| 2.2.1.2 | Install License and Accept EULA | 42 |
| 2.2.1.3 | Create Cluster | 45 |
| 2.2.1.4 | Install CA Certificates | 46 |
| 2.2.1.5 | Log In and Log Out using CLI | 48 |
| 2.2.2 | Initial Cluster Setup by using UI | 49 |
| 2.2.2.1 | Login using UI..... | 49 |
| 2.2.2.2 | Run Initial Setup Wizard | 49 |
| 2.2.2.3 | Manage and Monitor DDN Infinia from the Dashboard..... | 54 |
| 2.3 | Allocate Cluster Resources | 58 |
| 2.4 | Use DDN Infinia S3 DataService | 59 |
| 2.4.1 | Define Virtual Hosts on Nodes..... | 60 |
| 2.4.2 | Setup Access to S3 Service..... | 61 |

| | | |
|---------|---|-----|
| 2.4.3 | Read and Write Object Data with S3 Client Utility | 65 |
| 2.5 | Use DDN Infinia Block DataService | 68 |
| 2.6 | Setup DDN Infinia Client. | 69 |
| 2.7 | Verify Deployment. | 70 |
| 2.7.1 | Verify redsetup Daemon | 71 |
| 2.7.2 | Verify redcli Package Installation on Client Nodes | 71 |
| 2.7.3 | Verify Realm Status | 71 |
| 2.7.4 | Verify the DDN Infinia Cluster State | 71 |
| 2.7.5 | Verify Network Connectivity and Measure Performance. | 72 |
| 2.8 | Upgrade DDN Infinia | 74 |
| 3. | Cluster Management | 76 |
| 3.1 | Propose New Runtime Configuration | 78 |
| 3.2 | Manage Nodes | 81 |
| 3.2.1 | Add One or More Nodes to a DDN Infinia Cluster | 82 |
| 3.2.2 | Delete Nodes from a DDN Infinia Cluster | 83 |
| 3.2.3 | Add or Delete a Device in a Node | 84 |
| 3.3 | Manage Core Affine Storage Targets | 85 |
| 3.3.1 | Add One or More CATs to DDN Infinia | 86 |
| 3.3.2 | Delete One or More CATs from DDN Infinia. | 87 |
| 3.4 | Manage Networks | 88 |
| 3.5 | Manage Volumes | 89 |
| 3.5.1 | Create Volume. | 90 |
| 3.5.2 | List Volumes. | 90 |
| 3.5.3 | Show Details of a Volume | 91 |
| 3.5.4 | Resize Volumes | 92 |
| 3.6 | Manage Tenant Structure | 93 |
| 3.6.1 | Create a Tenant. | 94 |
| 3.6.2 | Delete a Tenant. | 95 |
| 3.6.3 | Update Tenant Properties | 95 |
| 3.6.4 | List Tenants | 96 |
| 3.6.5 | Show Details of a Tenant | 96 |
| 3.7 | Manage Subtenant Structure | 97 |
| 3.7.1 | Create Subtenant | 98 |
| 3.7.2 | Delete Subtenant | 99 |
| 3.7.3 | Update Subtenant Properties | 99 |
| 3.7.4 | List Subtenants | 100 |
| 3.7.5 | Show Details of a Subtenant | 100 |
| 3.8 | Manage Datasets | 101 |
| 3.8.1 | Create Dataset. | 102 |
| 3.8.2 | Update Dataset Properties. | 102 |
| 3.8.3 | List Datasets | 103 |
| 3.8.4 | Show Details of a Dataset | 104 |
| 3.8.5 | Manage Dataset Profiles | 105 |
| 3.8.5.1 | List Dataset Profiles | 105 |
| 3.8.5.2 | Show the Dataset Profile Details | 105 |

| | | |
|-------|--|-----|
| 4. | User Management and Authorization..... | 107 |
| 4.1 | Manage Realm Users and Groups | 108 |
| 4.1.1 | Add Realm Users..... | 109 |
| 4.1.2 | Add Realm User Groups | 109 |
| 4.1.3 | Update Realm User and Group Capabilities..... | 109 |
| 4.2 | Manage Tenant or Subtenant Users and Groups | 110 |
| 4.2.1 | Add Tenant or Subtenant Users | 111 |
| 4.2.2 | Add Tenant or Subtenant User Groups | 111 |
| 4.2.3 | Update Tenant or Subtenant User and Group Capabilities | 111 |
| 4.2.4 | Define Equivalence Rule | 112 |
| 4.3 | Manage Multi-tenant Users and Groups..... | 113 |
| 4.3.1 | Add Multi-tenant Users | 114 |
| 4.4 | Manage Realm Identity Providers | 115 |
| 4.4.1 | Create External Identity Provider for Realm..... | 116 |
| 4.4.2 | Create Group-based Identity Provider for Realm | 116 |
| 4.5 | Manage Tenant Identity Providers..... | 117 |
| 4.5.1 | Create External Identity Provider for Tenant..... | 118 |
| 4.5.2 | Create Group-based Identity Provider for Tenant | 118 |
| 4.6 | Use Client Certificate for Authentication | 119 |
| 4.6.1 | Generate and Use Client Certificate..... | 120 |
| 4.6.2 | Revoke Client Certificate | 122 |
| 4.7 | Password Management Using Local Authentication..... | 123 |
| 5. | Data Service Management | 124 |
| 5.1 | Manage S3 Access Records | 125 |
| 5.1.1 | Add S3 Access Record | 126 |
| 5.1.2 | Remove S3 Access Record | 126 |
| 5.1.3 | Update S3 Access Record | 127 |
| 5.1.4 | List S3 Access Record..... | 127 |
| 5.2 | Manage Data Services..... | 128 |
| 5.2.1 | Create a Block Data Service | 129 |
| 5.2.2 | Create an Object Data Service | 129 |
| 5.2.3 | Delete a Data Service..... | 130 |
| 5.2.4 | Update a Data Service..... | 130 |
| 5.2.5 | List Data Services | 130 |
| 5.2.6 | Show Data Service Status..... | 132 |
| 5.3 | Manage Buckets..... | 133 |
| 5.3.1 | Create Buckets | 134 |
| 5.3.2 | List Buckets | 134 |
| 6. | Hardware Management..... | 137 |
| 6.1 | Upgrade Server, Drive, and NIC Firmware | 138 |
| 6.1.1 | Upgrade BMC Firmware | 139 |
| 6.1.2 | Upgrade BIOS Firmware | 141 |
| 6.1.3 | Upgrade Drive Firmware..... | 143 |
| 6.1.4 | Upgrade Network Interface Controller Firmware | 143 |
| 6.2 | Manage BIOS Configuration | 146 |

| | | |
|------------|--|-----|
| 6.2.1 | Get the BIOS Configuration | 146 |
| 6.2.2 | Update the BIOS Configuration..... | 147 |
| 7. | Cluster Administration..... | 148 |
| 7.1 | List All Clusters | 149 |
| 7.2 | Show Details of a Cluster | 149 |
| 7.3 | List Current Network Configurations | 150 |
| 7.4 | Start a Cluster..... | 151 |
| 7.5 | Stop a Cluster | 151 |
| 7.6 | Show Status and Health of a Cluster..... | 152 |
| 7.6.1 | Show Status of a Cluster..... | 152 |
| 7.6.2 | Show Health of a Cluster..... | 154 |
| 7.7 | Show Inventory Details..... | 157 |
| 8. | Monitoring and Failure Handling..... | 159 |
| 8.1 | Verify the Operational State of Your Cluster | 160 |
| 8.2 | Monitor Realm, Realm Agent, and Node Status | 161 |
| 8.2.1 | Monitor Realm Status | 162 |
| 8.2.2 | Monitor Realm Agents Status | 163 |
| 8.2.3 | Monitor Node Status | 164 |
| 8.3 | Check Installed Versions | 166 |
| 8.4 | Watch for DDN Infinia Events | 168 |
| 8.4.1 | Admin Events..... | 169 |
| 8.4.2 | Cluster Events | 169 |
| 8.4.3 | Hardware Events | 170 |
| 8.4.4 | System Events..... | 171 |
| 8.5 | Check Configuration Endpoint Health Status..... | 172 |
| 8.6 | Collect and View DDN Infinia Logs..... | 173 |
| 8.6.1 | Upload Cluster Logs..... | 173 |
| 8.6.2 | Manage Logs on Unhealthy Cluster | 175 |
| 8.7 | Debug Failures Using Tracing | 176 |
| 8.8 | Query CAT Health..... | 178 |
| Appendix A | Realm Configuration Input Parameters | 180 |
| Appendix B | DDN Infinia S3 DataService API Endpoints | 182 |
| Appendix C | Using the DDN Infinia REST API..... | 248 |
| Appendix D | nettest Utility..... | 257 |
| Appendix E | Using the DDN Infinia User Interface | 266 |
| Appendix F | Terminology | 286 |
| Appendix G | redsupport | 290 |
| Appendix H | redsetup | 292 |
| Appendix I | Manage BIOS Configuration using SUM..... | 295 |

1. Overview

DDN Infinia is an easy-to-deploy, easy-to-manage cluster based Modern Data Store with the ability to store, share, and manage data via Multi-tenant, Software-defined Storage.

The following sections provide an overview of the software:

- [Features and Capabilities](#) (Section [1.1](#) on page [10](#))
- [Deployments](#) (Section [1.2](#) on page [11](#))
- [Instance](#) (Section [1.3](#) on page [12](#))
- [Realm and Cluster](#) (Section [1.4](#) on page [13](#))
- [Tenants, Subtenants, and Data Services](#) (Section [1.5](#) on page [14](#))
- [Core Affine Storage Targets](#) (Section [1.6](#) on page [16](#))
- [User Scope and Capability](#) (Section [1.7](#) on page [17](#))
- [Identity Providers](#) (Section [1.8](#) on page [20](#))
- [Command Line Interface](#) (Section [1.10](#) on page [23](#))
- [REST API](#) (Section [1.11](#) on page [24](#))

1.1 Features and Capabilities

DDN Infinia 1.1 supports elastic data services for object and block as follows:

- Objects
 - ❖ S3 compatible Storage Interface
 - ❖ No restrictions on Object size
 - ❖ No restrictions on Object Metadata size
- Block
 - ❖ Provided via NVMe over Fabrics (NVMe-oF, NVMeoF, NVMF)
 - ❖ Created via Container Storage Interface (CSI)

Its capabilities include the following:

- Cloud-like Tenancy Support (Tenants, Subtenants, and Data Services) with Service-level Agreements (SLA)
- Role-based Access Control (RBAC) Management using REST API, CLI, and GUI
- Comprehensive Remote Access Service (RAS) and Call Home Capabilities
- Remote Support Capability
- Erasure Encoding
- Compression, Encryption, and Thin Provisioning (whole Tenant in DDN Infinia 1.1)
- QLC (or other consumer drive) Flash Support
- Elastic Scalability for Expansion Without Downtime

Compression is a cluster-wide tunable, supporting LZ4, ZSTD, and IGZIP (on intel platforms) compressors.

DDN Infinia supports two compression modes:

- Fast compression – Default compression algorithm is lz4.
- High compression – Default compression algorithm is zstd.

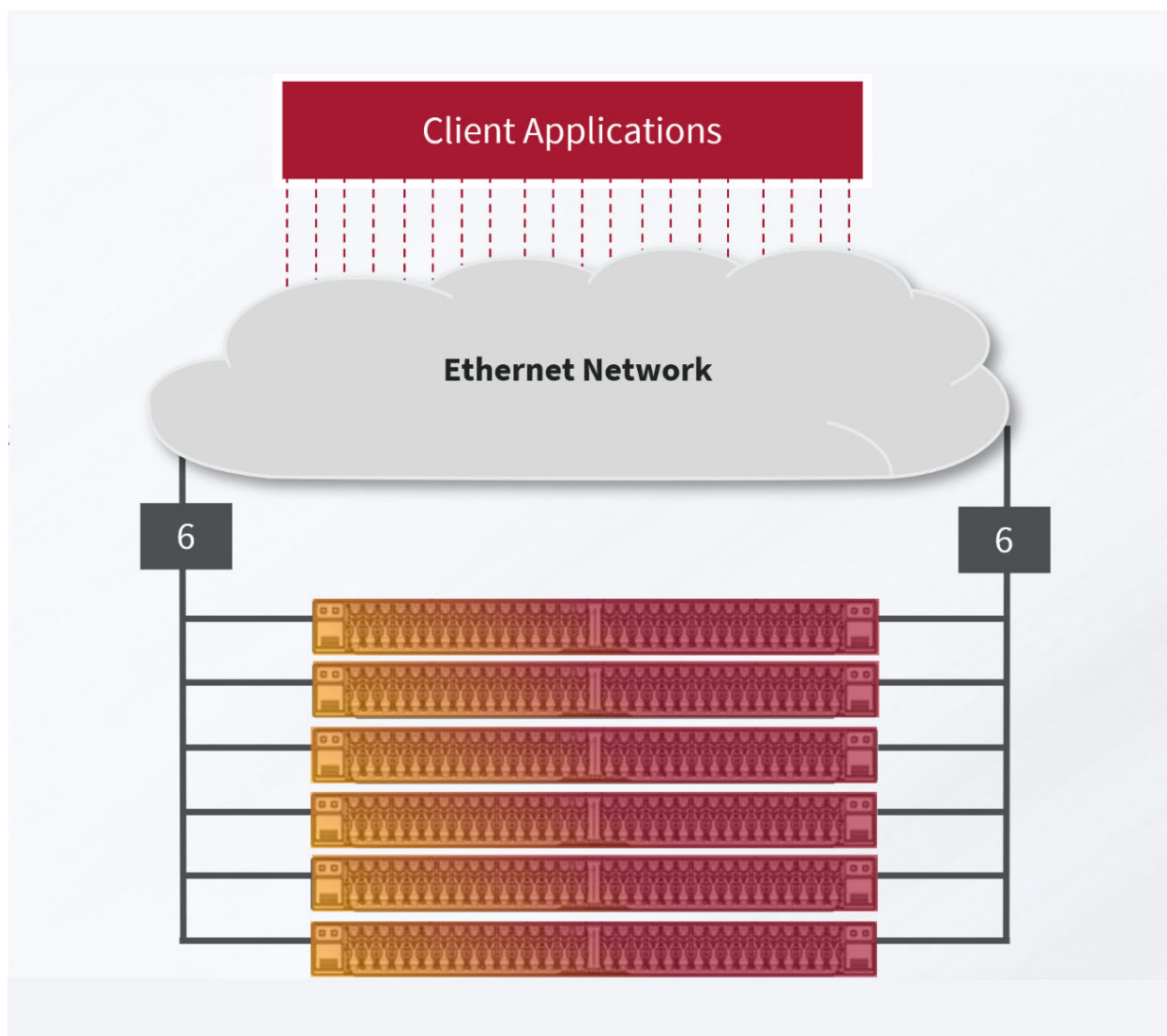
By default, fast compression is enabled. If you need to change the compression mode, contact DDN Support (via [Support Portal](#)).

DDN Infinia is a software-based solution that applies to many applications. However, it is especially suited for Enterprise computing where ease of installation, ease of management, and ease of cluster growth (space and performance) are paramount.

1.2 Deployments

DDN Infinia supports on-premise and cloud deployments through an easy-to-use Command Line Interface (DDN Infinia CLI) or Graphical User Interface (DDN Infinia GUI). Deployment is fast, taking less than 10 minutes after the hardware is provisioned. Rolling upgrades are also supported without downtime. [Figure 1](#) shows a DDN Infinia deployment example.

Figure 1. DDN Infinia Six Node Example



For detailed instructions to deploy DDN Infinia, see [Installation and Setup](#) (Section 2 on page 25).

IMPORTANT: For supported hardware, see the latest *DDN Infinia Product Release Notes*.

1.3 Instance

Throughout this document, *server* is a physical term that refers to a physical machine (on-premise) or Virtual Machine (in the Cloud) running one supported OS Instance; while *node* is a logical term that refers to an OS instance running either on a single *physical server* (on-premise) or an OS instance running on a Virtual Machine in the Cloud. *Instance* is a logical term that refers to the collection of services (CATs, DataServices, infrastructure services), running on a single node.

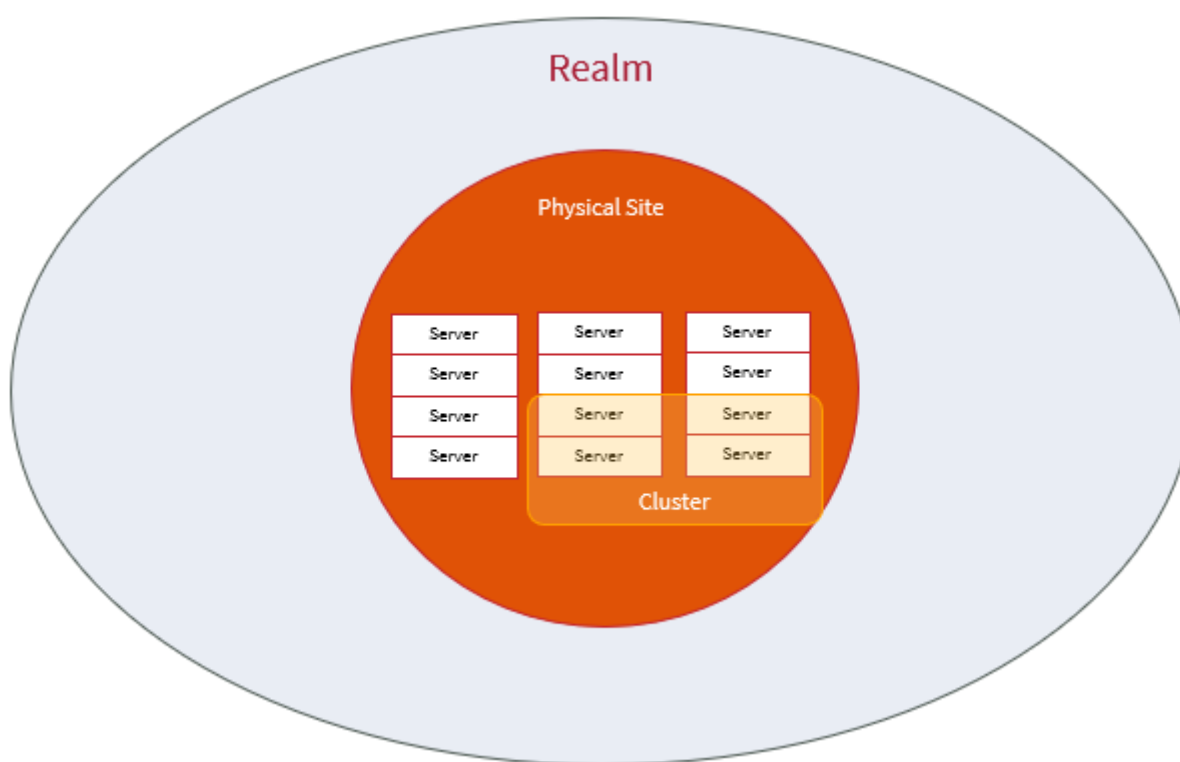
| | |
|-------------------|---|
| IMPORTANT: | In DDN Infinia 1.1, all services on a node belong to a single instance. |
| | In summary, one server = one node = one instance. |
| | DDN Infinia 1.1 also supports only one <i>cluster</i> , which can span one <i>physical site</i> . |

1.4 Realm and Cluster

A **realm** is the DDN Infinia management domain, including all physical resources that are available for use in DDN Infinia clusters and DDN Infinia data services. A realm can span one physical **site**, which usually defines geographical location (such as “New York”, “Los Angeles”, “Tokyo” etc.) but might also be used to define a finer granularity, in case of large locations. In Cloud environments, physical site can be mapped to availability zones.

Operations such as upgrade are performed at the realm level. The hardware resources of a realm are managed by a realm administrator (**realm admin**). A realm admin acquires servers, storage media, and network infrastructure then allocates them to tenants. A realm admin also monitors the hardware health to fix the issues and monitors resource usage to add more resources, if required. The **realm configuration** specifies the collection of containers (and their resource limits) that are managed by DDN Infinia. Figure 2. shows an example DDN Infinia realm.

Figure 2. DDN Infinia Realm



DDN Infinia groups instances that mutually participate to serve resources into a *cluster*. The cluster can include instances running on nodes running on servers in a physical site. The cluster is a named object created by an administrator into which storage resources are placed, grouped mostly for resiliency management of data store and other services (infrastructure and data services). DDN Infinia utilizes these resources within a cluster as a collective group. The cluster is both an administrative container, as well as the total collection of Instances and **Core Affine storage Targets (CATs)** that participate as a group to serve data.

1.5 Tenants, Subtenants, and Data Services

IMPORTANT:

In DDN Infinia 1.1, cluster capacity can be divided between tenants and subtenants. Other resources cannot be divided.

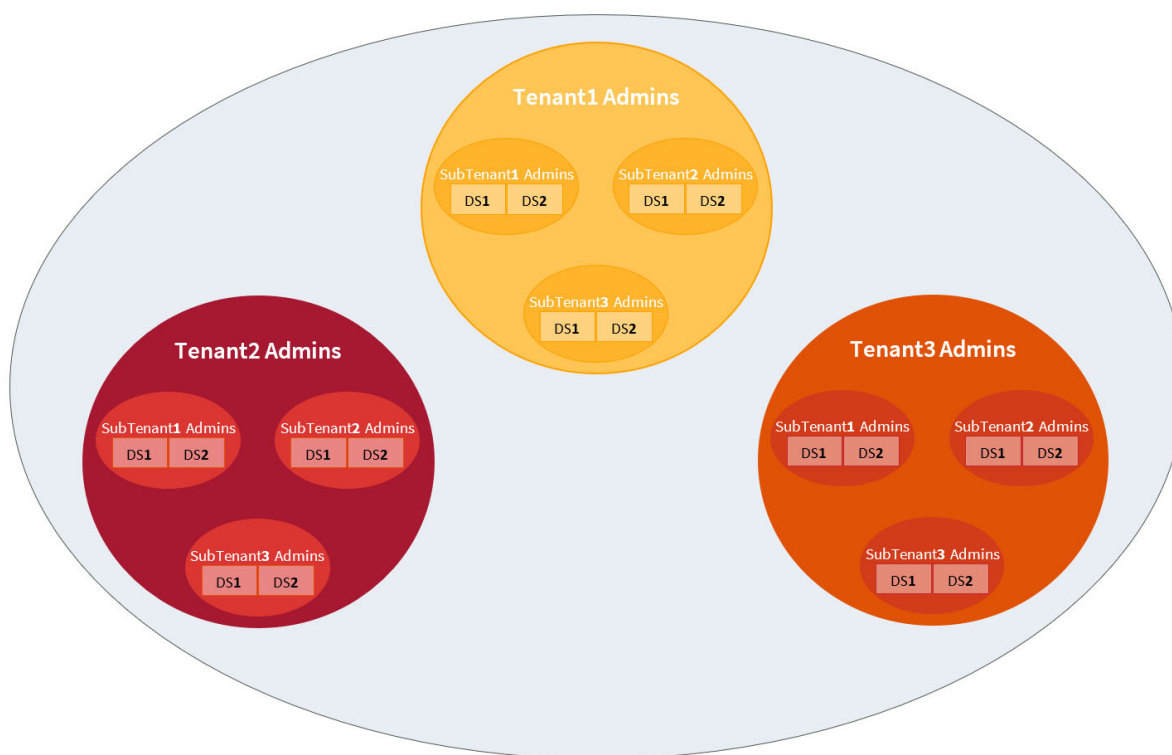
DDN Infinia supports cloud-like tenancy support (Tenants, Subtenants, and Data Services). The *tenant/subtenant* is a DDN Infinia cluster abstraction, which allows multiple organizations to share single physical DDN Infinia storage securely and efficiently. The inventory and configuration objects of a cluster define physical resources that provide storage (servers, network, solid-state drives) and tenants/subtenants/datasets (as well as volume/objects in a dataset) allocate that storage to consumers/clients.

A **tenant** is managed by a tenant administrator (**tenant admin**). A tenant admin requests resources from a realm admin, allocates them to subtenants, and monitors resource usage to request additional resources from a realm admin, if required. A **subtenant** is a subset of a tenant. It is used with customers that require more than one level of service management. A subtenant is managed by a subtenant administrator (**subtenant admin**). A subtenant admin defines data services (which implies datasets) based on resources received from a tenant admin and monitors those services for health and capacity to request more resources, if required.

If a DDN Infinia cluster is used by a cloud storage provider, it is important to split the cluster by tenants for CompanyA, CompanyB, etc. All CompanyA's storage must be kept separate and private from CompanyB's storage. CompanyA may want to allocate storage to various teams, which would be defined as subtenants. CompanyB may have a different set of subtenants. Each subtenant can carve their storage into datasets and create volumes or objects in these datasets.

Figure 3. shows an example of DDN Infinia tenants, subtenants, and data services.

Figure 3. DDN Infinia Tenants, Subtenant, and Data Services



Dataset is the unit of virtual storage management. It defines the **dataservices** that are provided, the security and resilience under which it operates, and the SLAs that determine performance and capacity. DDN Infinia supports two data service types:

- Object - contains objects exposed via S3.
For details, see [Use DDN Infinia S3 DataService](#) (Section 2.4 on page 59).
- Block - collection of Block devices provided via NVMeoF and created via [Container Storage Interface \(CSI\)](#).
For details, see [Use DDN Infinia Block DataService](#) (Section 2.5 on page 68).

For additional details about datasets and instructions to create a data service, see [Manage Data Services](#) (Section 5.2 on page 128) and [Manage Datasets](#) (Section 3.8 on page 101).

1.6 Core Affine Storage Targets

[Core Affine storage Targets \(CATs\)](#) are single/multiple storage devices, affine to single/multiple cores. Each CAT contains an Intent log (OPREC journal), copy-on-write B-epsilon tree and Bulk data. DDN Infinia analyzes the storage nodes that are available for incorporating into a cluster and aggregates the components of each server into the DDN Infinia hardware *inventory*. This process of updating the DDN Infinia inventory is called *discovery*. For additional details, see [Propose New Runtime Configuration](#) (Section 3.1 on page 78). When a CAT is discovered and incorporated into DDN Infinia, it is given a unique identifier called a DDN Infinia *uuid*. This *uuid* is a globally unique identifier assigned by the system to an object in DDN Infinia and stored with the object attributes. For example, both DDN Infinia instances and DDN Infinia CATs are uniquely identified by *uuid*.

1.7 User Scope and Capability

In DDN Infinia, users are defined by scope and capability (`scope:capability`). Scope defines the breadth of control assigned to a user or users group. The scope can be a combination of realm, one or more tenants, one or more subtenants, and one or more data services. The following are valid formats for scope:

- `realm` scope designates the realm.
- `<tenant>` scope designates a tenant.
- `<tenant/subtenant>` scope designates a subtenant.
- `<tenant/subtenant/dataservice>` scope designates a data service.

When assigning a user with a scope of tenant, subtenant, or data service, it is possible to specify `[all]` or a list of entries to define the extent of the management scope:

- `[all]` scope designates all tenants.
- `<tenant>/[all]` designates all subtenants.
- `<tenant>/<subtenant>/[all]` designates all data services.

Capability defines the permissions assigned for a particular scope. The capability can be the following:

- `admin`
- `viewer`
- `data-access`
- `service-user`

Service users are a special class of system users. These service users are not associated with a user identity but rather a data service. The purpose of the service user is to create and manage data services through the CSI interface and ensure that these services have an appropriate scope and capability. This service user is not associated with an individual user, allowing services to persist across user and identity provider changes.

Users can have multiple scopes and capabilities assigned to extend available operations. However, tenant users cannot have `realm` capability or `<other tenant>` capability. The following are the guidelines for assigning scope and capability:

- If a user is created with no `scope:capability` defined and no group assigned, there is no default. DDN Infinia will not be able to authenticate the user and the user cannot login.
- When only scope is provided, the capability defaults to `admin`.
- When only capability is provided, the scope defaults to the scope of the current admin.

When managing users in the CLI, the `-r (--scope)` option specifies the user `scope:capability`. Specify multiple scopes and capabilities in comma separated form in the `redcli` command.

Table 1 defines DDN Infinia's `scope:capability` model.

Table 1. User Scope and Capability

| scope | capability | | | |
|-------------|---|---|---|--|
| | admin | viewer (note 1) | data-access | service-user |
| realm | <ul style="list-style-type: none"> Manage realm Manage license Manage configuration Manage cluster resources Manage tenant structure Manage realm users/identities/groups Run configuration tests Upgrade DDN Infinia | <ul style="list-style-type: none"> View realm resources | | |
| tenant | <ul style="list-style-type: none"> List cluster name Manage tenant for themselves Manage subtenant structure Manage tenant/subtenant users/identities/groups List/show data services for their own tenant | <ul style="list-style-type: none"> View tenant resources | | |
| subtenant | <ul style="list-style-type: none"> List cluster name List/show subtenant for themselves Show user for themselves List subtenant users Manage datasets Manage data services | <ul style="list-style-type: none"> View subtenant resources | | |
| dataservice | <ul style="list-style-type: none"> List cluster name List/show subtenant for themselves Show user for themselves Manage data service for themselves Manage buckets | <ul style="list-style-type: none"> View data service resources | <ul style="list-style-type: none"> List cluster name Show user for themselves List/show data service for themselves Access data in datasets/buckets owned by the data service. (note 2) | <ul style="list-style-type: none"> List cluster name Show user for themselves List/show data service for themselves Execute API commands exposed in block driver interface. (note 3) |

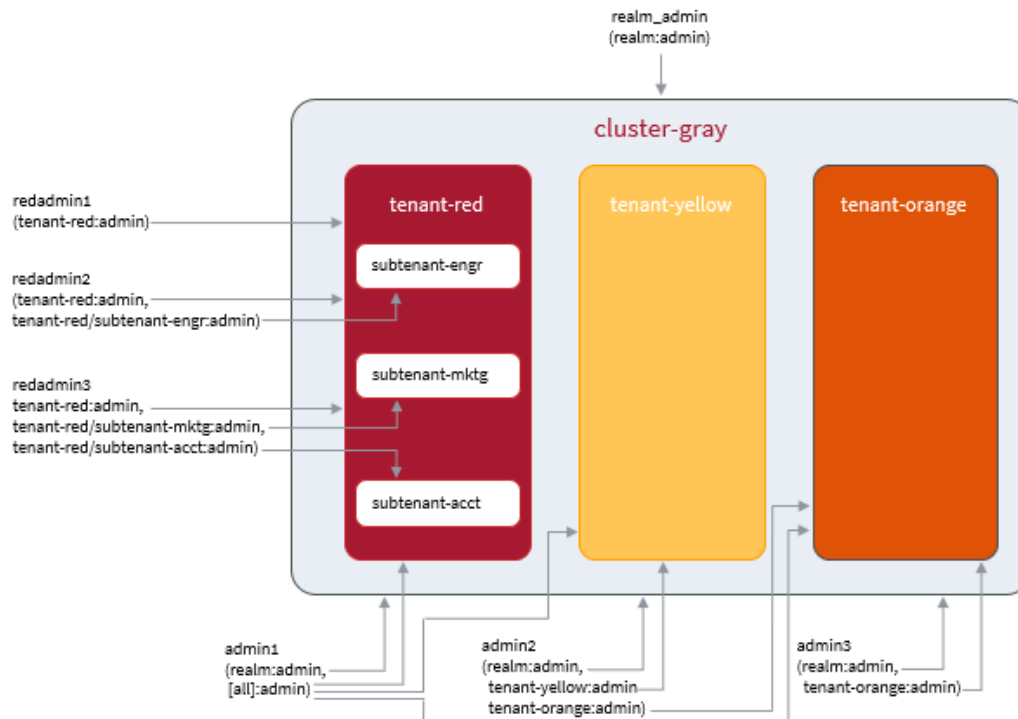
1. Viewer capability only allows looking at the managed layer.
2. The level of access is controlled by bucket ACLs. S3 secret/key can be generated for any user with `<tenant/subtenant/dataservice>:data-access` capability.
3. Users with this capability cannot issue CLI commands or access the GUI. The API commands available are limited to volume create, delete, resize, and clone for datasets that are in the dataservice scope for this user.

This user [scope](#) and [capability](#) model supports the key points defined in the following sections:

- [Cluster Management](#) (Section 3 on page 76)
- [User Management and Authorization](#) (Section 4 on page 107)
- [Data Service Management](#) (Section 5 on page 124)

Example

The following diagram shows the scopes and capabilities of an example cluster named **cluster-gray**.



where the admin roles are as follows:

realm_admin – Defined with **realm** scope and **admin** capability. This user is the default realm admin user and has administrative capability for the realm, but not for any tenants or subtenants.

admin1 – Defined with **realm** and **[all]** scopes and **admin** capability. This user has administrative capability for the realm plus all tenants.

admin2 – Defined with **realm**, **tenant-yellow**, and **tenant-orange** scopes and **admin** capability. This user has administrative capability for the realm, plus “tenant-yellow” and “tenant-orange”.

admin3 – Defined with **realm** and **tenant-orange** scopes and **admin** capability. This user has administrative capability for the realm, plus “tenant-orange”.

redadmin1 – Defined with **tenant-red** scope and **admin** capability. This user has administrative capability for “tenant-red” only.

redadmin2 – Defined with **tenant-red** and **subtenant_engr** scopes and **admin** capability. This user has administrative capability for the “tenant-red”, plus “subtenant-engr”.

redadmin3 – Defined with **tenant-red**, **subtenant_mktg**, and **subtenant_acct** scopes and **admin** capability. This user has administrative capability for the “tenant-red”, plus “subtenant-mktg” and “subtenant-acct”.

For details about the cluster created during deployment and initial setup, see [Deploy the Cluster](#) (Section 2.2 on page 38).

1.8 Identity Providers

DDN Infinia supports the following types of internal and external identity providers that can be assigned to the realm, as well as to tenants. Up to two unique identity providers are supported for the realm and per tenant:

- Local/internal – DDN Infinia supports an internal [identity provider](#), reflecting the internal multi-tenancy model on both the realm level (physical) and the tenant level (logical).
- Active Directory (AD) – DDN Infinia allows for the authentication and authorization of AD users to be given management permissions on DDN Infinia entities (both physical and logical).
- Lightweight Directory Access Protocol (LDAP) – DDN Infinia allows for the authentication and authorization of LDAP users to be given management permissions on DDN Infinia entities (both physical and logical).
- OpenID – DDN Infinia supports OpenID protocol for authentication and ID mapping.

For details about creating identity providers for the realm and tenant, see [Manage Realm Identity Providers](#) (Section 4.4 on page 115) and [Manage Tenant Identity Providers](#) (Section 4.5 on page 117), respectively.

1.9 Quotas

NOTE: DDN Infinia uses the terms *quota* and *soft quotas*; however, their use is different from the common use of the terms in filesystems (user/group quotas), as described in this section.

DDN Infinia's Quota system enables administrators to place limits on storage usage for tenants, subtenants, and datasets. To allow for high performance and scalability, a time-based system is implemented. This method allows the limit to be exceeded during a short window of time.

Implementation

The quota system is implemented by using time-limited quota authorization tokens that are specific to the tenant, subtenant, or dataset to which they are writing. These quota limits are checked and enforced by verifying that the token expiration time has not elapsed and that the limits bound to the token have not changed since the token was issued. Quota token expirations are chosen based on the number of free resources the tenant, subtenant, or dataset has available at the time of token creation. The amount that the limit can be exceeded by is dependent upon the write bandwidth of the workload. The data overrun will never be more than a very small percentage of the quota.

For details about setting quota when creating a tenant, subtenant, and dataset, see the following sections:

- [Create a Tenant](#) (Section 3.6.1 on page 94)
- [Create Subtenant](#) (Section 3.7.1 on page 98)
- [Create a Block Data Service](#) (Section 5.2.1 on page 129)
- [Create Buckets](#) (Section 5.3.1 on page 134)

Each of these limits is checked when determining the effective limit. The minimum limit for any of these entities will dictate the enforced limit. [Table 2](#) illustrates the effective quota limit for a variety of configurations set on tenants, subtenants, and datasets.

Table 2. Effective Quota Limit

| Tenant | Subtenant | Dataset | Effective Limit |
|-------------|--------------|--------------|-----------------|
| T1 – 1 TiB | S1 – 512GiB | D1 – 256 GiB | 256 GiB |
| | S2 – 512 GiB | D2 – 1 TiB | 512 GiB |
| | S3 – 2 TiB | D3 – 128 GiB | 128 GiB |
| T2 – 16 TiB | S4 – 8TiB | D4 – 4 TiB | 4 TiB |
| | S5 – 16TiB | D5 – 16TiB | 16 TiB |

Administrators can adjust quota limits dynamically. For details about modifying the quota for a tenant, subtenant, and dataset, see the following sections:

- [Update Tenant Properties](#) (Section 3.6.3 on page 95)
- [Update Subtenant Properties](#) (Section 3.7.3 on page 99)
- [Update Dataset Properties](#) (Section 3.8.2 on page 102)

Upon changing a quota limit, tokens for other quota configurations for that tenant, subtenant, or dataset are automatically invalidated. New tokens will automatically be acquired based on the entity's current usage and the new quota limit configuration.

Example

Dynamically changing a dataset quota limit to a level that is below a dataset's current usage would

- ❖ invalidate any previously issued tokens and
- ❖ return invalid tokens that prevent user IO requests from putting more data onto the storage cluster.

Alternatively, increasing the dataset usage beyond the dataset's current usage would

- ❖ invalidate any previously issued tokens and
- ❖ issue tokens that re-enable the user IO requests or extend the expiration lifetime of tokens.

As previously mentioned, DDN Infinia's quota limits are soft limits and the limits can be slightly overrun because of the time-based expiration policy of the tokens. Tokens are granted and usable for a limited amount of time based upon the availability of a tenant's, subtenant's, or dataset's quota. As the amount of free quota shrinks for a quota limited entity, the expiration for newly granted tokens shrinks. Once the entities for which the quota is defined (tenant, subtenant, or dataset) exceed a quota limit and requires a new token, they are not granted valid tokens until their quota usage no longer exceeds the limit. While the token expiration decreases to a small valid lifetime as a tenant, subtenant, or dataset approaches its quota limit, the user is still allowed to issue IO requests if a valid, non-expired token is provided to the storage cluster. This behavior may lead to small overruns of the quota limit when the limit has been breached but a valid token is still available. Hence, the DDN Infinia quota limits are soft limits and not hard limits that cutoff users as soon as the limit is breached.

Quotas are monitored as part of the relevant entity's monitoring:

- [List Tenants](#) (Section 3.6.4 on page 96)
- [List Subtenants](#) (Section 3.7.4 on page 100)
- [List Data Services](#) (Section 5.2.5 on page 130)

If a quota is exceeded by an entity, DDN Infinia raises an event, as described in [Watch for DDN Infinia Events](#) (Section 8.4 on page 168).

1.10 Command Line Interface

DDN Infinia CLI (**redcli**) is a command line tool that enables configuring, starting, and stopping one or more DDN Infinia clusters. It acts as the client of a REST server that supports a set of operations through an OpenAPI specification. The **redcli** is part of the DDN Infinia product. If you are running **redcli** commands from a DDN Infinia storage node, it is installed during the [Run Setup](#) (Section 2.1.3 on page 36) step. No further action is required. If you are running **redcli** commands from remote nodes, you must download and install the **redcli** package, as described in [Setup DDN Infinia Client](#) (Section 2.6 on page 69).

For details about log in and log out, see [Log In and Log Out using CLI](#) (Section 2.2.1.5 on page 48).

For a complete list of commands, refer to the *DDN Infinia CLI Reference*.

1.11 REST API

The DDN Infinia REST API server provides a management interface for DDN Infinia storage clusters for management clients such as the DDN Infinia CLI, DDN Infinia GUI, and customer and partner management applications. For details about REST API environment, user authentication, and basic REST operations, see [Using the DDN Infinia REST API](#) (Appendix C on page 248).

Once the REST server is started, you can inspect the OpenAPI Specification from your browser at the following URL: `https://<your_server_ip>/redapi/v1/ui/`. Using the OpenAPI Specification UI, you can view the supported resources and execute sample requests. This UI also shows `curl` commands for each sample request, as well as the URL used to access data from browser (GET requests).

For a complete list of available resources, refer to the *DDN Infinia REST API Reference*.

2. Installation and Setup

Install and setup DDN Infinia following the instructions in these sections:

- [Deploy DDN Infinia](#) (Section 2.1 on page 26)
- [Deploy the Cluster](#) (Section 2.2 on page 38)
- [Allocate Cluster Resources](#) (Section 2.3 on page 58)
- [Use DDN Infinia S3 DataService](#) (Section 2.4 on page 59)
- [Use DDN Infinia Block DataService](#) (Section 2.5 on page 68)
- [Setup DDN Infinia Client](#) (Section 2.6 on page 69)
- [Verify Deployment](#) (Section 2.7 on page 70)

Upgrade from an existing DDN Infinia installation following the steps in [Upgrade DDN Infinia](#) (Section 2.8 on page 74).

NOTE:

If required, to completely remove DDN Infinia from the set of nodes on which it has been deployed, contact DDN Support (via [Support Portal](#)).

2.1 Deploy DDN Infinia

NOTE: This section applies to realm admins only.

This section provides the commands to perform the installation on a server for use as a DDN Infinia storage node using **redsetup** (node bootstrap and resource manager). Typically, the binaries will be pre-loaded on your server. If you need to download the **redsetup** package, contact DDN Support (via [Support Portal](#)).

IMPORTANT: If a failure occurs during initial installation, run `sudo redsetup --reset` before retrying the **redsetup** command. For details about this option, see [Reset](#) (Section 2.1.1.5.1 on page 28).

The **redsetup** bootstrap process described in this section may reboot your storage nodes to activate changes in kernel parameters and other system settings.

Complete this bootstrap process after you have completed your networking configuration.

This section describes how to deploy DDN Infinia on a collection of servers. The procedures apply to bare metal physical servers or virtual machines hosted by any virtual machine hosting service.

Deploy DDN Infinia using the following steps:

1. [Plan Your Configuration](#) (Section 2.1.1 on page 27)
2. [Complete Configuration Prerequisites](#) (Section 2.1.2 on page 31)
3. [Run Setup](#) (Section 2.1.3 on page 36)

DDN Infinia should be configured with either a proxy or load balancer to distribute S3 data traffic. If there is no option to configure a load balancer/proxy, the CA certificate must be installed on client nodes after you create your cluster. For instructions, see [Install CA Certificates](#) (Section 2.2.1.4 on page 46).

For instructions to log in, query current login, and log out, see [Log In and Log Out using CLI](#) (Section 2.2.1.5 on page 48). If the **redcli** version differs from the **redapi** version, you will get an error/warning message upon log in, as described in that section.

The following procedures provide the minimum steps to install and to start a DDN Infinia cluster on your hardware. For examples showing how to verify the success of each deployment step, see [Verify Deployment](#) (Section 2.7 on page 70).

2.1.1 Plan Your Configuration

The following sections describe the configuration requirements for DDN Infinia:

- [Internet Access](#) (Section 2.1.1.1 on page 27)
- [Containers](#) (Section 2.1.1.2 on page 27)
- [Additional Software Utilities Required](#) (Section 2.1.1.3 on page 27)
- [Hardware and Networking Requirements](#) (Section 2.1.1.4 on page 27)

NOTE: For demonstration and testing purposes, you can configure multiple DDN Infinia devices from a single NVMe device using namespaces or configure HDD devices for use by DDN Infinia if NVMe devices are not available.

2.1.1.1 Internet Access

At this time, Internet access to DDN Infinia external package repositories and supported container registry is required. If you want to deploy DDN Infinia on a site with limited Internet access, contact DDN Support (via [Support Portal](#)).

2.1.1.2 Containers

A typical setup includes the following DDN Infinia containers running on each storage node:

- Storage server (**redagent**)
- Cluster management REST (**redapi**)
- GUI (**redui**)
- Object service (**reds3**)
- Utility (**registry**, **events**, and **stats**)

Note that **redcli** requires that **redapi** and **redagent** must be running correctly on the nodes. If the **redcli** version differs from the **redapi** version, you will get an error/warning message upon log in. For details, see [Log In and Log Out using CLI](#) (Section 2.2.1.5 on page 48).

IMPORTANT: DDN Infinia 1.1 supports the **config** container running on *three* storage nodes.

2.1.1.3 Additional Software Utilities Required

Examples in this guide use the **wget** tool to download the **redsetup** package. Note that the default installation of Linux OS may **not** contain this tool. If needed, install **wget** on each node where you want to install DDN Infinia.

2.1.1.4 Hardware and Networking Requirements

Prior to running **redsetup**, ensure that your hardware is set up and your network is configured as detailed in the *Hardware Installation and Maintenance Guide* for your platform.

2.1.1.5 redsetup Options

By default, the **redsetup** executable performs the following:

- Performs all bootstrapping required to make a server ready for use in a DDN Infinia cluster with no other prerequisite setup steps.
- Pulls required packages from the package repositories and container images from a supported container registry.
- Creates a default administrator named **realm_admin** for the realm, with the initial password specified during deployment. This user has administrative capability for the realm, but not for any tenants or subtenants.

The **redsetup** command in [Run Setup](#) (Section 2.1.3 on page 36) details a typical deployment; however, your deployment may require usage of additional **redsetup** options. Prior to running **redsetup**, determine which options your deployment requires. The following sections describe select options.

For a complete list of available options, see [redsetup](#) (Appendix H on page 292).

2.1.1.5.1 Reset

If a failure occurs during initial installation, run **sudo redsetup --reset** before retrying the **redsetup** command. Note that this option removes the generated file at `/opt/ddn/red/node_description/location.json`. For details about this file, see [Specify Server Location Information](#) (Section 2.1.1.5.5 on page 29).

2.1.1.5.2 Skip Reboot

To skip the reboot in step 1, pass the **--skip-reboot** option when running **redsetup**. However, before the DDN Infinia cluster can be used, all nodes must be rebooted to activate the kernel and system changes.

2.1.1.5.3 Specify Management IPs Explicitly

By default, all IP addresses will be listed as management addresses when running **redsetup**. If there are transient addresses, exclude these addresses by specifying the management IPs explicitly as a comma-separated list with the **--management-ip** option in step 1.

An example of this case is when you are using a deployment tool such as Canonical MAAS.

2.1.1.5.4 NTP Configuration

By default, the NTP server is **pool.ntp.org**. If your deployment requires alternative NTP servers, pass a comma-separated list of NTP servers with the **--ntp-servers** option when running **redsetup** in step 1.

2.1.1.5.5 Specify Server Location Information

NOTE: Running **redsetup** with the **--reset** option removes the generated file **/opt/ddn/red/node_description/location.json**.

By default, DDN Infinia looks for failure domain information in the **/opt/ddn/red/node-description/location.json** file when running **redsetup** in step 1. If required, realm admins can specify this failure domain information (location information) for nodes and devices by passing a json file using the **--node-description-file** option.

Specifying a node-description-file during **redsetup** overwrites the default file contents if the file exists or creates the default **location.json** file if it does not exist. If a default file does not exist and a node-description-file is not defined, DDN Infinia does not populate failure domain information for the node and attached devices during setup.

redsetup accepts a node-description-file in the following json format:

```
{
  "hostname": "hostname-value",
  "location": {
    "site": "site-value",
    "hall": "hall-value",
    "rack": "rack-value",
    "subrack": "subrack-value",
    "slot": "slot-value",
    "size": "size-value"
  }
}
```

This file details the failure domain information for the *current* node, as follows:

- Each node present in the cluster must have a corresponding node-description file, which contains information for that specific node. Pass this file when running **redsetup** for that node.
- Fields that are not specified or empty in the node-description file are assigned the default value.

Example

If you want nodes to be assigned the default size and default slot, remove the size and slot fields from the node description file, as shown in the following example file:

```
{
  "hostname": "hostname-value",
  "location": {
    "site": "site-value",
    "hall": "hall-value",
    "rack": "rack-value",
    "subrack": "subrack-value"
  }
}
```

After initial setup in [Deploy the Cluster](#) (Section 2.2 on page 38) is complete, view location information using the commands `redcli realm config show` or `redcli inventory show`.

Example

In the following example, the `hall` field was not provided in the node-description-file for the last node, therefore the default value (`default_hall`) for the `hall` field is displayed.

```
$ redcli realm config show
...
```

| REALM MEMBERS | | | |
|---------------|---------------|--|--------------------------------------|
| ADDRESS | HOSTNAME | COMPONENTS | LOCATION |
| 250.59.13.153 | user-redtest1 | [etcd hmi loki redagent redapi reds3 redsupport redui rregistry tsdb] | site1 / datahall1 / dev2 / top |
| 250.59.41.17 | user-redtest2 | [etcd hmi loki redagent redapi reds3 redsupport redui tsdb] | site1 / datahall1 / dev2 / bot |
| 250.59.63.77 | user-redtest3 | | site1 / default_hall / dev2 / bot |

2.1.2 Complete Configuration Prerequisites

IMPORTANT:

Each server is set up with one administrative account, configured with sudo privileges. The default user name is **nodeadmin** and password is **ddn infinia**. Login into the storage node using this account.

It is important that you change the default login password immediately to ensure security for your account.

Prior to running **redsetup**, complete the following configuration prerequisites on all storage nodes:

- [Set a Static BMC IP Address](#) (Section 2.1.2.1 on page 31)
- [Setup Host Name](#) (Section 2.1.2.2 on page 32)
- [Specify IPv4 Address/Subnet for Management Network](#) (Section 2.1.2.3 on page 32)
- [Specify IPv4 Address/Subnet for HSN Active Link #1 and #2](#) (Section 2.1.2.4 on page 33)
- [Setup Time Zone and NTP](#) (Section 2.1.2.5 on page 34)
- [Enable RoCE on the High-Speed Network Switch \(optional\)](#) (Section 2.1.2.6 on page 35)

Note that the storage nodes require external connectivity to pull and update the latest DDN Infinia container images and necessary software dependency packages, as defined in [Internet Access](#) (Section 2.1.1.1 on page 27).

2.1.2.1 Set a Static BMC IP Address

Each server has a Baseboard Management Controller (BMC), which is useful for remote management. This interface is accessible via the Linux utility **ipmitool**.

Set the BMC to a static IP address, as shown in the following example.

Example

This example configures the BMC with a static IP address using **ipmitool** commands. Note that your IP addresses may be different.

```
sudo ipmitool lan set 1 ipsrc static
sudo ipmitool lan set 1 ipaddr 192.168.1.200
sudo ipmitool lan set 1 netmask 255.255.255.0
sudo ipmitool lan set 1 defgw ipaddr 192.168.1.1
```

Verify the BMC network settings, as shown in the following example:

Example

This example gets the network settings on channel 1.

```
sudo ipmitool lan print 1
```

For additional information about the **ipmitool** utility, refer to its man page.

2.1.2.2 Setup Host Name

Set the host name for the server using the Linux tool **hostnamectl**:

```
sudo hostnamectl set-hostname <new-hostname>
```

Example

This example sets the host name to **node-01**. Note that your host names may differ.

```
sudo hostnamectl set-hostname node-01
```

Verify the new host name using **hostnamectl**:

```
hostnamectl
```

For additional information about the **hostnamectl** tool, refer to its man page.

2.1.2.3 Specify IPv4 Address/Subnet for Management Network

On Ubuntu, network configuration is handled through **netplan**, which uses a YAML configuration file to define your network setup.

Configure a static IP address for the management interface using **netplan**, as described in the following steps.

Procedure

1. Run the **./check-network-interface.sh** script with the **--mgmt** flag to identify the management network interface name. This script checks for the 1Gbps Management interface with link detected and displays the interface name, as shown in the following example.

Example

```
# ./check-network-interface.sh --mgmt
Checking management interfaces with link detected and speed = 1Gbps...
Management Interface enp7s0f0 is running at 1000Mb/s with link detected.
```

2. Using the management interface name displayed in the previous step, edit the **netplan** configuration file located in **/etc/netplan/**.

```
sudo vi /etc/netplan/00-mgmt-config.yaml
```

The following is an example **netplan** configuration for a static IP on the **enp7s0f0** management interface.

Example

```
network:
  version: 2
  ethernets:
    enp7s0f0:
      addresses:
        - 192.168.1.100/24
      routes:
        - to: default
          via: 10.168.1.1
      nameservers:
        search: [example.com, internal.example.com]
        addresses: [8.8.8.8, 1.1.1.1]
```


3. Apply the configuration to set the default route.

```
sudo netplan apply
```

4. Verify that the node can communicate to some public IP addresses, as shown in the following example.

Example

```
$ ping -c 3 ddn.com
PING ddn.com (141.193.213.11) 56(84) bytes of data.
64 bytes from 141.193.213.11: icmp_seq=1 ttl=54 time=2.38 ms
64 bytes from 141.193.213.11: icmp_seq=2 ttl=54 time=2.40 ms
64 bytes from 141.193.213.11: icmp_seq=3 ttl=54 time=2.41 ms

--- ddn.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2001ms
rtt min/avg/max/mdev = 2.384/2.398/2.413/0.011 ms

$ ping -c 3 google.com
PING google.com (142.250.191.110) 56(84) bytes of data.
64 bytes from ord38s28-in-f14.1e100.net (142.250.191.110): icmp_seq=1 ttl=113 time=22.3 ms
64 bytes from ord38s28-in-f14.1e100.net (142.250.191.110): icmp_seq=2 ttl=113 time=22.4 ms
64 bytes from ord38s28-in-f14.1e100.net (142.250.191.110): icmp_seq=3 ttl=113 time=22.3 ms

--- google.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2002ms
rtt min/avg/max/mdev = 22.260/22.319/22.357/0.042 ms
```

2.1.2.4 Specify IPv4 Address/Subnet for HSN Active Link #1 and #2

Similar to the management network, configure the high-speed network (HSN) interfaces using **netplan**, as described in the following steps.

Procedure

1. Run the **./check-network-interface.sh** script with the **--hsn** flag to identify the HSN interface names for Link #1 and Link #2. This script checks for the 100Gbps high-speed interfaces with links detected and displays the interface names, as shown in the following example.

Example

```
# ./check-network-interface.sh --hsn
Checking high-speed network interfaces with link detected and speed ≥ 100Gbps...
HSN Interface enp193s0f0np0 is running at 100000Mb/s with link detected.
HSN Interface enp65s0flnp1 is running at 100000Mb/s with link detected.
```

For HSN interfaces, it is mandatory to segment traffic by using VLANs for each interface, especially when dealing with multiple networks. Each VLAN must have its own subnet IP, which ensures the segregation of network traffic and facilitates efficient routing.

2. Using the HSN interface names displayed in the previous step, edit the **netplan** configuration file located in **/etc/netplan/**.

```
sudo vi /etc/netplan/01-hsn-config.yaml
```

The following is an example **netplan** configuration for the **enp193s0f0np0** and **enp65s0f1np1** HSN interfaces.

Example

```
network:
  version: 2
  ethernet:
    enp193s0f0np0:
      addresses:
        - 10.10.10.10/24
      mtu: 9000
    enp65s0f1np1:
      addresses:
        - 10.10.20.10/24
      mtu: 9000
```

3. Apply the configuration to set the default route.

```
sudo netplan apply
```

4. Enable Jumbo Frames in the high-speed network switches corresponding to the DDN Infinia storage nodes.

2.1.2.5 Setup Time Zone and NTP

Set your time zone using the Linux utility **timedatectl**:

```
sudo timedatectl set-timezone <your-timezone>
```

Example

This example sets the time zone to **Asia/Kolkata**. Note that your time zone may differ.

```
sudo timedatectl set-timezone Asia/Kolkata
```

Verify the time zone using **timedatectl**:

```
timedatectl
```

For additional information about the **timedatectl** utility, refer to its man page.

Install and configure your Network Time Protocol (NTP), as described in the following steps.

Procedure

1. Edit the **/etc/ntp.conf** file with your NTP server configuration.

```
sudo nano /etc/ntp.conf
```

The following is an example NTP server configuration.

Example

```
pool 0.ubuntu.pool.ntp.org iburst
pool 1.ubuntu.pool.ntp.org iburst
pool 2.ubuntu.pool.ntp.org iburst
```

2. Restart the NTP service:

```
sudo systemctl restart ntp
```

2.1.2.6 Enable RoCE on the High-Speed Network Switch (optional)

If your network switch supports RoCE, enable it on the HSN network switch corresponding to HSN Active Link #1 and HSN Active Link #2.

Example

The following example shows the commands to enable RoCE with PFC and ECN to ensure lossless Ethernet traffic for RoCE on an NVidia switch (e.g. SN3700).

```
nv set qos roce

nv config apply

nv config save

cumulus@sn3700-01:mgmt:~$ nv show qos
               operational  applied
-----
[advance-buffer-config]      default-global
[congestion-control]         default-global
[egress-queue-mapping]       default-global
[egress-scheduler]           default-global
[egress-shaper]
[link-pause]
[mapping]                     default-global
[pfc]                         default-global
pfc-watchdog
  polling-interval            0:00:00.100000
  robustness                  3
[remark]                     default-global
roce
  enable                      on
  mode                        lossless
[traffic-pool]                default-lossy
[traffic-pool]                roce-lossless
```

Once enabled, verify that RoCE is active in the storage nodes corresponding to the HSN interfaces, as shown in the following example.

Example

The following example shows that RoCe is active on the **enp193s0f0np0** and **enp65s0f1np1** HSN interfaces.

```
$ rdma link show | grep ACTIVE
link rocep193s0f0/1 state ACTIVE physical_state LINK_UP netdev enp193s0f0np0
link rocep65s0f1/1 state ACTIVE physical_state LINK_UP netdev enp65s0f1np1
```

2.1.3 Run Setup

During deployment, you must choose a node to be the **realm-entry** node. This node can be any node in your cluster. The procedure executes the **redsetup** steps on the realm-entry node first, which then establishes itself as the node to which subsequently installed nodes reference to obtain their configuration and container images.

Execute **redsetup** on all nodes using the following procedure. Note that these steps detail a typical deployment; however, your deployment may require usage of additional **redsetup** options. Prior to running **redsetup**, review the available options in **redsetup Options** (Section 2.1.1.5 on page 28) and **redsetup** (Appendix H on page 292).

Procedure

1. Choose any node in the cluster to be the realm-entry node. On this realm-entry node, execute **redsetup**, passing the **--realm-entry** option, a realm-entry secret, an initial password for the default realm admin user (**realm_admin**), and the **--ctrl-plane-ip** option.

The **--ctrl-plane-ip** option is required to bootstrap this realm-entry node and is optional for the remaining nodes. This option defines the IP address that will be used to communicate with other nodes:

- ❖ In virtual environments, use your VM IP.
- ❖ In bare metal systems, use the physical node IP.

No validation is required for **<realm_secret>**.

redsetup installs all required packages, performs OS kernel configuration, extracts DDN Infinia deployment tools, and reboots the node if needed:

```
sudo redsetup --realm-entry --realm-entry-secret <realm_secret> \
--admin-password <password> --ctrl-plane-ip <ip_address>
```

2. On the realm-entry node, check that bootstrap was successful by running **docker ps** and verify that the following containers are running: **redagent**, **redapi**, and **redui**.

Example

```
user@test:~$ docker ps
```

| CONTAINER ID | IMAGE | COMMAND | CREATED |
|--------------|------------------------|--------------------------|------------|
| 3f413ad169fd | red/redui:1.1.1_amd64 | "nginx -g 'daemon of..." | 2 days ago |
| e28074c9ac9a | red/core:1.1.1_amd64 | "/usr/bin/dumb-init ..." | 2 days ago |
| 324d2d0e9aae | red/config:1.0.0_amd64 | "/usr/local/bin/etcd" | 2 days ago |
| 270994aa56c4 | red/events:1.0.0_amd64 | "/usr/bin/loki -conf..." | 2 days ago |
| 4befc7b89aef | red/proxy:1.0.0_amd64 | "/docker-entrypoint..." | 2 days ago |
| ab67d160a576 | red/images:1.0.0_amd64 | "/entrypoint.sh /etc..." | 2 days ago |

| STATUS | PORTS | NAMES |
|-------------|-------|----------------------|
| Up 16 hours | | redui-redui-1 |
| Up 16 hours | | redapi-redapi-1 |
| Up 16 hours | | loki-loki-1 |
| Up 16 hours | | etcd-etcd-1 |
| Up 16 hours | | proxy-rproxy-1 |
| Up 16 hours | | registry-rregistry-1 |

3. **redsetup** provides self-signed certificates (**/etc/red/certs/redsetup_*.pem** files) that are used by the REST server to communicate with clients using HTTPS protocol. The setup bootstrap phase also generates the CA certificate file **/etc/red/certs/ca.pem** on the realm-entry node.

For security reasons, DDN recommends backing up and removing the **/etc/red/certs/ca.pem** file from the realm-entry node after bootstrap success.

4. On all remaining nodes, execute the following **redsetup** command, passing the **-realm-entry-secret** and **--realm-entry-address** options. Use the **<realm_secret>** used previously.

Note that this step can be run in parallel across all of the remaining nodes.

The IP address that is used to communicate with the realm-entry node is detected automatically for the other realm member nodes. The **--ctrl-plane-ip** option is optional. Use the option to manually set a specific IP address that overrides the automatically detected IP address.

```
sudo redsetup --realm-entry-secret <realm_secret> \  
--realm-entry-address <realm_entry_node_ctrl_plane_ip> [--ctrl-plane-ip <ip_address>]
```

5. If required, install the **redcli** package:
 - ❖ If you are running **redcli** commands from a DDN Infinia storage node, no further action is required.
 - ❖ If you are running **redcli** commands from remote nodes, you must download and install the **redcli** package, as described in [Setup DDN Infinia Client](#) (Section 2.6 on page 69).

2.2 Deploy the Cluster

NOTE: This section applies to realm admins only.

During the initial setup, the first system-level object created is the DDN Infinia cluster, which is the parent of all user named configurations and runtime. The cluster is both an administrative container, as well as the total collection of resources that participate as a group to serve data.

- ❖ In the [Initial Cluster Setup by using CLI](#), the realm admin specifies the name of the cluster.
- ❖ In the [Initial Cluster Setup by using UI](#), the default cluster name is `cluster1`.

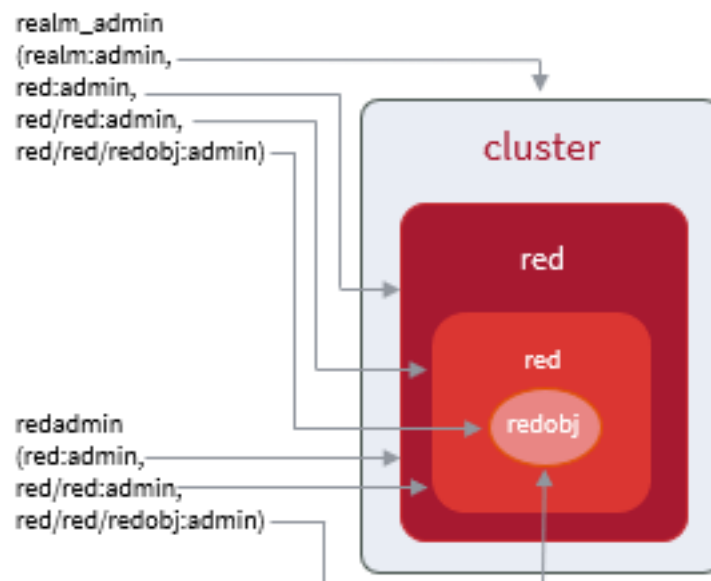
By default, the initial setup uses all available NVMe drives on each node, as well as all devices identified by `/dev/sd*`, `/dev/hd*` and all virtual devices identified by `/dev/vd*`, `/dev/xvd*`.

For configurations where it is beneficial to create a single tenant, subtenant, and object data service, pass the `-z` flag when creating the cluster. This flag performs the following:

- Creates a cluster that includes a tenant, subtenant, and dataset (all named `red`), which are available and ready for use, and adds a `redobj` object data service.
- Grants the `realm_admin` the following scopes and capabilities: `realm:admin`, `red:admin`, `red/red:admin`, `red/red/redobj:admin`
- Creates a primary tenant administrator named `redadmin` for the `red` tenant with the following scopes and capabilities: `red:admin`, `red/red:admin`, `red/red/redobj:admin`
 - ❖ The `redadmin` password is auto-generated at the time of cluster creation and must be changed upon first login of the `redadmin` user. Note that tenant users must pass the tenant name with the `-t (--tenant)` option at log in (see [Log In and Log Out using CLI](#) (Section 2.2.1.5 on page 48)).

Figure 4 shows the cluster created when the `-z` flag is used.

Figure 4. Cluster Created with `-z` Flag



After installation of DDN Infinia binaries, complete the initial setup for DDN Infinia by using either of the following methods:

- [Initial Cluster Setup by using CLI](#) (Section [2.2.1](#) on page [40](#))
- [Initial Cluster Setup by using UI](#) (Section [2.2.2](#) on page [49](#))

| | |
|-------------------|---|
| IMPORTANT: | DDN Infinia 1.1 supports only one cluster, which can span one physical site. The following sections provide instructions to create your initial cluster. If you must create a new cluster with different options, you must first delete this initial cluster. |
|-------------------|---|

2.2.1 Initial Cluster Setup by using CLI

The following steps describe the initial setup process for DDN Infinia using the `redcli` commands at the command line:

1. [Generate and Deploy Realm Configuration](#) (Section 2.2.1.1 on page 40)
2. [Install License and Accept EULA](#) (Section 2.2.1.2 on page 42)
3. [Create Cluster](#) (Section 2.2.1.3 on page 45)
4. [Install CA Certificates](#) (Section 2.2.1.4 on page 46)
5. [Log In and Log Out using CLI](#) (Section 2.2.1.5 on page 48)

2.2.1.1 Generate and Deploy Realm Configuration

IMPORTANT: DDN Infinia will check the disks in the realm configuration for the presence of file systems, partitions, or multi-path configuration. If detected, the setup process will describe the disks that have been skipped as part of the inventory generation.

A realm configuration specifies the collection of containers (and their resource limits) that are managed by DDN Infinia. The following section describes how to generate and deploy the realm configuration using the DDN Infinia CLI. Note that the bootstrap process previously described must be successfully completed before the configuration can be fully deployed.

By default, the `redcli realm config generate` command performs the following:

- Generates the `realm_config.yaml` file in the current working directory.
- Enables the DDN Infinia S3 data service (`reds3`). If your deployment does not require this data service, disable it by passing the `-d` flag in step 3 (`redcli realm config generate -d`). If `reds3` is not enabled, it will not be shown as running in step 5.

As realm admin, generate and deploy your realm configuration using the following procedure:

Procedure

1. After the first node bootstrap completes, log in using the default realm admin user name (`realm_admin`) and the initial password specified previously in the `redsetup --realm-entry` command (Section 2.1.3 on page 36, step 1):

```
redcli user login realm_admin -p <password>
```

2. Show the current hardware inventory to view the list of nodes that have finished their bootstrap process and are available for use in the DDN Infinia cluster. Wait until all the expected nodes have registered:

```
redcli inventory show
```


3. Generate a realm configuration file from the hardware inventory:

```
redcli realm config generate
```

For S3 multi-tenant configurations, pass the **reds3** server certificate domains with the **-x** (**--addcertdns**) option and the additional **reds3** server certificate IPs with the **-y** (**--addcertips**) option. Pass the parameters as comma-separated lists. Using these settings, DDN Infinia generates a **reds3** certificate file **/etc/red/certs/reds3_cert.pem**.

```
redcli realm config generate -x <domain1, domain2, domain3, ...> -y <IP1,IP2,IP3, ...>
```

4. Deploy this realm configuration. When the following command completes, all services will be configured and started on all the nodes as specified in the input file:

```
redcli realm config update -f realm_config.yaml
```

Note that you can edit the **realm_config.yaml** file to change attributes, such as node assignment, before applying the configuration; however, this change is not necessary in typical cases.

For a description of the available configuration parameters, see [Realm Configuration Input Parameters](#) (Appendix A on page 180).

5. Run **docker ps -a** and verify that the output shows all the listed services are running.

Example

```
user@test:~$ docker ps -a
```

| CONTAINER ID | IMAGE | COMMAND | CREATED | STATUS |
|--------------|--------------------|---------------------------|-------------|------------|
| b363c22f3a77 | red/core:latest | "/usr/bin/dumb-init ..." | 4 hours ago | Up 4 hours |
| d2d611b3be64 | red/core:latest | "/usr/bin/dumb-init ..." | 4 hours ago | Up 4 hours |
| 7140ab2a89e0 | red/core:latest | "/usr/bin/dumb-init ..." | 4 hours ago | Up 4 hours |
| cec8c4867a4b | red/config:latest | "/usr/local/bin/etcd" | 4 hours ago | Up 4 hours |
| 7bd3a12217d1 | red/core:latest | "/usr/bin/dumb-init ..." | 4 hours ago | Up 4 hours |
| 4cc5aa6b172c | red/core:latest | "/usr/bin/dumb-init ..." | 4 hours ago | Up 4 hours |
| 114c863d1955 | red/reui:latest | "nginx -g 'daemon of...'" | 4 hours ago | Up 4 hours |
| 952af5e0d8b7 | red/metrics:latest | "/entrypoint.sh infl..." | 4 hours ago | Up 4 hours |
| e13206d7ad4f | red/events:latest | "/usr/bin/loki -conf..." | 4 hours ago | Up 4 hours |
| b4d8a36c5e6a | red/proxy:latest | "/docker-entrypoint..." | 4 hours ago | Up 4 hours |
| c061f2dfd5b0 | red/images:latest | "/entrypoint.sh /etc..." | 4 hours ago | Up 4 hours |


```
PORTS
```



```
0.0.0.0:2379-2380->2379-2380/tcp, :::2379-2380->2379-2380/tcp
```



```
127.0.0.200:5000->5000/tcp
```

6. For S3 multi-tenant configurations, check the **reds3** certificate.

Example

```

sudo openssl x509 -noout -text -in /etc/red/certs/reds3_cert.pem
...
        X509v3 Subject Alternative Name:
            DNS:*.my.com, IP Address:127.0.0.1, IP Address:10.32.115.48,
            IP Address:10.32.115.49, IP Address:10.32.115.50,
            IP Address:172.23.0.1, IP Address:172.24.0.1, IP Address:172.25.0.1,
            IP Address:172.28.120.77, IP Address:192.172.143.1,
            IP Address:250.58.156.34
...

```

7. Install your DDN Infinia software license following the instructions in [Install License and Accept EULA](#) (Section 2.2.1.2 on page 42).

2.2.1.2 Install License and Accept EULA

The DDN Infinia software license includes an activation token, which is required to generate your DDN Infinia cluster. Your activation token is available to download from the DDN Customer Support Portal (support.ddn.io) at any time. The following is an example of the activation token and GUID format:

```

Activation token: "A1234B5C-678D-9876-5432-E505012345G6"
Realm (cluster) GUID: "E1234Q6X123ABCDE1234X51234567XYZ"

```

As realm admin, install the license using either of the following procedures:

- [Procedure 1](#) – For deployments with Internet access available from a node running **redcli**
- [Procedure 2](#) – For deployments without Internet access available from a node running **redcli**

Procedure 1

For deployments with Internet access available from a node running **redcli**, complete the following steps:

1. From a node running **redcli**, log in the DDN Customer Support Portal at support.ddn.io.
2. Enter your DDN Infinia GUID to generate the license file.
3. Download and save the license in text file format at a location that is accessible by the node.
4. Run **redcli license install** and pass the string of your activation token with the **-a** (**--activation**) option:

```
redcli license install -a <activation_token>
```

5. When prompted, review and accept the Infinia End User License Agreement (EULA). The EULA is shipped (paper version) with DDN Infinia purchases and is available at ddn.com/resources/legal/ddn-infinia-end-user-license-agreement/
6. Create and start your cluster following the instructions in [Create Cluster](#) (Section 2.2.1.3 on page 45).

Example

```
$ redcli license install -a 74QB2E18-84HK-79L5-DF62-1A90M503L207
```

| | |
|------------------|--------------------------------------|
| ProductName | DDNInfinia |
| ProductVersion | 1.1.0-beta.1 |
| GUID | B6L8269B-A00Q-00LG-369B-37L2D85N4264 |
| Activation Token | 74QB2E18-84HK-79L5-DF62-1A90M503L207 |

| | |
|-----------|-------------------------------|
| INSTALLED | true |
| VALID | true |
| CHECKED | 2024-06-17 18:24:31 +0000 UTC |
| COMPLIANT | true |
| EXPIRES | 2024-08-01 00:00:00 +0000 UTC |

Please review and accept Infinia End User License Agreement (EULA) defined here: <https://www.ddn.com/resources/legal/ddn-infinia-end-user-license-agreement>.

Accept Infinia EULA terms and conditions [Yes/No]? yes
6:24PM INF End User License Agreement (EULA) is accepted.

Procedure 2

For deployments without Internet access available from a node running **redcli**, complete the following steps:

1. From a node running **redcli**, run **redcli license request** and pass the string of your activation token with the **-a (--activation)** option:

```
redcli license request -a <activation_token>
```

2. From a system that has access to the Internet, log in DDN Customer Support Portal at support.ddn.io.
3. Enter your token and DDN Infinia GUID to generate the license file.
4. Download and save the license in text file format at a location that is accessible by the node running **redcli**.
5. From the node, run **redcli license install** and pass the string of your license file with the **-f (--licenseFile)** option:

```
redcli license install -f <license_file>
```

6. When prompted, review and accept the Infinia End User License Agreement (EULA). The EULA is shipped (paper version) with DDN Infinia purchases and is available at [ddn.com/resources/legal/ddn-infinia-end-user-license-agreement/](https://www.ddn.com/resources/legal/ddn-infinia-end-user-license-agreement/)
7. Create and start your cluster following the instructions in [Create Cluster](#) (Section 2.2.1.3 on page 45).

If you did not accept the EULA as part of license installation, run `redcli license eula` to accept it. You cannot create a cluster without accepting the EULA.

Example

```
$ redcli license eula

Please review and accept Infinia End User License Agreement (EULA) defined here: https://
www.ddn.com/resources/legal/ddn-infinia-end-user-license-agreement.

Accept Infinia EULA terms and conditions [Yes/No]? Yes
6:38PM INF End User License Agreement (EULA) is accepted.
```

2.2.1.3 Create Cluster

As realm admin, create a cluster using the following steps.

Procedure

1. Create an initial runtime and start your cluster.

```
redcli cluster create <your_cluster>
```

By default, this command enables subnet enforcement, which is required for multiple networking interfaces.

Optionally, you can automatically create a single tenant, subtenant, and object data service by passing the **-z** flag when creating the cluster. For details about this cluster, see Figure 4 on page 38. For an example, see [Example](#) at the end of this section.

2. If required, propose subsequent changes to the runtime by submitting a revised user named configuration with new or deleted components, settings, and parameters, as described in [Propose New Runtime Configuration](#) (Section 3.1 on page 78).

Example

In the first example, the command creates a cluster named **cluster-gray** using all discovered devices in inventory, without automatically creating a **red/red** tenant/subtenant combination.

```
$ redcli cluster create cluster-gray
5:07AM INF Cluster cluster-gray created.
5:07AM INF Inventory discovered.
5:07AM INF Cluster cluster-gray config auto_config created from inventory.
5:07AM INF Applying HW specific defaults for SKU AI2000
5:07AM INF Config auto_config created.
5:08AM INF Config auto_config proposed as runtime.
5:08AM INF Cluster cluster-gray starting.
5:08AM INF Cluster cluster-gray is running
5:08AM INF Initial logs will be uploaded at s3://inbound-w/ddnred/C955AAB5-99C2-91CE-957D-
283FD7B8D3B6/red_realm_001/init_20240910050838-4ce97e2a-baa7-432f-8b26-0ef0bd485876.tar.
TaskId: [6c2f1a27-33a7-4b4d-ad6d-98c005adalca]
```

In the second example, the command creates a cluster named **cluster-gray** using all discovered devices in inventory, and automatically creates a tenant, subtenant, and dataset (all named **red**) and adds a **redobj** object data service.

```
$ redcli cluster create cluster-gray -z
```

For additional details about the **redcli cluster create** command, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

2.2.1.4 Install CA Certificates

This section provides instructions to install CA certificates on the server and client nodes:

- [Install Server Certificate](#) (Section 2.2.1.4.1 on page 46)
- [Install Client Certificate](#) (Section 2.2.1.4.2 on page 47)

2.2.1.4.1 Install Server Certificate

As realm admin, install the CA certificate on the server nodes using the following steps.

Procedure

1. Log in using the default realm admin user name (**realm_admin**) and the initial password specified previously in the **redsetup --realm-entry** command (Section 2.1.3 on page 36, step 1):

```
redcli user login realm_admin -p <password>
```

2. Prepare the configuration file for future Signing Requests, using your information in the following template:

```
cat > csr.config <<-EOF
[req]
default_bits = 2048
prompt = no
default_md = sha256
req_extensions = req_ext
distinguished_name = dn

[ dn ]
C=US
ST=CA
L=Location
O=End Point
OU=Testing Domain
emailAddress=administrator@domain.com
CN=hostname.domain.com

[ req_ext ]
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = althostname.domain.com
DNS.2 = 10.10.10.10
EOF
```

3. Generate TLS key and Signing Request:

```
openssl req -new -sha256 -nodes -out reds3.csr -newkey rsa:2048 -keyout reds3key.pem -config
csr.config
```

4. Depending on your platform, sign the host certificate with CA using **reds3.csr** on the CA host side to generate the **reds3.pem** file.
5. On each DDN Infinia node, install the resulting key and certificate files:

```
redcli certificate install reds3key.pem reds3.pem
```

2.2.1.4.2 Install Client Certificate

IMPORTANT: This step is required only if the client needs direct access to **reds3** without load balancer/proxy. The main use case is to put load balancer/proxy in front of **reds3**.

As realm admin, install the CA certificate on the client nodes using the following steps.

Procedure

1. Retrieve the certificate.

```
sudo apt-get install -y ca-certificates
```

2. Log in using the default realm admin user name (**realm_admin**) and the initial password specified previously in the **redsetup --realm-entry** command (Section 2.1.3 on page 36, step 1):

```
redcli user login realm_admin -p <password>
```

3. Get the CA certificate and write it to a file **<your_ca_certificate>.cert**.

```
redcli certificate getca
```

4. Install the CA certificate on your client node:

```
sudo cp <your_ca_certificate>.cert /usr/local/share/ca-certificates  
sudo update-ca-certificates
```

2.2.1.5 Log In and Log Out using CLI

As a realm user, log in DDN Infinia by passing your credentials.

```
redcli user login <user_id> -p <your_password>
```

As a tenant user, log in DDN Infinia by passing your credentials and the tenant name with the `-t` (`--tenant`) option:

```
redcli user login <user_id> -p <your_password> -t <tenant-name>
```

Note that `<your_password>` must be at least 15 characters and can include special characters, uppercase letters, lowercase letters, and numbers. Possible minimum character sets for your password include the following:

- 3 uppercase letters, 4 lowercase letters, 3 numbers, 2 special characters
- 2 uppercase letters, 12 lowercase letters, 1 special character
- 15 lowercase letters

If the `redcli` version differs from the `redapi` version, you will get one of the following error/warning messages when you log in. Note that you cannot log in if `redcli` is incompatible.

- `redcli` is incompatible

Example

```
9:08PM FTL Server '127.0.0.1:443' redapi build version '2.1.0-beta.1' is incompatible with
redcli build version '1.1.0-beta.1'
Please install proper redcli client.
```

- `redcli` works in compatibility mode

Example

```
9:12PM WRN Server '127.0.0.1:443' redapi build version '1.0.0-beta.1' is older then redcli build
version '1.1.0-beta.1'
Redcli works in compatibility mode.
```

- `redapi` is a newer version than `redcli`

Example

```
9:09PM WRN Server '127.0.0.1:443' redapi build version '1.2.0-beta.1' is newer then redcli build
version '1.1.0-beta.1'
Client might miss new functionality, please update redcli.
```

Query current login context:

```
redcli user info [flags]
```

Log off the current user's session. The user's session stays active for 100 minutes, and then the user must re-authenticate.

```
redcli user logoff [flags]
```

For additional details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

2.2.2 Initial Cluster Setup by using UI

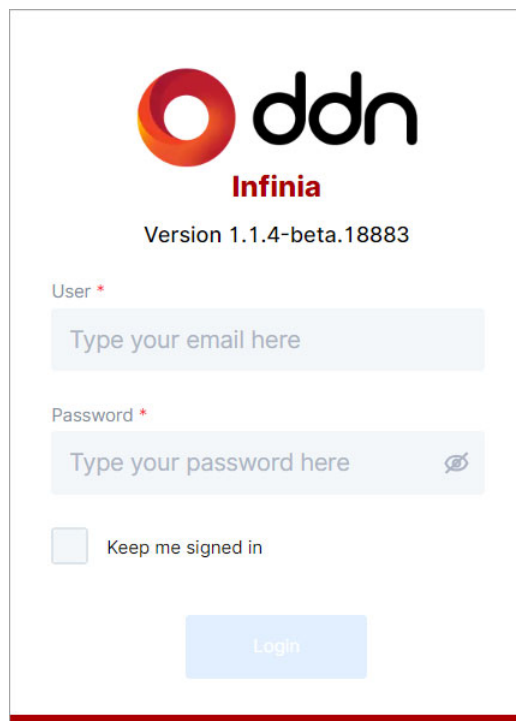
DDN Infinia provides an alternate method to complete the initial setup using the UI. The following steps describe the initial setup process for DDN Infinia using the Initial Setup wizard in UI:

1. [Login using UI](#) (Section 2.2.2.1 on page 49)
2. [Run Initial Setup Wizard](#) (Section 2.2.2.2 on page 49)
3. [Manage and Monitor DDN Infinia from the Dashboard](#) (Section 2.2.2.3 on page 54)

2.2.2.1 Login using UI

After installing the DDN Infinia binaries, log in DDN Infinia as a realm user by passing your credentials (see [Figure 5](#)).

Figure 5. DDN Infinia Login

The image shows the DDN Infinia login interface. At the top is the DDN Infinia logo, which consists of a stylized orange and red circular icon followed by the text 'ddn' in black and 'Infinia' in red. Below the logo is the version number 'Version 1.1.4-beta.18883'. The login form has two input fields: 'User' with a red asterisk and a placeholder 'Type your email here', and 'Password' with a red asterisk, a placeholder 'Type your password here', and an eye icon for toggling visibility. Below these fields is a checkbox labeled 'Keep me signed in'. At the bottom of the form is a blue 'Login' button. The entire form is enclosed in a light gray border with a red horizontal bar at the bottom.

2.2.2.2 Run Initial Setup Wizard

The Initial Setup wizard has three pages:

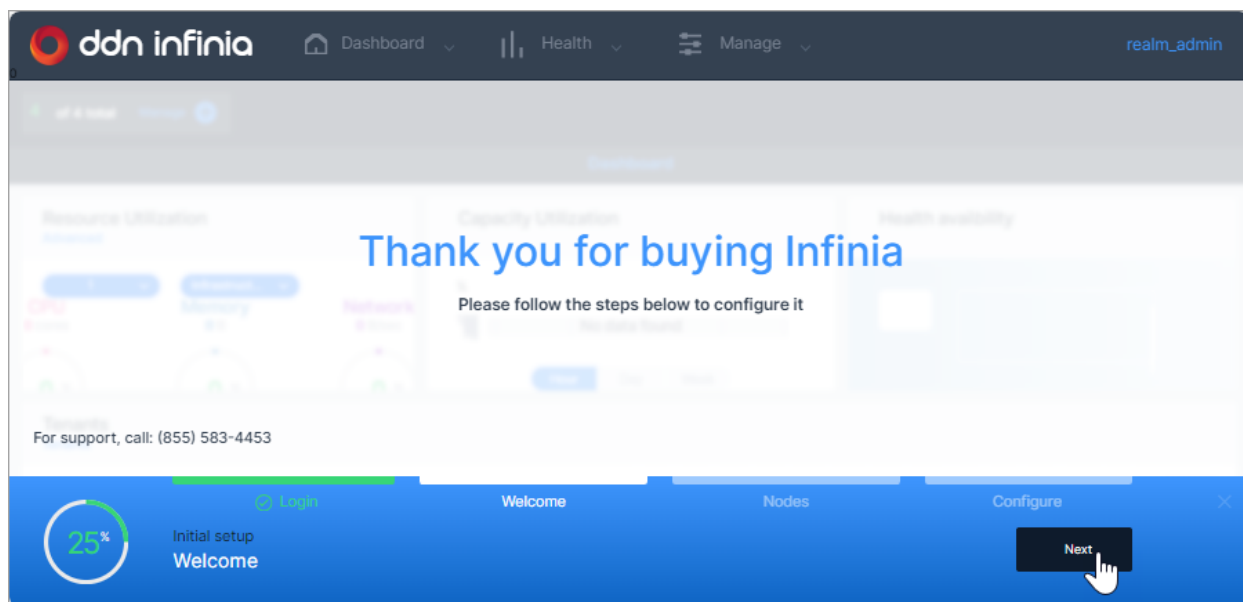
- [Welcome](#) (Section 2.2.2.2.1 on page 50)
- [Nodes Registration](#) (Section 2.2.2.2.2 on page 50)
- [Configure Authentication and Users](#) (Section 2.2.2.2.3 on page 51)

Follow the steps on a page and then click **Next** to move to the next page in the sequence.

2.2.2.2.1 Welcome

On initial login, you land on the **Welcome** page of the Initial Setup wizard (Figure 6). The tick mark on the Login tab signifies that the login process is successfully complete. A circle in the lower-left corner signifies that the total progress in initial setup is 25%. Click **Next** to move to the **Nodes Registration** page.

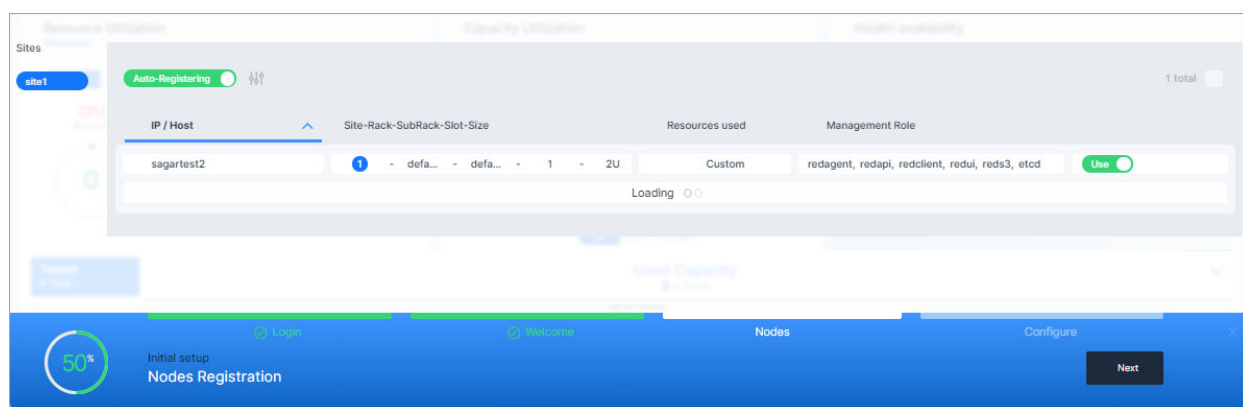
Figure 6. Initial Setup Wizard for DDN Infinia



2.2.2.2.2 Nodes Registration

In the **Nodes Registration** page, the registered nodes appear automatically with details such as sitename, rack number, subrack number, slot, and size. The nodes are auto-assigned with management roles (see Figure 7).

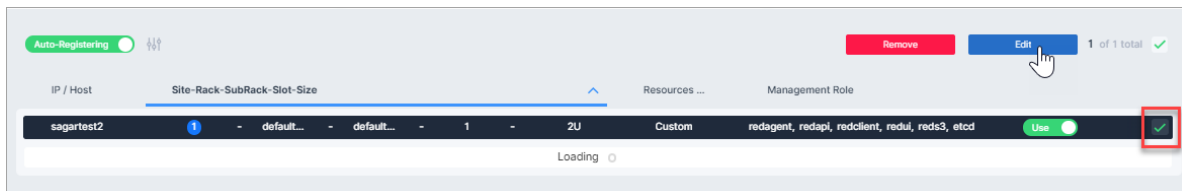
Figure 7. Node Registration Page



If required, edit the details of a node by completing the following steps.

Procedure

1. Select the check box next to the node and then click **Edit**.



2. In the **Modify_hostnodename** dialog box, modify the site location, other physical details of the host node, management roles played by the host node, and then click **Modify**.

3. After you are done with the required node modifications, click **Next** to move to the **Configure Authentication and Users** page.

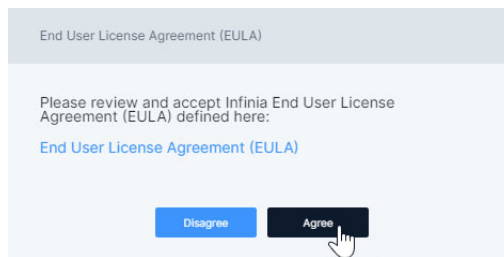
2.2.2.2.3 Configure Authentication and Users

In the **Configure Authentication and Users** page, complete the following steps to apply the license and to add authentication methods and users.

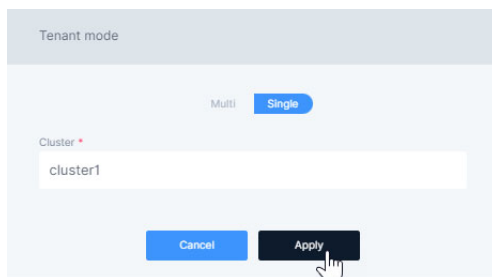
Procedure

1. Apply the license.
 - a. On landing, the License dialog box appears prompting for the DDN Infinia activation key. Enter the activation key and then click **Apply**.

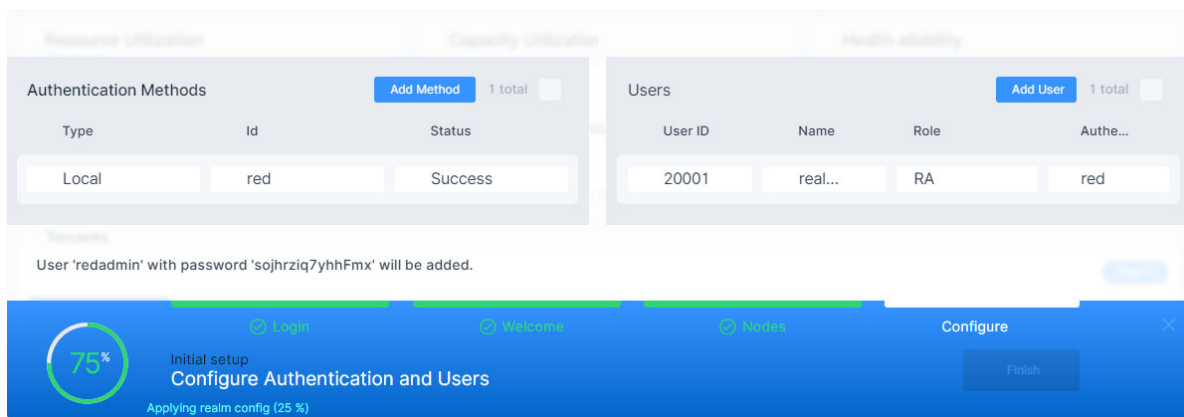
- b. In the **End User License Agreement (EULA)** dialog box that appears, click the End User License Agreement (EULA) link, which will open a page with End User License Agreement. Read the agreement and if acceptable click **Agree**.



2. In the Tenant mode dialog box that appears, select the type of tenant mode that DDN Infinia should be operated. For example, select *Multi* for multi mode tenant or *Single* for single mode tenant. If required, enter a name for cluster and then click **Apply**. Note that default cluster name is *cluster1*.



3. If required, add additional authentication methods and additional users.

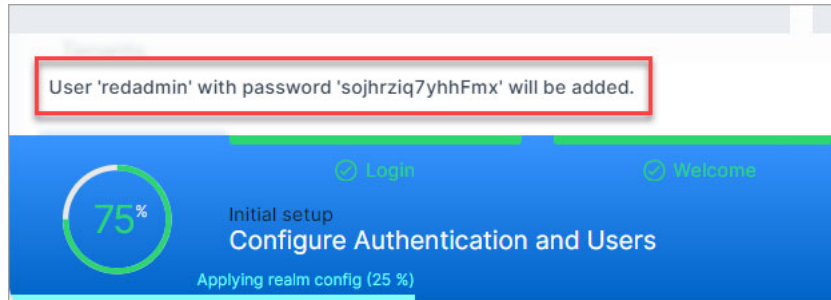


- a. By default, DDN Infinia adds a local identity provider named *red*. To add a new identity provider, complete the following steps:
- Click **Add Method**.
 - In **Add ID Provider** dialog box, provide the following details:

| Field | Description |
|---------|---|
| Type | Select the type of identity provider. For example LDAP. |
| Config* | Provide the configuration details for the identity provider |

| Field | Description |
|--------|---|
| Name* | Enter a name for the identity provider |
| Tenant | Select the tenant to which the authentication is to be assigned |
| Group | Select the group to which the authentication is to be assigned |

- b. During initial setup, DDN Infinia automatically creates a tenant called *red*, creates a primary admin user for the tenant as *redadmin*, and generates a password for *redadmin*, which is notified on the **Configure Authentication and Users** page as follows.



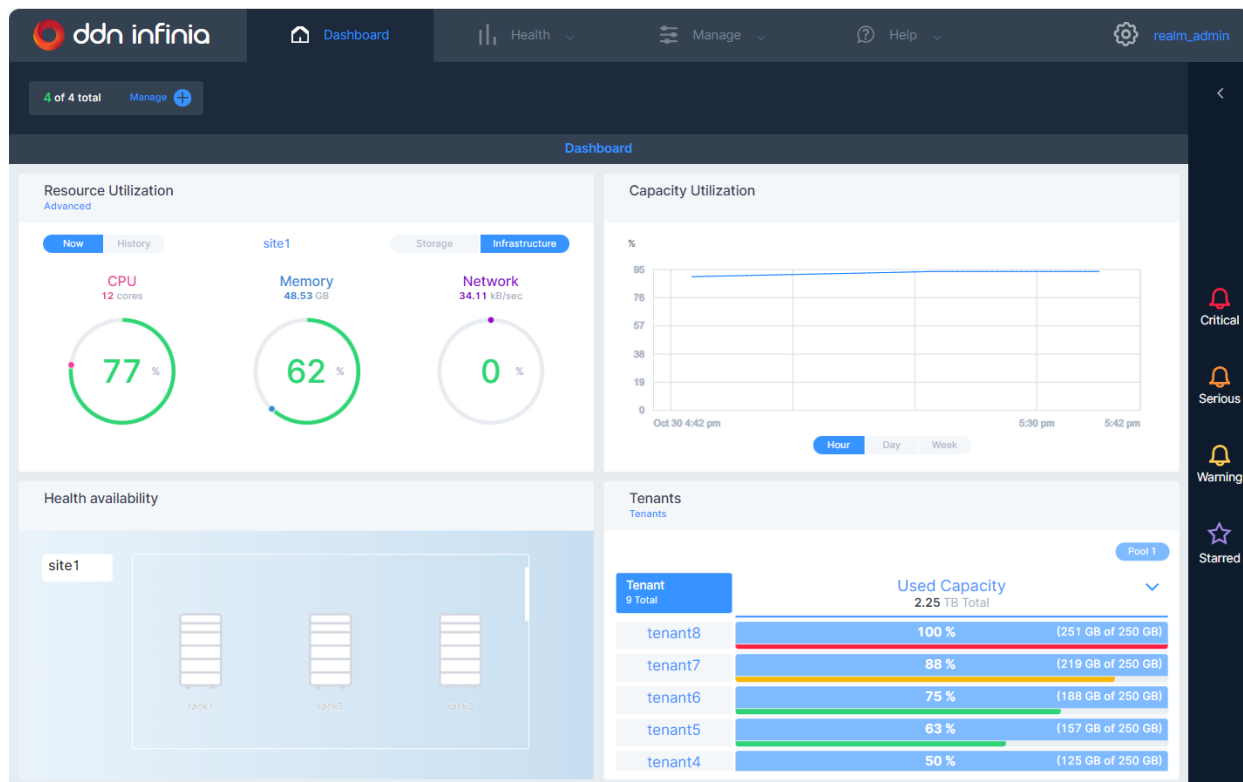
To add more users, click **Add User** and follow step 3 under [Add an User](#) (Appendix E.1 on page 267) to add details to the Add User wizard.

4. After you complete adding authentication methods or users, click **Finish** to complete the initial setup.

2.2.2.3 Manage and Monitor DDN Infinia from the Dashboard

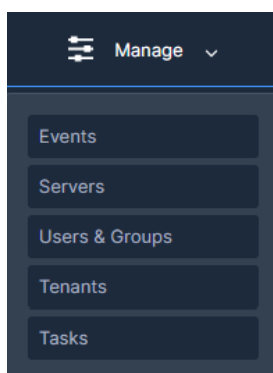
After finishing the initial setup, you land on the **Dashboard** page (see [Figure 8](#)). The dashboard displays the applicable widgets as per the scopes and capabilities assigned to a *realm* administrator (*realm admin*).

Figure 8. DDN Infinia Dashboard for Realm Administrator



Note that the menu commands, tabs, and widgets that appear on the UI varies as per the type of user and the scopes and capabilities assigned to the user. For example, [Figure 9](#) shows the menu items for a realm admin on the first login to DDN Infinia.

Figure 9. Initial Manage Menu for a Realm Administrator



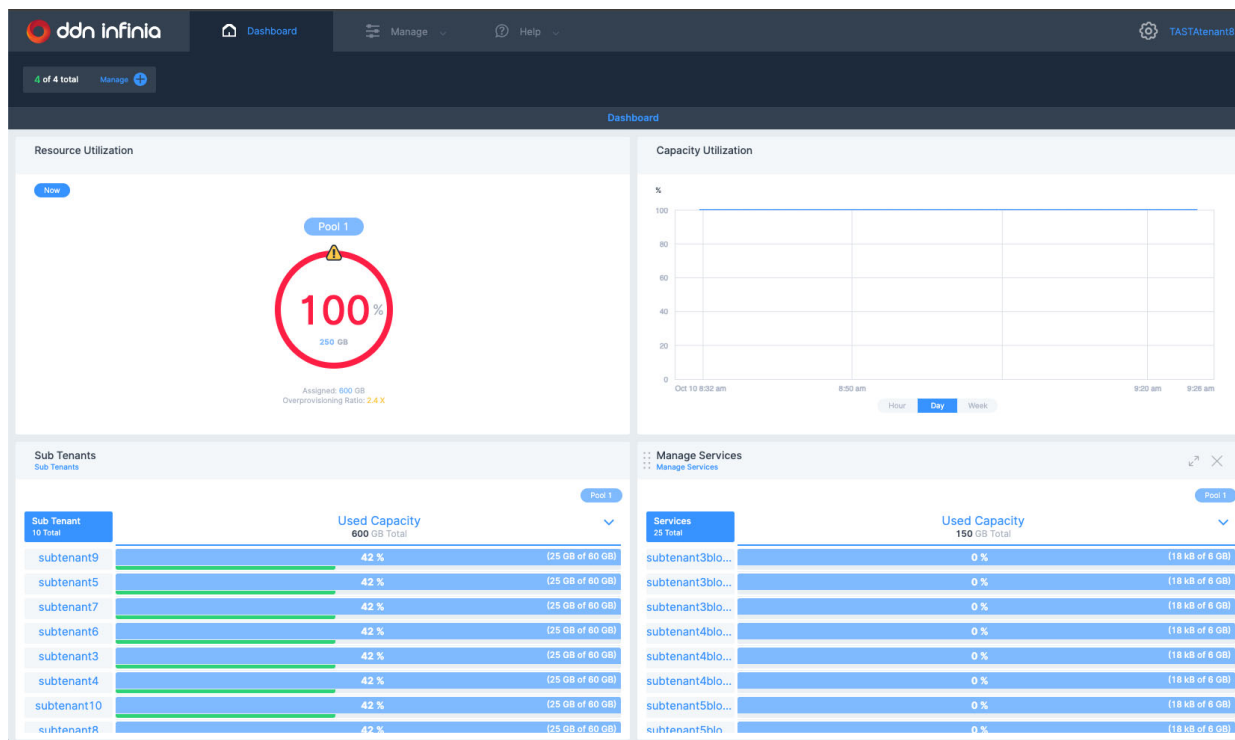
Perform various functions for storage management and monitoring using tabs and menu commands on the realm admin **Dashboard** page (see [Table 3](#)).

Table 3. Tabs and Menu Details on the Realm Administrator Dashboard

| Tab | | Description |
|-----------|----------------|--|
| Dashboard | | Manage Widgets for the realm administrator. |
| Health | | View the hardware resource (CPU, Memory, Network) utilization status per storage server in DDN Infinia. |
| Manage | Events | View logs of different types of DDN Infinia events arranged per their severity level. <ul style="list-style-type: none"> • View Events |
| | Servers | Manage the server resources in DDN Infinia. View the physical details of the servers under a site. View the management roles played by the servers. <ul style="list-style-type: none"> • Manage Servers • Change the Physical Details of a Server • Perform Various Operations On the Server |
| | Users & Groups | Manage users and groups along with their scopes and capabilities. Perform following activities using the menu: <ul style="list-style-type: none"> • Add an User • Add a Group • Add an Existing User to a Group • Delete an User • Delete a Group |
| | Tenants | Manage tenants. Perform following activities using the menu: <ul style="list-style-type: none"> • Add a Tenant |
| | Tasks | View the status of the tasks performed in DDN Infinia. <ul style="list-style-type: none"> • View Tasks |

To view the **Dashboard** page for a *tenant administrator* ([tenant admin](#)), log in DDN Infinia as a tenant user by passing your credentials (see [Figure 10](#)). The dashboard displays the applicable widgets as per the scopes and capabilities assigned to the tenant admin.

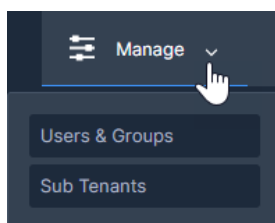
Figure 10. DDN Infinia Dashboard for Tenant Administrator



Note that the top-left widget shows tenant quota and how much of it is being used. The top-right widget shows historical view of the tenant quota. The bottom-left widget shows subtenant quotas. The bottom-right widget shows services quotas.

Following [Figure 11](#) shows the menu items for a tenant admin on the first login to DDN Infinia.

Figure 11. Initial Manage Menu for a Tenant Administrator



Perform various functions for tenant management using tabs and menu commands on the tenant admin **Dashboard** page (see [Table 4](#)).

Table 4. Tabs and Menu Details on the Tenant Administrator Dashboard

| Tab | | Description |
|-----------|----------------|--|
| Dashboard | | Manage Widgets for the tenant administrator. |
| Manage | Users & Groups | Manage users and groups along with their scopes and capabilities. Perform following activities using the menu: <ul style="list-style-type: none"> • Add an User • Add a Group • Add an Existing User to a Group • Delete an User • Delete a Group |
| | Sub Tenants | Manage sub tenants. Perform following activities using the menu: <ul style="list-style-type: none"> • Add a Sub Tenant • Add an Object Data Service • Add a Block Data Service • Add a Bucket |

For detailed instructions for some commonly performed administration tasks, see [Using the DDN Infinia User Interface](#) (Appendix E on page 266).

2.3 Allocate Cluster Resources

NOTE: This section applies to realm, tenant, and subtenant admins.

DDN Infinia is a multi-tenant system, allowing administrators to allocate resources to individual tenants and subtenants in a hierarchical structure. Tenant is the first level of multi-tenancy, separating realm resources. Realm admin users are responsible for creating tenants; while tenant admin users are responsible for creating subtenants.

Administrators allocate cluster resources, as follows:

- Realm admins manage resources for the realm as described in the following sections:
 - ❖ [Propose New Runtime Configuration](#) (Section 3.1 on page 78)
 - ❖ [Manage Nodes](#) (Section 3.2 on page 81)
 - ❖ [Manage Core Affine Storage Targets](#) (Section 3.3 on page 85)
 - ❖ [Manage Networks](#) (Section 3.4 on page 88)
 - ❖ [Manage Volumes](#) (Section 3.5 on page 89)
 - ❖ [Manage Tenant Structure](#) (Section 3.6 on page 93)
 - ❖ [Manage Realm Users and Groups](#) (Section 4.1 on page 108)
 - ❖ [Manage Realm Identity Providers](#) (Section 4.4 on page 115)
 - ❖ [Upgrade Server, Drive, and NIC Firmware](#) (Section 6.1 on page 138)
- Tenant admins manage resources for the tenant as described in the following sections:
 - ❖ [Update Tenant Properties](#) (Section 3.6.3 on page 95) for themselves
 - ❖ [Manage Subtenant Structure](#) (Section 3.7 on page 97)
 - ❖ [Manage Tenant or Subtenant Users and Groups](#) (Section 4.2 on page 110)
 - ❖ [Manage Tenant Identity Providers](#) (Section 4.5 on page 117)
- Subtenant admins manage datasets and data services as described in the following sections:
 - ❖ [Manage Datasets](#) (Section 3.8 on page 101)
 - ❖ [Manage S3 Access Records](#) (Section 5.1 on page 125)
 - ❖ [Manage Data Services](#) (Section 5.2 on page 128)
 - ❖ [Manage Buckets](#) (Section 5.3 on page 133) on the data service for which they have admin capability.

For detailed instructions about performing these administration tasks using the UI, see [Using the DDN Infinia User Interface](#) (Appendix E on page 266).

2.4 Use DDN Infinia S3 DataService

NOTE: This section applies to realm, tenant, and subtenant admins. To access the DDN Infinia S3 DataService, users must have `<tenant/subtenant/dataservice>:admin` or `<tenant/subtenant/dataservice>:data-access` scope and capability.

DDN Infinia provides object data services using the Amazon's Simple Storage Service (S3) protocol. Subtenant admins manage these DDN Infinia data services for S3 to provide object access for DDN Infinia users. DDN Infinia is limited to one object data service per subtenant.

S3 is a REST service and DDN Infinia provides support for a subset of S3 API commands. The following sections provide a brief overview of using the DDN Infinia S3 DataService. [DDN Infinia S3 DataService API Endpoints](#) (Appendix B on page 182) provides additional details, including supported S3 API endpoints.

By default, the DDN Infinia S3 DataService is enabled when DDN Infinia is deployed (Section 2.1 on page 26).

To use the DDN Infinia S3 DataService in *single-tenant* clusters, complete the steps in the following sections:

- [Setup Access to S3 Service](#) (Section 2.4.2 on page 61)
- [Read and Write Object Data with S3 Client Utility](#) (Section 2.4.3 on page 65)

To use the DDN Infinia S3 DataService in *multi-tenant* clusters, complete the steps in the following sections:

- [Define Virtual Hosts on Nodes](#) (Section 2.4.1 on page 60)
- [Setup Access to S3 Service](#) (Section 2.4.2 on page 61)
- [Read and Write Object Data with S3 Client Utility](#) (Section 2.4.3 on page 65)

For additional details about managing key/secret pairs, managing data services, and managing buckets, see [Data Service Management](#) (Section 5 on page 124).

2.4.1 Define Virtual Hosts on Nodes

IMPORTANT: Virtual hosts are used to select the proper service for multi-tenant access. This step is not required for one tenant access.

The virtual hosts (vhosts) are used to isolate traffic between different S3 services. The same IP addresses could be used for multiple S3 services. This isolation is required to separate tenant/subtenant specific traffic to their respective buckets for S3 access. Your vhosts must be defined on DDN Infinia and remote nodes. Also, the vhosts could be defined as DNS entries for specific nodes or for load balancer.

Add the vhosts to `/etc/hosts` on every DDN Infinia node and to `/etc/hosts` on every remote (client) node.

Example

The following example shows the vhosts in `/etc/hosts` on the nodes and validates access using `curl`.

```
# Display the vhosts in /etc/hosts on the DDN Infinia nodes.
$ cat /etc/hosts
127.0.0.1 localhost yellow.my.com orange.my.com

# Validate access
$ curl -k --max-time 10 https://yellow.my.com:8111/healthz
{}
$ curl -k --max-time 10 https://orange.my.com:8111/healthz
{}

# Display the vhosts in /etc/hosts on the client nodes.
$ cat /etc/hosts
127.0.0.1 localhost

10.25.118.88 yellow.my.com orange.my.com

# Validate access
$ curl -k --max-time 10 https://yellow.my.com:8111/healthz
{}
$ curl -k --max-time 10 https://orange.my.com:8111/healthz
{}
```

2.4.2 Setup Access to S3 Service

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, administrators can create new tenants and subtenants, and then add new users with the `s3` role permitting them to read and write data using the S3 protocol. To authenticate the REST requests, users must be created in DDN Infinia within a given subtenant scope and assigned an access key and secret key by the administrator.

Note that realm users cannot perform S3 data access operations. Subtenant admins manage the data services and users with `<tenant/subtenant/dataservice>:admin` or `<tenant/subtenant/dataservice>:data-access` scope and capability access the S3 service.

In general, defining users with S3 role and creating buckets requires these steps:

1. As realm admin, create a tenant structure following the instructions in [Manage Tenant Structure](#) (Section 3.6 on page 93).
2. As tenant admin, create a subtenant structure following the instructions in [Manage Subtenant Structure](#) (Section 3.7 on page 97).
3. As tenant admin, add new users or groups with subtenant admin capability or grant existing users or groups the capability following the instructions in [Manage Tenant or Subtenant Users and Groups](#) (Section 4.2 on page 110).
4. As subtenant admin, add an S3 access record for the user following the instructions in [Manage S3 Access Records](#) (Section 5.1 on page 125).

Note that S3 access records consist of a key/secret pair, which are required to read and write data using the S3 protocol.

5. As subtenant admin, create an object data service following the instructions in [Create an Object Data Service](#) (Section 5.2.2 on page 129). Only subtenant admins with admin capability on the data service can create buckets in step 7.

Note that it takes 30-60 seconds for S3 to pickup the new configuration after adding the new service.

Object naming follows [AWS compatibility standards](#). For guidelines about object naming, see [S3 Bucket and Object Management](#) (Appendix B.2 on page 183).

6. As tenant admin, add new users or groups with the capability to access the S3 service or grant existing users or groups the capability to access the S3 service following the instructions in [Manage Tenant or Subtenant Users and Groups](#) (Section 4.2 on page 110).
7. As a subtenant admin with admin capability on the data service, create buckets following the instructions in [Manage Buckets](#) (Section 5.3 on page 133).

DDN Infinia S3 DataServices conform to [S3 protocol standards](#). For rules about bucket naming, see [S3 Bucket and Object Management](#) (Appendix B.2 on page 183).

The following examples define the S3 service (`red/red/redobj`) for the `red` tenant (Figure 4 on page 38) and a new S3 service (`tenant-yellow/subtenant-yellow/yellowobj`) for a new `tenant-yellow` tenant, respectively.

Example

In the first example, the default realm admin (**realm_admin**) creates a new admin user (**new_admin**) with **red** tenant, **red/red** subtenant, and **red/red/redobj** dataservice scopes in the **red** tenant (Figure 4 on page 38). This new admin user then creates an S3 access record and bucket from within the **red** tenant domain.

```
# Login as the default realm admin.
$ redcli user login realm_admin -p rEalmsEcretwd4U
12:04AM INF User "realm_admin" logged in successfully.

# Add a new admin user with tenant, subtenant, and dataservice scopes in the red tenant.
$ redcli user add new_admin -p nEwsEcretwoRd4A -r red,red/red,red/red/redobj -t red
12:05AM INF User "new_admin" has been added.

# Login as the new admin user.
$ redcli user login new_admin -p nEwsEcretwoRd4A -t red
12:06AM INF User "new_admin" logged in successfully

# Add an S3 access record for the new admin user.
$ redcli s3 access add new_admin -e 1y
12:06AM INF Auto Selecting cluster: cluster-gray
```

| | |
|------------|--|
| S3_KEY | 7LR5RIKJGUKDMM2K6KYB |
| S3_SECRET | eZbiKgL9MdGsbXYwH4fA18fDVd71S467txIej8Uc |
| EXPIRATION | 2024-12-24 00:07:50 |
| CLUSTER | cluster-gray |
| TENANT | red |
| GROUP | |
| UID | 20002 |
| GID | 20000 |
| EXPIRED | false |

```
# Create a bucket for the new admin user.
$ redcli s3 bucket create redbckt
12:06AM INF Auto Selecting cluster: cluster-gray
12:06AM INF Auto Selecting tenant: red
12:06AM INF Auto Selecting subtenant: red
12:06AM INF Auto Selecting service: redobj
12:07AM INF S3 bucket created

# List buckets assigned to the new admin user.
$ redcli s3 bucket list
12:07AM INF Auto Selecting cluster: cluster-gray
12:07AM INF Auto Selecting tenant: red
12:07AM INF Auto Selecting subtenant: red
12:07AM INF Auto Selecting service: redobj
```

| NAME | OWNER | CAPACITY | USAGE |
|---------|-----------|----------|-------|
| redbckt | new_admin | 9.223 EB | 0 B |

Example

In the second example, the default realm admin (**realm_admin**) creates a new tenant named **tenant-yellow** with primary admin user **yellow_admin**. This new admin user then creates a subtenant (**subtenant-yellow**), an S3 service (**yellowobj**), and bucket (**yellowbckt**) from within the **tenant-yellow** tenant domain. Note that in this case, **yellow.my.com** must be routable to the **red** data network.

```
# Login as the default realm admin.
$ redcli user login realm_admin -p rEalmsEcretwd4U
11:08PM INF User "realm_admin" logged in successfully.

# Create a tenant with primary admin user. Note that the password will be auto-generated, if not specified.
$ redcli tenant create tenant-yellow -u yellow_admin
11:08PM INF Auto Selecting cluster: cluster-gray
11:08PM INF Successfully created tenant: tenant-yellow

# Login as the yellow_admin user.
$ redcli user login yellow_admin -p yeLlowseCretwd2 -t tenant-yellow
11:08PM INF User "yellow_admin" logged in successfully.

# Create a subtenant in the new tenant.
$ redcli subtenant create subtenant-yellow -t tenant-yellow
11:08PM INF Auto Selecting cluster: cluster-gray
11:08PM INF Successfully created subtenant: subtenant-yellow

# Grant the yellow_admin user access to the new subtenant.
$ redcli user grant yellow_admin tenant-yellow/subtenant-yellow
11:08PM INF User "yellow_admin" access to "tenant-yellow/subtenant-yellow" granted.

# Add an S3 access record for the yellow_admin user.
$ redcli s3 access add yellow_admin -e 10y -t tenant-yellow
11:08PM INF Auto Selecting cluster: cluster-gray
```

| | |
|------------|--|
| S3_KEY | 100WDCZGLCZ7Z9RI3DQ0 |
| S3_SECRET | xnVPqYm1ZiaavQiv9XikcQbFu7TnSp5M7x7dfW3K |
| EXPIRATION | 2033-10-19 23:08:33 |
| CLUSTER | cluster-gray |
| TENANT | tenant-yellow |
| GROUP | |
| UID | 20003 |
| GID | 20000 |
| EXPIRED | false |

```
# Create an S3 service assigned to vhost yellow.my.com.
# In this case, yellow.my.com must be routable to the red data network.
$ redcli service create yellowobj -T file-and-object -P s3 -t tenant-yellow -s subtenant-yellow -V yellow.my.com
11:08PM INF Auto Selecting cluster: cluster-gray
11:08PM INF Auto Selecting tenant: tenant-yellow
11:08PM INF Auto Selecting subtenant: subtenant-yellow
11:08PM INF Service 'yellowobj' has been added.

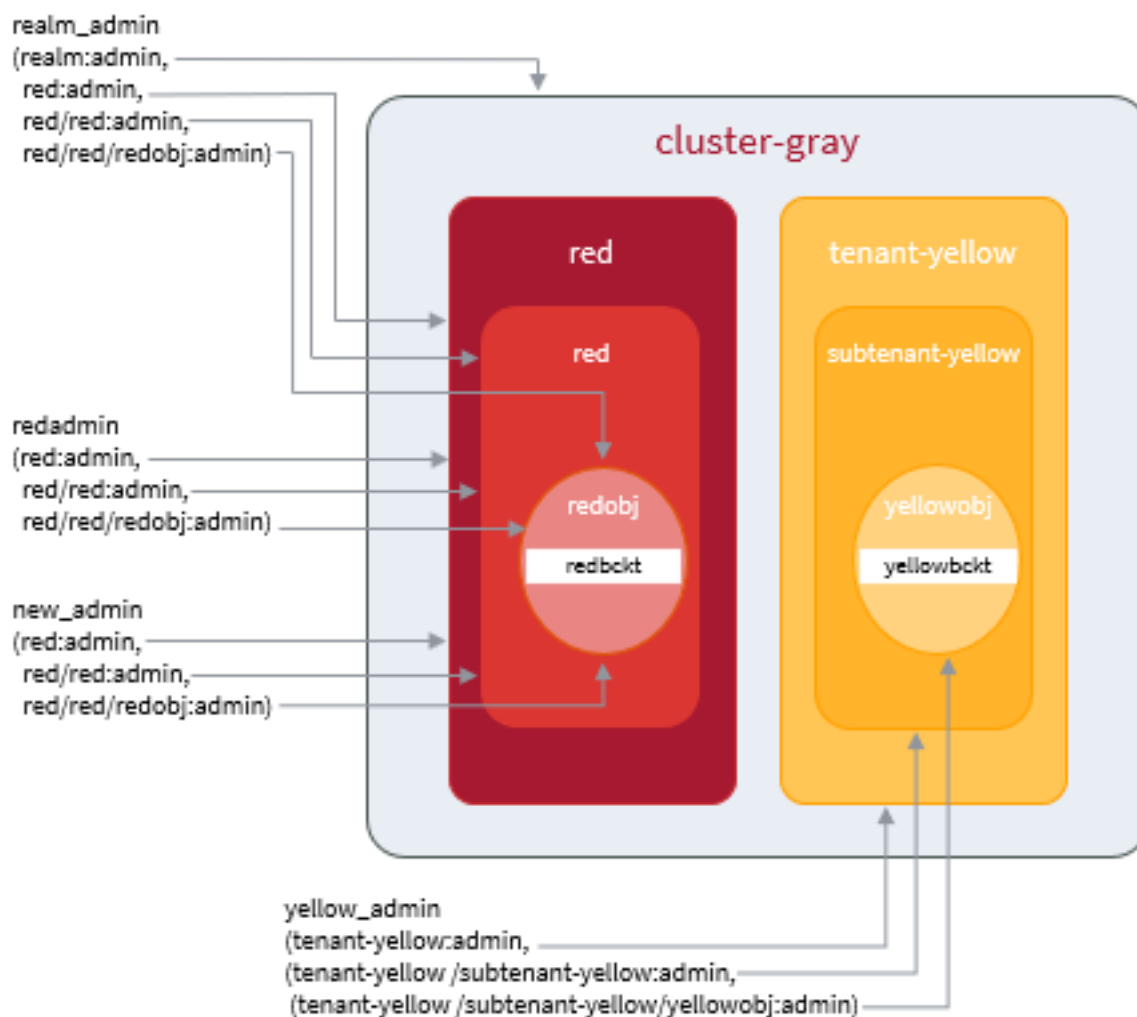
# Grant the yellow_admin user access to the new data service.
$ redcli user grant yellow_admin tenant-yellow/subtenant-yellow/yellowobj
11:08PM INF User "yellow_admin" access to "tenant-yellow/subtenant-yellow/yellowobj" granted.

# Create a bucket for the yellowobj service.
$ redcli s3 bucket create yellowbckt -r tenant-yellow/subtenant-yellow/yellowobj
12:26AM INF Auto Selecting cluster: cluster-gray
12:26AM INF S3 bucket created
```

```
# List buckets assigned to the yellowobj service.
$ redcli s3 bucket list -r tenant-yellow/subtenant-yellow/yellowobj
12:26AM INF Auto Selecting cluster: cluster-gray
```

| NAME | OWNER | CAPACITY | USAGE |
|------------|--------------|----------|-------|
| yellowbckt | yellow_admin | 9.223 EB | 0 B |

The following diagram shows the **red** tenant, **red/red** subtenant, **red/red/redobj** and newly created **redbckt**, along with the newly created **tenant-yellow**, **subtenant-yellow**, **yellowobj**, and **yellowbckt**.



2.4.3 Read and Write Object Data with S3 Client Utility

To execute **aws** commands from a supported AWS client, you must update the AWS Command Line Interface (AWS CLI) with your key/secret pair as follows.

Procedure

1. List the access records defined for the subtenant:

```
redcli s3 access list
```

Example

```
$ redcli s3 access list
```

| USER_ID | TENANT | SUBTENANT | S3_KEY | EXPIRATION | GROUP |
|-----------------|---------|------------|----------------------|------------|-------|
| subtenant_admin | tenant1 | subtenant1 | WN2UJ1GZJW3X8T2BDBJ0 | 2025-02-21 | |

2. Show the access record details using the S3_KEY value:

```
redcli s3 access show <S3_KEY_value>
```

Example

For the S3_KEY value of WN2UJ1GZJW3X8T2BDBJ0, the example output is as follows.

```
$ redcli s3 access show WN2UJ1GZJW3X8T2BDBJ0
```

| | |
|------------|----------------------|
| S3_KEY | WN2UJ1GZJW3X8T2BDBJ0 |
| S3_SECRET | <secret> |
| EXPIRATION | 2025-02-21 |
| CLUSTER | cluster-gray |
| TENANT | tenant1 |
| SUBTENANT | subtenant1 |
| GROUP | |
| UID | 20003 |
| GID | 20000 |

3. Update the AWS CLI with your key/secret pair. Edit the credentials file directly or use a utility such as **sed**, as shown in the following example.

Example

```
S3KEY=WN2UJ1GZJW3X8T2BDBJ0
S3SECRET=<secret>
sed -i 's/aws_access_key_id = .*/aws_access_key_id = "$S3KEY"/' ${HOME}/.aws/credentials
sed -i 's/aws_secret_access_key = .*/aws_secret_access_key = "$S3SECRET"/' ${HOME}/.aws/credentials
```

4. To read or write object data to the buckets you created, after you set the key/secret pair, execute the **aws** commands as shown in the following examples.

- ❖ List all the buckets in a specified storage host.

The **S3_ENDPOINT** environment variable is defined for convenience, and it can point to any of the DDN Infinia storage nodes or a load balancer that fronts the DDN Infinia S3 service:

Example

```
$ export S3_ENDPOINT=https://<s3-server-host-name-or-ip-address>:8111
$ aws --no-verify-ssl --endpoint-url $S3_ENDPOINT s3api list-buckets
{
  "Buckets": [
    {
      "Name": "bucket1",
      "CreationDate": "2023-02-21T23:12:50Z"
    },
    {
      "Name": "bucket3",
      "CreationDate": "2023-02-21T23:13:24Z"
    }
  ],
  "Owner": {
    "DisplayName": "subtenant_admin",
    "ID": "3f6bda07efaa463fb6e8312749a39cb91a3a5caef6f59bcfcefd32e2ed32b2be"
  }
}
```

- ❖ Write an object to a bucket.

Example

In the following example, object data from a sample object file is copied to the objects in **bucket1**:

```
$ cat sampleobj.txt
Wed Feb 22 12:00:43 AM UTC 2023

$ aws --no-verify-ssl --endpoint-url $S3_ENDPOINT s3api put-object --bucket bucket1 --key
first-object --body sampleobj.txt
{
  "ETag": "\"2706bc01df207bcdcab419685e42919d\"",
  "VersionId": "1pUcIO"
}
```

- ❖ List the objects in a bucket.

Example

In the following example, the objects in **bucket1** are listed:

```
$ aws --no-verify-ssl --endpoint-url $S3_ENDPOINT s3api list-objects --bucket bucket1
{
  "Contents": [
    {
      "Key": "first-object",
      "LastModified": "2023-02-22T00:02:44.000Z",
      "ETag": "\"2706bc01df207bcdcab419685e42919d\"",
      "Size": 32,
      "Owner": {
        "DisplayName": "subtenant_admin",
        "ID": "3f6bda07efaa463fb6e8312749a39cb91a3a5caef6f59bcfcefd32e2ed32b2be"
      }
    }
  ]
}
```

- ❖ Read the objects that were listed in the previous example.

Example

In the following example, the objects listed in **bucket1** are read:

```
$ aws --no-verify-ssl --endpoint-url $S3_ENDPOINT s3api get-object --bucket bucket1 --key
first-object object-read.txt
{
  "AcceptRanges": "bytes",
  "LastModified": "Wed, 22 Feb 2023 00:02:44 GMT",
  "ContentLength": 32,
  "ETag": "\"2706bc01df207bcdcab419685e42919d\"",
  "Metadata": {},
  "TagCount": 0
}
cat object-read.txt
Wed Feb 22 12:00:43 AM UTC 2023

--endpoint-url=https://userx86srv-1.virts.devintel.greenlab.company.com:8111 s3api put-
object \ --bucket bucket1111 --key abcd --body date.txt
```

2.5 Use DDN Infinia Block DataService

NOTE: This section applies primarily to realm, tenant, and subtenant admins; however, users with `<tenant/subtenant/dataservice>:service-user` scope and capability can execute commands exposed in the CSI driver, but no `redcli` or `redui` capabilities.

DDN Infinia supports block dataset types that can be exported using NVMeoF as the transport protocol, but managed using [Container Storage Interface \(CSI\)](#). This support includes more than one block data service associated with any subtenant. For each block service, DDN Infinia creates a service user.

For CSI, this service user is the user specified in the configuration, which can be authenticated by a certificate with extended (or none) expiration date. This user is created with `dataservice:service-user` scope and capability, allowing them to execute commands exposed in the CSI driver, but no `redcli` or `redui` capabilities. For additional details about service user, refer to [User Scope and Capability](#) (Section 1.7 on page 17).

The key parameters for both the dynamically provisioned and pre-provisioned volumes are the specification of tenant, subtenant, and dataset names. With this information, administrators can adjust the quota limit and IO priority for the tenant and subtenant and adjust the size limit for the dataset containing the DDN Infinia block devices exported as NVMF devices.

Prior to using the DDN Infinia block dataservice, setup the CSI driver following the procedure in the *DDN Infinia CSI Driver User Guide*.

For details about managing volumes and managing datasets, see [Manage Volumes](#) (Section 3.5 on page 89) and [Manage Datasets](#) (Section 3.8 on page 101), respectively.

2.6 Setup DDN Infinia Client

NOTE: During the client setup process, you may be asked to run one or more `sudo` commands. After running these commands, logout and log back in to the client node.

If you are running `redcli` commands from remote nodes, you must download and install the `redcli` package using the following steps on each remote node. A *remote node* is any node that is not a DDN Infinia storage node, i.e. not a member of the realm or cluster.

Procedure

1. Download and install `redcli_x.x.x` using the `wget` command, where `x.x.x` specifies the current version.

Example

The following is an example for servers running Ubuntu. Note that the filename `redcli_x.x.x` will vary based on your version.

```
export BUCKET_URL=https://storage.googleapis.com/ddn-redsetup-public/releases
export DISTRO="ubuntu/24.04"
echo ">>> url: $BUCKET_URL"
echo ">>> distro: $DISTRO"
wget "${BUCKET_URL}/${DISTRO}/redcli_1.0.4_${dpkg --print-architecture}.deb" -O /tmp/redcli.deb
sudo apt update -y && sudo apt install -y /tmp/redcli.deb
```

2. Define the environment variable `REDCLI_SERVER` to specify the address of one of the DDN Infinia storage nodes on which the DDN Infinia REST API containers are running.

Typically, this service is running on all nodes, so you can assign `REDCLI_SERVER` to any of the node IP addresses. Alternatively, setup DNS or a load balancer to select one IP from the list of IPs:

```
export REDCLI_SERVER=https://red-node-name-or-ip-address:443/redapi/v1
```

3. Log in to the DDN Infinia API server:

```
redcli user login <username> -p <password>
```

4. Users running `redcli client setup` must be a member of the `red` group. If you are not a member, run the following command. After running the `sudo` command, logout and log back in to the client node.

```
sudo usermod -a -G red <username>
```

5. Run `redcli client setup` to create a `redrc` file that will establish the DDN Infinia client environment:

```
redcli client setup
```

As indicated in the previous step, users running this command must be a member of the `red` group. If you are not a member, you will receive the following warning:

```
WRN User "<username>" who ran command, expected to be in "red" group. Hint: Run 'sudo usermod -a -G red <username>' to add yourself into "red" group. If red client app still returns an error, try to logout and log back in.
FTL Unable to run command further without required permissions.
```

6. Source the file generated in the previous step to complete the client setup.

```
source $HOME/.config/red/redrc
```

When complete, you can run `redcli` commands remotely or run DDN Infinia client utilities such as `nettest` or `nettest-agent` on a client node. For information about `nettest`, see [nettest Utility](#) (Appendix D on page 257).

2.7 Verify Deployment

NOTE: This section applies to realm admins only.

[Deploy DDN Infinia](#) (Section 2.1 on page 26) provides the minimum steps to install and to start a DDN Infinia cluster on your hardware.

Verify the success of each deployment step, following the instructions in these sections:

- [Verify redsetup Daemon](#) (Section 2.7.1 on page 71)
- [Verify redcli Package Installation on Client Nodes](#) (Section 2.7.2 on page 71)
- [Verify Realm Status](#) (Section 2.7.3 on page 71)
- [Verify the DDN Infinia Cluster State](#) (Section 2.7.4 on page 71)
- [Verify Network Connectivity and Measure Performance](#) (Section 2.7.5 on page 72)

Note that the instructions in this section use the DDN Infinia CLI (`redcli`). Add the `--verbose` flag to any command to view the REST commands that are sent to each `redsetup` server.

To run `redcli`, ensure that you log in DDN Infinia following the instructions in [Log In and Log Out using CLI](#) (Section 2.2.1.5 on page 48).

2.7.1 Verify redsetup Daemon

On each DDN Infinia storage node, verify the status of **redsetup** using the **systemctl** command.

Example

```
$ systemctl status redsetup
redsetup.service - RED Deployment Service
   Loaded: loaded (/etc/systemd/system/redsetup.service; enabled; vendor preset: enabled)
   Active: active (running) since Tue 2020-07-28 16:44:37 UTC; 19min ago
     Process: 139971 ExecStartPre=/bin/mkdir -p /var/log/red (code=exited, status=0/SUCCESS)
     Process: 139972 ExecStartPre=/bin/chown syslog:adm /var/log/red (code=exited, status=0/SUCCESS)
     Process: 139973 ExecStartPre=/bin/chmod 755 /var/log/red (code=exited, status=0/SUCCESS)
    Main PID: 139974 (redsetup)
      Tasks: 16 (limit: 38263)
     Memory: 11.7M
    CGroup: /system.slice/redsetup.service
            └─139974 /usr/bin/redsetup -s
```

2.7.2 Verify redcli Package Installation on Client Nodes

After installing the **redsetup** package on the storage nodes or the **redcli** package on the client nodes, verify that the **redcli** executable is installed and executable in the directory **/usr/bin**.

Example

```
$ /usr/bin/redcli -v
redcli version 1.0.0-alpha.1 01bd05e56 aarch64-linux-gnu engineering Sat Jun 24 06:28:04 UTC 2023
```

2.7.3 Verify Realm Status

Query the container status of all the nodes in a realm from any storage or client node that has the **redcli** packages installed using **redcli realm status**. For more details, see [Monitor Realm, Realm Agent, and Node Status](#) (Section 8.2 on page 161).

2.7.4 Verify the DDN Infinia Cluster State

Query the list of instances in your cluster, the status of each Instance, and overall cluster health using **redcli cluster show** command. For more details, see [Show Status and Health of a Cluster](#) (Section 7.6 on page 152).

2.7.5 Verify Network Connectivity and Measure Performance

Execute **nettest** using the **redcli** to validate and characterize the network connectivity and performance between DDN Infinia instances, and optionally, client nodes. Using the **redcli** utility to execute a **nettest** run provides a simple way to verify network state. For a detailed description of using the **nettest** utility, see [nettest Utility](#) (Appendix D on page 257).

Prior to running the test, ensure that you have logged in with **redcli** and, if running on a client node, have executed **redcli client setup** following the instructions in [Setup DDN Infinia Client](#) (Section 2.6 on page 69).

Perform a connectivity verification test to the specified endpoints by using **redcli nettest verify**. The command makes sure the endpoint is online, accessible, and responding to requests.

Example

The following example verifies redsetup.

```
$ redcli nettest verify redsetup
```

| SOURCE | DEST | IP | TYPE | RESULT | ERROR |
|--------|--------|----------------|----------|--------|-------|
| dvred1 | dvred2 | 250.60.28.88 | redsetup | ✓ | |
| | dvred3 | 250.60.236.183 | redsetup | ✓ | |
| | dvred4 | 250.60.206.243 | redsetup | ✓ | |
| dvred2 | dvred1 | 250.60.115.103 | redsetup | ✓ | |
| | dvred3 | 250.60.236.183 | redsetup | ✓ | |
| | dvred4 | 250.60.206.243 | redsetup | ✓ | |
| dvred3 | dvred1 | 250.60.115.103 | redsetup | ✓ | |
| | dvred2 | 250.60.28.88 | redsetup | ✓ | |
| | dvred4 | 250.60.206.243 | redsetup | ✓ | |
| dvred4 | dvred1 | 250.60.115.103 | redsetup | ✓ | |
| | dvred2 | 250.60.28.88 | redsetup | ✓ | |
| | dvred3 | 250.60.236.183 | redsetup | ✓ | |

Note that **redcli nettest verify** is run automatically on the red agents after you create a realm.

Pass data to the endpoint and measure the network performance by using `redcli nettest run`.

Example

The following example runs nettest with test case number 31.

```
$ redcli nettest run -f 31
11:07PM INF Auto Selecting cluster: cluster-gray
11:07PM INF Running tests...
11:07PM INF done.
Test: all clients -> all servers (serial) single-threaded 1m RPC (rdma)
```

| ENDPOINT | HOST | IOPS | IOPS RANGE [min/avg/max] | LATENCY (µs) [min/avg/max] |
|-----------|-------|------|-----------------------------|-------------------------------|
| 1_nettest | user1 | 2863 | [523/715/846] | [1181/1443/1910] |

2.8 Upgrade DDN Infinia

This section applies to realm admins only. During upgrade you cannot configure the realm or cluster. Also, you cannot add or remove nodes from the cluster.

NOTE:

If you need to upgrade the firmware for a specific component included in a server, see [Upgrade Server, Drive, and NIC Firmware](#) (Section 6.1 on page 138).

To upgrade DDN Infinia on a realm, complete the following steps on any node in a cluster:

Procedure

1. Update the cached release metadata from remote server and list the available releases:

```
redcli realm upgrade list -u
```

2. Download the required release bits from remote server to local container image registry:

```
redcli realm upgrade download <release_version>
```

If successful, the command returns a task UUID. Use this task UUID to monitor the progress.

```
redcli task show <task_uuid>
```

Example

The following example shows the progress of download task (with a given UUID) monitored till successful.

```
$ redcli realm upgrade download 1.1.0
6:36PM INF Realm upgrade-download task [c75e762e-3422-4dc0-a6ff-e9ed1cc6628e] has been started
$ redcli task show c75e762e-3422-4dc0-a6ff-e9ed1cc6628e
```

| UUID | NAME | TYPE | DURATION (sec) | STATE | MESSAGE | ERROR |
|--------------------------------------|------------------------------|----------|----------------|----------|-------------------------|-------|
| c75e762e-3422-4dc0-a6ff-e9ed1cc6628e | Realm upgrade task: download | rupgrade | 5 | complete | Close task as completed | |

| STAGE | NAME | TRANSITION TIME | STATE | MESSAGE | ERROR |
|-------|-----------------------------|-------------------------------|----------|---------|-------|
| 0 | STAGE_INIT | 2023-07-25 18:36:36 +0000 UTC | complete | | |
| 1 | STAGE_UPDATE_LOCAL_REGISTRY | 2023-07-25 18:36:41 +0000 UTC | complete | | |
| 3 | STAGE_DONE | 2023-07-25 18:36:41 +0000 UTC | complete | | |

3. Run the **redcli realm upgrade apply** command specifying the release version.

```
redcli realm upgrade apply <release_version>
```

If successful, the command returns a task UUID. Use this task UUID to monitor the progress.

```
redcli task show <task_uuid>
```

Example

The following example shows the progress of the upgrade task (with a given UUID) monitored till successful.

```
$ redcli realm upgrade apply 1.1.0
```

```
4:58PM INF Realm upgrade-apply task [97b1becf-236e-469f-81b6-b7967a6e78a3] has been started
```

```
$ redcli task show 97b1becf-236e-469f-81b6-b7967a6e78a3
```

| UUID | NAME | TYPE | DURATION (sec) | STATE | MESSAGE | ERROR |
|--------------------------------------|------------------------------|----------|-------------------|----------|----------------------------|-------|
| 97b1becf-236e-469f-81b6-b7967a6e78a3 | Realm upgrade task: apply | rupgrade | 80 | complete | Close task as completed | |

| STAGE | NAME | TRANSITION TIME | STATE | MESSAGE | ERROR |
|-------|--|-------------------------------|----------|---------|-------|
| 0 | STAGE_INIT | 2023-07-28 16:58:47 +0000 UTC | complete | | |
| 1 | STAGE_DISTRIBUTE_IMAGES _ACROSS_NODES | 2023-07-28 16:58:52 +0000 UTC | complete | | |
| 2 | STAGE_UPGRADE_SERVICES | 2023-07-28 16:58:52 +0000 UTC | complete | | |
| 3 | STAGE_UPDATE_REALM_CONFIG | 2023-07-28 17:00:07 +0000 UTC | complete | | |
| 4 | STAGE_DONE | 2023-07-28 17:00:07 +0000 UTC | complete | | |

3. Cluster Management

A *cluster configuration* specifies the collection of resources for a DDN Infinia cluster deployment including servers, CATs, network devices, and tuning parameters. DDN Infinia incorporates and utilizes devices and hosts into the system by consuming these configurations given to it by a realm admin. Realm admins can define any number of unique configurations that are saved as part of the DDN Infinia cluster definition. The current configuration in use is called the cluster *runtime configuration*.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, administrators manage the cluster, as follows:

- [Propose New Runtime Configuration](#) (Section 3.1 on page 78)
- [Manage Nodes](#) (Section 3.2 on page 81)
- [Manage Core Affine Storage Targets](#) (Section 3.3 on page 85)
- [Manage Networks](#) (Section 3.4 on page 88)
- [Manage Volumes](#) (Section 3.5 on page 89)
- [Manage Tenant Structure](#) (Section 3.6 on page 93)
- [Manage Subtenant Structure](#) (Section 3.7 on page 97)
- [Manage Datasets](#) (Section 3.8 on page 101)

Key Points

The following are key points about cluster management:

- To be valid, a configuration must have at least one network, one client group, one instance, one pool, and minimum four CATs.
- When creating a tenant, if the primary admin password is not specified, it will be auto-generated.
- When creating a tenant, if no quota is specified, the default quota for that tenant is set to the total capacity of the cluster.
- When creating a subtenant, if no quota is specified, the default quota for that subtenant is set to the parent tenant's quota.
- When creating a tenant, updating a tenant, or reducing pool capacity, the resulting storage quota ratio is calculated as: $\text{ratio} = (\text{sum of all tenant quotas}) / (\text{capacity of cluster})$
- When creating or updating a subtenant, the resulting storage quota ratio is calculated as: $\text{ratio} = (\text{sum of all subtenant quotas}) / (\text{sum of all tenant quotas})$
- When creating a dataset, if no quota is specified, the default quota for that dataset is set to the parent subtenant's quota.
- When creating or updating a dataset, the resulting storage quota ratio is calculated as: $\text{ratio} = (\text{sum of dataset quotas}) / (\text{sum of all subtenant quotas})$
- When creating an S3 bucket, if no quota is specified, the default quota for that bucket is set to the parent subtenant's quota.

- Recommended ratios are less than 3. If the calculated ratio is greater than 3, a warning will be displayed. For an example warning, see the example in [Create a Tenant](#) (Section 3.6.1 on page 94).

NOTE: The instructions in this section use the DDN Infinia CLI (`redcli`); however, you can also use the DDN Infinia GUI or DDN Infinia REST API to configure your DDN Infinia installation. To help explore the DDN Infinia REST API, issue `redcli` commands with the `--verbose` flag, which will log the underlying REST API calls to use as reference or refer to the *DDN Infinia REST API Reference*.

3.1 Propose New Runtime Configuration

NOTE: This section applies to realm admins only.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, realm admins propose runtime configurations, as described in this section.

Example

If you add new disk devices to the system, you must propose a user named configuration that contained all CATs in the current runtime configuration, amended with new CATs for the new devices to incorporate into the system. Likewise, for deleting CATs from the system, you must propose a configuration that contained all CATs from the runtime you wanted to keep, but remove CATs from the proposal that you wanted to remove from system use.

In general, proposing a new runtime configuration requires three steps:

1. Run discovery.
DDN Infinia analyzes the storage nodes that are available for incorporating into a cluster and aggregates the components of each server into the DDN Infinia hardware *inventory*. This process of updating the DDN Infinia inventory is called *discovery*. Prior to modifying your configuration, ensure that the current inventory of possible devices, hosts, and networks is up to date.
2. Generate a cluster configuration.
Initialize your new configuration by auto-creating a cluster configuration template, so you can examine the type of objects that are created within it. If required, you can modify this cluster configuration by selectively adding or deleting hardware or revising parameters to match your needs.
3. Add the new configuration to the collection of DDN Infinia cluster configurations.
Once satisfied that the modified cluster configuration matches the requirements, realm admins must propose it to effect actual change to an operational DDN Infinia cluster. As previously defined, changes to DDN Infinia clusters are managed through proposing changes to the runtime configuration.

To be valid, a configuration must have at least one network, one client group, one instance, one pool, and minimum four CATs.

The command `redcli config` is used to create these configurations from hardware discovery as well as logical configuration changes. For additional `redcli config` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

As a realm admin, propose a new runtime configuration, as described in the following steps:

Procedure

1. Run discovery to update the current hardware inventory:

```
redcli inventory discover
```

2. Generate a cluster configuration:

- a. Initialize your new configuration by creating a cluster configuration template with a name of your choice **<your_new_cfg>**:

```
redcli config init <your_new_cfg>
```

This step generates a file in yaml format **<your_new_cfg>.yaml**.

- b. If required, modify the cluster configuration:
 - i. Edit the **<your_new_cfg>.yaml** file generated after the last step to modify the tuning parameters or to adjust the resources allocated to DDN Infinia. Typically, no changes are required.
 - ii. Update the configuration by passing the changes in a configuration file using the **-y (--config-template-yaml)** option:

```
redcli config update <your_new_cfg> -y <your_new_config>.yaml
```

3. Add the new configuration to the collection of DDN Infinia cluster configurations and propose it as the new cluster runtime, using the **-y (--config-template-yaml)** option and **-p (--propose)** flag, respectively:

```
redcli config create <your_new_cfg> -y <your_new_config>.yaml -p
```

Note that you can add the new configuration and propose it as the cluster runtime in separate steps:

```
redcli config create <your_new_cfg> <your_new_config>.yaml
redcli config select <your_new_cfg>
```

4. Show the status of the cluster and verify the cluster state is running, using the **-s (--status)** flag:

```
redcli cluster show -s
```

For example outputs, see [Show Status and Health of a Cluster](#) (Section 7.6 on page 152).

In addition to the **redcli cluster show** command, you can review the newly-generated cluster configuration using one of more of the following methods:

- View the configurations in your DDN Infinia cluster:

```
redcli config list [flags]
```

Example

```
$ redcli config list
9:15PM INF Auto Selecting cluster: cluster-gray
```

| NAME | #INSTANCES | #CATS |
|--------------|------------|-------|
| auto_config | 3 | 1 |
| expanded_cfg | 4 | 1 |

- Explore the instances included in a cluster configuration:

```
redcli instance list [flags]
```

Example

```
$ redcli instance list
9:31PM INF Auto Selecting cluster: cluster-gray
```

| INSTANCE ID | INSTANCE UUID | HOSTNAME | HOST UUID |
|-------------|--------------------------------------|--------------|--------------------------------------|
| 1 | 78f0b8af-a34a-49e9-9021-6a50357efce6 | 1681779783-1 | e3d64664-dc93-481e-8026-c57f4a2da262 |
| 2 | 8b037558-afb4-4ae0-be00-68d107c1ecc6 | 1681779783-2 | 96d9e1bd-7303-4b30-834a-08b530f54d7b |
| 3 | 102f3646-7fd7-4984-b380-138838dbbfec | 1681779783-3 | a5d349b2-5dc3-462c-8f27-5e4e2333ffda |
| 4 | 2cd35d9b-af3b-4888-80b7-ac3673c7a4a8 | 1681939540 | 9971043f-eef2-438d-9c9e-7f5eaf1a07e9 |

- Explore the CATs included in a cluster configuration:

```
redcli cat list [flags]
```

Example

```
$ redcli cat list
9:28PM INF Auto Selecting cluster: cluster-gray
9:28PM INF Auto Selecting pool: 5a0bc788-b93a-49e4-93da-a4ddb4c55bf
```

| CAT | DEVICE NAME | RAW CAPACITY | HOSTNAME | UUID |
|-----|-----------------------|--------------|-----------------|--------------------------------------|
| 1 | 1681779783-1:/dev/sdb | 129.9 GB | ds-1681779783-1 | bbcbf3f2-2471-4555-8a84-72f73620ebc7 |
| 2 | 1681779783-1:/dev/sdc | 129.9 GB | ds-1681779783-1 | d4539dfb-17ef-49eb-9e5f-1c7aafc1ea6a |
| 3 | 1681779783-1:/dev/sdd | 129.9 GB | ds-1681779783-1 | 64e84f7e-b22f-40ee-8f0d-47b972fdf6ef |
| 4 | 1681779783-1:/dev/sde | 129.9 GB | ds-1681779783-1 | 701859fd-1fd2-4a48-9615-35f5f213a2a5 |
| 5 | 1681779783-2:/dev/sdb | 129.9 GB | ds-1681779783-2 | e4ddcf13-1fe2-48ef-8627-23f5cc316c29 |
| 6 | 1681779783-2:/dev/sdc | 129.9 GB | ds-1681779783-2 | f415b923-27f2-4924-83c1-a6057108d67e |
| 7 | 1681779783-2:/dev/sdd | 129.9 GB | ds-1681779783-2 | 015fa108-cc71-4088-b80a-e7cd89ab81f9 |
| 8 | 1681779783-2:/dev/sde | 129.9 GB | ds-1681779783-2 | a48574a7-cc6e-403e-b6b8-07f0ef300e75 |
| 9 | 1681779783-3:/dev/sdb | 129.9 GB | ds-1681779783-3 | 0f2d4081-a996-4a0c-8de4-15fbfb0567be |
| 10 | 1681779783-3:/dev/sdc | 129.9 GB | ds-1681779783-3 | dfb8d90d-bbd7-4d48-bcfd-0c34aff8eba0 |
| 11 | 1681779783-3:/dev/sdd | 129.9 GB | ds-1681779783-3 | e986e11a-3aa7-4516-95ef-4d86cbba73ae |
| 12 | 1681779783-3:/dev/sde | 129.9 GB | ds-1681779783-3 | 683b2168-e566-44ae-a112-5190d1fd882d |

Total capacity: 1.559 TB

For additional details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

3.2 Manage Nodes

NOTE: This section applies to realm admins only.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, realm admins manage the nodes in a cluster, as follows:

- [Add One or More Nodes to a DDN Infinia Cluster](#) (Section 3.2.1 on page 82)
- [Delete Nodes from a DDN Infinia Cluster](#) (Section 3.2.2 on page 83)
- [Add or Delete a Device in a Node](#) (Section 3.2.3 on page 84)

For a summary of key points about managing nodes, see [Key Points](#) in [Cluster Management](#) (Section 3 on page 76).

3.2.1 Add One or More Nodes to a DDN Infinia Cluster

As a realm admin, expand a DDN Infinia cluster by adding one or more new nodes, as described in the following procedure.

Procedure

1. On the new nodes to be added, install and execute **redsetup**, passing the **-realm-entry-secret** and **--realm-entry-address** options. For more details, see step 1, [Run Setup](#) (Section 2.1.3 on page 36).

Execute the remaining steps in the procedure from a node in the current cluster (excluding the new nodes).

2. Verify the new nodes are listed in the DDN Infinia inventory:

```
redcli inventory show
```

3. Add the node to the realm configuration by generating a new realm configuration and deploy the updated configuration:

```
redcli realm config generate
redcli realm config update -f realm_config.yaml
```

By default a file named **realm_config.yaml** is generated in the current working directory.

4. Propose changes to the runtime configuration to add the nodes to the cluster:

- a. Create an initial cluster configuration template with a name of your choice **<expanded_cfg>**:

```
redcli config init <your_expanded_cfg>
```

This step generates a file in yaml format **<expanded_cfg>.yaml**.

- b. If required, edit the **<expanded_cfg>.yaml** file generated after the last step to modify the tuning parameters or to adjust the resources allocated to DDN Infinia. Typically, no changes are required.
- c. Add the new configuration to the collection of DDN Infinia cluster configurations and propose it as the new cluster runtime, using the **-y (--config-template-yaml)** option and **-p (--propose)** flag, respectively:

```
redcli config create <expanded_cfg> -y <input_config_template_path_and_filename>.yaml -p
```

Note that you can add the new configuration and propose it as the cluster runtime in separate steps:

```
redcli config create <expanded_cfg> -y <input_config_template_path_and_filename>.yaml
redcli config select <expanded_cfg>
```

5. Verify the new nodes are added to the cluster and the cluster state is running, using the **-s (--status)** flag:

```
redcli cluster show -s
```

3.2.2 Delete Nodes from a DDN Infinia Cluster

IMPORTANT:

For important details about minimum cluster size, refer to the latest *DDN Infinia Product Release Notes*.

Only a single node must be deleted at a time and the cluster must be allowed to redistribute data before deleting additional nodes.

As a realm admin, contract a DDN Infinia cluster by deleting nodes, as described in the following procedure.

Procedure

1. Propose changes to the runtime configuration to delete the node from the cluster:

- a. Create an initial cluster configuration template with a name of your choice **<contracted_cfg>**:

```
redcli config init <contracted_cfg>
```

- b. Edit the **<contracted_cfg>.yaml** file generated in earlier step to delete the target node by removing the lines that make up target node stanza.
- c. Add the new configuration to the collection of DDN Infinia cluster configurations and propose it as the new cluster runtime, using the **-y (--config-template-yaml)** option and **-p (--propose)** flag, respectively:

```
redcli config create <contracted_cfg> -y <input_config_template_path_and_filename>.yaml -p
```

Note that you can add the new configuration and propose it as the cluster runtime in separate steps:

```
redcli config create <contracted_cfg> -y <input_config_template_path_and_filename>.yaml
redcli config select <contracted_cfg>
```

2. Show the status of a cluster:

```
redcli cluster show -s
```

Wait until the STATE field in the command output changes from *running (reconfiguring)* to *running* and the deleted node is not shown under HOSTNAME, which indicates that the node is removed from cluster. For an example output, see Section 7.6.1 on page 152.

3. Delete the target node from the realm configuration by specifying its hostname with the **-n (--node)** option and deploy the updated configuration:

```
redcli inventory delete -n <node_hostname>
redcli realm config generate
redcli realm config update -f realm_config.yaml
```

By default a file named **realm_config.yaml** is generated in the current working directory.

4. Verify the new contracted cluster shows the correct number of nodes, using the **-s (--status)** flag:

```
redcli cluster show -s
```

3.2.3 Add or Delete a Device in a Node

IMPORTANT: The device to be added must already be included in the runtime cluster configuration. No restrictions apply to deleting a device.

As a realm admin, add or delete a device in a node, as described in the following procedure.

Procedure

1. If you are adding a device to a node, verify that the new device is listed in the DDN Infinia inventory, from any node that has **redcli** enabled (including the node to which the device is to be added):

```
redcli inventory show
```

2. Propose changes to the runtime configuration to add or delete a device in a node:
 - a. Create an initial cluster configuration template with a name of your choice **<your_new_cfg>**:

```
redcli config init <your_new_cfg>
```

This step generates a file in yaml format **<your_new_cfg>.yaml**.

- b. Edit the **<your_new_cfg>.yaml** file generated after last step to add devices to a node or delete devices from a node.
 - c. Add the new configuration to the collection of DDN Infinia cluster configurations and propose it as the new cluster runtime, using the **-y (--config-template-yaml)** option and **-p (--propose)** flag, respectively:

```
redcli config create <your_new_cfg> -y <input_config_template_path_and_filename>.yaml -p
```

Note that you can add the new configuration and propose it as the cluster runtime in separate steps:

```
redcli config create <your_new_cfg> -y <input_config_template_path_and_filename>.yaml
redcli config select <your_new_cfg>
```

3. Verify that the new configuration is runtime:

```
redcli config show runtime
```

3.3 Manage Core Affine Storage Targets

NOTE: This section applies to realm admins only.

When assigning a device the first time to be used as a CAT, a realm admin must use the udid name to make the association to a DDN Infinia CAT udid. Once the device is in use, DDN Infinia uses the udid to reference it. The device ID (*udid*) is a globally unique device identifier that is assigned by the system to a storage device. DDN Infinia discovery reads the device manufacturer ID and serial number strings from each disk device to create the 60-character udid. Disk device specific attributes are saved in the inventory under the udid associated with the device.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, realm admins manage CATs, as follows:

- [Add One or More CATs to DDN Infinia](#) (Section 3.3.1 on page 86)
- [Delete One or More CATs from DDN Infinia](#) (Section 3.3.2 on page 87)

For instructions to show a list of assigned or unassigned CATs and view details about the CATs, see [Query CAT Health](#) (Section 8.8 on page 178).

For summary of key points about managing CATs, see [Key Points in Cluster Management](#) (Section 3 on page 76).

IMPORTANT:

If new storage devices are to be incorporated for use by the DDN Infinia software, ensure connectivity and readiness of the device before attempting to add them. For instructions about requesting DDN Infinia to reattempt to serve a repaired drive or adding a new drive to replace a failed drive, refer to the *Hardware Installation and Maintenance Guide* for your platform.

3.3.1 Add One or More CATs to DDN Infinia

If new or additional storage devices need to be added for use by the DDN Infinia software, add CATs by proposing a configuration that enumerates them as part of the runtime. You may need to do this task if you are adding a new drive to replace a failed drive.

As a realm admin, add one or more CATs for use by the DDN Infinia software using the following procedure.

Procedure

1. Ensure that the new device is available to incorporate into the cluster:

- a. Run discovery to update the current hardware inventory with the new device:

```
redcli inventory discover
```

- b. Verify the new CAT is listed in the DDN Infinia inventory:

```
redcli inventory show
```

- c. Verify that any device you intend to incorporate as a new CAT is not in use by any other software or other DDN Infinia configuration.

2. Add the CAT to the realm configuration by generating a new realm configuration and deploy the updated configuration:

```
redcli realm config generate
redcli realm config update -f realm_config.yaml
```

By default a file named **realm_config.yaml** is generated in the current working directory.

3. Propose changes to the runtime configuration to add the CAT to the cluster:

- a. Create an initial cluster configuration template with a name of your choice **<expanded_cfg>**:

```
redcli config init <expanded_cfg>
```

This step generates a file in yaml format **<expanded_cfg>.yaml**.

- b. If required, edit the **<expanded_cfg>.yaml** file generated after last step to modify the tuning parameters or to adjust the resources allocated to DDN Infinia. Typically, no changes are required.
- c. Add the new configuration to the collection of DDN Infinia cluster configurations and propose it as the new cluster runtime, using the **-y (--config-template-yaml)** option and **-p (--propose)** flag, respectively:

```
redcli config create <expanded_cfg> -y <input_config_template_path_and_filename>.yaml -p
```

Note that you can add the new configuration and propose it as the cluster runtime in separate steps:

```
redcli config create <expanded_cfg> -y <input_config_template_path_and_filename>.yaml
redcli config select <expanded_cfg>
```

4. Verify the new CAT is added to the cluster and the cluster state is running, using the **-s (--status)** flag:

```
redcli cluster show -s
```

3.3.2 Delete One or More CATs from DDN Infinia

IMPORTANT:

Only a single CAT must be deleted at a time. If more than one CAT needs to be deleted from a DDN Infinia cluster, repeat the procedure. Note that the cluster must be allowed to redistribute data before deleting any additional CATs.

If the number of storage devices used by the DDN Infinia software needs to shrink, delete CATs by proposing a configuration that no longer enumerates them as part of the runtime. You may need to do this task if you are decommissioning dead or unneeded devices.

When creating a tenant, updating a tenant, or reducing pool capacity, the resulting storage quota ratio is calculated as: $\text{ratio} = (\text{sum of all tenant quotas}) / (\text{capacity of cluster})$. Recommended ratios are less than 3. If the calculated ratio is greater than 3, a warning will be displayed.

As a realm admin, delete a CAT used by the DDN Infinia software using the following procedure.

Procedure

1. Propose changes to the runtime configuration to delete the CAT from the cluster:

- a. Create an initial cluster configuration template with a name of your choice **<contracted_cfg>**:

```
redcli config init <contracted_cfg>
```

This step generates a file in yaml format **<contracted_cfg>.yaml**.

- b. Edit the **<contracted_cfg>.yaml** file generated after last step to delete the target CAT by removing the lines that make up target CAT stanza.
- c. Add the new configuration to the collection of DDN Infinia cluster configurations and propose it as the new cluster runtime, using the **-y (--config-template-yaml)** option and **-p (--propose)** flag, respectively:

```
redcli config create <contracted_cfg> -y <input_config_template_path_and_filename>.yaml -p
```

Note that you can add the new configuration and propose it as the cluster runtime in separate steps:

```
redcli config create <contracted_cfg> -y <input_config_template_path_and_filename>.yaml
redcli config select <contracted_cfg>
```

2. Delete the target CAT from the realm configuration by specifying its uuid and deploy the updated configuration:

```
redcli cat delete <cat_uuid>
```

Note that apart from removing the target CAT, the **redcli cat delete** command also updates the realm configuration.

3. Verify the new contracted cluster shows the correct number of CATs, using the **-s (--status)** flag:

```
redcli cluster show -s
```

3.4 Manage Networks

NOTE: This section applies to realm admins only.

During cluster creation, two networks **verbs-auto** and **tcp-auto**, respectively with types **verbs** and **tcp**, are automatically created, and each network interface in the configuration is matched with those two networks types. If all the instances have at least one network interface attached to the verbs network, the tcp network is configured as a fallback network that will be used only if all the connections using the verbs network are down.

Each network has a weight assigned to it, which defines the priority of the network from 0 to 100. The higher the weight, the more likely the network is selected. Zero (0) is a special value to express that the network must be used as fallback if all the connections to primary networks are down, where primary networks are defined as networks with weight higher than 0. By default, all networks are created with a weight of 100.

As a realm admin, update network configurations using **redcli network update**.

For example, modify the weight of a specified network using **redcli network update** by passing the desired network weight with the **-w (--weight)** option. Note that this command will auto-select the cluster name.

```
redcli network update <network_name> -w <network_weight>
```

Example

The following example sets the network weight to 75.

```
$ redcli network update your-network -w 75
11:16PM INF Auto Selecting cluster: cluster-gray
11:16PM INF Auto Selecting config: auto_config
11:16PM INF Current weight: 100
11:16PM INF Setting weight to 75 on network your-network
11:16PM INF Cluster cluster-gray config auto_config validated.
11:16PM INF Cluster cluster-gray config auto_config updated.
11:16PM INF Cluster cluster-gray config auto_config proposed as runtime.
```

For details about additional configurable network parameters, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

For details about listing the current network configurations, see [List Current Network Configurations](#) (Section 7.3 on page 150).

For summary of key points about managing networks, see [Key Points in Cluster Management](#) (Section 3 on page 76).

3.5 Manage Volumes

NOTE: This section applies to realm admins only.

In block datasets, volume is a namespace object. Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, realm admins manage volumes, as described in the following sections:

- [Create Volume](#) (Section 3.5.1 on page 90)
- [List Volumes](#) (Section 3.5.2 on page 90)
- [Show Details of a Volume](#) (Section 3.5.3 on page 91)
- [Resize Volumes](#) (Section 3.5.4 on page 92)

For summary of key points about managing volumes, see [Key Points](#) in [Cluster Management](#) (Section 3 on page 76).

For additional `redcli volume` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

3.5.1 Create Volume

As a realm admin, create a general purpose volume using **redcli volume create** by specifying the volume name and passing the required parameters:

```
redcli volume create <volume_name> [flags]
```

To create a volume with a specified capacity, pass the capacity of block device with the **-C (--capacity)** option:

```
redcli volume create <volume_name> -C <capacity> [flags]
```

Example

The following example creates a general purpose volume in the default dataset (named **red**) with a 10TiB capacity.

```
redcli volume create vol1 -C 10T
```

To create a volume from a selected instance, pass the instance ID to be used for the NVMeoF device with the **-i (-nvmeof-instance-ids)** option:

```
redcli volume create <volume_name> -i <instance_id> [flags]
```

Example

The following example creates a general purpose volume from instance 2 with a 10TiB capacity.

```
redcli volume create vol2 -C 10T -i 2
```

3.5.2 List Volumes

As a realm admin, list all volumes using **redcli volume list**:

```
redcli volume list [flags]
```

By default, the output is in tabular format. To display the output in yaml format, pass the **-o yaml** flag.

Example

The following example lists volumes in the default tabular output format.

```
$ redcli volume list
```

| VOLUME NAME | CAPACITY | EXPORT? |
|-------------|----------|---------|
| vol1 | 10 TiB | |
| vol2 | 10 TiB | |

3.5.3 Show Details of a Volume

As a realm admin, show details about a specified volume using `redcli volume show` and specifying the name of the volume:

```
redcli volume show <volume_name> [flags]
```

By default, the output is in tabular format. To display the output in yaml format, pass the `-o yaml` flag.

Example

The following example shows details for `vol1` in the default tabular output format.

```
$ redcli volume show vol1
11:24PM INF Auto Selecting cluster: cluster-gray
11:24PM INF Auto Selecting tenant: red
11:24PM INF Auto Selecting subtenant: red
11:24PM INF Auto Selecting dataset: red
```

| VOLUME NAME | UUID | NBLOCKS | BLOCK SIZE | CAPACITY | WWN | XATTRS | |
|----------------|--------------------------------------|------------|---------------|----------|------------------|--------|---|
| | | | | | | KEY | V |
| vol1 | 67a50f9c-3223-46d4-94e4-65b71dbf235c | 2684354560 | 4096 | 10 TiB | 27d50001ff615a54 | | |

3.5.4 Resize Volumes

NOTE: Increasing the size of a volume is permitted, but decreasing the size of a volume is not permitted.

As a realm admin, dynamically resize existing DDN Infinia volumes using the `redcli volume resize` command, as described in the following procedure.

Procedure

1. View the volume properties using `redcli volume show` by passing the tenant name for which volume details needs to be seen with the `-t (--tenant)` option and the dataset name for which volume details needs to be seen with the `-d (--dataset)` option:

```
redcli volume show <volume_name> -t <tenant_name> -d <dataset_name>
```

Example

```
$ redcli volume show vol3 -t tenant-orange -d dataset1
4:59PM INF Auto Selecting cluster: cluster-gray
4:59PM INF Auto Selecting subtenant: subtenant-orange
```

| VOLUME NAME | UUID | NBLOCKS | BLOCK SIZE | CAPACITY | WWN |
|----------------|--------------------------------------|---------|---------------|----------|------------------|
| vol3 | c602f9d3-b588-4519-91ab-284754d15fsw | 100000 | 4096 | 409.6 MB | 265c0001ff1eaf01 |

| XATTRS | | EXPORT | |
|--------|-------|--------|-------------|
| KEY | VALUE | TYPE | INSTANCE ID |
| | | | |

2. Run `redcli volume resize`, passing the total number of volume blocks with the `-n (--nblocks)` option:

```
redcli volume resize <volume_name> -n <total_number_of_volume_blocks>
```

Example

In the following example, the command will return the updated size of the volume upon success.

```
redcli volume resize vol3 -t tenant-orange -d dataset1 -n 500000
```

3.6 Manage Tenant Structure

NOTE: This section primarily applies to realm admins; however, tenant admins can manage the tenant for themselves.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, realm and tenant admins manage the tenant structure, as described in the following sections:

- [Create a Tenant](#) (Section 3.6.1 on page 94) (realm admins only)
- [Delete a Tenant](#) (Section 3.6.2 on page 95) (realm admins only)
- [Update Tenant Properties](#) (Section 3.6.3 on page 95) (realm and tenant admins)
- [List Tenants](#) (Section 3.6.4 on page 96) (realm and tenant admins)
- [Show Details of a Tenant](#) (Section 3.6.5 on page 96) (realm and tenant admins)

For a summary of the key points about managing the tenant structure, see [Key Points](#) in [Cluster Management](#) (Section 3 on page 76).

For additional `redcli tenant` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

3.6.1 Create a Tenant

When creating a tenant, the following properties define the tenant:

- Tenant name (**tenant-name**) – Unique (across realm) name for the tenant (required).
- Primary admin user ID (**-u, --primary-admin-user** option) – First user that will manage that tenant (required).
 - ❖ Tenants must have a user specified as a *primary administrator* for the tenant. The *primary administrator* is a special user role, which is a designated administration point for tenant users, subtenants, and data services.
- Primary admin password (**-p, --primary-password** option) – Primary admin password.
 - ❖ When creating a tenant, if the primary admin password is not specified, it will be auto-generated.
- Bulk quota (**-b, --quota** option) – Limit quota for the tenant (optional).
 - ❖ When creating a tenant, if no quota is specified, the default quota for that tenant is set to the total capacity of the cluster.
 - ❖ When creating a tenant, updating a tenant, or reducing pool capacity, the resulting storage quota ratio is calculated as: $\text{ratio} = (\text{sum of all tenant quotas}) / (\text{capacity of cluster})$
 - ❖ Recommended ratios are less than 3. If the calculated ratio is greater than 3, a warning will be displayed.
- IO Priority (**-i, --io-priority** option) - IO priority (optional).

Note that the properties listed are a subset of the available flags and options. For a complete list, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

As a realm admin, create tenants using **redcli tenant create** by specifying the tenant name and passing the required properties. By default, the primary admin has scope and capability set to **tenant:admin**. These users are linked to the local identity provider for the tenant for which they are created.

```
redcli tenant create <tenant_name> -u <primary_admin> [-b <bulk_quota>] [-i <io_priority>] [flags]
```

In addition, the newly created primary admin user is not required to change their password on first login. To require the new user to change their password, pass the **-w (--newpass)** flag.

Example

The first example creates globally unique tenants named **orange**, **yellow**, and **gray** with their respective primary admin users, IO priority, and quota.

```
redcli tenant create tenant-orange -u orange_admin -i high -b 1G
redcli tenant create tenant-yellow -u yellow_admin -i high -b 2G
redcli tenant create tenant-blue -u blue_admin -i low -b 100M
```

The second example shows a case in which the over-provisioning ratio is greater than the recommend value of 3.

```
$ redcli tenant create tenant-green -u green_admin
3:26PM INF Auto Selecting cluster: cluster-gray
3:26PM INF Successfully created tenant: tenant-green
3:26PM WRN WARNING: Overprovisioning ratio of 4.000 is higher than the maximum recommended ratio
of 3. Reduce bulk quota allocations for tenants in realm to approximately 974.1 GB.
```

3.6.2 Delete a Tenant

IMPORTANT: In DDN Infinia 1.1, deleting a tenant also deletes the contained resources, including datasets. To delete a tenant, you must have the appropriate realm capability, and the action requires the admin to explicitly acknowledge the deletion of the resources.

As a realm admin, delete a tenant using **redcli tenant delete** and specifying the name of the tenant. Note tenants must be empty to be deleted. In other words, delete all subtenants prior to deleting a tenant:

```
redcli tenant delete <tenant_name> [flags]
```

3.6.3 Update Tenant Properties

Realm and tenant admins can update tenant properties, which include IO priority, quota, and xattrs. Note that tenant admins can update properties for their own tenant only.

Modify quota limits on a tenant using **redcli tenant update** by passing the bulk quota with the **-b (--quota)** option:

```
redcli tenant update <tenant_name> -b <bulk_quota> [flags]
```

When creating a tenant, updating a tenant, or reducing pool capacity, the resulting storage quota ratio is calculated as: $\text{ratio} = (\text{sum of all tenant quotas}) / (\text{capacity of cluster})$. Recommended ratios are less than 3. If the calculated ratio is greater than 3, a warning will be displayed.

Set the IO relative priority of tenants using **redcli tenant update** by passing the IO priority with the **-i (--io-priority)** option.

```
redcli tenant update <tenant_name> -i <io_priority> [flags]
```

3.6.4 List Tenants

As a realm or tenant admin, list tenants using **redcli tenant list**. The output displays name, quota, usage, provisioning ratio, IO priority, and primary admin for the tenants. Note that tenant admins can list their own tenant only.

```
redcli tenant list [flags]
```

Example

This example lists all tenants in the **cluster-gray** cluster.

```
$ redcli tenant list
6:17AM INF Auto Selecting cluster: cluster-gray
```

| NAME | QUOTA | USAGE | PROVISIONING RATIO | IO PRIORITY | PRIMARY ADMIN |
|---------|--------|-----------------|-----------------------|-------------|---------------|
| tenant1 | 250 GB | 31.33 GB (13%) | 2.40 | medium | TASTAtenant1 |
| tenant2 | 250 GB | 62.65 GB (25%) | 2.40 | medium | TASTAtenant2 |
| tenant3 | 250 GB | 93.97 GB (38%) | 2.40 | medium | TASTAtenant3 |
| tenant4 | 250 GB | 125.3 GB (50%) | 2.40 | medium | TASTAtenant4 |
| tenant5 | 250 GB | 156.6 GB (63%) | 2.40 | medium | TASTAtenant5 |
| tenant6 | 250 GB | 187.9 GB (75%) | 2.40 | medium | TASTAtenant6 |
| tenant7 | 250 GB | 219.3 GB (88%) | 2.40 | medium | TASTAtenant7 |
| tenant8 | 250 GB | 250.6 GB (100%) | 2.40 | medium | TASTAtenant8 |
| tenant9 | 250 GB | 37.69 GB (15%) | 2.40 | medium | TASTAtenant9 |

3.6.5 Show Details of a Tenant

As a realm or tenant admin, show details of a single tenant using **redcli tenant show** and specifying the name of the tenant. Note that tenant admins can show details of their own tenant only.

```
redcli tenant show <tenant_name> [flags]
```

Example

This example shows details for the **red** tenant.

```
$ redcli tenant show red
3:42PM INF Auto Selecting cluster: cluster-gray
```

| TENANT NAME | TENANT ID | IO PRIORITY | BULK QUOTA | USAGE | PRIMARY ADMIN | XATTRS | |
|-------------|-----------|-------------|------------|-------|---------------|--------|-------|
| | | | | | | KEY | VALUE |
| red | 17 | 0 | 324.7 GB | 0 B | redadmin | -- | -- |

3.7 Manage Subtenant Structure

NOTE: This section primarily applies to tenant admins; however, subtenant admins can list and show details of their own subtenant only.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, tenant admins manage the subtenant structure, as described in the following sections:

- [Create Subtenant](#) (Section 3.7.1 on page 98)
- [Delete Subtenant](#) (Section 3.7.2 on page 99)
- [Update Subtenant Properties](#) (Section 3.7.3 on page 99)
- [List Subtenants](#) (Section 3.7.4 on page 100)
- [Show Details of a Subtenant](#) (Section 3.7.5 on page 100)

For a summary of the key points about managing the subtenant structure, see [Key Points](#) in [Cluster Management](#) (Section 3 on page 76).

For additional `redcli subtenant` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

3.7.1 Create Subtenant

If required, a tenant admin can provision tenant resources into subtenants. When creating a subtenant, the following properties define the subtenant:

- Subtenant name (**subtenant-name**) – Unique (across tenant) name for the subtenant (required).
- Admin groups or users (**-a, --admins** option) – Comma-separated list of admin groups or users that will manage the subtenant (optional).
 - ❖ Optionally, subtenants can have a user specified as a *primary administrator* for the subtenant. The *primary administrator* is a special user role, which is a designated administration point for tenant users, subtenants, and data services.
- Bulk quota (**-b, --quota** option) – Limit quota for the subtenant (optional).
 - ❖ When creating a subtenant, if no quota is specified, the default quota for that subtenant is set to the parent tenant's quota.
 - ❖ When creating or updating a subtenant, the resulting storage quota ratio is calculated as: $\text{ratio} = (\text{sum of all subtenant quotas}) / (\text{sum of all tenant quotas})$
 - ❖ Recommended ratios are less than 3. If the calculated ratio is greater than 3, a warning will be displayed.
- IO Priority (**-p, --priority** option) - IO priority (optional).

Note that the properties listed are a subset of the available flags and options. For a complete list, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

As a tenant admin, create subtenants using **redcli subtenant create** by specifying the subtenant name and passing the required properties:

```
redcli subtenant create <subtenant_name> [-b <bulk_quota>] [-p <io_priority>] [flags]
```

Example

The following example creates the subtenant named **ops** in the tenant named **tenant-orange** with a quota of 1GB.

```
redcli user login orange_admin -p oRangeSecretWd5
redcli subtenant create ops -p med -b 1g
```

Optionally, a tenant admin can create a subtenant and specify the admin groups or users that are granted to the subtenant scope by passing these groups or users separated by a comma with the **-a (--admins)** option. If not specified, the default password is used. The subtenants are assigned within the scope of the logged-in admin, which is auto-selected.

```
redcli subtenant create <subtenant_name> -a <group_or_users> [flags]
```

Example

The following example creates the subtenants named **engr**, **mktg**, and **acct** in the **red** tenant (Figure 4 on page 38) and assigns specified groups and users to the respective subtenant scope.

```
redcli user login redadmin -p redSecrEtwoRd4D
redcli subtenant create engr -a EngrAdmins -b 2P -p high
redcli subtenant create mktg -a john@ad.red.ddn.com,jane@red.ddn.com -b 500T -p low
redcli subtenant create acct -a AcctAdmins -v sam@ldap.red.ddn.com -b 500T -p med
```

3.7.2 Delete Subtenant

IMPORTANT: In DDN Infinia 1.1, deleting a subtenant also deletes the contained resources, including datasets. To delete a subtenant, you must have the appropriate tenant capability, and the action requires the admin to explicitly acknowledge the deletion of the resources.

As a tenant admin, delete a subtenant using **redcli subtenant delete** and specifying the name of the subtenant:

```
redcli subtenant delete <subtenant_name> [flags]
```

3.7.3 Update Subtenant Properties

Tenant admins can update subtenant properties, which include IO priority, quota and xattrs.

Modify quota limits on the subtenant using **redcli subtenant update** by passing the bulk quota with the **-b** (**-quota**) option:

```
redcli subtenant update <subtenant_name> -b <bulk_quota> [flags]
```

When creating or updating a subtenant, the resulting storage quota ratio is calculated as: $\text{ratio} = (\text{sum of all subtenant quotas}) / (\text{sum of all tenant quotas})$. Recommended ratios are less than 3. If the calculated ratio is greater than 3, a warning will be displayed.

Set the IO relative priority subtenant using **redcli subtenant update** by passing the IO priority with the **-p** (**--priority**) option:

```
redcli subtenant update <subtenant_name> -p <io_priority> [flags]
```

3.7.4 List Subtenants

As a tenant or subtenant admin, list subtenants using **redcli subtenant list**. The output displays name, quota, usage, provisioning ratio, IO priority, and primary admin for the subtenants. Note that tenant admins can list their own subtenants only.

```
redcli subtenant list [flags]
```

Example

This example lists all subtenants in **tenant6**.

```
$ redcli subtenant list
6:18AM INF Auto Selecting cluster: cluster-gray
6:18AM INF Auto Selecting tenant: tenant6
```

| NAME | QUOTA | USAGE | PROVISIONING RATIO | IO PRIORITY | PRIMARY ADMIN |
|-------------|-------|----------------|-----------------------|-------------|------------------|
| subtenant1 | 60 GB | 18.79 GB (31%) | 1.50 | medium | subtenant1admin |
| subtenant2 | 60 GB | 18.79 GB (31%) | 1.50 | medium | subtenant2admin |
| subtenant3 | 60 GB | 18.79 GB (31%) | 1.50 | medium | subtenant3admin |
| subtenant4 | 60 GB | 18.79 GB (31%) | 1.50 | medium | subtenant4admin |
| subtenant5 | 60 GB | 18.79 GB (31%) | 1.50 | medium | subtenant5admin |
| subtenant6 | 60 GB | 18.79 GB (31%) | 1.50 | medium | subtenant6admin |
| subtenant7 | 60 GB | 18.79 GB (31%) | 1.50 | medium | subtenant7admin |
| subtenant8 | 60 GB | 18.79 GB (31%) | 1.50 | medium | subtenant8admin |
| subtenant9 | 60 GB | 18.79 GB (31%) | 1.50 | medium | subtenant9admin |
| subtenant10 | 60 GB | 18.79 GB (31%) | 1.50 | medium | subtenant10admin |

3.7.5 Show Details of a Subtenant

As a tenant or subtenant admin, show details of a single subtenant using **redcli subtenant show** and specifying the name of the subtenant. Note that tenant admins can show details of their own subtenants only.

```
redcli subtenant show <tenant_name> [flags]
```

3.8 Manage Datasets

NOTE: This section applies to subtenant admins only.

In DDN Infinia, a dataset is the store data building block. Everything that stores data must have a dataset defined. This dataset can be visible through S3 protocol if the particular attribute/xatts is set to be a S3 bucket. Therefore, everything is a dataset. but it may not be a S3 bucket. For additional details about the S3 protocol, see [Use DDN Infinia S3 DataService](#) (Section 2.4 on page 59).

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, subtenant admins manage datasets, as described in the following sections:

- [Create Dataset](#) (Section 3.8.1 on page 102)
- [Update Dataset Properties](#) (Section 3.8.2 on page 102)
- [List Datasets](#) (Section 3.8.3 on page 103)
- [Show Details of a Dataset](#) (Section 3.8.4 on page 104)
- [Manage Dataset Profiles](#) (Section 3.8.5 on page 105)

For details about modifying the quota limits and IO relative priority of the parent tenant or subtenant, see [Update Tenant Properties](#) (Section 3.6.3 on page 95) and [Update Subtenant Properties](#) (Section 3.7.3 on page 99), respectively.

For summary of key points about managing datasets, see [Key Points](#) in [Cluster Management](#) (Section 3 on page 76).

For additional `redcli dataset` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

3.8.1 Create Dataset

Datasets are either created when creating a data service or a bucket. For details, see [Create a Block Data Service](#) (Section 5.2.1 on page 129) and [Create Buckets](#) (Section 5.3.1 on page 134), respectively.

3.8.2 Update Dataset Properties

As a subtenant admin, update the properties of the underlying DDN Infinia dataset:

```
redcli dataset update <dataset_name> [flags]
```

For example, change the size of a specified dataset using **redcli dataset update** by passing the dataset size with the **-b (--quota)** option.

When creating or updating a dataset, the resulting storage quota ratio is calculated as: $\text{ratio} = (\text{sum of dataset quotas}) / (\text{sum of all subtenant quotas})$. Recommended ratios are less than 3. If the calculated ratio is greater than 3, a warning will be displayed.

```
redcli dataset update <dataset_name> -b <dataset_size> [flags]
```

Example

In the following example, the quota for **bucket1** is set to 2TiB. Note that only datasets with the special attribute **xatts** set to be a S3 bucket are included in the output.

```
$ redcli dataset update bucket1 -b 2T
$ redcli s3 bucket list -o json
{
  "data": {
    "Buckets": [
      {
        "CreationDate": "2023-02-21T23:12:50Z",
        "Name": "bucket1",
        "Quota": 2199023255552,
        "Usage": -1
      },
      {
        "CreationDate": "2023-02-21T23:13:24Z",
        "Name": "bucket2",
        "Quota": 2199023255552,
        "Usage": -1
      }
    ],
    "Owner": {
      "DisplayName": "subtenant_admin",
      "ID": "3f6bda07efaa463fb6e8312749a39cb91a3a5caef6f59bcfced32e2ed32b2be"
    },
    "ResultMetadata": {}
  },
  "detail": "",
  "epoch": 1677024803,
  "timestamp": "Wed, 22 Feb 2023 00:13:23 GMT",
  "title": "List s3 buckets"
}
```

3.8.3 List Datasets

As a subtenant admin, list datasets to which the logged-in admin has access using **redcli dataset list**. Note that this command lists all datasets, even those with the special attribute/xatts set to be a S3 bucket.

```
redcli dataset list [flags]
```

By default, the output is in tabular format. To display the output in yaml format, pass the **-o yaml** flag.

Example

The following example lists all datasets for the subtenant **red/red**. Note that datasets with the special attribute/xatts set to be a S3 bucket are included in the output also.

```
$ redcli dataset list -t red -s red
11:06AM INF Auto Selecting cluster: cluster-gray
```

| DATASET LIST |
|---------------|
| bucket1 |
| bucket2 |
| user1bucket1 |
| user1bucket10 |
| user1bucket11 |
| user1bucket12 |
| user1bucket2 |
| user1bucket3 |
| user1bucket4 |
| user1bucket5 |
| user1bucket6 |
| user1bucket7 |
| user1bucket8 |
| user1bucket9 |
| red |
| user2bucket1 |
| user2bucket4 |
| user3bucket1 |
| user3bucket2 |
| user3bucket3 |
| user3bucket4 |

3.8.4 Show Details of a Dataset

As a subtenant admin, show details of a datasets to which the logged-in admin has access using **redcli dataset show** and specifying the name of the dataset.

```
redcli dataset show <dataset_name> [flags]
```


3.8.5 Manage Dataset Profiles

DDN Infinia widely stripes data across all available Instances and CATs in the DDN Infinia cluster. With default data redundancy schemes, the distributed placement ensures that the loss of one or more CATs and/or the instances hosting those CATs, up to the specified resiliency, does not result in loss of data.

Example

In a properly configured cluster with three instances, using the 3-parity EC encoding, you may lose up to two storage devices and still maintain access to your data without loss.

DDN Infinia automatically manages data placement using a default system data placement profile. This section provides instructions to list these profiles and show their details:

- [List Dataset Profiles](#) (Section 3.8.5.1 on page 105)
- [Show the Dataset Profile Details](#) (Section 3.8.5.2 on page 105)

IMPORTANT: The default EC schema is 3-parity for the cluster, which is created as part of the policy during cluster configuration creation. This schema cannot be changed in DDN Infinia 1.1.

3.8.5.1 List Dataset Profiles

As a subtenant admin, list dataset profiles using **redcli dataset profile list**:

```
redcli dataset profile list
```

3.8.5.2 Show the Dataset Profile Details

As a subtenant admin, show the dataset profile for a data service using **redcli dataset profile show** and specifying the profile name:

```
redcli dataset profile show <profile_name>
```

Example

The following example shows details for the **SYSTEM_DATA** profile.

```
$ redcli dataset profile show SYSTEM_DATA
9:03PM INF Auto Selecting cluster: cluster-gray
9:03PM INF Auto Selected config: auto_config
```

| ID | NAME | Metapool ID | Meta LTID | Datapool ID | Data LTID | Efficiency | Protection |
|----|-------------|-------------|-----------|-------------|-----------|------------|------------|
| 1 | SYSTEM_DATA | 1 | 2 | 1 | 3 | 87 | 11 |

| Availability | MinStripe | MaxStripe |
|--------------|-----------|-----------|
| 8 | 32768 | 262144 |

Table 5 describes the fields provided in the command output.

Table 5. Show the Dataset Profile Details

| Field | Description |
|--------------|---|
| ID | Identifier of the profile whose details are shown |
| Name | Name of profile |
| Metapool ID | Identifier of the pool on which metadata will be stored |
| Meta LTID | Identifier of the profile to be used on the metadata |
| Datapool ID | Identifier of the pool on which data will be stored |
| Data LTID | Identifier of the profile to be used on the data |
| Efficiency | Default maximum efficiency of the Erasure Parity Schema |
| Protection | Level of data endurance |
| Availability | Which failure data will survive, for example, device, instance, rack, or site |
| MinStripe | Minimum data targeted per device - 32768 bytes |
| MaxStripe | Maximum data targeted per device in bytes |

4. User Management and Authorization

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, administrators manage users and authorization, as follows:

- Realm admins manage users, groups, and identity providers for the realm:
 - ❖ [Manage Realm Users and Groups](#) (Section 4.1 on page 108)
 - ❖ [Manage Realm Identity Providers](#) (Section 4.4 on page 115)
- Tenant admins manage users, groups, and identity providers for the tenant:
 - ❖ [Manage Tenant or Subtenant Users and Groups](#) (Section 4.2 on page 110)
 - ❖ [Manage Tenant Identity Providers](#) (Section 4.5 on page 117)
- Admins with `red/[a11]:admin` scope and capability manage multi-tenant users:
 - ❖ [Manage Multi-tenant Users and Groups](#) (Section 4.3 on page 113)
- Realm, tenant, and subtenant admins generate client certificates for authentication and authorization:
 - ❖ [Use Client Certificate for Authentication](#) (Section 4.6 on page 119)
- [Password Management Using Local Authentication](#) (Section 4.7 on page 123)

Key Points

The following are key points about user management and authorization:

- [User Scope and Capability](#) (Section 1.7 on page 17) defines DDN Infinia's user model. Review the information in this section prior to managing users.
- The realm must have a `realm_admin` user; while tenants must have a user specified as a *primary administrator* for the tenant.
 - ❖ The *primary administrator* is a special user role, which is a designated administration point for tenant users, subtenants, and data services.
 - ❖ This *primary administrator* user is optional for subtenants.
- Administrators can assign capabilities to users by either of the following methods:
 - ❖ Capabilities are assigned directly to the user when added or later granted to the user.
 - ❖ Capabilities are assigned indirectly from groups to which the user is a member.

The best practice is to create groups, assign scope and capability to the groups, and assign users as members of those groups.
- If multiple tenant or subtenant scopes are defined for a user, the first applicable tenant or subtenant is selected. However, the user could be requested to provide the tenant and subtenant explicitly when selection could cause confusion.
- Administrators cannot remove themselves. The `realm_admin` user cannot be removed and admin capability cannot be revoked. For all other users, user records will be deactivated when a user is removed; however, the user record remains in the DDN Infinia system. As soon as the deactivated flag is set, no operations are allowed for this user, regardless of whether tokens exist. The deactivated user can not login, but the token could be valid up to 10 minutes until the next revalidation. The S3 access key/secret could be valid up to 5 minutes from deactivation.

4.1 Manage Realm Users and Groups

NOTE: This section applies to realm admins only.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, realm admins manage users and groups with realm scope, as described in the following sections:

- [Add Realm Users](#) (Section 4.1.1 on page 109)
- [Add Realm User Groups](#) (Section 4.1.2 on page 109)
- [Update Realm User and Group Capabilities](#) (Section 4.1.3 on page 109)

Administrators can assign capabilities to users by either of the following methods:

- Capabilities are assigned directly to the user when added or later granted to the user.
- Capabilities are assigned indirectly from groups to which the user is a member.

The best practice is to create groups, assign scope and capability to the groups, and assign users as members of those groups. When a group is created with realm scope, realm admins specify the capability of the group.

For summary of key points about managing users and groups, see [Key Points](#) in [User Management and Authorization](#) (Section 4 on page 107).

For additional `redcli` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

4.1.1 Add Realm Users

As a realm admin, add a new realm user using **redcli user add** by passing the user **scope:capability** with the **-r (--scope)** option. If a user is created with no **scope:capability** defined, no default **scope:capability** is applied. When only scope is provided, the capability defaults to **admin**. When only capability is provided, the scope defaults to the scope of the current admin. In this case if only capability is provided, the scope defaults to **realm**.

```
redcli user add <user_id> -p <password> -r <scope:capability> [flags]
```

By default, the new user is not required to change their password on first login. To require the new user to change their password, pass the **-w (--newpass)** flag.

Example

In the following example, a realm admin (**realm_admin**) logs in and adds a realm admin user (**new_realm_admin**) and a realm viewer user (**realm_viewer**).

```
redcli user login realm_admin -p rEalmsEcretwd4U
redcli user add new_realm_admin -p tmPaDminSecret8 -r admin
redcli user add realm_viewer -p temPvieWersEcr2 -r viewer
```

4.1.2 Add Realm User Groups

As a realm admin, add realm user groups using **redcli user group add**. Users from a group have the scope and capability of the specified group.

```
redcli user group add <group_id> -r <scope:capability> [flags]
```

4.1.3 Update Realm User and Group Capabilities

Realm admins grant capabilities to realm groups or individual realm users when they are created. The best practice is to create groups, assign scope and capability to the groups, and assign users as members of those groups. Note that admin users can grant capabilities to themselves.

Update realm groups to add or remove capabilities using **redcli user group update** by passing the group scope and capability with the **-r (--scope)** option. Specify multiple scopes and capabilities in comma separated form.

```
redcli user group update <group_id> -r <scope:capability> [flags]
```

Optionally, realm admins can grant capabilities to individual realm users:

```
redcli user grant <user_id> -r <scope:capability> [flags]
```

4.2 Manage Tenant or Subtenant Users and Groups

NOTE: This section applies to tenant admins only.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, tenant admins manage users and groups with tenant and subtenant scopes, as described in the following sections:

- [Add Tenant or Subtenant Users](#) (Section 4.2.1 on page 111)
- [Add Tenant or Subtenant User Groups](#) (Section 4.2.2 on page 111)
- [Update Tenant or Subtenant User and Group Capabilities](#) (Section 4.2.3 on page 111)
- [Define Equivalence Rule](#) (Section 4.2.4 on page 112)

Administrators can assign capabilities to users by either of the following methods:

- Capabilities are assigned directly to the user when added or later granted to the user.
- Capabilities are assigned indirectly from groups to which the user is a member.

The best practice is to create groups, assign scope and capability to the groups, and assign users as members of those groups. When a group is created with tenant or subtenant scope, tenant admins specify the capability of the group.

For summary of key points about managing users and groups, see [Key Points](#) in [User Management and Authorization](#) (Section 4 on page 107).

For additional `redcli` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

4.2.1 Add Tenant or Subtenant Users

As a tenant admin, add a new tenant or subtenant user using **redcli user add** by passing the user **scope:capability** with the **-r (--scope)** option. If a user is created with no **scope:capability** defined, no default **scope:capability** is applied. When only scope is provided, the capability defaults to **admin**. When only capability is provided, the scope defaults to the scope of the current admin.

```
redcli user add <user_id> -p <password> -r <scope:capability> [flags]
```

By default, the new user is not required to change their password on first login. To require the new user to change their password, pass the **-w (--newpass)** flag.

Example

In the following example, a **red** tenant admin (**redadmin**) logs in and adds a new tenant admin user (**redadmin1**) and a tenant viewer user (**tenant_viewer**).

```
redcli user login redadmin -p reDsecrEtwoRd4D -t red
redcli user add redadmin1 -p tmPaDminSecret8 -r admin
redcli user add tenant_viewer -p temPvieWersEcr3 -r viewer
```

4.2.2 Add Tenant or Subtenant User Groups

As a tenant admin, add tenant or subtenant user groups using **redcli user group add**. Users from a group have the scope and capability of the specified group.

```
redcli user group add <group_id> -r <scope:capability> [flags]
```

Examples

In the first example, a **red** tenant admin (**redadmin**) creates the **RedAdmins** group with **admin** capability and **RedViewers** group with **viewer** capability within the **red** tenant, respectively.

```
redcli user login redadmin -p reDsecrEtwoRd4D
redcli user group add RedAdmins -r red:admin
redcli user group add RedViewers -r red:viewer
```

In the second example, a tenant admin creates a group of users with data access to the **red/red/redobj** data service and then the tenant admin adds an individual user to the **s3users** group.

```
redcli user group add s3users -r red/red/redobj:data-access
redcli user add user1 -p uSerlseCretPwrD -g s3users
```

4.2.3 Update Tenant or Subtenant User and Group Capabilities

Tenant admins grant capabilities to groups or individual users for their own tenant or subtenants. The best practice is to create groups, assign scope and capability to the groups, and assign users as members of those groups. Note that admin users can grant capabilities to themselves.

Update tenant or subtenant groups to add or remove capabilities using **redcli user group update** by passing the group scope and capability with the **-r (--scope)** option. Specify multiple scopes and capabilities in comma separated form.

```
redcli user group update <group_id> -r <scope:capability> [flags]
```

Optionally, tenant admins can grant capabilities to individual tenant or subtenant users:

```
redcli user grant <user_id> -r <scope:capability> [flags]
```

4.2.4 Define Equivalence Rule

Optionally, tenant admins can define equivalence rules.

Example

Assume that an admin with **red** tenant scope defines an equivalence rule where **johnsmith@red.ldap.com** maps to **john_smith@red.ad.com**, so that logins as **johnsmith@red.ldap.com/<ldap_provider_pwd>** maps to the same user as **john_smith@red.ad.com**.

In this case, the following two commands would show the same output:

```
redcli user show john_smith@red.ad.com
redcli user show johnsmith@red.ldap.com
```


4.3 Manage Multi-tenant Users and Groups

NOTE: This section applies to realm and tenant admins only.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, admins with `red/ [a11]` scope manage multi-tenant users and groups, as described in the following sections:

- [Add Multi-tenant Users](#) (Section 4.3.1 on page 114)

For a summary of key points about managing users and groups, see [Key Points](#) in [User Management and Authorization](#) (Section 4 on page 107).

For additional `redcli` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

4.3.1 Add Multi-tenant Users

To add multi-tenant users, realm or tenant admins must first grant themselves **red/[all]:admin** and then add users with the specified **scope:capability**.

Procedure

1. As realm or tenant admin, grant yourself **red/[all]:admin** scope and capability:

```
redcli user grant <admin> red/[all]:admin
```

2. Add multi-tenant users using **redcli user add** by passing multiple scopes and capabilities for the user with the **-r (--scope)** option.

```
redcli user add <multi-tenant_user> -r <scope:capability,scope:capability,...>
```

Examples

In the first example, **realm_admin** grants them self **red/[all]:admin** scope and capability. Then they create **new_admin** with **admin** capability for **red/[all]** and **new-user** with **viewer** capability for **tenant-yellow** and subtenant **tenant-orange/ops**, respectively.

```
redcli user login realm_admin -p rEalmsEcretwd4U
redcli user grant realm_admin red/[all]:admin
redcli user add new_admin -r red/[all]:admin
redcli user add new-user -r tenant-yellow:viewer,tenant-orange/ops:viewer
```

In the second example, a **red** tenant admin (**redadmin**) grants them self access to all subtenants (**red/[all]**) and creates three tenant admins. Tenant admins **redadmin2** and **redadmin3** also have subtenant scope.

```
redcli user login redadmin -p reDsecrEtwoRd4D
redcli user grant redadmin red/[all]:admin
redcli user add redadmin1 -r red
redcli user add redadmin2 -r red,red/engr
redcli user add redadmin3 -r red,red/mktg,red/acct
```

4.4 Manage Realm Identity Providers

NOTE: This section applies to realm admins only.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, realm admins manage identity providers for the realm, as described in the following sections:

- [Create External Identity Provider for Realm](#) (Section 4.4.1 on page 116)
- [Create Group-based Identity Provider for Realm](#) (Section 4.4.2 on page 116)

For summary of key points about managing identity providers, see [Key Points](#) in [User Management and Authorization](#) (Section 4 on page 107).

For additional `redcli` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

4.4.1 Create External Identity Provider for Realm

NOTE: The following sections describe how to authenticate using the DDN Infinia CLI (**redcli**).

As a realm admin, create an external identify provider for the realm using **redcli identity create** and passing **ldap** or **ad** with the required parameters. Note that the configuration file is in yaml format, and the command auto-selects the scope of the logged-in admin.

For AD:

```
redcli identity create <identity_id> ad -f <identity_config>.yaml [flags]
```

For LDAP:

```
redcli identity create <identity_id> ldap -f <identity_config>.yaml [flags]
```

4.4.2 Create Group-based Identity Provider for Realm

As a realm admin, create a group-based identity provider for the realm using **redcli identity create** by passing the group name and description with the **-g** (**--group**) and **-a** (**--about**) flags, respectively.

For AD:

```
redcli identity create <identity_id> ad -g <group> -a <descr> -f <identity_config>.yaml [flags]
```

For LDAP:

```
redcli identity create <identity_id> ldap -g <group> -a <descr> -f <identity_config>.yaml [flags]
```

If the specified group does not exist, it will be created automatically. When a user logs in using an ID that authenticates against the server specified by **<identify_id>**, the user will automatically be added to the group specified by **<group>** and gain the scope and capabilities defined for that group.

4.5 Manage Tenant Identity Providers

NOTE: This section applies to tenant admins only.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, realm admins manage identity providers for the tenant, as described in the following sections:

- [Create External Identity Provider for Tenant](#) (Section 4.5.1 on page 118)
- [Create Group-based Identity Provider for Tenant](#) (Section 4.5.2 on page 118)

For a summary of key points about managing identity providers, see [Key Points](#) in [User Management and Authorization](#) (Section 4 on page 107).

For additional `redcli` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

4.5.1 Create External Identity Provider for Tenant

NOTE: The following sections describe how to authenticate using the DDN Infinia CLI (**redcli**).

As a tenant admin, create an external identity provider for the tenant using **redcli identity create** and passing **ldap** or **ad** with the required parameters. Note that the configuration file is in yaml format, and the command auto-selects the scope of the logged-in admin.

For AD:

```
redcli identity create <identity_id> ad -f <identity_config>.yaml [flags]
```

For LDAP:

```
redcli identity create <identity_id> ldap -f <identity_config>.yaml [flags]
```

Example

In the following example, the **red** tenant primary admin logs in and creates two external identity providers with domains of **ad.red.ddn.com** and **ldap.red.ddn.com**, respectively.

```
redcli user login redadmin -p reDsecrEtwoRd4D
redcli identity create red_ad_server ad -f ad_cfg.yaml
redcli identity create red_ldap_server ldap -f ldap_cfg.yaml
```

4.5.2 Create Group-based Identity Provider for Tenant

As a tenant admin, create a group-based identity provider for the tenant using **redcli identity create** by passing the group name and description with the **-g (--group)** and **-a (--about)** options, respectively.

For AD:

```
redcli identity create <identity_id> ad -g <group> -a <descr> -f <identity_config>.yaml [flags]
```

For LDAP:

```
redcli identity create <identity_id> ldap -g <group> -a <descr> -f <identity_config>.yaml [flags]
```

If the specified group does not exist, it will be created automatically. When a user logs in using an ID that authenticates against the server specified by **<identity_id>**, the user will automatically be added to the group specified by **<group>** and gain the scope and capabilities defined for that group.

Example

In the following example, the **red** tenant primary admin logs in and creates two group-based identity providers for the **EngrAdmins** and **AcctAdmins**, respectively.

```
redcli user login redadmin -p reDsecrEtwoRd4D
redcli subtenant create engr
redcli user group add EngrAdmins -r red/engr
redcli identity create engr_ldap_server ldap -g EngrAdmins -a Engineering -f ldap_engr_cfg.yaml
redcli user group add AcctAdmins -r red/acct
redcli identity create acct_ldap_server ldap -g AcctAdmins -a Accounting -f ldap_acct_cfg.yaml
```

4.6 Use Client Certificate for Authentication

NOTE: This section applies to realm, tenant, and subtenant admins. DDN Infinia 1.0 supports client certificates for **redcli** only.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, admins can generate client certificates for authentication and authorization, as follows:

- [Generate and Use Client Certificate](#) (Section 4.6.1 on page 120)
- [Revoke Client Certificate](#) (Section 4.6.2 on page 122)

Key Points

The following are key points about client certificates:

- The certificate does not guarantee access. The **redapi** revalidation process can deny access.
- When using a certificate, the user does not need to enter their password every time.
- Admins can set a certificate expiration (days/months/years).
- Admin can include additional capability restrictions in the certificate.
- Admins can revoke a certificate.
- If the certificate is stolen/lost, access could be compromised or lost.
- Admins must remember the location of the certificate file for access.

4.6.1 Generate and Use Client Certificate

To generate and use a client certificate, complete the following procedure.

Procedure

1. Login as a realm, tenant, or subtenant admin.
2. Generate a client certificate using **redcli security generate** by specifying the certificate name and passing the expiration with the **-e (--expiration)** option and the **scope:capability** with the **-r (--scope)** option.

If the scope is not specified, it defaults to the scope of the user issuing the command.

```
redcli security generate <certificate_name> -e <expiration> -r <scope:capability>
```

Note that this command generates two files **<certificate_name>.cert** and **<certificate_name>.key**. Both files must be available at the same location to use the certificate.

3. List the certificates to verify the details:

```
redcli security list
```

4. Download the specified certificate:

```
redcli security get <certificate_name>
```

5. Export the environment variable **REDCLI_CLIENT_CERTIFICATE** with the **<certificate_name>.cert** file. Note that this environment variable must point to the **<certificate_name>.cert** file to use the certificate for authentication and authorization:

```
export REDCLI_CLIENT_CERTIFICATE='pwd'/'<certificate_name>.cert'
```

6. Display the user authentication information to verify the certificate:

```
redcli user info
```

Example

In the following example, a **red** tenant admin (**redadmin**) logs in and generates two certificates: the first with the default scope and capability of **red:admin** and an expiration of one year and the second for a subtenant admin with **red/red** scope and an expiration of one year. This example shows the usage of both certificates.

```
# Login as the red tenant admin (redadmin).
$ redcli user login redadmin -p reDsecrEtwoRd4D -t red
4:14PM INF User "redadmin" logged in successfully.

# Create a certificate with the default scope of red:admin and an expiration of one year.
$ redcli security generate admin -e 1y
4:15PM INF Client certificate 'admin' has been created.
4:15PM INF Client certificate written to file 'admin.crt'
4:15PM INF Client certificate key written to file 'admin.key'

# Create a certificate with red/red:admin scope and capability and an expiration of one year.
$ redcli security generate user -e 1y -r red/red
4:16PM INF Client certificate 'user' has been created.
4:16PM INF Client certificate written to file 'user.crt'
4:16PM INF Client certificate key written to file 'user.key'
```



```
# List the newly created certificates.
$ redcli security list
```

| NAME | VALID | NUMBER | ACCESS SCOPE |
|-------|-------------------------|------------------|------------------------|
| admin | 2023-10-04 - 2024-10-06 | 76db2c16ba408d7b | <creator capabilities> |
| user | 2023-10-04 - 2024-10-06 | 57e48a567d158b2c | red/red |

```
# Download the admin certificate.
$ redcli security get admin
4:29PM INF Client certificate written to file 'admin.crt'
4:29PM INF Client certificate key written to file 'admin.key'
```

```
# Use the admin client certificate.
$ export REDCLI_CLIENT_CERTIFICATE=`pwd`/admin.crt
```

```
$ redcli user info
4:45PM INF Using 'redadmin' certificate '/file_path/admin.crt' at level cluster 'cluster-gray',
tenant 'red'
```

```
$ redcli user list
```

| USER | IDENTITY | ACCESS SCOPE | NAME | EMAIL | GROUPS |
|----------|----------|------------------------------|------|-------|--------|
| redadmin | red | red, red/red, red/red/redobj | | | |

```
# Download the user certificate.
$ redcli security get user
4:29PM INF Client certificate written to file 'user.crt'
4:29PM INF Client certificate key written to file 'user.key'
```

```
# Use the user client certificate.
$ export REDCLI_CLIENT_CERTIFICATE=`pwd`/user.crt
```

```
$ redcli user info
4:49PM INF Using 'redadmin' (red/red) certificate '/file_path/user.crt' at level cluster
'cluster-gray', tenant 'red'
```

```
$ redcli user list
4:49PM FTL Error: operation unauthorized for scope 'red/red'
```

```
$ redcli dataset list
4:49PM INF Auto Selecting cluster: cluster-gray
```

| DATASET LIST |
|--------------|
| red |

4.6.2 Revoke Client Certificate

To revoke a client certificate complete the following procedure.

Procedure

- 1. Login as admin with proper realm or tenant access or export the environment variable REDCLI_CLIENT_CERTIFICATE with proper access scope.

For example, you must have tenant admin privileges to revoke certificate for a subtenant user.

```
export REDCLI_CLIENT_CERTIFICATE=`pwd`/<certiciate_name.crt>
```

- 2. Revoke the specified client certificate:

```
redcli security revoke <certificate_name>
```

- 3. List the certificates to verify that the specified certificate is revoked:

```
redcli security list
```

Example

Continuing the previous example, a **red** tenant admin (**redadmin**) uses their certificate to revoke the subtenant **user.crt** certificate.

```
# Use the admin client certificate.
$ export REDCLI_CLIENT_CERTIFICATE=`pwd`/admin.crt

# Revoke the user client certificate
$ redcli security revoke user
4:52PM INF Client certificate 'user' has been revoked.

# List the available certificates and verify that the user certificate has been revoked.
$ redcli security list
```

| NAME | VALID | NUMBER | ACCESS SCOPE |
|-------|-------------------------|------------------|------------------------|
| admin | 2023-10-04 - 2024-10-06 | 76db2c16ba408d7b | <creator capabilities> |

4.7 Password Management Using Local Authentication

When using DDN Infinia local authentication, passwords can be reset either by the user directly or administrators with administrative capabilities on the management entity (realm, tenant). For new users, this is default behavior.

5. Data Service Management

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, administrators manage data services, as described in the following sections:

- [Manage S3 Access Records](#) (Section 5.1 on page 125)
- [Manage Data Services](#) (Section 5.2 on page 128)
- [Manage Buckets](#) (Section 5.3 on page 133)

Key Points

The following are key points about data service management:

- Only subtenant admins with admin capability on the data service can create buckets.
- A dataset is the store data building block in DDN Infinia, and thus everything is a dataset. A dataset can be visible through S3 protocol if the particular attribute/xatts is set to be a S3 bucket. For additional details including instructions to change the properties of the underlying dataset, see [Manage Datasets](#) (Section 3.8 on page 101).

5.1 Manage S3 Access Records

NOTE: This section applies to subtenant admins only. To access the DDN Infinia S3 DataService, users must have the `<tenant/subtenant/dataservice>:admin` or `<tenant/subtenant/dataservice>:data-access` scope and capability.

S3 access records consist of a key/secret pair, which are required to authenticate REST requests. When a subtenant admin assigns a scope that includes DDN Infinia S3 DataService to a user, the key/secret pair is automatically created with default values for that user. In this case, the user can login with `redcli` and query the access record to get the key/secret pair or the admin can pass the key/secret pair to the user. Optionally, subtenant admins can set an expiration for the S3 access records.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, subtenant admins manage S3 access records (key/secret pair), as follows:

- [Add S3 Access Record](#) (Section 5.1.1 on page 126)
- [Remove S3 Access Record](#) (Section 5.1.2 on page 126)
- [Update S3 Access Record](#) (Section 5.1.3 on page 127)
- [List S3 Access Record](#) (Section 5.1.4 on page 127)

For a summary of key points about managing S3 access records, see [Key Points](#) in [Data Service Management](#) (Section 5 on page 124).

For additional `redcli s3 access` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

5.1.1 Add S3 Access Record

As a subtenant admin, add an S3 access record (key/secret pair) for a user using **redcli s3 access add**. For tenants other than the **red** tenant (Figure 4 on page 38), specify the tenant by passing the **-t (--tenant)** option. This command generates an S3 key and S3 secret for the specified user, along with a date when the access expires. The output of the command will display the key pair, which must be passed to the user.

```
redcli s3 access add <new_user> [-t <tenant_name>] [flags]
```

If required, set an expiration by passing the expiration period in **ymd** format with the **-e (--expiration)** option.

Example

The first example shows the commands to create access records with an expiration of ninety days and ten years, respectively, within the default **red** tenant domain.

```
redcli s3 access add s3user -e 90d
redcli s3 access add john@red.ad.com -e 10y
```

The second example adds an S3 access record for a new user (**yellow_admin**) within the **tenant-yellow** tenant domain.

```
$ redcli s3 access add yellow_admin -e 10y -t tenant-yellow
11:08PM INF Auto Selecting cluster: cluster-gray
```

| | |
|------------|--|
| S3_KEY | 100WDCZGLCZ7Z9RI3DQ0 |
| S3_SECRET | xnVPqYM1ZiaavQiv9XikcQbFu7TnSp5M7x7dfW3K |
| EXPIRATION | 2033-10-19 23:08:33 |
| CLUSTER | cluster-gray |
| TENANT | tenant-yellow |
| GROUP | |
| UID | 20003 |
| GID | 20000 |
| EXPIRED | false |

5.1.2 Remove S3 Access Record

As a subtenant admin, remove an S3 access record using **redcli s3 access remove** and specifying the S3 key.

```
redcli s3 access remove <s3-key> [flags]
```

5.1.3 Update S3 Access Record

As a subtenant admin, update an S3 access record using **redcli s3 access update** by specifying the S3 key and passing the options for the attributes to be changed. For a complete list of flags and options, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

```
redcli s3 access update <s3-key> [flags]
```

5.1.4 List S3 Access Record

As a subtenant admin, list the S3 access records using **redcli s3 access list**:

```
redcli s3 access list [flags]
```

5.2 Manage Data Services

NOTE: This section primarily applies to subtenant admins; however, tenant admins can list and show data services for themselves. Dataservice admins can manage their own data service.

Each instance of a DDN Infinia data service has the following attributes, which uniquely identify it:

- **Name** – The data service name per subtenant should be unique. Different subtenants/tenants can have services with the same name.
- **uuid** – This *uuid* is a globally unique identifier assigned by the system to an object in DDN Infinia and stored with the object attributes.
- **Type** – DDN Infinia supports object and block data services.
- **Protocol** – Object data services contain objects exposed via S3; while block data services are a collection of block devices provided via NVMeoF and created via [Container Storage Interface \(CSI\)](#).

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, subtenant admins create and update data services, as described in the following sections. DDN Infinia data services are defined at the tenant and subtenant level and support the following operations:

- [Create a Block Data Service](#) (Section 5.2.1 on page 129)
- [Create an Object Data Service](#) (Section 5.2.2 on page 129)
- [Delete a Data Service](#) (Section 5.2.3 on page 130)
- [Update a Data Service](#) (Section 5.2.4 on page 130)
- [List Data Services](#) (Section 5.2.5 on page 130)
- [Show Data Service Status](#) (Section 5.2.6 on page 132)

For a summary of key points about managing data services, see [Key Points](#) in [Data Service Management](#) (Section 5 on page 124).

For additional details about `redcli service` and available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

5.2.1 Create a Block Data Service

DDN Infinia supports creating and updating block dataset types that can be exported using NVMeoF as the transport protocol, but managed using [Container Storage Interface \(CSI\)](#). For additional details about the DDN Infinia block data service, see [Use DDN Infinia Block DataService](#) (Section 2.5 on page 68).

When creating a block data service, the following properties define the service:

- Service name (**service-name**) – Unique name for the data service (required).
- Service type (**-T, --type option**) – The **block** service type (required).
- Protocol (**-P, --protocol option**) – The **csi** protocol (required).
- Dataset size (**-b, --dataset-size option**) – Size of the dataset created (optional).
 - ❖ When creating a dataset, if no quota is specified, the default quota for that dataset is set to the parent subtenant's quota.
 - ❖ When creating or updating a dataset, the resulting storage quota ratio is calculated as: $\text{ratio} = (\text{sum of dataset quotas}) / (\text{sum of all subtenant quotas})$
 - ❖ Recommended ratios are less than 3. If the calculated ratio is greater than 3, a warning will be displayed.

Note that the properties listed are a subset of the available flags and options. For a complete list, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

As a subtenant admin, create a block data service using **redcli service create** by specifying the service name and passing the required parameters. Note that the command creates the service and service user simultaneously:

```
redcli service create <service_name> -T block -P csi [-b <dataset_size>] [flags]
```

5.2.2 Create an Object Data Service

NOTE: Object naming follows [AWS compatibility standards](#). For guidelines about object naming, see [S3 Bucket and Object Management](#) (Appendix B.2 on page 183).

DDN Infinia supports creating and updating S3 data services. For additional details about the DDN Infinia S3 data service, see [Use DDN Infinia S3 DataService](#) (Section 2.4 on page 59).

When creating an object data service, the following properties define the service:

- Service name (**service-name**) – Unique name for the data service (required).
- Service type (**-T, --type option**) – The **file-and-object** service type (required).
- Protocol (**-P, --protocol option**) – The **s3** protocol (required).

Note that the properties listed are a subset of the available flags and options. For a complete list, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

As a subtenant admin, create an object data service using **redcli service create** by specifying the service name and passing the required parameters:

```
redcli service create <service_name> -T file-and-object -P s3 [flags]
```

5.2.3 Delete a Data Service

As a subtenant admin, delete an existing data service using **redcli service delete** and specifying the name of the service. Note that the **-T (--type)** option is required only if there are block and object data services with the same name.

```
redcli service delete <service_name> [flags]
```

To bypass confirmation of the deletion, pass the **-f (--force)** flag:

```
redcli service delete <service_name> -f
```

To delete the data service but not the associated datasets, pass the **-k (--keep-dataset)** flag:

```
redcli service delete <service_name> -k
```

Example

In the following example, the command deletes the **obsolete-service** data service but retains the datasets defined by that data service.

```
$ redcli service delete obsolete-service -k
11:00PM INF Auto Selecting cluster: cluster-gray
11:00PM INF Auto Selecting tenant: red
11:00PM INF Auto Selecting subtenant: engr
Deleting service obsolete-service. Confirm? [y/N] y

11:00PM INF Successfully deleted service: obsolete-service
```

5.2.4 Update a Data Service

As a subtenant or dataservice admin, update a data services using **redcli service update** by specifying the service name and passing the options for the attributes to be changed. Note that admins can update attributes for their own data services only.

The **-T (--type)** option is required only if there are block and object data services with the same name. For a complete list of flags and options, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

```
redcli service update <service_name> [flags]
```

5.2.5 List Data Services

As a subtenant, tenant, or dataservice admin, list data services using **redcli service list**. Note that admins can list their own data services only.

The **-T (--type)** option is optional for the **list** subcommand. If this option is not provided, the command lists all data services for all types. For block services, the output displays the datasets used by the services. For object services, the output displays all buckets for the services.

```
redcli service list [-T <service_type>] [flags]
```

Example

The first example lists block services in the **cluster-gray** cluster.

```
$ redcli service list -T block
6:19AM INF Auto Selecting cluster: cluster-gray
```

| NAME | SCOPE | TYPE | PROTOCOL | DATASET | QUOTA | USAGE |
|----------|--------------------|-------|----------|-----------------|-------|---------------|
| sub1blk1 | tnt6/sub1/sub1blk1 | block | csi | sub1blkdataset1 | 6 GB | 18.05 kB (0%) |
| sub1blk2 | tnt6/sub1/sub1blk2 | block | csi | sub1blkdataset2 | 6 GB | 18.05 kB (0%) |
| sub1blk3 | tnt6/sub1/sub1blk3 | block | csi | sub1blkdataset3 | 6 GB | 18.14 kB (0%) |
| sub1blk4 | tnt6/sub1/sub1blk4 | block | csi | sub1blkdataset4 | 6 GB | 18.14 kB (0%) |
| sub1blk5 | tnt6/sub1/sub1blk5 | block | csi | sub1blkdataset5 | 6 GB | 18.14 kB (0%) |
| sub2blk1 | tnt6/sub2/sub2blk1 | block | csi | sub2blkdataset1 | 6 GB | 18.05 kB (0%) |
| sub2blk2 | tnt6/sub2/sub2blk2 | block | csi | sub2blkdataset2 | 6 GB | 18.05 kB (0%) |
| sub2blk3 | tnt6/sub2/sub2blk3 | block | csi | sub2blkdataset3 | 6 GB | 18.14 kB (0%) |
| sub2blk4 | tnt6/sub2/sub2blk4 | block | csi | sub2blkdataset4 | 6 GB | 18.14 kB (0%) |
| sub2blk5 | tnt6/sub2/sub2blk5 | block | csi | sub2blkdataset5 | 6 GB | 18.14 kB (0%) |
| sub3blk1 | tnt6/sub3/sub3blk1 | block | csi | sub3blkdataset1 | 6 GB | 18.14 kB (0%) |
| sub3blk2 | tnt6/sub3/sub3blk2 | block | csi | sub3blkdataset2 | 6 GB | 18.14 kB (0%) |
| sub3blk3 | tnt6/sub3/sub3blk3 | block | csi | sub3blkdataset3 | 6 GB | 18.4 kB (0%) |
| sub3blk4 | tnt6/sub3/sub3blk4 | block | csi | sub3blkdataset4 | 6 GB | 18.4 kB (0%) |
| sub3blk5 | tnt6/sub3/sub3blk5 | block | csi | sub3blkdataset5 | 6 GB | 18.4 kB (0%) |
| sub4blk1 | tnt6/sub4/sub4blk1 | block | csi | sub4blkdataset1 | 6 GB | 18.14 kB (0%) |
| sub4blk2 | tnt6/sub4/sub4blk2 | block | csi | sub4blkdataset2 | 6 GB | 18.14 kB (0%) |
| sub4blk3 | tnt6/sub4/sub4blk3 | block | csi | sub4blkdataset3 | 6 GB | 18.4 kB (0%) |
| sub4blk4 | tnt6/sub4/sub4blk4 | block | csi | sub4blkdataset4 | 6 GB | 18.4 kB (0%) |
| sub4blk5 | tnt6/sub4/sub4blk5 | block | csi | sub4blkdataset5 | 6 GB | 18.4 kB (0%) |
| sub5blk5 | tnt6/sub5/sub5blk5 | block | csi | sub5blkdataset5 | 6 GB | 18.4 kB (0%) |
| sub5blk2 | tnt6/sub5/sub5blk2 | block | csi | sub5blkdataset2 | 6 GB | 18.14 kB (0%) |
| sub5blk3 | tnt6/sub5/sub5blk3 | block | csi | sub5blkdataset3 | 6 GB | 18.4 kB (0%) |
| sub5blk4 | tnt6/sub5/sub5blk4 | block | csi | sub5blkdataset4 | 6 GB | 18.4 kB (0%) |
| sub5blk1 | tnt6/sub5/sub5blk1 | block | csi | sub5blkdataset1 | 6 GB | 18.14 kB (0%) |

The second example lists S3 services in the **cluster-gray** cluster.

```
$ redcli service list -T file-and-object
6:19AM INF Auto Selecting cluster: cluster-gray
```

| NAME | SCOPE | TYPE | PROTOCOL | DATASET | QUOTA | USAGE |
|---------|--------------------|-----------------|----------|---------|-------|----------------|
| sub1s3 | tnt6/sub1/sub1s3 | file-and-object | s3 | | 60 GB | 18.79 GB (31%) |
| sub2s3 | tnt6/sub2/sub2s3 | file-and-object | s3 | | 60 GB | 18.79 GB (31%) |
| sub3s3 | tnt6/sub3/sub3s3 | file-and-object | s3 | | 60 GB | 18.79 GB (31%) |
| sub4s3 | tnt6/sub4/sub4s3 | file-and-object | s3 | | 60 GB | 18.79 GB (31%) |
| sub5s3 | tnt6/sub5/sub5s3 | file-and-object | s3 | | 60 GB | 18.79 GB (31%) |
| sub6s3 | tnt6/sub6/sub6s3 | file-and-object | s3 | | 60 GB | 18.79 GB (31%) |
| sub7s3 | tnt6/sub7/sub7s3 | file-and-object | s3 | | 60 GB | 18.79 GB (31%) |
| sub8s3 | tnt6/sub8/sub8s3 | file-and-object | s3 | | 60 GB | 18.79 GB (31%) |
| sub9s3 | tnt6/sub9/sub9s3 | file-and-object | s3 | | 60 GB | 18.79 GB (31%) |
| sub10s3 | tnt6/sub10/sub10s3 | file-and-object | s3 | | 60 GB | 18.79 GB (31%) |

5.2.6 Show Data Service Status

As a subtenant, tenant, or dataservice admin, show the current status of a data service using **redcli service show** and specifying the name of the service. Note that admins can show their own data services only.

The **-T (--type)** option is required only if there are block and object data services with the same name.

```
redcli service show <service_name> [flags]
```

5.3 Manage Buckets

NOTE: Only subtenant admins with admin capability on the data service can create buckets.

A dataset is the store data building block in DDN Infinia, and thus everything is a dataset. A dataset can be visible through S3 protocol if the particular attribute/xatts is set to be a S3 bucket.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, subtenant admins manage buckets to write and read object data using the S3 protocol, as described in this section:

- [Create Buckets](#) (Section 5.3.1 on page 134)
- [List Buckets](#) (Section 5.3.2 on page 134)

For a summary of key points about managing data services, see [Key Points](#) in [Data Service Management](#) (Section 5 on page 124).

For additional `redcli s3 bucket` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

NOTE: The following section discusses creating buckets using the DDN Infinia CLI (`redcli`). You also can create buckets using the DDN Infinia REST API, DDN Infinia GUI, or DDN Infinia S3 DataService API. For details about using the DDN Infinia S3 DataService API, see [DDN Infinia S3 DataService API Endpoints](#) (Appendix B on page 182).

5.3.1 Create Buckets

NOTE: DDN Infinia S3 DataServices conform to [S3 protocol standards](#). For rules about bucket naming, see [S3 Bucket and Object Management](#) (Appendix B.2 on page 183).

As a subtenant admin with admin capability on the data service, create a bucket to write and read object data using the S3 protocol using **redcli s3 bucket create** by specifying the bucket name and, optionally, passing the bulk quota with the **-b** (**--quota**) option. For data services other than the **red\red\redobj**, specify the data service scope by passing the **-r** (**--scope**) option.

```
redcli s3 bucket create <bucket_name> [-b <bulk_quota>] [-r <scope>] [flags]
```

Note that creating a bucket also create the dataset. When creating an S3 bucket, if no quota is specified, the default quota for that bucket is set to the parent subtenant's quota.

Example

In this example, a subtenant admin (**subtenant_admin**) with admin capability on the **red\red\redobj** data service (Figure 4 on page 38) creates two buckets, one with a 1 TiB quota and other with 2 TiB quota, for that service.

```
redcli user login subtenant_admin -p suBteNantsecWD3
redcli s3 bucket create bucket1 -b 1T
redcli s3 bucket create bucket2 -b 2T
```

5.3.2 List Buckets

As a subtenant admin, list s3 buckets to which the logged-in admin has access using **redcli s3 bucket list**. Note that this command only lists datasets with the special attribute/xatts set to be a S3 bucket.

```
redcli s3 bucket list [flags]
```

By default, the output is in tabular format. To display the output in yaml format, pass the **-o yaml** flag.

Example

The first example lists buckets for the subtenant **red/red** in the default tabular output format. Note that only datasets with the special attribute/xatts set to be a S3 bucket are included in the output.

```
$ redcli s3 bucket list -t red -s red
```

```
6:22AM INF Auto Selecting cluster: cluster-gray
```

```
6:22AM INF Auto Selecting tenant: red
```

```
6:22AM INF Auto Selecting subtenant: red
```

```
6:22AM INF Auto Selecting service: redobj
```

| NAME | OWNER | CAPACITY | USAGE |
|---------------|-------|----------|--------------------|
| bucket1 | admin | 9.223 EB | 1.238070970414e+12 |
| user1bucket1 | admin | 9.223 EB | 2.56712020583e+11 |
| user1bucket10 | admin | 9.223 EB | 0 |
| user1bucket11 | admin | 9.223 EB | 0 |
| user1bucket12 | admin | 9.223 EB | 0 |
| user1bucket2 | admin | 9.223 EB | 1.27757934009e+11 |
| user1bucket3 | admin | 9.223 EB | 0 |
| user1bucket4 | admin | 9.223 EB | 0 |
| user1bucket5 | admin | 9.223 EB | 0 |
| user1bucket6 | admin | 9.223 EB | 0 |
| user1bucket7 | admin | 9.223 EB | 0 |
| user1bucket8 | admin | 9.223 EB | 0 |
| user1bucket9 | admin | 9.223 EB | 0 |
| user2bucket4 | admin | 9.223 EB | 2.04754415903e+11 |
| user3bucket1 | admin | 9.223 EB | 0 |
| user3bucket2 | admin | 1 TB | 0 |
| user3bucket3 | admin | 3 TB | 0 |
| user3bucket4 | admin | 4 TB | 0 |

The second example lists the buckets in json format by passing the `-o json` flag. Note that only datasets with the special attribute/xatts set to be a S3 bucket are included in the output.

```
$ redcli s3 bucket list -o json
{
  "data": {
    "Buckets": [
      {
        "CreationDate": "2023-02-21T23:12:50Z",
        "Name": "bucket1",
        "Quota": 1099511627776,
        "Usage": -1
      },
      {
        "CreationDate": "2023-02-21T23:13:24Z",
        "Name": "bucket2",
        "Quota": 2199023255552,
        "Usage": -1
      }
    ],
    "Owner": {
      "DisplayName": "subtenant_admin",
      "ID": "3f6bda07efaa463fb6e8312749a39cb91a3a5caef6f59bcfcfd32e2ed32b2be"
    },
    "ResultMetadata": {}
  },
  "detail": "",
  "epoch": 1677024803,
  "timestamp": "Wed, 22 Feb 2023 00:13:23 GMT",
  "title": "List s3 buckets"
}
```


6. Hardware Management

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, administrators manage the servers, drives, and network interface controllers (NICs) in a cluster, as described in the following sections:

- [Upgrade Server, Drive, and NIC Firmware](#) (Section 6.1 on page 138)
- [Manage BIOS Configuration](#) (Section 6.2 on page 146)

6.1 Upgrade Server, Drive, and NIC Firmware

This section applies to realm admins only. Realm admin user scope and capability is required to upgrade the firmware on a server in a DDN Infinia cluster.

NOTE:

To upgrade the firmware using the `redcli`, you must have completed the installation and setup steps described in [Deploy DDN Infinia](#) (Section 2.1 on page 26) and [Deploy the Cluster](#) (Section 2.2 on page 38) to configure your realm and create a cluster.

As a realm administrator, you can upgrade the BMC, BIOS, drive, and network interface controller (NIC) firmware for the servers in your cluster using the `redcli` commands described in the following sections.

For additional details about the `redcli` commands, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

On startup, the latest firmware bundle specific to the platform is downloaded by HMI under `/opt/ddn/red/mounts/fw_rep`. This bundle includes the respective firmware image file for the specific components to be upgraded.

For each server, upgrade the firmware for a specific component using the commands in the following sections.

- ❖ [Upgrade BMC Firmware](#) (Section 6.1.1 on page 139)
- ❖ [Upgrade BIOS Firmware](#) (Section 6.1.2 on page 141)
- ❖ [Upgrade Drive Firmware](#) (Section 6.1.3 on page 143)
- ❖ [Upgrade Network Interface Controller Firmware](#) (Section 6.1.4 on page 143)

6.1.1 Upgrade BMC Firmware

IMPORTANT: `redcli` does not support BMC upgrades on the AI2000. For a list of platforms that support BMC upgrades, contact DDN Support.

Display the current BMC firmware version using `redcli server show` by specifying the host name or UUID of the server:

```
redcli server show <host-name>
```

or

```
redcli server show <uuid>
```

Example

The following is an example output.

```
# Display the current BMC firmware version.
$ redcli server show ddn_server
```

| NAME | UUID | DRIVES | ADAPTERS | SENSORS | POWER |
|------------|--------------------------------------|--------|----------|---------|-------|
| ddn_server | 9050b2cc-c229-4675-aad1-f785ed42eb33 | 13 | 10 | 74 | on |

| CPU INFO | | | | |
|----------|---------------------------------|---------|--------------------|------------|
| ARCH | MODEL | SOCKETS | CORES (Per SOCKET) | NUMA NODES |
| x86_64 | AMD EPYC 9254 24-Core Processor | 1 | 24 | 4 |

| MEMORY DEVICES | | | | |
|----------------|-------------|-------|-----------|------|
| BANK | LOCATION | SIZE | SPEED | TYPE |
| P0 CHANNEL A | P0_CHA_DIM1 | 32 GB | 4800 MT/s | DDR5 |
| P0 CHANNEL K | P0_CHK_DIM1 | 32 GB | 4800 MT/s | DDR5 |

| BMC INFO | | | | |
|-------------|---------------|------------|-----------|-------------------|
| ADDRESS | NETMASK | GATEWAY | SUPPORTED | FIRMWARE REVISION |
| 10.25.61.40 | 255.255.255.0 | 10.25.61.1 | true | 1.13 |

| BIOS INFO | | | |
|---|----------------|--------------|----------|
| VENDOR | VERSION | RELEASE DATE | REVISION |
| American Megatrends International, LLC. | V1.13v4 (0x32) | 10/02/2023 | 5.27 |

Upgrade the BMC firmware using `redcli server bmcfw-update` by passing the path to the BMC firmware image file with the `-f (--file-path)` option and, optionally, specifying the host name or UUID of the server. If the host name or UUID is not specified, the command auto selects the server from which the command is running by default. You can override this default by passing the `--server` global option or by setting the `REDCLI_SERVER` environment variable. Note that this command resets the BMC of the target server.

```
redcli server bmcfw-update <host-name> -f <FW_Filepath>
```

or

```
redcli server bmcfw-update <uuid> -f <FW_Filepath>
```

Example

The following is an example output of a BMC upgrade. In this example, the command `auto` selects the server from which the command is running.

```
# Upgrade the BMC firmware.
ddn@ddn_server:~/user$ redcli server bmcfw-update -f /opt/ddn/red/mounts/fw_repo/BMC_example_fw_file.bin
Please dont interrupt the Firmware Update call. Wait for it to finish.
2:35AM INF Auto selecting server: ddn-server. Auto selecting target server: ddn_server. Success
HMI Server BMC Firmware Update
Option: preserve CFG
Option: preserve SDR
Option: preserve SSL
*****
WARNING!
Firmware upgrade must not be interrupted once it is started.
Once you get error after Upgrading, please use local KCS tool
for recovery.
*****
Check firmware file... Done (ver:1.02.14)
Check BMC status... Done (ver:1.02.14)
Uploading file >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 100%
Updating      >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 100%
Resetting BMC
Done
Total Elapsed Time: 3 min 14 sec(s)
Fri, 30 Aug 2024 09:35:39 GMT
```

6.1.2 Upgrade BIOS Firmware

IMPORTANT: `redcli` does not support BIOS upgrades on the AI2000. For a list of platforms that support BIOS upgrades, contact DDN Support.

Display the current BIOS firmware version using `redcli server show` by specifying the host name or UUID of the server:

```
redcli server show <host-name>
```

or

```
redcli server show <uuid>
```

Example

The following is an example output.

```
# Display the current BIOS firmware version.
$ redcli server show ddn_server
```

| NAME | UUID | DRIVES | ADAPTERS | SENSORS | POWER |
|------------|--------------------------------------|--------|----------|---------|-------|
| ddn_server | 9050b2cc-c229-4675-aad1-f785ed42eb33 | 13 | 10 | 74 | on |

| CPU INFO | | | | |
|----------|---------------------------------|---------|--------------------|------------|
| ARCH | MODEL | SOCKETS | CORES (Per SOCKET) | NUMA NODES |
| x86_64 | AMD EPYC 9254 24-Core Processor | 1 | 24 | 4 |

| MEMORY DEVICES | | | | |
|----------------|-------------|-------|-----------|------|
| BANK | LOCATION | SIZE | SPEED | TYPE |
| P0 CHANNEL A | P0_CHA_DIM1 | 32 GB | 4800 MT/s | DDR5 |
| P0 CHANNEL K | P0_CHK_DIM1 | 32 GB | 4800 MT/s | DDR5 |

| BMC INFO | | | | |
|-------------|---------------|------------|-----------|-------------------|
| ADDRESS | NETMASK | GATEWAY | SUPPORTED | FIRMWARE REVISION |
| 10.25.61.40 | 255.255.255.0 | 10.25.61.1 | true | 1.13 |

| BIOS INFO | | | |
|---|----------------|--------------|----------|
| VENDOR | VERSION | RELEASE DATE | REVISION |
| American Megatrends International, LLC. | V1.13v4 (0x32) | 10/02/2023 | 5.27 |

Upgrade the BIOS firmware using `redcli server biosfw-update` by passing the path to the BIOS firmware image file with the `-f` (`--file-path`) option and, optionally, specifying the host name or UUID of the server. If the host name or UUID is not specified, the command auto selects the server from which the command is running by default. You can override this default by passing the `--server` global option or by setting the `REDCLI_SERVER` environment variable.

```
redcli server biosfw-update <host-name> -f <FW_Filepath>
```

or

```
redcli server biosfw-update <uuid> -f <FW_Filepath>
```

By default, this command reboots the server at the end of upgrade. To skip this automatic reboot and manually reboot the server at a later time, pass the `--reboot=false` flag when running the command. Note that the server must be rebooted to activate the updated BIOS.

Example

The following is an example output of a BIOS upgrade. In this example, the command `auto` selects the server from which the command is running.

```
# Upgrade the BIOS FW firmware
ddn@ddn_server:~/user$ redcli server biosfw-update -f /opt/ddn/red/mounts/fw_repo/BIOS_example_fw_file.bin
Please dont interrupt the Firmware Update call. Wait for it to finish.
2:52AM INF Auto selecting server: ddn_server. Auto selecting target server: ddn_server. Success
HMI Server BIOS Firmware Update
Option: preserve NVRAM
Option: preserve SMBIOS
Option: preserve ME
Option: Force power down to proceed update if needed
        Perform reset after update
=====
BIOS Image info
=====
Date      = 12/06/2023
MB Type   = H13SSH
Size      = 32 MB
Rev       = 1.7a

=====
Start BIOS Upgrade
=====

Uploading file >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 100%
Updating    >>>>>>>>>>>>>>>>>>>>>>>>>>>>>>> 100%
Done
Update Time: 2 min 20 sec(s)

Total Elapsed Time: 2 min 20 sec(s)
System will perform reset soon.
Fri, 30 Aug 2024 09:52:35 GMT
```

6.1.3 Upgrade Drive Firmware

Display current drive firmware versions using the following command:

```
redcli drive list
```

Upgrade the firmware for a specific drive using **redcli drive fw-update** by specifying the host name or UUID of the server, specifying the drive ID, and passing the path to the drive firmware image file with the **-f (--file-path)** option:

```
redcli drive fw-update <host-name> <drive-id> -f <FW_Filepath>
```

or

```
redcli drive fw-update <uuid> <drive-id> -f <FW_Filepath>
```

Note that this upgrade is asynchronous. The command will return immediately while the upgrade continues. To check the status of the upgrade, pass the **-s (--status)** flag when running the command. When the upgrade is complete, the status will display **FIRMWARE_UPDATE_STATUS_COMPLETE**, as shown in the following example.

Example

The following is an example output of a drive upgrade.

```
# Upgrade a drive firmware
$ redcli drive firmware-update ddn_server drive-103 -f /opt/ddn/red/mounts/fw_repo/5CV10203.bin
11:56AM INF Drive firware update status: FIRMWARE_UPDATE_STATUS_INITIATED

$ redcli drive firmware-update ddn_server drive-103 --status
11:56AM INF Drive firware update status: FIRMWARE_UPDATE_STATUS_IN_PROGRESS

$ redcli drive firmware-update ddn_server drive-103 --status
11:56AM INF Drive firware update status: FIRMWARE_UPDATE_STATUS_COMPLETE
```

6.1.4 Upgrade Network Interface Controller Firmware

IMPORTANT: The **redcli** commands support Mellanox CX-6 and CX-7 cards only.

Display current NIC firmware versions using the following command:

```
redcli nic list
```

Upgrade the firmware for a specific NIC using **redcli nic fw-update** by specifying the NIC ID, passing the host name or UUID of the server with the **-H (--hostname)** option, and passing the path to the NIC firmware image file with the **-f (--file-path)** option:

```
redcli nic fw-update <nic-id> -H <host-name> -f <FW_Filepath>
```

or

```
redcli nic fw-update <nic-id> -H <uuid> -f <FW_Filepath>
```

Note that this upgrade is asynchronous. The command will return immediately while the upgrade continues.

To check the status of the upgrade, pass the **-s** (**--status**) flag:

```
redcli nic fw-update <nic-id> -H <host-name> --status
```

or

```
redcli nic fw-update <nic-id> -H <uuid> --status
```


Example

The following is an example output of a NIC upgrade.

```
# Upgrade NIC firmware
$ redcli nic fw-update mlx5_0 -f /opt/ddn/red/mounts/fw_repo/nic_example.deb -H ddn_server
5:26AM INF Auto Selecting cluster: cluster1
5:26AM INF Auto Selecting config: auto_config
5:26AM INF Success
Update NIC Firmware
Status: FIRMWARE_UPDATE_STATUS_INITIATED

Fri, 27 Sep 2024 05:26:36 GMT
$ redcli nic fw-update mlx5_0 -H ddn_server -s
5:26AM INF Auto Selecting cluster: cluster1
5:26AM INF Auto Selecting config: auto_config
5:26AM INF Success
Update NIC Firmware
Status: FIRMWARE_UPDATE_STATUS_IN_PROGRESS

Fri, 27 Sep 2024 05:26:41 GMT
$ redcli nic fw-update mlx5_0 -H ddn_server -s
5:27AM INF Auto Selecting cluster: cluster1
5:27AM INF Auto Selecting config: auto_config
5:27AM INF Success
Update NIC Firmware
Status: FIRMWARE_UPDATE_STATUS_COMPLETE
WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
Reading package lists...
Building dependency tree...
Reading state information...
The following NEW packages will be installed:
  mlnx-fw-updater
0 upgraded, 1 newly installed, 0 to remove and 47 not upgraded.
Need to get 0 B/55.7 MB of archives.
After this operation, 88.8 MB of additional disk space will be used.
Get:1 file:/usr/share/doca-host/repo main/ mlnx-fw-updater 24.07-0.6.1.0 [55.7 MB]
debconf: delaying package configuration, since apt-utils is not installed
Selecting previously unselected package mlnx-fw-updater.
(Reading database ... 47669 files and directories currently installed.)
Preparing to unpack .../mlnx-fw-updater_24.07-0.6.1.0_amd64.deb ...
Unpacking mlnx-fw-updater (24.07-0.6.1.0) ...
Setting up mlnx-fw-updater (24.07-0.6.1.0) ...
Initializing...
Attempting to perform Firmware update...
Querying Mellanox devices firmware ...
Device #1:
-----
Device Type:      ConnectX7
Part Number:      MCX755106AC-HEA_Ax
Description:      NVIDIA ConnectX-7 HHHL adapter Card
PSID:             MT_0000001045
PCI Device Name:  41:00.0
Base GUID:        946dae03007e61ca
Base MAC:         946dae7e61ca
Versions:
  FW              28.42.1000      Available 28.42.1000
  PXE             3.7.0500       N/A
  UEFI            14.35.0015       N/A
Status:          Up to date
Log File: /tmp/NVK7zVrns3
Real log file: /tmp/mlnx_fw_update.log
Fri, 27 Sep 2024 05:27:01 GMT
```

6.2 Manage BIOS Configuration

This section applies to realm admins only.

NOTE: DDN Infinia 1.1 does not support BIOS configuration on the AI2000. For a list of platforms that support BIOS configuration, contact DDN Support. For improved performance on these platforms, DDN recommends setting **NUMA Nodes Per Socket = NPS4** in the BIOS configuration. This recommend setting is available under `/opt/ddn/red/mounts/` on these nodes.

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, realm admins manage the BIOS configuration for a server, as follows:

- [Get the BIOS Configuration](#) (Section 6.2.1 on page 146)
- [Update the BIOS Configuration](#) (Section 6.2.2 on page 147)

For instructions to manage the BIOS configuration using the SUM tool, see [Manage BIOS Configuration using SUM](#) (Appendix I on page 295).

For additional `redcli server` commands and details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

6.2.1 Get the BIOS Configuration

As a realm admin, get the BIOS configuration of a specific server using `redcli server bioscfg-get` by specifying the host name or UUID of the server and passing the path for the BIOS configuration file to be saved with the `-f(--file-path)` option. If the file path is not specified, the command prints to stdout.

```
redcli server bioscfg-get <host-name> [-f <BIOS_config_filepath>]
```

or

```
redcli server bioscfg-get <uuid> [-f <BIOS_config_filepath>]
```

Example

The following example saves the BIOS configuration to `/opt/ddn/red/mounts/smc-a9-u1_BIOS.cfg`.

```
$ redcli server bioscfg-get -f /opt/ddn/red/mounts/smc-a9-u1_BIOS.cfg
2:35PM INF Auto selecting server: smc-a9-u1. Auto selecting target server: smc-a9-u1. Success
2:35PM INF Supermicro Update Manager (for UEFI BIOS) 2.14.0 (2024/02/15) (x86_64)
Copyright(C) 2013-2024 Super Micro Computer, Inc. All rights reserved.

File "/opt/ddn/red/mounts/smc-a9-u1_BIOS.cfg" is created.
```

6.2.2 Update the BIOS Configuration

As a realm admin, update the BIOS configuration of a specific server using `redcli server bioscfg-update` by specifying the host name or UUID of the server and passing the path to BIOS configuration file with the `-f(--file-path)` option.

```
redcli server bioscfg-update <host-name> -f <BIOS_config_filepath>
```

or

```
redcli server bioscfg-update <uuid> -f <BIOS_config_filepath>
```

Note that you must manually reboot the server to activate the new BIOS configuration.

Example

The following example updates the BIOS configuration file at `/opt/ddn/red/mounts/fw_repo/SMC-BIOS-NUMA4only.cfg`.

```
$ redcli server bioscfg-update -f /opt/ddn/red/mounts/fw_repo/SMC-BIOS-NUMA4only.cfg
Please dont interrupt the BIOS Config Update call. Wait for it to finish.
2:11PM INF Auto selecting server: smc-a9-u1. Auto selecting target server: smc-a9-u1. Success
2:11PM INF Supermicro Update Manager (for UEFI BIOS) 2.14.0 (2024/02/15) (x86_64)
Copyright(C) 2013-2024 Super Micro Computer, Inc. All rights reserved.

Status: Start updating the BIOS configuration for the managed system

*****WARNING*****
Do not remove AC power from the server.
*****

Note: No BIOS setting has been changed.

Status: The BIOS configuration is updated for the managed system

Note: You have to reboot or power up the system for the changes to take effect.
```

7. Cluster Administration

Based on the [User Scope and Capability](#) (Section 1.7 on page 17) model, some commonly performed administration tasks include the following:

- [List All Clusters](#) (Section 7.1 on page 149)
- [Show Details of a Cluster](#) (Section 7.2 on page 149)
- [List Current Network Configurations](#) (Section 7.3 on page 150)
- [Start a Cluster](#) (Section 7.4 on page 151)
- [Stop a Cluster](#) (Section 7.5 on page 151)
- [Show Status and Health of a Cluster](#) (Section 7.6 on page 152)
- [Show Inventory Details](#) (Section 7.7 on page 157)

Key Points

The following are key points about `redcli` commands:

- DDN Infinia 1.1 supports only one cluster, which can span one physical site. In this case, you can omit the cluster name in most `redcli` commands and the command will auto-select your cluster. Operations that could disrupt service are except from this auto-select. You must include the cluster name in the following commands:
 - ❖ `redcli cluster start <cluster_name>`
 - ❖ `redcli cluster stop <cluster_name>`
 - ❖ `redcli cluster delete <cluster_name>`
- If your cluster has only one cluster configuration, the `redcli` command will auto-select it and the configuration name can be omitted. If your cluster has more than one cluster configuration, you must specify the cluster configuration in the command.

For a complete list of `redcli` commands, refer to the *DDN Infinia CLI Reference*.

The instructions in this section use the DDN Infinia CLI (`redcli`); however, you can also use the DDN Infinia GUI or DDN Infinia REST API for DDN Infinia cluster administration.

NOTE: To help explore the DDN Infinia REST API, issue `redcli` commands with the `--verbose` flag, which will log the underlying REST API calls to use as reference or refer to the *DDN Infinia REST API Reference*.

7.1 List All Clusters

NOTE: Realm, tenant, subtenant, and dataservice admins can list their cluster name.

List all DDN Infinia clusters that are defined:

```
redcli cluster list [flags]
```

Example

This example shows one cluster named **cluster-gray**.

```
$ redcli cluster list
```

| CLUSTER NAME |
|--------------|
| cluster-gray |

7.2 Show Details of a Cluster

NOTE: This section applies to realm admins only.

As realm admin, show details of a single cluster:

```
redcli cluster show [flags]
```

Example

This example shows details for the **cluster-gray** cluster.

```
$ redcli cluster show
5:40PM INF Auto Selecting cluster: cluster-gray
```

| CLUSTER NAME | RUNTIME CONFIG NAME | #INSTANCES | #CATS | NETWORK TYPE(S) |
|--------------|---------------------|------------|-------|-----------------|
| cluster-gray | auto_config | 4 | 4 | [verbs tcp] |

7.3 List Current Network Configurations

NOTE: This section applies to realm admins only.

As realm admin, list all current networking configurations:

```
redcli network list [flags]
```

If your cluster has only one cluster configuration, this command will auto-select it and the configuration name can be omitted from the command. If your cluster has more than one cluster configuration, you must specify the cluster configuration by passing the **-C** (**--config**) option.

Example

In the first example, the command lists all current networking configurations for the auto-selected cluster configuration **only_config**.

```
$ redcli network list
9:09PM INF Auto Selecting cluster: cluster-gray
9:09PM INF Auto Selecting config: only_config
```

| UUID | NAME | TYPE | ENCRYPTION | EQ NENTRIES | CQ NENTRIES |
|--------------------------------------|------------|-------|------------|-------------|-------------|
| 9ba9b885-a04f-45d7-b8b6-2efd862cfadd | tcp-auto | tcp | false | 100 | 8192 |
| beb941a0-c688-42ef-838f-11fda4ef589f | verbs-auto | verbs | false | 100 | 8192 |

| MAX CQ COUNT | WEIGHT | TAGS |
|--------------|--------|------|
| 64 | 100 | |
| 64 | 100 | |

In the second example, the command lists all current networking configurations for the specified cluster configuration **second_config**.

```
$ redcli network list -C second_config
11:19PM INF Auto Selecting cluster: cluster-gray
```

| UUID | NAME | TYPE | ENCRYPTION | EQ NENTRIES | CQ NENTRIES |
|--------------------------------------|-----------|------|------------|-------------|-------------|
| 425953ac-c40b-4da2-9c7d-b81528b28f52 | lo-auto | tcp | false | 100 | 16384 |
| decd24e3-b11a-452c-9b59-bf13e510152c | roce-auto | roce | false | 100 | 16384 |
| fc59e2f0-1853-414d-a456-69eda3044831 | tcp-auto | tcp | false | 100 | 16384 |

| MAX CQ COUNT | WEIGHT | TAGS |
|--------------|--------|-------------------|
| 64 | 100 | NVMF_DATA_SERVICE |
| 64 | 100 | NVMF_DATA_SERVICE |
| 64 | 0 | NVMF_DATA_SERVICE |

7.4 Start a Cluster

NOTE: This section applies to realm admins only.

As realm admin, start the cluster. Note that you must include the cluster name in the command:

```
redcli cluster start <cluster_name> [flags]
```

Example

The following example starts the **cluster-gray** cluster.

```
$ redcli cluster start cluster-gray
5:44PM INF Cluster cluster-gray is starting.
5:44PM INF Cluster cluster-gray status boot-join.
5:44PM INF Cluster cluster-gray status boot-join.
5:44PM INF Cluster cluster-gray status boot-replay.
5:44PM INF Cluster cluster-gray status running.
```

7.5 Stop a Cluster

NOTE: This section applies to realm admins only.

As realm admin, stop the cluster. Note that you must include the cluster name in the command:

```
redcli cluster stop <cluster_name> [flags]
```

Example

The following example stops the **cluster-gray** cluster.

```
$ redcli cluster stop cluster-gray
5:42PM INF Cluster cluster-gray stopping.
5:42PM INF Cluster cluster-gray is down.
```

7.6 Show Status and Health of a Cluster

NOTE: This section applies to realm admins only.

Use `redcli cluster show` to show the status or health of a cluster by passing the `-s (--status)` or `-H (--health)` flag, respectively.

7.6.1 Show Status of a Cluster

As realm admin, show the status of a cluster using `redcli cluster show` by passing the `-s (--status)` flag. Note that DDN Infinia 1.1 supports only one cluster, and you can omit the cluster name in the command:

```
redcli cluster show -s [flags]
```

Example

The following example shows status for the auto-selected cluster named `cluster-gray`.

```
$ redcli cluster show -s
11:20PM INF Auto Selecting cluster: cluster-gray
```

| CLUSTER NAME | CLUSTER STATE | FEATURE VERSION | CSE | SCE | CCE | RDE | #INSTANCES | #CATS | CAPACITY |
|--------------|---------------|-----------------|-----|-----|-----|-----|------------|-------|----------|
| cluster-gray | running | 1.1.0-alpha.1 | 12 | 15 | 15 | 8 | 4 | 16 | 2.078 TB |

| INSTANCE ID | HOSTNAME | INSTANCE UUID | SW VERSION | LIVENESS | INSTANCE STATE |
|-------------|------------|--------------------------------------|------------|----------|----------------|
| 1 | test-doc-1 | 087d11c7-47ba-44c0-8c95-d891a7ecacb1 | 1.1.0 | present | Joined |
| 2 | test-doc-2 | cd44611d-8574-497e-a3de-5726eef5e55d | | | Joined |
| 3 | test-doc-3 | f8ad1435-59ea-465e-89cb-79bbc76c334e | | | Joined |
| 4 | test-doc-4 | 84527239-c018-498b-b3a5-94b93f40266a | | | Joined, Leader |

Table 6 describes the fields provided in the command output.

Table 6. Show Status of a Cluster

| Field | Description |
|---------------|--|
| Cluster Name | Cluster for which status is provided |
| Cluster State | Current state of the cluster. Valid cluster states: <ul style="list-style-type: none"> • boot-join—booting (waiting for joins) • boot-replay—cluster wide replay occurring • dlm-init—cluster wide Distributed Lock Manager (DLM) lock initialization and replay • service-init—cluster wide meta/app initialization • running—running state • shutting-down—cluster is in shutting down state • shutdown—cluster has shutdown |

Table 6. Show Status of a Cluster

| Field | Description |
|-----------------|--|
| Feature Version | The admin-controlled version that is fully deployed across the cluster and instances are allowed to enable features |
| CSE | Cluster state epoch |
| SCE | System Configuration Epoch is incrementing the version number of the cluster state |
| CCE | CAT Configuration Epoch or Committed Configuration Epoch is the SCE in which all CAT configuration has been enacted by all instances |
| RDE | Replay Done Epoch is the SCE in which replay is completed |
| #Instances | Number of instances in the cluster |
| #CATS | Number of CATs in the cluster |
| Capacity | Total storage capacity |

| Field | Description |
|----------------|--|
| Instance ID | Instance identifier |
| Host Name | Host name |
| Instance UUID | Universal unique identifier for the instance |
| SW Version | Software version |
| Liveness | <p>Liveness of an instance. Provides relative state of communication with other instances in the cluster. If the instance,</p> <ul style="list-style-type: none"> • Responds promptly then the field shows present. • Falls behind then the field shows laggard. • Falls behind too far then the field shows absent. |
| Instance State | <p>Current state of the instance.</p> <p>Following are available instance states:</p> <ul style="list-style-type: none"> • Joined—Instance has joined the cluster • Joined (Replay in progress)—Instance has joined the cluster and replay is in progress • Evicted—Instance has evicted from the cluster |

7.6.2 Show Health of a Cluster

As realm admin, show the health of a cluster using **redcli cluster show** by passing the **-H (--health)** flag. Note that DDN Infinia 1.1 supports only one cluster, and you can omit the cluster name in the command:

```
redcli cluster show -H [flags]
```

Example

The following example shows health for the auto-selected cluster named **cluster-gray**.

```
$ redcli cluster show -H
```

```
11:20PM INF Auto Selecting cluster: cluster-gray
```

| CLUSTER | | | | | | INSTANCES | | | | | |
|---------|-------|-----|-----|-----|-----|-----------|--------|--------|---------|------|--------|
| STATE | EPOCH | SCE | CCE | RDE | EDE | Valid | Joined | Replay | Evicted | Left | Failed |
| running | 54 | 54 | 54 | 50 | 0 | 6 | 6 | 0 | 0 | 0 | 0 |

| CAT STATUS | | | | | | | |
|------------|----------------|--------------|---------|--------|----------|-------|-----------|
| Degraded | Delete Pending | Delete Ready | Evicted | Joined | Readable | Valid | Writeable |
| 0 | 0 | 0 | 0 | 72 | 72 | 72 | 72 |

| TABLES | | | | | | |
|---------|-----------|----------|------|---------|--------|------------------|
| POOL ID | POOL NAME | TABLE ID | CATS | EVICTED | HEALTH | REBALANCING CATS |
| 1 | SYSTEM | 1 | 72 | 0 | Green | 0 |
| 1 | SYSTEM | 2 | 72 | 0 | Green | 0 |
| 1 | SYSTEM | 3 | 72 | 0 | Green | 0 |

| REBUILDING CATS | REDUNDANCY | WIDTH |
|-----------------|------------|-------|
| 0 | 3 | 4 |
| 0 | 2 | 3 |
| 0 | 3 | 4 |

Table 7 describes the fields provided in the command output.

Table 7. Show Health of a Cluster

| Field | | Description |
|------------|----------------|--|
| Cluster | State | Current state of the cluster. For the valid cluster states, see Valid cluster states : |
| | Epoch | Cluster state epoch |
| | SCE | System Configuration Epoch is incrementing the version number of the cluster state |
| | CCE | CAT Configuration Epoch or Committed Configuration Epoch is the SCE in which all CAT configuration has been enacted by all instances |
| | RDE | Replay Done Epoch is the SCE in which replay is completed |
| | EDE | An Erasure Done Epoch (EDE) is a cluster epoch in which all broken erasure-coded bulk data affected by failed CATs are rebuilt |
| Instances | Valid | Number of valid instances in the cluster |
| | Joined | Number of instances joined to the cluster |
| | Replay | Number of instances that needs replay |
| | Evicted | Number of instances evicted from the cluster |
| | Left | Number of instances that left the cluster |
| | Failed | Number of instances that are not communicating with the rest of the instances in the cluster |
| CAT Status | Degraded | Number of CATs in a degraded state (evicted or lost either read or write capability) |
| | Delete Pending | Number of CATs in which the delete operation is pending |
| | Delete Ready | Number of CATs that are ready to be deleted from the cluster |
| | Evicted | Number of CATs evicted from the cluster |
| | Joined | Number of CATs that have joined the cluster |
| | Readable | Number of CATs with read capability |
| | Valid | Number of valid CATs |
| | Writeable | Number of CATs with write capability |

Table 7. Show Health of a Cluster

| Field | | Description |
|--------|------------------|--|
| Tables | Pool ID | Pool identifier |
| | Pool Name | Pool name |
| | Table ID | Table identifier |
| | CATs | Number of CATs |
| | Evicted | Number of CATs evicted |
| | Health | Current health status |
| | Rebalancing CATs | Number of CATs involved in re-balancing as part of the planned deletions of CATs |
| | Rebuilding CATs | Number of CATs undergoing failure recovery |
| | Redundancy | Maximum CAT failures allowed without causing a data loss |
| | Width | Width corresponding to number of replica or data + parity |

7.7 Show Inventory Details

NOTE: This section applies to realm admins only.

As realm admin, show a summary of inventory data. By default the output is in tabular format:

```
redcli inventory show [flags]
```

Example

The following example shows the inventory data in the default tabular output format.

```
$ redcli inventory show
```

| NODE | HOSTNAME | CPU COUNT | MEMORY | DEVICE COUNT | TOTAL CAPACITY |
|--------------------------------------|----------|-----------|-----------|--------------|----------------|
| 6fb335f6-4b01-4ea5-9f60-ca6296f90ebd | server1 | 2 | 12.47 GiB | 1 | 375 GiB |
| 079bf455-6d6b-4001-bc20-a02af635136e | server2 | 2 | 12.47 GiB | 1 | 375 GiB |
| 068c5bbf-974a-4d92-84db-8aaa50963519 | server3 | 2 | 12.47 GiB | 1 | 375 GiB |
| 6d797365-0065-412a-b166-0cfc69057dda | server 4 | 2 | 12.47 GiB | 1 | 375 GiB |

SUMMARY:

- Total Nodes: 4
- Total Cores: 8
- Total Memory: 49.86 GiB
- Total Devices: 4
- Total Capacity: 1.465 TiB

Table 8 describes the fields provided in the command output.

Table 8. Show Inventory Details

| Field | Description |
|----------------|--|
| Node | Universal unique identifier for the node |
| Hostname | Hostname |
| CPU Count | Number of CPUs on the node |
| Memory | Total available memory on the node |
| Device Count | Total storage devices on the node |
| Total Capacity | Total storage capacity of the node |

To show inventory details in yaml format, run **redcli inventory show** by passing **yaml** with the **-o (--output-format)** flag:

```
redcli inventory show -o yaml [flags]
```

Example

The following example shows the inventory data in the yaml output format.

```
$ redcli inventory show -o yaml

path: inventory/
tree:
  devices:
    'red-h9tw-0000:00:04.0-Google-NVMe':
      nodes:
        46110a51-7017-481f-a802-38115e266532:
          bus_address: "0000:00:04.0"
          device_name: vfio0
      ...
```

For additional details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

8. Monitoring and Failure Handling

DDN Infinia provides extensive monitoring and failure handling capabilities, as described in the following sections:

- [Verify the Operational State of Your Cluster](#) (Section 8.1 on page 160)
- [Monitor Realm, Realm Agent, and Node Status](#) (Section 8.2 on page 161)
- [Check Installed Versions](#) (Section 8.3 on page 166)
- [Watch for DDN Infinia Events](#) (Section 8.4 on page 168)
- [Check Configuration Endpoint Health Status](#) (Section 8.5 on page 172)
- [Collect and View DDN Infinia Logs](#) (Section 8.6 on page 173)
- [Debug Failures Using Tracing](#) (Section 8.7 on page 176)
- [Query CAT Health](#) (Section 8.8 on page 178)

8.1 Verify the Operational State of Your Cluster

NOTE: This section applies to realm admins only.

Verify your cluster is operational by executing the following procedure. Note that `kvcltest` in the procedure validates IO operations across cluster nodes, without actually issuing IO to storage devices. The test is useful to check that rpc communications are working between storage nodes.

Procedure

1. Run `redcli cluster show -s` command to show status of your cluster. For more information on cluster status, see [Show Status and Health of a Cluster](#) (Section 7.6 on page 152).
2. Run `redcli cluster show -H` command to show health of your cluster. For more information on cluster health, see [Show Status and Health of a Cluster](#) (Section 7.6 on page 152).
3. Run the `kvcltest` command to perform key value operations in your cluster:
 - a. Run `redcli kvcltest start` to start the KV Client test.

Example

```
$ redcli kvcltest start
10:21PM INF Auto Selecting cluster: cluster-gray
10:21PM INF Auto Selecting tenant: red
10:21PM INF Auto Selecting subtenant: red
10:21PM INF Auto Selecting dataset: red
10:21PM INF Successfully started redkvcl_test_rpc
```

- b. Run `redcli kvcltest status` and verify that the test started completes with no errors.

Example

```
$ redcli kvcltest status
10:21PM INF Auto Selecting cluster: cluster-gray
{
  "data": {
    "error": "3000 ops executed, 0 of them failed\n\n",
    "output": "T* I --- Completing all iterations: 1389 assign ops/sec; 2550 lookup ops/sec; ---\n",
    "status": "Complete"
  },
  "detail": "",
  "epoch": 1693606901,
  "timestamp": "Fri, 01 Sep 2023 22:21:41 GMT",
  "title": "Get KVCL test results"
}
```


8.2 Monitor Realm, Realm Agent, and Node Status

NOTE: This section applies to realm admins only.

Use the `redcli` to monitor realm, realm agent, and node status, as described in this section:

- [Monitor Realm Status](#) (Section 8.2.1 on page 162)
- [Monitor Realm Agents Status](#) (Section 8.2.2 on page 163)
- [Monitor Node Status](#) (Section 8.2.3 on page 164)

8.2.1 Monitor Realm Status

Query the container status of all the nodes in a realm from any storage or client node that has the **redcli** packages installed:

```
redcli realm status [flags]
```

Example

The following example shows the status in the default tabular output format.

```
$ redcli realm status
```

| Hostname | Component | State | Status | Image |
|----------------|------------|----------|-------------|------------------------------|
| infinia_server | etcd | running | Up 14 hours | red/config:1.0.0_amd64 |
| | hmi | running | Up 14 hours | red/core:1.0.0-user.1_amd64 |
| | loki | running | Up 14 hours | red/events:1.0.0_amd64 |
| | redagent | running* | Up 14 hours | red/core:1.0.0-user.1_amd64 |
| | redapi | running | Up 14 hours | red/core:1.0.0-user.1_amd64 |
| | reds3 | running | Up 14 hours | red/core:1.0.0-user.1_amd64 |
| | redsupport | down | down | unavailable |
| | redui | running | Up 14 hours | red/redui:1.0.0-user.1_amd64 |
| | rregistry | running | Up 14 hours | red/images:1.0.0_amd64 |
| | tsdb | running | Up 14 hours | red/metrics:1.0.0_amd64 |

8.2.2 Monitor Realm Agents Status

Query the status of the components from any storage or client node that has the **redcli** packages installed:

```
redcli realm agent-status [flags]
```

Example

The following example shows the status in the default tabular output format.

```
$ redcli realm agent-status
```

| Cluster: cluster1 | | | |
|-------------------|--------------|--------|--------|
| AGENT | | | |
| Hostname | SubType / Id | Status | Uptime |
| infinia_server | reds3 | Online | 13h55m |
| | instance [1] | Online | 13h56m |

| DAEMON | | | | | | |
|--------|---------|---------------|------------------|---------|--------|-------------|
| PID | Respawn | Respawn Count | Respawn Interval | State | Uptime | Core Dumped |
| 59 | true | 0 | 10 | running | 13h54m | no |
| 853 | true | 1 | 10 | running | 00m30s | yes |

8.2.3 Monitor Node Status

Query the status of a node from any storage or client node that has the **redcli** packages installed:

```
redcli realm node-status [flags]
```

Example

The following example shows the status in the default tabular output format.

```
$ redcli realm node-status
```

| NODE STATUS DETAILS | | | | |
|---------------------|-------------|---------------------|-----------------|--------------------|
| HOSTNAME | HOSTIP | UPSINCE | KERNELRELEASE | OPERATINGSYSTEM |
| server1 | 10.138.0.4 | 2023-05-18 18:52:18 | 5.19.0-1022-gcp | Ubuntu 24.04.1 LTS |
| server2 | 10.138.0.6 | 2023-05-18 18:50:53 | | |
| server3 | 10.138.0.8 | 2023-05-18 18:50:46 | | |
| server4 | 10.138.0.31 | 2023-05-18 18:59:01 | | |

| COMPONENT STATUS | | | | |
|------------------|------------|--------------------|------------|---------------------|
| HOSTNAME | HOSTIP | IMAGE | NAME | STATE |
| server1 | 10.138.0.4 | | redsupport | unknown |
| | | red/core:1.0.0.3 | redapi | Up 4 days |
| | | red/redui:1.0.0.3 | redui | |
| | | red/core:1.0.0.3 | redagent | |
| | | | hmi | |
| | | red/metrics:latest | tsdb | |
| | | red/events:latest | loki | |
| | | red/core:1.0.0.3 | reds3 | Up 4 days (healthy) |
| | | red/images:latest | rregistry | Up 4 days |
| | | red/config:latest | etcd | |
| server2 | 10.138.0.6 | | redsupport | unknown |
| | | red/core:1.0.0.3 | reds3 | Up 4 days (healthy) |
| | | red/metrics:latest | tsdb | Up 4 days |
| | | red/core:1.0.0.3 | hmi | |
| | | | redagent | |
| | | red/redui:1.0.0.3 | redui | |
| | | red/core:1.0.0.3 | redapi | |
| | | red/events:latest | loki | |
| | | red/config:latest | etcd | |

| | | | | |
|---------|-------------|--------------------|------------|---------------------|
| server3 | 10.138.0.8 | red/events:latest | loki | |
| | | red/config:latest | etcd | |
| | | red/core:1.0.0.3 | redapi | |
| | | red/reui:1.0.0.3 | reui | |
| | | red/core:1.0.0.3 | redagent | |
| | | | hmi | |
| | | red/metrics:latest | tsdb | |
| | | | redsupport | unknown |
| | | red/core:1.0.0.3 | reds3 | Up 4 days (healthy) |
| server4 | 10.138.0.31 | red/reui:1.0.0.3 | reui | Up 4 days |
| | | | redsupport | unknown |
| | | red/core:1.0.0.3 | reds3 | Up 4 days (healthy) |
| | | red/events:latest | loki | Up 4 days |
| | | red/metrics:latest | tsdb | |
| | | red/core:1.0.0.3 | hmi | |
| | | | redagent | |
| | | | redapi | |

8.3 Check Installed Versions

NOTE: This section applies to realm admins only.

Query the installed versions:

```
redcli version show [flags]
```

Example

The following example shows the installed versions in the default tabular output format.

| \$ redcli version show | | | | |
|------------------------|----------------|---------------------------------|-----------|--------------|
| NAME | HOST | IMAGE BUILD DATE | COMMIT ID | VERSION |
| etcd | 250.60.68.181 | - | - | - |
| | 250.60.28.69 | - | - | - |
| | 250.60.208.188 | - | - | - |
| hmi | 250.60.68.181 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.28.69 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.115.73 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.208.188 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| loki | 250.60.208.188 | - | - | - |
| | 250.60.28.69 | - | - | - |
| | 250.60.68.181 | - | - | - |
| | 250.60.115.73 | - | - | - |
| redagent | 250.60.208.188 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.68.181 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.115.73 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.28.69 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| redapi | 250.60.28.69 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.68.181 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.115.73 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.208.188 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| redcli | Local node | Fri Mar 22 10:56:53 PM UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| reds3 | 250.60.28.69 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.68.181 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.208.188 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.115.73 | Fri Mar 22 22:50:19 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| redsetup | Local node | Fri Mar 22 10:56:50 PM UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |

| | | | | |
|------------|----------------|------------------------------|----------|--------------|
| redsupport | 250.60.68.181 | - | - | - |
| | 250.60.208.188 | - | - | - |
| | 250.60.115.73 | - | - | - |
| | 250.60.28.69 | - | - | - |
| redui | 250.60.28.69 | Fri Mar 22 23:02:36 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.68.181 | Fri Mar 22 23:02:36 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.115.73 | Fri Mar 22 23:02:36 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| | 250.60.208.188 | Fri Mar 22 23:02:36 UTC 2024 | 6ed0a65a | 1.0.5-beta.1 |
| rregistry | 250.60.68.181 | - | - | - |
| tsdb | 250.60.208.188 | - | - | - |
| | 250.60.68.181 | - | - | - |
| | 250.60.28.69 | - | - | - |
| | 250.60.115.73 | - | - | - |

8.4 Watch for DDN Infinia Events

NOTE: This section applies to realm admins only.

DDN Infinia produces output for important *events* that can be read in machine readable or user-friendly formats. Each DDN Infinia instance logs these events first in its local log file on the host. By default, all events are persisted in a local event database.

Events are categorized as:

- **admin** – [Admin Events](#) (Section 8.4.1 on page 169)
- **cluster** – [Cluster Events](#) (Section 8.4.2 on page 169)
- **hardware** – [Hardware Events](#) (Section 8.4.3 on page 170)
- **system** – [System Events](#) (Section 8.4.4 on page 171)

View these different types of events using `redcli events {admin|cluster|hardware|system}`.

Key Points

The following are key points about events:

- By default, the `redcli events` command returns all events. To specify a time frame, use the `-s, --starttime` and `-e, --endtime` options. The format for the time is `yyyy-mm-dd_hh:mm:ss`.
- Severity of the events include **EMERG**, **ALERT**, **CRITC**, **ERROR**, **WARNG**, **NOTIC**, **INFO**, and **DEBUG**.

8.4.1 Admin Events

View admin events:

```
redcli events admin [flags]
```

Example

The following example output shows all admin events in the default tabular output format.

```
$ redcli events admin
```

| DATE | SEVERITY | SOURCE | MESSAGE |
|--|----------|--------------------------------------|--|
| 2023-06-09 14:21:17.686320128+0000 UTC | INFO | f7f44122-c419-4f2a-a662-4f52ada25390 | User realm_admin with level realm executed GET /redapi/v1/events/system, response code: 200. |

For additional details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

8.4.2 Cluster Events

View cluster events:

```
redcli events cluster [flags]
```

The following are boot wait cluster events:

- Cluster <cluster_name> startup is waiting for CAT uuid <cat_uuid> hosted by instance uuid <instance_uuid>. The cluster is waiting on a CAT that was present in the prior boot of the cluster and may have up-to-date copies that can contribute to the data resiliency and continuity of the system.
- Cluster <cluster_name> startup is waiting for instance uuid <instance_uuid> hostname <hostname>. The cluster is waiting on an instance that was present in the prior boot of the cluster and may have hosted one or more CATs with up-to-date copies that can contribute to the data resiliency and continuity of the system.

| | |
|------------------------|--|
| BEST PRACTISES: | Make every effort to rectify the boot and join issues experienced by CATs and instances that are being waited on during cluster boot. The cluster boot waits on these issues because they may contribute to the data resiliency of the system. The loss of one or multiple instances or CATs (depending on the configured resiliency of the system) may result in data loss. |
|------------------------|--|

For additional details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

8.4.3 Hardware Events

NOTE:

For instructions about requesting DDN Infinia to reattempt to serve a repaired device or adding a new drive to replace a failed device, refer to the *Hardware Installation and Maintenance Guide* for your platform.

View hardware events:

```
redcli events hardware [flags]
```

When DDN Infinia encounters issues with a disk device, the local instance that attaches to the disk may report one of the following hardware events:

- *CAT device uuid <cat_uuid> failed format. CAT device is now marked broken. Verify the physical device is attached and operational.*

The device has an operational or connection error that prevented it being formatted and added to the system. Check that the device is present at the host where format was attempted and that it can be successfully read and written. If the device is no longer operational, it should be removed from the DDN Infinia configuration.

- *CAT device uuid <cat_uuid> failed mount. CAT device is now marked broken. Verify the physical device is attached and operational.*

The device has an operational or connection error that prevented it being mounted to the system. Check that the device is present at the host where mount was attempted and that it can be successfully read and written. If the device is no longer operational, it should be removed from the DDN Infinia configuration.

- *CAT device uuid <cat_uuid> failed device enablement. Verify the physical device is attached and operational and the device setup string present in the runtime configuration of the CAT is correct.*

The device has an operational or connection error that prevented it being enabled to the system. Check that the device is present at the host where enabled was attempted and that it can be successfully read and written. If the device is no longer operational, it should be removed from the DDN Infinia configuration.

For additional details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the `-h` flag at the command line.

8.4.4 System Events

View system events:

```
redcli events system [flags]
```

Example

The following example output shows all system events in the default tabular output format.

```
$ redcli events system
```

| DATE | SEVERITY | SOURCE | MESSAGE |
|--|----------|--------------------------------------|--|
| 2023-09-28 14:33:04.14977792 +0000 UTC | INFO | 57372353-026a-4ff3-ace9-eed8b257e8e0 | config create request config_name=auto_config |
| 2023-09-28 14:32:07.506082048 +0000 UTC | NOTICE | e9a9c1ff-b6b3-473c-8727-93e2849b4b80 | Could not fully determine a valid JRPC port range from etcd - Setting to [3001-3001] |

For additional details about available flags and options, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

8.5 Check Configuration Endpoint Health Status

NOTE: This section applies to realm admins only.

From any of the nodes running the `config` container run the following commands and check the health status:

```
source /opt/ddn/red/red_aliases
ec endpoint health
```

Example

The following example output shows all endpoints are in the healthy state.

```
Creating etcd_etcd_run ... done
10.142.0.30:2379 is healthy: successfully committed proposal: took = 2.570627ms
10.142.0.31:2379 is healthy: successfully committed proposal: took = 1.922887ms
10.142.0.28:2379 is healthy: successfully committed proposal: took = 2.188093ms
```

8.6 Collect and View DDN Infinia Logs

NOTE: This section applies to realm admins only.

IMPORTANT: Log upload temporary space uses the `/var/tmp/red/uploads` directory. DDN recommends `/var/tmp` be a separate file system, not root fs, and depending on the size of realm be the greater of 20GB or 0.5GB allocated per node.

DDN Infinia produces a variety of debug and log data included in the following logs:

- setup log – `/var/log/red/redsetup-<hostuuid>/redsetup.log`
- instance log – `/var/log/red/<cluster_name>-<cluster_uuid>/<instance_number>/instance.log`
- REST API log – `/var/log/red/redapi-<hostuuid>/redapi.log`

By default, this data is stored under the `/var/log/red` directory.

8.6.1 Upload Cluster Logs

If required, collect log data to assist DDN Support in analyzing problems using `redcli logs upload` by passing the appropriate option:

```
redcli logs upload [flags]
```

By default, logs are uploaded to DDN Infinia Central backend, where they can be accessed by DDN development and support teams. When contacting DDN Support, provide the complete URL returned by this command.

`redcli logs upload` includes the following options:

- Specify a log collection level of heavy, medium, or light using the `-l` argument. The default level is medium:

```
redcli logs upload -l <level>
```

- Encode a unique case number or defect number in the log package using the `-C` option.

```
redcli logs upload -l <level> -C <case_id>
```

- Specify a local backend using the `-b` and `-d` options.

```
redcli logs upload -b <local_directory> -d /tmp/redlogs/
```

Examples

The following example collects the heavy level of logs.

```
redcli logs upload -R -l heavy
```

The following example encodes the case number RED-1234 into the name of the log package that is uploaded.

```
$ redcli logs upload -R -l heavy -C RED-1234
11:58PM INF Uploading log bundle
URL: gs://red-debug-data/red_realm/RED-1234/realm_20230307235804-5f6aa522-3699-43fd-b720-
fbb51b057832.tar.gz
```

For certain errors, more detailed log tracing may be required. For advanced log collection options, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

8.6.2 Manage Logs on Unhealthy Cluster

The log collection can fail because of multiple reasons. Some examples are failure in JRPC on an agent, failure of tar command, failure due to low memory, or failure due to network issues.

If the *redlog* file is missing in a log bundle, get the list of failed agents using **redcli task show <task_id>** and then run **sudo -E redcli logs upload -l currentnode** on the failed agents.

Example

The following example shows the log collection failed for site-10 agents and the log upload is in complete state.

| \$ redcli task list | | | | | | |
|--------------------------------------|-----------|----------------|----------|--|-------|--|
| Tasks | | | | | | |
| UUID | TYPE | DURATION (sec) | STATE | MESSAGE | ERROR | |
| 68230e59-3667-436c-b680-38763004d991 | logupload | 32 | complete | | | |
| cb7e0192-d156-4763-a783-a2fb5c5ce717 | logupload | 77 | complete | One or more log upload stage(s) failed. See 'redcli task show cb7e0192-d156-4763-a783-a2fb5c5ce717' for details. | | |

| \$ redcli task show cb7e0192-d156-4763-a783-a2fb5c5ce717 | | | | | | |
|--|--|-----------|----------------|----------|--|-------|
| UUID | NAME | TYPE | DURATION (sec) | STATE | MESSAGE | ERROR |
| cb7e0192-d156-4763-a783-a2fb5c5ce717 | Logs upload task: "/var/log/red/uploads/light_20240320084743-b8d43e4f-fcad-4e01-93eb-32162e3e0fbc.tar" | logupload | 77 | complete | One or more log upload stage(s) failed. See 'redcli task show cb7e0192-d156-4763-a783-a2fb5c5ce717' for details. | |

| STAGE | NAME | TRANSITION TIME | STATE | MESSAGE | ERROR |
|-------|------------------------|-------------------------------|----------|---------|--|
| 0 | Generate SOS data | 2024-03-20 08:48:46 +0000 UTC | complete | | |
| 1 | Collect log data | 2024-03-20 08:48:58 +0000 UTC | failed | | CallHome: Failed to gather agent logs for 'site-10' agents. You could try 'sudo -E redcli logs upload -l currentnode' on those agents. |
| 2 | Generate manifest File | 2024-03-20 08:48:58 +0000 UTC | complete | | |
| 3 | Compress and upload | 2024-03-20 08:48:59 +0000 UTC | complete | | |
| 4 | Cleanup files | 2024-03-20 08:49:00 +0000 UTC | complete | | |

Check for such local failures and if required, collect local logs using **sudo -E redcli logs upload -l currentnode** command.

8.7 Debug Failures Using Tracing

NOTE: This section applies to realm admins only.

DDN Infinia provides its own tracing facility for enhanced introspection of the system's operation. Many DDN Infinia code paths have unique tracing callouts (*tracepoints*) inserted at important junctures for recording the operation and changing the state of the system.

Individual tracepoints are usually categorized by *class* and *level*:

- *class* – A group of tracepoints usually encompassing a single code component within DDN Infinia or a small set of components performing tightly intertwined functions. For instance, there are trace classes defined for only tracing cluster management functions.
- *level* – Each tracepoint is assigned a level regarding its importance and frequency. Usually, a tracepoint assigned at level 0 conveys the most important system changes and occurs the least seldom in output frequency. Tracepoints at level 1 or higher may occur at higher output frequencies and contain additional detail about system operations.

DDN Support may direct administrators to enable or disable certain trace classes or trace levels.

Additionally, the tracing facility operates in multiple modes regarding how trace data is persisted for post trace examination. Internal to the DDN Infinia processes are memory buffer areas where trace data is initially recorded. If an enabled tracepoint is passed during program execution, the tracepoint identity and the data it wishes to record is saved into the buffer area, based on its *mode*:

- *stream* mode – Ensures all enabled classes and levels in this mode are persistently saved to the specified file. If the enabled tracepoint and level are in *stream* mode, the buffered data is saved persistently to a file periodically.
- *wrap* mode – Only the last trace data that is present in the memory buffer after the last wrap is saved to the specified file on program exit. If the enabled tracepoint and level are in *wrap* mode, new trace data wraps back to the beginning of the available buffer space when it is filled. Only when the program exits will the trace data in wrap mode be written.

The **redcli trace {dump|inject|set|show}** provides the following command options:

- ❖ **dump** – Dump traces
- ❖ **inject** – Inject traces
- ❖ **set** – Set traces
- ❖ **show** – Show trace status

For details about these options, refer to the *DDN Infinia CLI Reference* or use the **-h** flag at the command line.

The following are example commands that DDN Support may request an administrator to invoke:

- List the trace settings, along with all the trace classes:

```
redcli trace show [flags]
```

- Set a trace:

```
redcli trace set {action|tag|fmtoption|stream|wrap} [flags]
```


Example

In the first example, the command enables streaming traces with an internal memory buffer size of 256k:

```
redcli trace set stream -e -z 262144 -y 1
```

In the second example, the command requests all classes to wrap tracepoints at level 1, 2, and 3:

```
redcli trace set action wrap -w 1,2,3
```

8.8 Query CAT Health

NOTE:

This section applies to realm admins only. For instructions about requesting DDN Infinia to reattempt to serve a repaired drive or adding a new drive to replace a failed drive, refer to the *Hardware Installation and Maintenance Guide* for your platform.

When DDN Infinia encounters issues with a disk device, the local instance that attaches to the disk may report hardware events noting the issue with the CAT and underlying device. For a list of these events, see [Hardware Events](#) (Section 8.4.3 on page 170). If no Instance can successfully serve the CAT, the CAT health will be noted with a failed status both on all failed Instances that attempted to serve it and on the CAT global health itself.

View the CAT health details in your DDN Infinia cluster using the following procedure.

Procedure

1. List all CATs in your DDN Infinia cluster:

```
redcli cat list
```

Example

```
$ redcli cat list
```

| CAT | DEVICE NAME | RAW CAPACITY | HOSTNAME | POOL | STATE | CAPABILITIES |
|-----|------------------------------------|--------------|----------|--------|--------|--------------|
| 1 | server1-0000:00:04.0n0-Google-NVMe | 402.7 GB | server1 | SYSTEM | Joined | READ & WRITE |
| 2 | server2-0000:00:04.0n0-Google-NVMe | 402.7 GB | server2 | SYSTEM | Joined | READ & WRITE |
| 3 | server3-0000:00:04.0n0-Google-NVMe | 402.7 GB | server3 | SYSTEM | Joined | READ & WRITE |
| 4 | server4-0000:00:04.0n0-Google-NVMe | 402.7 GB | server4 | SYSTEM | Joined | READ & WRITE |

| UUID |
|--------------------------------------|
| 372b81fa-9be7-433a-9d3e-9def293cf03f |
| 737a63c6-4f4c-459c-be6b-97bf291c2723 |
| 748b2b64-f62f-4c19-8796-a2b071000add |
| abc879f2-e044-4ec5-9794-da64e9b046b0 |

Total capacity: 1.611 TB

2. If a CAT has an issue, view details of the CAT by specifying its UUID, device name, or index number:

```
redcli cat show <uuid>
```

or

```
redcli cat show <device_name>
```

or

```
redcli cat show -i <cat_id>
```

Example

The following example shows the details of the CAT by specifying its device name.

```
$ redcli cat show server1-0000:00:04.0n0-Google-NVMe
```

| CAT | DEVICE NAME | RAW CAPACITY | HOSTNAME | POOL | STATE | CAPABILITIES |
|-----|------------------------------------|--------------|----------|--------|--------|--------------|
| 1 | server1-0000:00:04.0n0-Google-NVMe | 402.7 GB | server1 | SYSTEM | Joined | READ & WRITE |

| |
|------|
| UUID |
|------|

| |
|--------------------------------------|
| 372b81fa-9be7-433a-9d3e-9def293cf03f |
|--------------------------------------|

Appendix A Realm Configuration Input Parameters

The supported parameters of the realm configuration file are documented in the realm config schema.

The following is an example in yaml format.

```
## Request Schema (application/json)
properties:
  type: object
  realm:
    required:
      - etcd_addresses
    type: object
    properties:
      name:
        type: string
        description: "realm name"
      etcd_addresses:
        description: "Etcd addresses internal or external"
        type: array
        items:
          type: string
          example: '["10.128.0.41:2379", "10.128.0.42:2379", "10.128.0.43:2379"]'
      realm_keyspace_name:
        type: string
        description: "Identifier for ETCD keyspace for use at this realm."
        default: storage_001
      admin_user:
        type: string
        description: "Identifier for ETCD admin user for this realm."
        default: realm_admin
      unsecured:
        description: "Set true to disable etcd security"
        default: false
        type: boolean
      etcd_image_version:
        type: string
        description: "ETCD image version."
        default: latest
      red_image_version:
        type: string
        description: "DDN Infinia image version."
        default: latest
    nodes:
      type: array
      description: "realm nodes"
      items:
        type: object
        properties:
          address:
            description: "Node internal address"
            type: string
            example: "10.128.0.41"
          roles:
            type: array
            description: "List of services that should be started on this node."
            items:
              type: string
              example: '["etcd", "hmi", "redapi", "redagent", "redclient", "reui", "tsdb", "wbb"]'
          spdk_allowed:
            type: array
```

```

description: "List of device pci addresses to use for DDN Infinia
storage. If no value is passed, all unused, unpartitioned devices will
be used by default. This can be modified after initial configuration
is complete."
items:
  type: string
example: '["0000:08:00.0", "0000:09:00.0", "0000:0a:00.0",
"0000:0b:00.0"]'
hugemem:
  type: integer
description: "Memory, in MB, to allocate in hugepages for DDN Infinia
daemon. If no value is passed, 2GB per device, plus 2GB additional
will be allocated by default, limited to 70% of system memory. This
can be modified after initial configuration is complete."
example: 21504
client_net_cpulist:
  type: string
description: "Integer list, including ranges, specifying CPU codes
allocated to the client network pollers." example: "24,28,30-31"

```

Appendix B DDN Infinia S3 DataService API Endpoints

This appendix provides details about the following:

- [S3 Permissions and Access Management](#) (Appendix B.1 on page 183)
- [S3 Bucket and Object Management](#) (Appendix B.2 on page 183)
- [Key Name Characters](#) (Appendix B.3 on page 184)
- [General Limitations](#) (Appendix B.4 on page 185)
- [Supported S3 API Calls](#) (Appendix B.5 on page 186)
- [S3 API Details](#) (Appendix B.6 on page 188)
- [S3-Compatible API Error Codes](#) (Appendix B.7 on page 243)
- [URL Processing Rules](#) (Appendix B.8 on page 243)
- [AWS CLI Compatibility](#) (Appendix B.9 on page 244)

B.1 S3 Permissions and Access Management

DDN Infinia supports defining policies via the S3 API using the PutBucketPolicy API (see [PutBucketPolicy](#)). S3 ACLs also can be used to define user or group access at the bucket or object level. Buckets or objects created in DDN Infinia are created with a default ACL that assumes full control of the object or bucket by the creator. If required, a non-default ACL can be supplied in the create to override the default ACL creation.

B.2 S3 Bucket and Object Management

DDN Infinia S3 DataServices conform to [S3 protocol standards](#), namely:

- Bucket names must be between 3 (min) and 63 (max) characters long.
- Bucket names can consist only of lowercase letters, numbers, dots (.), and hyphens (-).
- Bucket names must begin and end with a letter or number.
- Bucket names must not contain two adjacent periods.
- Bucket names must not be formatted as an IP address (for example, 192.168.5.4).
- Bucket names must not start with the prefix xn--.
- Bucket names must not end with the suffix -s3alias. This suffix is reserved for access point alias names.

NOTE: DDN Infinia supports up to 10,000 buckets per subtenant. Bucket tagging and bucket logging are not supported in DDN Infinia 1.1.

Object naming also follows [AWS compatibility standards](#), and the following guidelines are recommended to ensure compatibility:

- Objects with a prefix of "/" must be uploaded or downloaded with the AWS Command Line Interface (AWS CLI), AWS SDKs, or REST API. You cannot use the Amazon S3 console.
- Objects with a prefix of "../" cannot be uploaded using the AWS Command Line Interface (AWS CLI) or Amazon S3 console.

B.3 Key Name Characters

Character Special Handling

The following characters in a key name might require additional code handling and likely need to be URL encoded or referenced as HEX. Some of these are non-printable characters that your browser might not handle, which also requires special handling:

- Ampersand ("&")
- Dollar ("\$")
- ASCII character ranges 00–1F hex (0–31 decimal) and 7F (127 decimal)
- 'At' symbol ("@")
- Equals ("=")
- Semicolon (";")
- Forward slash ("/")
- Colon (":")
- Plus ("+")
- Space – Significant sequences of spaces might be lost in some uses (especially multiple spaces)
- Comma (",")
- Question mark ("?")

Characters to Avoid

Avoid the following characters in a key name because of significant special handling for consistency across all applications.

- Backslash ("\")
- Left curly brace ("{")
- Non-printable ASCII characters (128–255 decimal characters)
- Caret ("^")
- Right curly brace ("}")
- Percent character ("%")
- Grave accent / back tick ("`")
- Right square bracket ("]")
- Quotation marks
- 'Greater Than' symbol (">")
- Left square bracket ("[")
- Tilde ("~")
- 'Less Than' symbol ("<")
- 'Pound' character ("#")
- Vertical bar / pipe ("|")

B.4 General Limitations

DDN Infinia supports encryption at rest on a per cluster basis and does not support SSE via the S3 API.

The following S3 features are not supported in DDN Infinia:

- Storage classes
- Object checksums
- Bucket or object expiration
- S3 website endpoints
- IPv6
- Bucket tagging

The following are key length restrictions:

- Maximum key length is 1024 bytes
- Maximum key component ("file/dir name") length is 234 bytes

NOTE: The DDN Infinia S3 DataService supports only HTTPS connections.

B.5 Supported S3 API Calls

Table 9. Supported S3 API Calls

| Name | Method | URI | Source File |
|---|--------|---|------------------------|
| AbortMultipartUpload | DELETE | /bucket/{key}?uploadId={upload_id} | s3_abortmultiup.cpp |
| CompleteMultipartUpload | POST | /bucket/{key}?uploadId={upload_id} | s3_completemultiup.cpp |
| CopyObject | PUT | /bucket/{key} (header: x-amz-copy-source) | s3_copyobject.cpp |
| CreateBucket | PUT | /bucket | s3_createbucket.cpp |
| CreateMultipartUpload | POST | /bucket/{key}?uploads | s3_createmultiup.cpp |
| DeleteBucket | DELETE | /bucket | s3_deletebucket.cpp |
| DeleteBucketPolicy | DELETE | /bucket?policy | s3_deletesubres.cpp |
| DeleteBucketTagging | DELETE | /bucket?tagging | s3_deletesubres.cpp |
| DeleteObject | DELETE | /bucket/{key} | s3_deleteobject.cpp |
| DeleteObjectTagging | DELETE | /bucket/{key}?tagging | s3_deletesubres.cpp |
| DeleteObjects | POST | /bucket/?delete | s3_deleteobjects.cpp |
| GetBucketAcl | GET | /bucket?acl | s3_getsubres.cpp |
| GetBucketLocation | GET | /bucket?location | s3_getsubres.cpp |
| GetBucketPolicy | GET | /bucket?policy | s3_getsubres.cpp |
| GetBucketTagging | GET | /bucket?tagging | s3_getsubres.cpp |
| GetBucketVersioning | GET | /bucket?versioning | s3_getsubres.cpp |
| GetObject | GET | /bucket/{key} | s3_getobject.cpp |
| GetObjectTagging | GET | /bucket/{key}?tagging | s3_getsubres.cpp |
| GetObjectAcl | GET | /bucket/{key}?acl | s3_getsubres.cpp |
| HeadBucket | HEAD | /bucket | s3_headbucket.cpp |
| HeadObject | HEAD | /bucket/{key} | s3_getobject.cpp |
| ListBuckets | GET | / | s3_listbuckets.cpp |
| ListMultipartUploads | GET | /bucket?uploads | s3_listobjects.cpp |
| ListObjects | GET | /bucket | s3_listobjects.cpp |
| ListParts | GET | /bucket/{key}?uploadId={upload_id} | s3_listparts.cpp |
| PutBucketAcl | PUT | /bucket?acl | s3_putsubres.cpp |
| PutBucketPolicy | PUT | /bucket?policy (Policy in JSON format) | s3_putsubres.cpp |

Table 9. Supported S3 API Calls

| Name | Method | URI | Source File |
|---------------------|--------|---|-------------------|
| PutBucketTagging | PUT | /[bucket]?tagging | s3_putsubres.cpp |
| PutBucketVersioning | PUT | /[bucket]?versioning | s3_putsubres.cpp |
| PutObject | PUT | /[bucket]/[key] | s3_putobject.cpp |
| PutObjectTagging | PUT | /[bucket]/[key]?tagging | s3_putsubres.cpp |
| PutObjectAcl | PUT | /[bucket]/[key]?acl | s3_putsubres.cpp |
| UploadPart | PUT | /[bucket]/[key]?partNumber={part_number}&uploadId={upload_id} | s3_putobject.cpp |
| UploadPartCopy | PUT | /[bucket]/[key]?partNumber={part_number}&uploadId={upload_id} (header: x-amz-copy-source) | s3_copyobject.cpp |
| PostObject | POST | /[bucket] | s3_putobject.cpp |

B.6 S3 API Details

This section provides details about the following S3 API:

- [ListBuckets](#)
- [CreateBucket](#)
- [DeleteBucket](#)
- [HeadBucket](#)
- [GetBucketLocation](#)
- [PutBucketAcl](#)
- [GetBucketAcl](#)
- [ListObjects](#)
- [ListObjectsV2](#)
- [ListMultipartUploads](#)
- [PutObject](#)
- [PutObjectTagging](#)
- [GetObject](#)
- [GetObjectTagging](#)
- [HeadObject](#)
- [DeleteObject](#)
- [DeleteObjectTagging](#)
- [DeleteObjects](#)
- [PutObjectAcl](#)
- [GetObjectAcl](#)
- [CreateMultipartUpload](#)
- [UploadPart](#)
- [CompleteMultipartUpload](#)
- [AbortMultipartUpload](#)
- [ListParts](#)
- [PostObject](#)
- [CopyObject](#)
- [UploadPartCopy](#)
- [PutBucketPolicy](#)
- [PutBucketVersioning](#)
- [GetBucketVersioning](#)
- [GetBucketPolicy](#)
- [DeleteBucketPolicy](#)

B.6.1 ListBuckets

List user buckets

[AWS Reference](#)

Request Syntax

```
GET / HTTP/1.1
Host: endpoint
```

Response Syntax

```
HTTP/1.1 200 OK
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ListAllMyBucketsResult>
  <Buckets>
    <Bucket>
      <CreationDate>timestamp</CreationDate>
      <Name>string</Name>
    </Bucket>
  </Buckets>
  <Owner>
    <DisplayName>string</DisplayName>
    <ID>string</ID>
  </Owner>
</ListAllMyBucketsResult>
```

Permissions

Anonymous requests are rejected with 403 Unauthorized. Only buckets owned by authenticated user are included into the response.

Response

| Parameter | Implemented | Comments |
|-------------------------|-------------|----------|
| Buckets | yes | |
| Buckets >> CreationDate | yes | |
| Buckets >> Name | yes | |
| Owner | yes | |
| Owner >> DisplayName | yes | |
| Owner >> ID | yes | |

B.6.2 CreateBucket

Create new bucket

[AWS Reference](#)

Request Syntax

```
PUT /bucket HTTP/1.0
Host: endpoint
x-amz-acl: canned-acl
x-amz-grant-full-control: grantee
x-amz-grant-read: grantee
x-amz-grant-read-acl: grantee
x-amz-grant-write: grantee
x-amz-grant-write-acp: grantee

<?xml version="1.0" encoding="UTF-8"?>
<CreateBucketConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <LocationConstraint>string</LocationConstraint>
</CreateBucketConfiguration>
```

Response Syntax

```
HTTP/1.1 200 OK
```

Permissions

Any authenticated user is permitted to create a bucket.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| ACL - x-amz-acl | Yes | |
| GrantFullControl - x-amz-grant-full-control | Yes | |
| GrantRead - x-amz-grant-read | Yes | |
| GrantReadACP - x-amz-grant-read-acp | Yes | |
| GrantWrite - x-amz-grant-write | Yes | |
| GrantWriteACP - x-amz-grant-write-acp | Yes | |
| ObjectLockEnabledForBucket - x-amz-bucket-object-lock-enabled | No | |
| ObjectOwnership - x-amz-object-ownership | No | |

Request Body

| Parameter | Implemented | Comments |
|---|-------------|---|
| CreateBucketConfiguration | Yes | XML document validated and parsed but ignored |
| CreateBucketConfiguration >> LocationConstraint | Yes | |

Response

| Parameter | Implemented | Comments |
|-----------|-------------|----------|
| Location | No | |

B.6.3 DeleteBucket

Delete empty bucket

[AWS Reference](#)

Request Syntax

```
DELETE /bucket HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.1 204 No Content
```

Permissions

Bucket owner or any user granted FULL_CONTROL permission on the bucket is permitted to delete the bucket.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

B.6.4 HeadBucket

Check if the bucket exists

[AWS Reference](#)

Request Syntax

```
HEAD /bucket HTTP/1.0
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.1 200 OK
```

Permissions

Bucket owner or any user granted READ permission on the bucket is permitted to check the bucket exists.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

B.6.5 GetBucketLocation

[AWS Reference](#)

Request Syntax

```
GET /bucket?location HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.0 200 OK

<?xml version="1.0" encoding="UTF-8"?>
<LocationConstraint>string</LocationConstraint>
```

Permissions

Only bucket owner is permitted to get bucket location.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

Response

| Parameter | Implemented | Comments |
|--------------------|-------------|--------------------------|
| LocationConstraint | Yes | Value is always location |

B.6.6 PutBucketAcl

[AWS Reference](#)

Request Syntax

```
PUT /bucket?acl HTTP/1.0
Host: endpoint
x-amz-acl: canned-acl
x-amz-grant-full-control: grantee
x-amz-grant-read: grantee
x-amz-grant-read-acl: grantee
x-amz-grant-write: grantee
x-amz-grant-write-acp: grantee
x-amz-expected-bucket-owner: owner-ID

<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <AccessControlList>
    <Grant>
      <Grantee>
        <DisplayName>string</DisplayName>
        <EmailAddress>string</EmailAddress>
        <ID>string</ID>
        <xsi:type>string</xsi:type>
        <URI>string</URI>
      </Grantee>
      <Permission>string</Permission>
    </Grant>
    ...
  </AccessControlList>
  <Owner>
    <DisplayName>string</DisplayName>
    <ID>string</ID>
  </Owner>
</AccessControlPolicy>
```

Permissions

Bucket owner or any user granted WRITE_ACP permission on the bucket is permitted to set bucket ACL.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| ACL - x-amz-acl | Yes | |
| ChecksumAlgorithm - x-amz-sdk-checksum-algorithm | No | |
| Content-MD5 | No | |
| GrantFullControl - x-amz-grant-full-control | Yes | |
| GrantRead - x-amz-grant-read | Yes | |
| GrantReadACP - x-amz-grant-read-acp | Yes | |
| GrantWrite - x-amz-grant-write | Yes | |
| GrantWriteACP - x-amz-grant-write-acp | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

Request Body

| Parameter | Implemented | Comments |
|-------------------------------|-------------|---|
| AccessControlPolicy | Yes | You cannot specify both the body and the request headers, request is rejected |
| AccessControlPolicy >> Grants | Yes | |
| AccessControlPolicy >> Owner | Yes | |

B.6.7 GetBucketAcl

[AWS Reference](#)

Request Syntax

```
GET /bucket?acl HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <AccessControlList>
    <Grant>
      <Grantee>
        <DisplayName>string</DisplayName>
        <EmailAddress>string</EmailAddress>
        <ID>string</ID>
        <xsi:type>string</xsi:type>
        <URI>string</URI>
      </Grantee>
      <Permission>string</Permission>
    </Grant>
    ...
  </AccessControlList>
  <Owner>
    <DisplayName>string</DisplayName>
    <ID>string</ID>
  </Owner>
</AccessControlPolicy>
```

Permissions

Bucket owner or any user granted READ_ACP permission on the bucket is permitted to get bucket ACL.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

Response

| Parameter | Implemented | Comments |
|-----------|-------------|----------|
| Grants | Yes | |
| Owner | Yes | |

B.6.8 ListObjects

[AWS Reference](#)

Request Syntax

```
GET /bucket?delimiter=/&encoding-type=url&marker=string&max-keys=integer&prefix=string HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Request Syntax

```
HTTP/1.1 200 OK
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
  <IsTruncated>boolean</IsTruncated>
  <Marker>string</Marker>
  <NextMarker>string</NextMarker>
  </Contents>
  <ETag>etag</ETag>
  <Key>string</Key>
  <LastModified>timestamp</LastModified>
  <Owner>
    <DisplayName>string</DisplayName>
    <ID>string</ID>
  </Owner>
  <Size>integer</Size>
</Contents>
...
<Name>string</Name>
<Prefix>string</Prefix>
<Delimiter>/</Delimiter>
<MaxKeys>integer</MaxKeys>
<CommonPrefixes>
  <Prefix>string</Prefix>
</CommonPrefixes>
...
<EncodingType>url</EncodingType>
</ListBucketResult>
```

Permissions

Bucket owner or any user granted READ permission on the bucket is permitted to list bucket objects.

Request

| Parameter | Implemented | Comments |
|---|-------------|---|
| Bucket | Yes | |
| Delimiter - delimiter | Yes | Only supported / - otherwise 400 Bad request. |
| EncodingType | Yes | |
| Marker - marker | Yes | |
| MaxKeys - max-keys | Yes | |
| Prefix - prefix | Yes | |
| RequestPayer - x-amz-request-payer | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

Response

| Parameter | Implemented | Comments |
|----------------------------------|-------------|---------------------------|
| IsTruncated | Yes | |
| Marker | Yes | |
| NextMarker | Yes | When response IsTruncated |
| Contents | Yes | |
| Contents >> Key | Yes | |
| Contents >> LastModified | Yes | |
| Contents >> ETag | Yes | |
| Contents >> ChecksumAlgorithm | No | |
| Contents >> Size | Yes | |
| Contents >> StorageClass | No | |
| Contents >> Owner | Yes | |
| Contents >> Owner >> DisplayName | Yes | |
| Contents >> Owner >> ID | Yes | |
| Name | Yes | |
| Prefix | Yes | |
| Delimiter | Yes | |
| MaxKeys | Yes | |
| CommonPrefixes | Yes | |
| EncodingType | Yes | |

B.6.9 ListObjectsV2

[AWS Reference](#)

Request Syntax

```
GET /bucket?list-type=2&continuation-token=string&delimiter=/&encoding-type=url&fetch-owner=boolean&max-keys=integer&prefix=string&start-after=string HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8"?>
<ListBucketResult>
  <IsTruncated>boolean</IsTruncated>
  </Contents>
    <ETag>etag</ETag>
    <Key>string</Key>
    <LastModified>timestamp</LastModified>
    <Owner>
      <DisplayName>string</DisplayName>
      <ID>string</ID>
    </Owner>
    <Size>integer</Size>
  </Contents>
  ...
  <Name>string</Name>
  <Prefix>string</Prefix>
  <Delimiter></Delimiter>
  <MaxKeys>integer</MaxKeys>
  <CommonPrefixes>
    <Prefix>string</Prefix>
  </CommonPrefixes>
  ...
  <EncodingType>url</EncodingType>
  <KeyCount>integer</KeyCount>
  <ContinuationToken>string</ContinuationToken>
  <NextContinuationToken>string</NextContinuationToken>
  <StartAfter>string</StartAfter>
</ListBucketResult>
```

Permissions

Bucket owner or any user granted READ permission on the bucket is permitted to list bucket objects.

Request

| Parameter | Implemented | Comments |
|--|-------------|---|
| Bucket | Yes | |
| Delimiter - delimiter | Yes | Only supported / - otherwise 400 Bad request. |
| EncodingType | Yes | |
| MaxKeys - max-keys | Yes | |
| Prefix - prefix | Yes | |
| ContinuationToken - continuation-token | Yes | |
| FetchOwner - fetch-owner | Yes | |
| StartAfter - start-after | Yes | |
| RequestPayer - x-amz-request-payer | No | |
| ExpectedBucketOwner | Yes | |

Response

| Parameter | Implemented | Comments |
|----------------------------------|-------------|----------|
| IsTruncated | Yes | |
| Contents | Yes | |
| Contents >> Key | Yes | |
| Contents >> LastModified | Yes | |
| Contents >> ETag | Yes | |
| Contents >> ChecksumAlgorithm | No | |
| Contents >> Size | Yes | |
| Contents >> StorageClass | No | |
| Contents >> Owner | Yes | |
| Contents >> Owner >> DisplayName | Yes | |
| Contents >> Owner >> ID | Yes | |
| Name | Yes | |
| Prefix | Yes | |
| Delimiter | Yes | |
| MaxKeys | Yes | |
| CommonPrefixes | Yes | |
| EncodingType | Yes | |
| EncodingType | Yes | |
| KeyCount | Yes | |
| ContinuationToken | Yes | |
| NextContinuationToken | Yes | |
| StartAfter | Yes | |

B.6.10 ListMultipartUploads

[AWS Reference](#)

Request Syntax

```
GET /bucket?uploads&delimiter=/&encoding-type=url&key-marker=string&max-
uploads=integer&prefix=string&upload-id-marker=string HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Request Syntax

```
HTTP/1.1 200 OK
```

```
<?xml version="1.0" encoding="UTF-8"?>
<ListMultipartUploadsResult>
  <Name>string</Name>
  <KeyMarker>string</KeyMarker>
  <UploadIdMarker>string</UploadIdMarker>
  <NextKeyMarker>string</NextKeyMarker>
  <Prefix>string</Prefix>
  <Delimiter>/</Delimiter>
  <NextUploadIdMarker>string</NextUploadIdMarker>
  <MaxUploads>integer</MaxUploads>
  <IsTruncated>boolean</IsTruncated>
  </Upload>
    <Initiated>timestamp</Initiated>
    <Initiator>
      <DisplayName>string</DisplayName>
      <ID>string</ID>
    </Initiator>
    <Key>string</Key>
    <Owner>
      <DisplayName>string</DisplayName>
      <ID>string</ID>
    </Owner>
    <UploadId>string</UploadId>
  </Upload>
  ...
  <CommonPrefixes>
    <Prefix>string</Prefix>
  </CommonPrefixes>
  ...
  <EncodingType>url</EncodingType>
</ListBucketResult>
```

Permissions

Bucket owner or any user granted READ permission on the bucket is permitted to list multipart uploads.

Request

| Parameter | Implemented | Comments |
|---|-------------|---|
| Bucket | Yes | |
| Delimiter - delimiter | Yes | Only supported / - otherwise 400 Bad request. |
| EncodingType - encoding-type | Yes | |
| KeyMarker - key-marker | Yes | |
| MaxUploads - max-uploads | Yes | |
| Prefix - prefix | Yes | |
| UploadIdMarker - upload-id-marker | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

Response

| Parameter | Implemented | Comments |
|------------------------------|-------------|----------|
| Bucket | Yes | |
| KeyMarker | Yes | |
| UploadIdMarker | Yes | |
| NextKeyMarker | Yes | |
| Prefix | Yes | |
| Delimiter | Yes | |
| NextUploadIdMarker | Yes | |
| MaxUploads | Yes | |
| IsTruncated | Yes | |
| Uploads | Yes | |
| Uploads >> UploadId | Yes | |
| Uploads >> Key | Yes | |
| Uploads >> Initiated | Yes | |
| Uploads >> StorageClass | No | |
| Uploads >> Owner | Yes | |
| Uploads >> Initiator | Yes | |
| Uploads >> ChecksumAlgorithm | No | |
| CommonPrefixes | Yes | |
| EncodingType | Yes | |

B.6.11 PutObject

[AWS Reference](#)

Request Syntax

```
PUT /bucket/key HTTP/1.1
Host: endpoint
Cache-Control: string
Content-Disposition: string
Content-Encoding: string
Content-Language: string
Content-Type: string
Expires: string
x-amz-acl: canned-acl
x-amz-grant-full-control: grantee
x-amz-grant-read: grantee
x-amz-grant-read-acl: grantee
x-amz-grant-write-acp: grantee
x-amz-expected-bucket-owner: owner-ID
x-amz-meta-*: string
x-amz-tagging: key=value&key2=value2
```

Body

Response Syntax

```
HTTP/1.1 200 OK
ETag: etag
x-amz-version-id: string
```

Permissions

Bucket owner or any user granted WRITE permission on the bucket is permitted to put object into the bucket. No permissions on existing object needed to replace it with new content and metadata.

Request

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Bucket | Yes | |
| ACL - x-amz-acl | Yes | |
| Body | Yes | |
| CacheControl - Cache-Control | Yes | |
| ContentDisposition - Content-Disposition | Yes | |
| ContentMD5 - Content-MD5 | No | |
| ContentEncoding - Content-Encoding | Yes | |
| ContentLanguage - Content-Language | Yes | |
| ContentLength - Content-Length | Yes | |
| ContentType - Content-Type | Yes | |
| ChecksumAlgorithm - x-amz-sdk-checksum-algorithm | No | |
| ChecksumCRC32 - x-amz-checksum-crc32 | No | |
| ChecksumCRC32C - x-amz-checksum-crc32c | No | |
| ChecksumSHA1 - x-amz-checksum-sha1 | No | |
| ChecksumSHA256 - x-amz-checksum-sha256 | No | |
| Expires | Yes | |
| GrantFullControl - x-amz-grant-full-control | Yes | |
| GrantRead - x-amz-grant-read | Yes | |
| GrantReadACP - x-amz-grant-read-acp | Yes | |
| GrantWriteACP - x-amz-grant-write-acp | Yes | |
| Key | Yes | |
| Metadata - x-amz-meta-* | Yes | |
| ServerSideEncryption - x-amz-server-side-encryption | No | |
| StorageClass - x-amz-storage-class | No | |
| WebsiteRedirectLocation - x-amz-website-redirect-location | No | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKey - x-amz-server-side-encryption-customer-key | No | |
| SSEKMSKeyId - x-amz-server-side-encryption-aws-kms-key-id | No | |
| SSEKMSEncryptionContext - x-amz-server-side-encryption-context | No | |
| BucketKeyEnabled - x-amz-server-side-encryption-bucket-key-enabled | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |
| RequestPayer - x-amz-request-payer | No | |
| Tagging - x-amz-tagging | Yes | |
| ObjectLockMode - x-amz-object-lock-mode | No | |
| ObjectLockRetainUntilDate - x-amz-object-lock-retain-until-date | No | |
| ObjectLockLegalHoldStatus - x-amz-object-lock-legal-hold | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

Response

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Expiration - x-amz-expiration | No | |
| ETag | Yes | |
| ChecksumCRC32 - x-amz-checksum-crc32 | No | |
| ChecksumCRC32C - x-amz-checksum-crc32c | No | |
| ChecksumSHA1 - x-amz-checksum-sha1 | No | |
| ChecksumSHA256 - x-amz-checksum-sha256 | No | |
| ServerSideEncryption - x-amz-server-side-encryption | No | |
| VersionId - x-amz-version-id | Yes | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |
| SSEKMSKeyId - x-amz-server-side-encryption-aws-kms-key-id | No | |
| SSEKMSEncryptionContext - x-amz-server-side-encryption-context | No | |
| BucketKeyEnabled - x-amz-server-side-encryption-bucket-key-enabled | No | |
| RequestCharged - x-amz-request-charged | No | |

B.6.12 PutObjectTagging

[AWS Reference](#)

Request Syntax

```
PUT /bucket/key?tagging&versionId=VersionId HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
x-amz-version-id: string

<?xml version="1.0" encoding="UTF-8"?>
<Tagging xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <TagSet>
    <Tag>
      <Key>string</Key>
      <Value>string</Value>
    </Tag>
    ...
  </TagSet>
</Tagging>
```

Response Syntax

```
HTTP/1.1 200 OK
x-amz-version-id: string
```

Permissions

To use this operation, you must have permission to perform the `s3:PutObjectTagging` action. By default, the bucket owner has this permission and can grant this permission to others.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| VersionId - versionId | Yes | |
| ContentMD5 - Content-MD5 | No | |
| ChecksumAlgorithm - x-amz-sdk-checksum-algorithm | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |
| RequestPayer - x-amz-request-payer | No | |

Request Body

| Parameter | Implemented | Comments |
|-------------------|-------------|----------|
| Tagging | Yes | |
| Tagging >> TagSet | Yes | |

Response

| Parameter | Implemented | Comments |
|------------------------------|-------------|----------|
| VersionId - x-amz-version-id | Yes | |

B.6.13 GetObject

[AWS Reference](#)

Request Syntax

```
GET /bucket/key?versionId=VersionId HTTP/1.1
Host: endpoint
If-Match: etag
If-Modified-Since: timestamp
If-None-Match: etag
If-Unmodified-Since: timestamp
Range: bytes=integer-integer
x-amz-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Content-Range: bytes integer-integer/integer
Last-Modified: timestamp
ETag: etag
Cache-Control: string
Content-Disposition: string
Content-Encoding: string
Content-Language: string
Content-Type: string
Expires: string
x-amz-tagging-count: string
x-amz-version-id: string
```

Body

Permissions

Object owner or any user granted READ permission on the object is permitted to get object.

Request

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| IfMatch - If-Match | Yes | |
| IfModifiedSince - If-Modified-Since | Yes | |
| IfNoneMatch - If-None-Match | Yes | |
| IfUnmodifiedSince - If-Unmodified-Since | Yes | |
| Range | Yes | |
| ResponseCacheControl - response-cache-control | No | |
| ResponseContentDisposition - response-content-disposition | No | |
| ResponseContentEncoding - response-content-encoding | No | |
| ResponseContentLanguage - response-content-language | No | |
| ResponseContentType - response-content-type | No | |
| ResponseExpires - response-expires | Yes | |
| VersionId - versionId | Yes | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |

| | |
|---|-----|
| SSECustomerKey - x-amz-server-side-encryption-customer-key | No |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No |
| RequestPayer - x-amz-request-payer | No |
| PartNumber - partNumber | No |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes |
| ChecksumMode - x-amz-checksum-mode | No |

Response

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Body | Yes | |
| DeleteMarker - x-amz-delete-marker | Yes | |
| AcceptRanges - accept-ranges | Yes | |
| Expiration - x-amz-expiration | No | |
| Restore - x-amz-restore | No | |
| LastModified - Last-Modified | Yes | |
| ContentLength - Content-Length | Yes | |
| ETag | Yes | |
| ChecksumCRC32 - x-amz-checksum-crc32 | No | |
| ChecksumCRC32C - x-amz-checksum-crc32c | No | |
| ChecksumSHA1 - x-amz-checksum-sha1 | No | |
| ChecksumSHA256 - x-amz-checksum-sha256 | No | |
| MissingMeta - x-amz-missing-meta | No | |
| VersionId - x-amz-version-id | Yes | |
| CacheControl - Cache-Control | Yes | |
| ContentDisposition - Content-Disposition | Yes | |
| ContentEncoding - Content-Encoding | Yes | |
| ContentLanguage - Content-Language | Yes | |
| ContentRange - Content-Range | Yes | |
| ContentType - Content-Type | Yes | |
| Expires | Yes | |
| WebsiteRedirectLocation - x-amz-website-redirect-location | No | |
| ServerSideEncryption - x-amz-server-side-encryption | No | |
| Metadata - x-amz-meta- | Yes | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |
| SSEKMSKeyId - x-amz-server-side-encryption-aws-kms-key-id | No | |
| BucketKeyEnabled - x-amz-server-side-encryption-bucket-key-enabled | No | |
| StorageClass - x-amz-storage-class | No | |
| RequestCharged - x-amz-request-charged | No | |
| ReplicationStatus - x-amz-replication-status | No | |
| PartsCount - x-amz-mp-parts-count | No | |

| | |
|---|-----|
| TagCount - x-amz-tagging-count | Yes |
| ObjectLockMode - x-amz-object-lock-mode | No |
| ObjectLockRetainUntilDate - x-amz-object-lock-retain-until-date | No |
| ObjectLockLegalHoldStatus - x-amz-object-lock-legal-hold | No |

B.6.14 GetObjectTagging

[AWS Reference](#)

Request Syntax

GET /bucket/key?tagging&versionId=VersionId HTTP/1.1
x-amz-expected-bucket-owner: owner-id

Response Syntax

HTTP/1.1 200
x-amz-version-id: string

<?xml version="1.0" encoding="UTF-8"?>
<Tagging>
 <TagSet>
 <Tag>
 <Key>string</Key>
 <Value>string</Value>
 </Tag>
 ...
 </TagSet>
</Tagging>

Permissions

To use this operation, you must have permission to perform the s3:GetObjectTagging action. By default, the bucket owner has this permission and can grant this permission to others.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| VersionId - versionId | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |
| RequestPayer - x-amz-request-payer | No | |

Response

| Parameter | Implemented | Comments |
|------------------------------|-------------|----------|
| VersionId - x-amz-version-id | Yes | |
| TagSet | Yes | |

B.6.15 HeadObject

[AWS Reference](#)

Request Syntax

```
HEAD /bucket/key?versionId=string HTTP/1.1
Host: endpoint
If-Match: etag
If-Modified-Since: timestamp
If-None-Match: etag
If-Unmodified-Since: timestamp
Range: bytes=integer-integer
x-amz-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.1 200 OK
Accept-Ranges: bytes
Last-Modified: timestamp
ETag: etag
Cache-Control: string
Content-Disposition: string
Content-Encoding: string
Content-Language: string
Content-Type: string
Expires: string
x-amz-version-id: string
```

Permissions

Same as for GetObject.

Request

| Parameter | Implemented | Comments |
|--|-------------|---|
| Bucket | Yes | |
| Key | Yes | |
| IfMatch - If-Match | Yes | |
| IfModifiedSince - If-Modified-Since | Yes | |
| IfNoneMatch - If-None-Match | Yes | |
| IfUnmodifiedSince - If-Unmodified-Since | Yes | |
| Range | Yes | Because HeadObject returns only the metadata, this parameter has no effect. |
| VersionId - versionId | Yes | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKey - x-amz-server-side-encryption-customer-key | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |
| RequestPayer - x-amz-request-payer | No | |
| PartNumber - partNumber | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |
| ChecksumMode - x-amz-checksum-mode | No | |

Response

| Parameter | Implemented | Comments |
|--|-------------|---------------------------------|
| DeleteMarker - x-amz-delete-marker | Yes | |
| AcceptRanges - accept-ranges | No | GetObject returns such header |
| Expiration - x-amz-expiration | No | |
| Restore - x-amz-restore | No | |
| LastModified - Last-Modified | Yes | |
| CacheControl - Cache-Control | Yes | |
| ContentDisposition - Content-Disposition | Yes | |
| ContentEncoding - Content-Encoding | Yes | |
| ContentLanguage - Content-Language | Yes | |
| ContentLength - Content-Length | Yes | |
| ETag | Yes | |
| ChecksumCRC32 - x-amz-checksum-crc32 | No | |
| ChecksumCRC32C - x-amz-checksum-crc32c | No | |
| ChecksumSHA1 - x-amz-checksum-sha1 | No | |
| ChecksumSHA256 - x-amz-checksum-sha256 | No | |
| MissingMeta - x-amz-missing-meta | No | |
| VersionId - x-amz-version-id | Yes | |
| ContentType - Content-Type | Yes | |
| Expires | Yes | |
| WebsiteRedirectLocation - x-amz-website-redirect-location | No | |
| ServerSideEncryption - x-amz-server-side-encryption | No | |
| Metadata - x-amz-meta-* | Yes | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |
| SSEKMSKeyId - x-amz-server-side-encryption-aws-kms-key-id | No | |
| BucketKeyEnabled - x-amz-server-side-encryption-bucket-key-enabled | No | |
| StorageClass - x-amz-storage-class | No | |
| RequestCharged - x-amz-request-charged | No | |
| ReplicationStatus - x-amz-replication-status | No | |
| PartsCount - x-amz-mp-parts-count | No | |
| TagCount - x-amz-tagging-count | Yes | AWS does not return such header |
| ObjectLockMode - x-amz-object-lock-mode | No | |
| ObjectLockRetainUntilDate - x-amz-object-lock-retain-until-date | No | |
| ObjectLockLegalHoldStatus - x-amz-object-lock-legal-hold | No | |
| ArchiveStatus - x-amz-archive-status | No | |

B.6.16 DeleteObject

[AWS Reference](#)

Request Syntax

```
DELETE /bucket/key?versionId=string HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.1 204 No Content
x-amz-delete-marker: bool
x-amz-version-id: string
```

Permissions

Bucket owner or any user granted WRITE permission on the bucket can delete any object from bucket. No specific permission on the object needed.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| MFA - x-amz-mfa | No | |
| VersionId - versionId | Yes | |
| RequestPayer - x-amz-request-payer | No | |
| BypassGovernanceRetention - x-amz-bypass-governance-retention | No | |
| ExpectedBucketOwner | Yes | |

Response

| Parameter | Implemented | Comments |
|--|-------------|----------|
| DeleteMarker - x-amz-delete-marker | Yes | |
| VersionId - x-amz-version-id | Yes | |
| RequestCharged - x-amz-request-charged | No | |

B.6.17 DeleteObjectTagging

[AWS Reference](#)

Request Syntax

```
DELETE /bucket/key?tagging&versionId=string HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Request Syntax

```
HTTP/1.1 204 No Content
x-amz-version-id: string
```

Permissions

To use this operation, you must have permission to perform the s3:DeleteObjectTagging action.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| VersionId - versionId | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

Response

| Parameter | Implemented | Comments |
|------------------------------|-------------|----------|
| VersionId - x-amz-version-id | Yes | |

B.6.18 DeleteObjects

[AWS Reference](#)

Request Syntax

```
POST /bucket?delete HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID

<?xml version="1.0" encoding="UTF-8"?>
<Delete xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Object>
    <Key>string</Key>
    <VersionId>string</VersionId>
  </Object>
  ...
  <Quiet>boolean</Quiet>
</Delete>
```

Response Syntax

```
HTTP/1.1 200 OK
x-amz-request-charged: string

<?xml version="1.0" encoding="UTF-8"?>
<DeleteResult>
  <Deleted>
    <DeleteMarker>bool</DeleteMarker>
    <DeleteMarkerVersionId>string</DeleteMarkerVersionId>
    <Key>string</Key>
    <VersionId>string</VersionId>
  </Deleted>
  ...
  <Error>
    <Code>string</Code>
    <Key>string</Key>
    <VersionId>string</VersionId>
  </Error>
  ...
</DeleteResult>
```

Permissions

Same as for DeleteObject.

Request

| Parameter | Implemented | Comments |
|---|-------------|---------------------------|
| Bucket | Yes | |
| Delete | Yes | Container for the request |
| MFA - x-amz-mfa | No | |
| RequestPayer - x-amz-request-payer | No | |
| BypassGovernanceRetention - x-amz-bypass-governance-retention | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |
| ChecksumAlgorithm - x-amz-sdk-checksum-algorithm | Yes | |

Request Body

| Parameter | Implemented | Comments |
|---------------------|-------------|----------------------------|
| Object | Yes | |
| Object >> VersionId | Yes | |
| Quiet | Yes | Element parsed but ignored |

Response

| Parameter | Implemented | Comments |
|--|-------------|---|
| Deleted | Yes | Container element for a successful delete |
| RequestCharged - x-amz-request-charged | No | |
| Error | Yes | If error is happened |

Response Body

| Parameter | Implemented | Comments |
|----------------------------------|-------------|----------|
| Deleted | Yes | |
| Deleted >> Key | Yes | |
| Deleted >> VersionId | Yes | |
| Deleted >> DeleteMarker | Yes | |
| Deleted >> DeleteMarkerVersionId | Yes | |
| Errors | Yes | |
| Errors >> Key | Yes | |
| Errors >> VersionId | Yes | |
| Errors >> Code | Yes | |

B.6.19 PutObjectAcl

[AWS Reference](#)

Request Syntax

```
PUT /bucket/key?acl&versionId=string HTTP/1.1
Host: endpoint
x-amz-acl: canned-acl
x-amz-grant-full-control: grantee
x-amz-grant-read: grantee
x-amz-grant-read-acl: grantee
x-amz-grant-write-acp: grantee
x-amz-expected-bucket-owner: owner-ID

<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <AccessControlList>
    <Grant>
      <Grantee>
        <DisplayName>string</DisplayName>
        <EmailAddress>string</EmailAddress>
        <ID>string</ID>
        <xsi:type>string</xsi:type>
        <URI>string</URI>
      </Grantee>
      <Permission>string</Permission>
    </Grant>
    ...
  </AccessControlList>
  <Owner>
    <DisplayName>string</DisplayName>
    <ID>string</ID>
  </Owner>
</AccessControlPolicy>
```

Response Syntax

```
HTTP/1.1 200 OK
x-amz-version-id: string
```

Permissions

Object owner or any user granted WRITE_ACP permission on the object is permitted to set object ACL.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| ACL - x-amz-acl | Yes | |
| ChecksumAlgorithm - x-amz-sdk-checksum-algorithm | No | |
| GrantFullControl - x-amz-grant-full-control | Yes | |
| GrantRead - x-amz-grant-read | Yes | |
| GrantReadACP - x-amz-grant-read-acp | Yes | |
| GrantWrite - x-amz-grant-write | Yes | |
| GrantWriteACP - x-amz-grant-write-acp | Yes | |
| Key | Yes | |
| RequestPayer - x-amz-request-payer | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |
| VersionId - versionId | Yes | |
| ContentMD5 - Content-MD5 | No | |

Request Body

| Parameter | Implemented | Comments |
|-------------------------------|-------------|--|
| AccessControlPolicy | Yes | You cannot specify both the body and the request headers |
| AccessControlPolicy >> Grants | Yes | |
| AccessControlPolicy >> Owner | Yes | |

Response

| Parameter | Implemented | Comments |
|--|-------------|---|
| RequestCharged - x-amz-request-charged | No | |
| VersionId - x-amz-version-id | Yes | Header is not listed in AWS docs, but it presents in AWS response |

B.6.20 GetObjectAcl

[AWS Reference](#)

Request Syntax

```
GET /bucket/key?acl&versionId=string HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.1 200 OK
x-amz-version-id: string

<?xml version="1.0" encoding="UTF-8"?>
<AccessControlPolicy xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <AccessControlList>
    <Grant>
      <Grantee>
        <DisplayName>string</DisplayName>
        <EmailAddress>string</EmailAddress>
        <ID>string</ID>
        <xsi:type>string</xsi:type>
        <URI>string</URI>
      </Grantee>
      <Permission>string</Permission>
    </Grant>
    ...
  </AccessControlList>
  <Owner>
    <DisplayName>string</DisplayName>
    <ID>string</ID>
  </Owner>
</AccessControlPolicy>
```

Permissions

Object owner or any user granted READ_ACP permission on the object is permitted to get object ACL.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| VersionId - versionId | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |
| RequestPayer - x-amz-request-payer | No | |

Response

| Parameter | Implemented | Comments |
|--|-------------|---|
| Owner | Yes | |
| Grants | Yes | |
| RequestCharged - x-amz-request-charged | No | |
| VersionId - x-amz-version-id | Yes | Header is not listed in AWS docs, but it presents in AWS response |

B.6.21 CreateMultipartUpload

[AWS Reference](#)

Request Syntax

```
POST /bucket/key?uploads HTTP/1.1
Host: endpoint
Cache-Control: string
Content-Disposition: string
Content-Encoding: string
Content-Language: string
Content-Type: string
Expires: string
x-amz-acl: canned-acl
x-amz-grant-full-control: grantee
x-amz-grant-read: grantee
x-amz-grant-read-acl: grantee
x-amz-grant-write-acp: grantee
x-amz-expected-bucket-owner: owner-ID
x-amz-meta-*: string
x-amz-tagging: key=value&key2=value2
```

Response Syntax

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8"?>
<InitiateMultipartUploadResult>
  <Bucket>string</Bucket>
  <Key>string</Key>
  <UploadId>string</UploadId>
</InitiateMultipartUploadResult>
```

Permissions

Bucket owner or any user granted WRITE permission on the bucket is permitted to create upload in the bucket.

Request

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Bucket | Yes | |
| ACL - x-amz-acl | Yes | |
| CacheControl - Cache-Control | Yes | |
| ContentDisposition - Content-Disposition | Yes | |
| ContentEncoding - Content-Encoding | Yes | |
| ContentLanguage - Content-Language | Yes | |
| ContentType - Content-Type | Yes | |
| ChecksumAlgorithm - x-amz-sdk-checksum-algorithm | No | |
| Expires | Yes | |
| GrantFullControl - x-amz-grant-full-control | Yes | |
| GrantRead - x-amz-grant-read | Yes | |
| GrantReadACP - x-amz-grant-read-acp | Yes | |
| GrantWriteACP - x-amz-grant-write-acp | Yes | |
| Key | Yes | |

| | |
|--|-----|
| Metadata - x-amz-meta-* | Yes |
| ServerSideEncryption - x-amz-server-side-encryption | No |
| StorageClass - x-amz-storage-class | No |
| WebsiteRedirectLocation - x-amz-website-redirect-location | No |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No |
| SSECustomerKey - x-amz-server-side-encryption-customer-key | No |
| SSEKMSKeyId - x-amz-server-side-encryption-aws-kms-key-id | No |
| SSEKMSEncryptionContext - x-amz-server-side-encryption-context | No |
| BucketKeyEnabled - x-amz-server-side-encryption-bucket-key-enabled | No |
| RequestPayer - x-amz-request-payer | No |
| Tagging - x-amz-tagging | Yes |
| ObjectLockMode - x-amz-object-lock-mode | No |
| ObjectLockRetainUntilDate - x-amz-object-lock-retain-until-date | No |
| ObjectLockLegalHoldStatus - x-amz-object-lock-legal-hold | No |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes |

Response

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| UploadId | Yes | |
| AbortDate - x-amz-abort-date | No | |
| AbortRuleId - x-amz-abort-rule-id | No | |
| ChecksumCRC32 - x-amz-checksum-crc32 | No | |
| ChecksumAlgorithm - x-amz-checksum-algorithm | No | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |
| SSEKMSKeyId - x-amz-server-side-encryption-aws-kms-key-id | No | |
| SSEKMSEncryptionContext - x-amz-server-side-encryption-context | No | |
| BucketKeyEnabled - x-amz-server-side-encryption-bucket-key-enabled | No | |
| RequestCharged - x-amz-request-charged | No | |

B.6.22 UploadPart

[AWS Reference](#)

Request Syntax

```
PUT /bucket/key?partNumber=integer&uploadId=string HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Body

Response Syntax

```
HTTP/1.1 200 OK
Etag: etag
```

Permissions

Upload owner or any user granted WRITE permission on the upload is permitted to upload part.

Request

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Body | Yes | |
| Bucket | Yes | |
| ContentLength - Content-Length | Yes | |
| ContentMD5 - Content-MD5 | No | |
| ChecksumAlgorithm - x-amz-sdk-checksum-algorithm | No | |
| ChecksumCRC32 - x-amz-checksum-crc32 | No | |
| ChecksumCRC32C - x-amz-checksum-crc32c | No | |
| ChecksumSHA1 - x-amz-checksum-sha1 | No | |
| ChecksumSHA256 - x-amz-checksum-sha256 | No | |
| Key | Yes | |
| PartNumber - partNumber | Yes | |
| UploadId - uploadId | Yes | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKey - x-amz-server-side-encryption-customer-key | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |
| RequestPayer - x-amz-request-payer | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

Response

| Parameter | Implemented | Comments |
|--|-------------|----------|
| ETag | Yes | |
| ChecksumCRC32 - x-amz-checksum-crc32 | No | |
| ChecksumCRC32C - x-amz-checksum-crc32c | No | |
| ChecksumSHA1 - x-amz-checksum-sha1 | No | |
| ChecksumSHA256 - x-amz-checksum-sha256 | No | |
| ServerSideEncryption - x-amz-server-side-encryption | No | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |
| SSEKMSKeyId - x-amz-server-side-encryption-aws-kms-key-id | No | |
| BucketKeyEnabled - x-amz-server-side-encryption-bucket-key-enabled | No | |
| RequestCharged - x-amz-request-charged | No | |

B.6.23 CompleteMultipartUpload

[AWS Reference](#)

Request Syntax

```
POST /bucket/key?uploadId=string HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID

<?xml version="1.0" encoding="UTF-8"?>
<CompleteMultipartUpload xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Part>
    <ETag>string</ETag>
    <PartNumber>integer</PartNumber>
  </Part>
  ...
</CompleteMultipartUpload>
```

Response Syntax

```
HTTP/1.1 200 OK
x-amz-version-id: string

<?xml version="1.0" encoding="UTF-8"?>
<CompleteMultipartUploadResult>
  <Bucket>string</Bucket>
  <Key>string</Key>
  <ETag>string</ETag>
</CompleteMultipartUploadResult>
```

Permissions

Bucket owner or any user granted WRITE permission on the bucket can complete upload. No permissions on upload is needed.

Request

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| UploadId - uploadId | Yes | |
| ChecksumCRC32 - x-amz-checksum-crc32 | No | |
| ChecksumCRC32C - x-amz-checksum-crc32c | No | |
| ChecksumSHA1 - x-amz-checksum-sha1 | No | |
| ChecksumSHA256 - x-amz-checksum-sha256 | No | |
| RequestPayer - x-amz-request-payer | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKey - x-amz-server-side-encryption-customer-key | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |

Request Body

| Parameter | Implemented | Comments |
|---|-------------|----------|
| MultipartUpload | Yes | |
| MultipartUpload >> Parts | Yes | |
| MultipartUpload >> Parts >> x-amz-checksum-crc32 | No | |
| MultipartUpload >> Parts >> x-amz-checksum-crc32c | No | |
| MultipartUpload >> Parts >> x-amz-checksum-sha1 | No | |
| MultipartUpload >> Parts >> x-amz-checksum-sha256 | No | |
| MultipartUpload >> Parts >> ETag | Yes | |
| MultipartUpload >> Parts >> PartNumber | Yes | |

Response

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| Expiration - x-amz-expiration | No | |
| Location | No | |
| ETag | Yes | |
| ChecksumCRC32 - x-amz-checksum-crc32 | No | |
| ChecksumCRC32C - x-amz-checksum-crc32c | No | |
| ChecksumSHA1 - x-amz-checksum-sha1 | No | |
| ChecksumSHA256 - x-amz-checksum-sha256 | No | |
| ServerSideEncryption - x-amz-server-side-encryption | No | |
| SSEKMSKeyId - x-amz-server-side-encryption-aws-kms-key-id | No | |
| BucketKeyEnabled - x-amz-server-side-encryption-bucket-key-enabled | No | |
| RequestCharged - x-amz-request-charged | No | |
| VersionId - x-amz-version-id | Yes | |

B.6.24 AbortMultipartUpload

[AWS Reference](#)

Request Syntax

```
DELETE /bucket/key?uploadId=string HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.1 200 OK
```

Permissions

Bucket owner or any user granted WRITE permission on the bucket can abort the upload. No permissions on upload is needed.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| UploadId - uploadId | Yes | |
| RequestPayer - x-amz-request-payer | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

Response

| Parameter | Implemented | Comments |
|--|-------------|----------|
| RequestCharged - x-amz-request-charged | No | |

B.6.25 ListParts

[AWS Reference](#)

Request Syntax

```
GET /bucket/key?max-parts=integer&part-number-marker=integer&uploadId=string HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8"?>
<ListPartsResult>
  <Bucket>string</Bucket>
  <Key>string</Key>
  <UploadId>string</UploadId>
  <PartNumberMarker>integer</PartNumberMarker>
  <NextPartNumberMarker>integer</NextPartNumberMarker>
  <MaxParts>integer</MaxParts>
  <IsTruncated>boolean</IsTruncated>
  <Part>
    <ETag>string</ETag>
    <LastModified>timestamp</LastModified>
    <PartNumber>integer</PartNumber>
    <Size>integer</Size>
  </Part>
  ...
  <Initiator>
    <DisplayName>string</DisplayName>
    <ID>string</ID>
  </Initiator>
  <Owner>
    <DisplayName>string</DisplayName>
    <ID>string</ID>
  </Owner>
</ListPartsResult>
```

Permissions

Upload owner or any user granted READ permission on the upload is permitted to list parts.

Request

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| MaxParts - max-parts | Yes | |
| PartNumberMarker - part-number-marker | Yes | |
| UploadId - uploadId | Yes | |
| RequestPayer - x-amz-request-payer | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKey - x-amz-server-side-encryption-customer-key | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |

Response

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| UploadId | Yes | |
| AbortDate - x-amz-abort-date | No | |
| AbortRuleId - x-amz-abort-rule-id | No | |
| PartNumberMarker | Yes | |
| NextPartNumberMarker | Yes | |
| MaxParts | Yes | |
| IsTruncated | Yes | |
| Parts | Yes | |
| Parts >> PartNumber | Yes | |
| Parts >> LastModified | Yes | |
| Parts >> ETag | Yes | |
| Parts >> Size | Yes | |
| Parts >> ChecksumCRC32 | No | |
| Parts >> ChecksumCRC32C | No | |
| Parts >> ChecksumSHA1 | No | |
| Parts >> ChecksumSHA256 | No | |
| Initiator | Yes | |
| Owner | Yes | |
| StorageClass | No | |
| RequestCharged - x-amz-request-charged | No | |
| ChecksumAlgorithm | No | |

B.6.26 PostObject

Forms-based upload

[AWS Reference](#)

Request Syntax

```
POST /bucket HTTP/1.1
Host: endpoint
Content-Type: multipart/form-data; boundary=BOUNDARY

--BOUNDARY
Content-Disposition: form-data; name="key"

string
--BOUNDARY
Content-Disposition: form-data; name="success_action_redirect"

string
--BOUNDARY
Content-Disposition: form-data; name="policy"

string
--BOUNDARY
Content-Disposition: form-data; name="signature"

string
--BOUNDARY
Content-Disposition: form-data; name="file"; filename="string"

file-content
--BOUNDARY
Content-Disposition: form-data; name="submit"

string
--BOUNDARY
Content-Disposition: form-data; name="tagging"

<Tagging><TagSet><Tag><Key>Tag Name</Key><Value>Tag Value</Value></Tag></TagSet></Tagging>
--BOUNDARY--
```

Response Syntax

```
HTTP/1.1 200 OK
ETag: etag
x-amz-version-id: string

<?xml version="1.0" encoding="UTF-8"?>
<PostObjectResult>
  <Bucket>string</Bucket>
  <Key>string</Key>
  <ETag>etag</ETag>
  <Location>string</Location>
</PostObjectResult>
```

Permissions

Anonymous PostObject requires WRITE permission on the bucket for AllUser group. Non-anonymous PostObject request should contain policy and signature form fields. Bucket owner or any user granted WRITE permission on the bucket is permitted for forms-based upload.

REST-specific headers

| Parameter | Implemented | Comments |
|---------------------|-------------|----------|
| Cache-Control | Yes | |
| Content-Type | Yes | |
| Content-Disposition | Yes | |
| Content-Encoding | Yes | |
| Expires | Yes | |

Form Fields

| Parameter | Implemented | Comments |
|---|-------------|--------------------------|
| acl | Yes | |
| file | Yes | The file or text content |
| key | Yes | |
| policy | Yes | |
| success_action_redirect | Yes | |
| success_action_status | Yes | |
| tagging | Yes | |
| x-amz-storage-class | No | |
| x-amz-meta-* | Yes | |
| x-amz-security-token | No | |
| x-amz-signature | Yes | |
| x-amz-credential | Yes | |
| x-amz-date | Yes | |
| x-amz-algorithm | Yes | |
| x-amz-website-redirect-location | No | |
| x-amz-checksum-algorithm | No | |
| x-amz-checksum-crc32 | No | |
| x-amz-checksum-crc32c | No | |
| x-amz-checksum-sha1 | No | |
| x-amz-checksum-sha256 | No | |
| x-amz-server-side-encryption | No | |
| x-amz-server-side-encryption-aws-kms-key-id | No | |
| x-amz-server-side-encryption-context | No | |
| x-amz-server-side-encryption-bucket-key-enabled | No | |
| x-amz-server-side-encryption-customer-algorithm | No | |
| x-amz-server-side-encryption-customer-key | No | |
| x-amz-server-side-encryption-customer-key-MD5 | No | |

Response Headers

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | No | |
| ETag | No | |
| Key | No | |
| Location | Yes | |
| Server | No | |
| x-amz-checksum-crc32 | No | |
| x-amz-checksum-crc32c | No | |
| x-amz-checksum-sha1 | No | |
| x-amz-checksum-sha256 | No | |
| x-amz-expiration | No | |
| success_action_redirect | No | |
| x-amz-server-side-encryption | No | |
| x-amz-server-side-encryption-aws-kms-key-id | No | |
| x-amz-server-side-encryption-bucket-key-enabled | No | |
| x-amz-server-side-encryption-customer-algorithm | No | |
| x-amz-server-side-encryption-customer-key-MD5 | No | |
| x-amz-version-id | Yes | |

B.6.27 CopyObject

[AWS Reference](#)

Request Syntax

```
PUT /bucket/key HTTP/1.1
Host: endpoint
Cache-Control: string
Content-Disposition: string
Content-Encoding: string
Content-Language: string
Content-Type: string
Expires: string
x-amz-copy-source: SourceBucket/SourceKey?versionId=SourceVersionId
x-amz-copy-source-if-match: etag
x-amz-copy-source-if-modified-since: timestamp
x-amz-copy-source-if-none-match: etag
x-amz-copy-source-if-unmodified-since: timestamp
x-amz-acl: canned-acl
x-amz-grant-full-control: grantee
x-amz-grant-read: grantee
x-amz-grant-read-acl: grantee
x-amz-grant-write-acp: grantee
x-amz-expected-bucket-owner: owner-ID
x-amz-source-expected-bucket-owner: owner-ID
x-amz-meta-*: string
```

Response Syntax

```
HTTP/1.1 200 OK
x-amz-copy-source-version-id: string
x-amz-version-id: string

<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult>
  <ETag>string</ETag>
  <LastModified>timestamp</LastModified>
</CopyObjectResult>
```

Permissions

Bucket owner or any user granted WRITE permission on target bucket and READ permission on source object is permitted to copy object.

Request

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Bucket | Yes | |
| ACL - x-amz-acl | Yes | |
| CacheControl - Cache-Control | Yes | |
| ContentDisposition - Content-Disposition | Yes | |
| ContentEncoding - Content-Encoding | Yes | |
| ContentLanguage - Content-Language | Yes | |
| ContentType - Content-Type | Yes | |
| ChecksumAlgorithm - x-amz-sdk-checksum-algorithm | No | |
| Expires | Yes | |

| | | |
|--|-----|--|
| GrantFullControl - x-amz-grant-full-control | Yes | |
| GrantRead - x-amz-grant-read | Yes | |
| GrantReadACP - x-amz-grant-read-acp | Yes | |
| GrantWriteACP - x-amz-grant-write-acp | Yes | |
| Key | Yes | |
| Metadata - x-amz-meta-* | Yes | |
| ServerSideEncryption - x-amz-server-side-encryption | No | |
| StorageClass - x-amz-storage-class | No | |
| WebsiteRedirectLocation - x-amz-website-redirect-location | No | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKey - x-amz-server-side-encryption-customer-key | No | |
| SSEKMSKeyId - x-amz-server-side-encryption-aws-kms-key-id | No | |
| SSEKMSEncryptionContext - x-amz-server-side-encryption-context | No | |
| BucketKeyEnabled - x-amz-server-side-encryption-bucket-key-enabled | No | |
| RequestPayer - x-amz-request-payer | No | |
| Tagging - x-amz-tagging | Yes | |
| ObjectLockMode - x-amz-object-lock-mode | No | |
| ObjectLockRetainUntilDate - x-amz-object-lock-retain-until-date | No | |
| ObjectLockLegalHoldStatus - x-amz-object-lock-legal-hold | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |
| CopySource - x-amz-copy-source | Yes | format: SourceBucket/ SourceKey?versionId=SourceVersionId |
| CopySourceIfMatch - x-amz-copy-source-if-match | Yes | |
| CopySourceIfModifiedSince - x-amz-copy-source-if-modified-since | Yes | |
| CopySourceIfNoneMatch - x-amz-copy-source-if-none-match | Yes | |
| CopySourceIfUnmodifiedSince - x-amz-copy-source-if-unmodified-since | Yes | |
| MetadataDirective - x-amz-metadata-directive | Yes | |
| TaggingDirective - x-amz-tagging-directive | Yes | |
| CopySourceSSECustomerAlgorithm - x-amz-copy-source-server-side-encryption-customer-algorithm | No | |
| CopySourceSSECustomerKey - x-amz-copy-source-server-side-encryption-customer-key | No | |
| CopySourceSSECustomerKeyMD5 - x-amz-copy-source-server-side-encryption-customer-key-MD5 | No | |
| ExpectedSourceBucketOwner - x-amz-source-expected-bucket-owner | Yes | |

CopySource body

| Parameter | Implemented | Comments |
|-----------|-------------|----------|
| Bucket | Yes | |
| Key | Yes | |
| VersionId | Yes | |

Response

| Parameter | Implemented | Comments |
|--|-------------|----------|
| ETag | No | |
| LastModified | No | |
| ChecksumCRC32 - x-amz-checksum-crc32 | No | |
| ChecksumCRC32C - x-amz-checksum-crc32c | No | |
| ChecksumSHA1 - x-amz-checksum-sha1 | No | |
| ChecksumSHA256 - x-amz-checksum-sha256 | No | |
| Expiration | No | |
| CopySourceVersionId | Yes | |
| VersionId | Yes | |
| ServerSideEncryption - x-amz-server-side-encryption | No | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |
| SSEKMSKeyId - x-amz-server-side-encryption-aws-kms-key-id | No | |
| SSEKMSEncryptionContext - x-amz-server-side-encryption-context | No | |
| BucketKeyEnabled - x-amz-server-side-encryption-bucket-key-enabled | No | |
| RequestCharged - x-amz-request-charged | No | |

B.6.28 UploadPartCopy

[AWS Reference](#)

Request Syntax

```
PUT /bucket/key?partNumber=integer&uploadId=string HTTP/1.1
Host: endpoint
x-amz-copy-source: string
x-amz-copy-source-if-match: etag
x-amz-copy-source-if-modified-since: timestamp
x-amz-copy-source-if-none-match: etag
x-amz-copy-source-if-unmodified-since: timestamp
x-amz-copy-source-range: bytes=integer-integer
x-amz-expected-bucket-owner: owner-ID
x-amz-source-expected-bucket-owner: owner-ID
```

Response Syntax

```
HTTP/1.1 200 OK
x-amz-copy-source-version-id: string

<?xml version="1.0" encoding="UTF-8"?>
<CopyObjectResult>
  <ETag>string</ETag>
  <LastModified>timestamp</LastModified>
</CopyObjectResult>
```

Permissions

Upload owner or any user granted WRITE permission on target upload and READ permission on source object is permitted to copy (part of) object to upload part.

Request

| Parameter | Implemented | Comments |
|--|-------------|----------|
| Bucket | Yes | |
| CopySource - x-amz-copy-source | Yes | |
| CopySourceIfMatch - x-amz-copy-source-if-match | Yes | |
| CopySourceIfModifiedSince - x-amz-copy-source-if-modified-since | Yes | |
| CopySourceIfNoneMatch - x-amz-copy-source-if-none-match | Yes | |
| CopySourceIfUnmodifiedSince - x-amz-copy-source-if-unmodified-since | Yes | |
| CopySourceRange - x-amz-copy-source-range | Yes | |
| Key | Yes | |
| PartNumber - partNumber | Yes | |
| UploadId - uploadId | Yes | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKey - x-amz-server-side-encryption-customer-key | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |
| CopySourceSSECustomerAlgorithm - x-amz-copy-source-server-side-encryption-customer-algorithm | No | |
| CopySourceSSECustomerKey - x-amz-copy-source-server-side-encryption-customer-key | No | |

| | |
|---|-----|
| CopySourceSSECustomerKeyMD5 - x-amz-copy-source-server-side-encryption-customer-key-MD5 | No |
| RequestPayer - x-amz-request-payer | No |
| ExpectedBucketOwner -- x-amz-expected-bucket-owner | Yes |
| ExpectedSourceBucketOwner - x-amz-source-expected-bucket-owner | Yes |

Response

| Parameter | Implemented | Comments |
|--|-------------|----------|
| CopySourceVersionId - x-amz-copy-source-version-id | Yes | |
| LastModified | Yes | |
| ETag | Yes | |
| ChecksumCRC32 - x-amz-checksum-crc32 | No | |
| ChecksumCRC32C - x-amz-checksum-crc32c | No | |
| ChecksumSHA1 - x-amz-checksum-sha1 | No | |
| ChecksumSHA256 - x-amz-checksum-sha256 | No | |
| ServerSideEncryption - x-amz-server-side-encryption | No | |
| SSECustomerAlgorithm - x-amz-server-side-encryption-customer-algorithm | No | |
| SSECustomerKeyMD5 - x-amz-server-side-encryption-customer-key-MD5 | No | |
| SSEKMSKeyId - x-amz-server-side-encryption-aws-kms-key-id | No | |
| BucketKeyEnabled - x-amz-server-side-encryption-bucket-key-enabled | No | |
| RequestCharged - x-amz-request-charged | No | |

B.6.29 PutBucketPolicy

[AWS Reference](#)

[Policies and Permissions in Amazon S3](#)

Request Syntax

```
PUT /?policy HTTP/1.1
Host: endpoint
Content-MD5: ContentMD5
x-amz-sdk-checksum-algorithm: ChecksumAlgorithm
x-amz-confirm-remove-self-bucket-access: ConfirmRemoveSelfBucketAccess
x-amz-expected-bucket-owner: ExpectedBucketOwner
```

```
{ Policy in JSON format }
```

Response Syntax

```
HTTP/1.1 200 OK
```

Permissions

Required PutBucketPolicy permissions. Bucket owner can always use this operation, even if the policy explicitly denies the root user the ability to perform this action.

Differences from AWS S3

| Parameter | AWS S3 | RED S3 |
|-----------------------|--|--|
| aws:PrincipalAccount | Returns AWS Account Id | Forbidden |
| aws:PrincipalArn | Returns the ARN based on AWS Account Id | Forbidden |
| aws:PrincipalType | Returns Account, User, AssumedRole and so on according to the spec | Always returns 'Account' |
| aws:userId | Returns AWS account Id of the Principal | Returns Canonical Id of the Principal |
| aws:username | Returns user name of the Principal | Returns display name (i.e. email address) of the Principal |
| s3:locationconstraint | Returns the value according the spec | Forbidden |
| s3:ResourceAccount | Returns AWS Account Id of the bucket owner | Returns Canonical Id of the bucket owner |
| Policy structure | AWS S3 does not allow duplicated keys | RED S3 allows duplicated keys |

Request

| Parameter | Implemented | Comments |
|---|-------------|---------------------|
| Bucket | Yes | |
| ContentMD5 - Content-MD5 | No | |
| ConfirmRemoveSelfBucketAccess - x-amz-confirm-remove-self-bucket-access | No | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |
| ChecksumAlgorithm - x-amz-sdk-checksum-algorithm | No | |
| Policy | Yes | Body in JSON format |

Policy Element

| Parameter | Implemented | Comments |
|-----------|-------------|--|
| Resources | Yes | Supported: bucket and object |
| Actions | Yes | List of supported actions is available |
| Effect | Yes | Valid values are Allow and Deny |
| Principal | Yes | |
| Condition | Yes | Condition keys are available |

Policy Action - only supported

| Parameter | Implemented | Comments |
|----------------------------|-------------|----------|
| AbortMultipartUpload | Yes | |
| DeleteBucket | Yes | |
| DeleteBucketPolicy | Yes | |
| DeleteObject | Yes | |
| DeleteObjectTagging | Yes | |
| DeleteObjectVersion | Yes | |
| DeleteObjectVersionTagging | Yes | |
| GetBucketAcl | Yes | |
| GetBucketCORS | Yes | |
| GetBucketLocation | Yes | |
| GetBucketLogging | Yes | |
| GetBucketPolicy | Yes | |
| GetBucketTagging | Yes | |
| GetBucketVersioning | Yes | |
| GetObject | Yes | |
| GetObjectAcl | Yes | |
| GetObjectTagging | Yes | |
| GetObjectVersion | Yes | |
| GetObjectVersionAcl | Yes | |
| GetObjectVersionTagging | Yes | |
| ListBucket | Yes | |
| ListBucketMultipartUploads | Yes | |
| ListBucketVersions | Yes | |
| ListMultipartUploadParts | Yes | |
| PutBucketAcl | Yes | |
| PutBucketCORS | Yes | |
| PutBucketLogging | Yes | |
| PutBucketPolicy | Yes | |
| PutBucketTagging | Yes | |
| PutBucketVersioning | Yes | |
| PutObject | Yes | |

| | |
|-------------------------|-----|
| PutObjectAcl | Yes |
| PutObjectTagging | Yes |
| PutObjectVersionAcl | Yes |
| PutObjectVersionTagging | Yes |

Policy Condition Key - only supported

| Parameter | Implemented | Comments |
|-----------------------------|-------------|----------|
| aws:CurrentTime | Yes | |
| aws:EpochTime | Yes | |
| aws:PrincipalType | Yes | |
| aws:SecureTransport | Yes | |
| aws:SourceIp | Yes | |
| aws:user-id | Yes | |
| aws:username | Yes | |
| s3:authType | Yes | |
| s3:delimiter | Yes | |
| s3:max-keys | Yes | |
| s3:prefix | Yes | |
| s3:signatureAge | Yes | |
| s3:signatureversion | Yes | |
| s3:TlsVersion | Yes | |
| s3:versionid | Yes | |
| s3:x-amz-acl | Yes | |
| s3:s3:x-amz-content-sha256 | Yes | |
| s3:x-amz-copy-source | Yes | |
| s3:x-amz-grant-full-control | Yes | |
| s3:x-amz-grant-read | Yes | |
| s3:x-amz-grant-read-acp | Yes | |
| s3:x-amz-grant-write | Yes | |
| s3:x-amz-grant-write-acp | Yes | |
| s3:x-amz-metadata-directive | Yes | |

Response

| Parameter | Implemented | Comments |
|-----------|-------------|----------|
| Bucket | No | |
| Policy | Yes | |

B.6.30 PutBucketVersioning

[AWS Reference](#)

Request Syntax

```
PUT /?versioning HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: ExpectedBucketOwner

<?xml version="1.0" encoding="UTF-8"?>
<VersioningConfiguration xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
  <Status>string</Status>
</VersioningConfiguration>
```

Response Syntax

```
HTTP/1.1 200 OK
```

VersioningConfiguration Element

| Parameter | Implemented | Comments |
|-----------|-------------|----------|
| MFADelete | No | |
| Status | Yes | |

B.6.31 GetBucketVersioning

[AWS Reference](#)

Request Syntax

```
GET /?versioning HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: ExpectedBucketOwner
```

Response Syntax

```
HTTP/1.1 200 OK

<?xml version="1.0" encoding="UTF-8"?>
<VersioningConfiguration>
  <Status>string</Status>
</VersioningConfiguration>
```

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

Response body

| Parameter | Implemented | Comments |
|-----------|-------------|----------|
| Status | Yes | |
| MFADelete | No | |

B.6.32 GetBucketPolicy

[AWS Reference](#)

Request Syntax

```
GET /?policy HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: ExpectedBucketOwner
```

Response Syntax

```
HTTP/1.1 200
{ Policy in JSON format }
```

Permissions

Required GetBucketPolicy permissions. Bucket owner can always use this operation, even if the policy explicitly denies the root user the ability to perform this action.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

Response

| Parameter | Implemented | Comments |
|-----------|-------------|-----------------------|
| Policy | Yes | Policy in JSON format |

Policy structure

| Parameter | Implemented | Comments |
|------------------------|-------------|----------------------------|
| Version | Yes | Supported value 2012-10-17 |
| Id | No | |
| Statement | Yes | |
| Statement >> Sid | Yes | |
| Statement >> Effect | Yes | |
| Statement >> Principal | Yes | |
| Statement >> Action | Yes | |
| Statement >> Resource | Yes | |

B.6.33 DeleteBucketPolicy

[AWS Reference](#)

Request Syntax

```
DELETE /?policy HTTP/1.1
Host: endpoint
x-amz-expected-bucket-owner: ExpectedBucketOwner
```

Response Syntax

```
HTTP/1.1 204
```

Permissions

Required DeleteBucketPolicy permissions. Bucket owner can always use this operation, even if the policy explicitly denies the root user the ability to perform this action.

Request

| Parameter | Implemented | Comments |
|---|-------------|----------|
| Bucket | Yes | |
| ExpectedBucketOwner - x-amz-expected-bucket-owner | Yes | |

B.7 S3-Compatible API Error Codes

DDN Infinia supports all S3 compatible xml error codes.

B.8 URL Processing Rules

All endpoints obey the URL processing rules.

B.9 AWS CLI Compatibility

The following are AWS CLI compatible commands:

- `cp`
- `ls`
- `mb`
- `mv`
- `presign`
- `rb`
- `rm`
- `sync`

cp

Copies a local file or S3 object to another location locally or in S3.

Command Syntax

cp <Path source> <Path destination> [Options]

| Paths | Implemented | Comments |
|------------------|-------------|------------------------|
| Path source | yes | <LocalPath> or <S3Uri> |
| Path destination | yes | <LocalPath> or <S3Uri> |
| Options | Implemented | Comments |
| --acl | yes | |
| --grants | yes | |

ls

List S3 objects and common prefixes under a prefix or all S3 buckets.

Command Syntax

ls <S3Uri> [Options]

| Paths | Implemented | Comments |
|-------------|-------------|----------|
| S3Uri | yes | |
| Options | Implemented | Comments |
| --page-size | yes | |

mb

Creates an S3 bucket.

Command Syntax

mb <S3Uri>

| Paths | Implemented | Comments |
|-------|-------------|----------|
| S3Uri | yes | |

mv

Moves a local file or S3 object to another location locally or in S3.

Command Syntax

mv <Path source> <Path destination> [Options]

| Paths | Implemented | Comments |
|------------------|-------------|------------------------|
| Path source | yes | <LocalPath> or <S3Uri> |
| Path destination | yes | <LocalPath> or <S3Uri> |

| Options | Implemented | Comments |
|----------|-------------|----------|
| --acl | yes | |
| --grants | yes | |

presign

Generate a pre-signed URL for an Amazon S3 object. DDN Infinia supports pre-signed URLs for Auth v4 only.

Command Syntax

presign <S3Uri> [Options]

| Path | Implemented | Comments |
|-------|-------------|----------|
| S3Uri | yes | |

| Options | Implemented | Comments |
|--------------|-------------|----------|
| --expires-in | yes | |

rb

Deletes an empty S3 bucket.

Command Syntax

rb <S3Uri> [Options]

| Paths | Implemented | Comments |
|-------|-------------|----------|
| S3Uri | yes | |

| Options | Implemented | Comments |
|---------|-------------|--|
| --force | yes | RED S3 with RED FS can delete a non-empty bucket, that is an old bug |

rm

Deletes an S3 object.

Command Syntax

rm <S3Uri> [Options]

| Paths | Implemented | Comments |
|-------------|-------------|----------|
| S3Uri | yes | |
| Option | Implemented | Comments |
| --recursive | yes | |

sync

Syncs directories and S3 prefixes. Recursively copies new and updated files from the source directory to the destination.

Command Syntax

sync <Path source> <Path destination> [Options]

| Paths | Implemented | Comments |
|------------------|-------------|------------------------|
| Path source | yes | <LocalPath> or <S3Uri> |
| Path destination | yes | <LocalPath> or <S3Uri> |
| Options | Implemented | Comments |
| --acl | yes | |
| --grants | yes | |

Appendix C Using the DDN Infinia REST API

This appendix expands the information provided in [REST API](#) (Section 1.11 on page 24) and provides details about the following:

- [REST API Environment](#) (Appendix C.1 on page 249)
- [User Authentication](#) (Appendix C.2 on page 249)
- [Basic REST API Operations](#) (Appendix C.3 on page 250)

NOTE: The examples in this section use `curl`; however, many other command line and browser-based tools are available to simplify these steps. Any modern programming language can be used for developing a REST client.

To help explore the DDN Infinia REST API, issue `redcli` commands with the `--verbose` flag, which will log the underlying REST API calls to use as reference or refer to the *DDN Infinia REST API Reference*.

C.1 REST API Environment

During DDN Infinia and realm configuration, a default realm administrator named `realm_admin` is created with a password specified by the installer.

The REST API server uses the `RED_ETCD`, `RED_USER` and `RED_PASSWORD` environment variables to bootstrap and find the DDN Infinia system with which it is affiliated. The `RED_PASSWORD` environment variable sets the initial REST API realm password for the first REST API run.

The following are the default settings:

```
RED_USER=realm_admin
RED_PASSWORD=changeme1234
```

Administrators must change these credentials to their own secure settings by a user-update call.

The DDN Infinia REST API logs are available on the host node running `redapi` and in the container itself:

```
/var/log/red/redapi-<hostuuid>/redapi.log
```

For additional details about logs, see [Collect and View DDN Infinia Logs](#) (Section 8.6 on page 173).

C.2 User Authentication

Authenticate users by calling the `/auth_user` endpoint and passing `user_id` and `password` as header parameters.

Example

This request example retrieves an authentication token for the `realm_admin` user.

```
$ curl -k -X GET "https://<hostname>/redapi/v1/auth_user" \
-H "accept: */*" -H "user_id: realm_admin" -H "password: <password>"
```

Save the token that is retrieved for use in subsequent commands by altering the request in the previous example as follows:

```
$ TOKEN=$(curl -s -X GET -H "User_id: realm_admin" \
-H "Password: <password>" https://edge02:8499/redapi/v1/auth_user | jq -r '.token')
```

C.3 Basic REST API Operations

This section provides the following examples of basic operations performed using the DDN Infinia REST API:

- ❖ [List All Clusters](#)
- ❖ [Create a Cluster](#)
- ❖ [Show Details of a Cluster](#)
- ❖ [Start a Cluster](#)
- ❖ [Stop a Cluster](#)
- ❖ [Query Cluster Status](#)
- ❖ [Delete a Cluster](#)
- ❖ [Show Inventory Details](#)
- ❖ [Run Discovery](#)
- ❖ [Auto-create a Configuration](#)
- ❖ [Propose a New Configuration](#)

For a complete list of all available resources, see the *DDN Infinia REST API Reference*.

C.3.1 List All Clusters

Retrieve a list of all clusters by sending a GET request to the `/clusters` endpoint.

Example

This response example shows one cluster named `cluster-gray`.

```
$ curl https://edge02:8499/redapi/v1/clusters -H "Authorization: Bearer $TOKEN"
{
  "cluster-gray": "https://edge02:8499/redapi/v1/clusters/cluster-gray"
}
```

C.3.2 Create a Cluster

Create an empty cluster by posting a JSON object with the name of the new cluster to the `/clusters` endpoint.

Example

This example creates a new cluster named `cluster-gray`.

```
$ curl -X POST https://edge02:8499/redapi/v1/clusters \
--data '{"path":"clusters/cluster-gray/"}' -H "Authorization: Bearer $TOKEN" \
-H "Content-Type: application/json"
{
  "cluster-gray": "https://edge02:8499/redapi/v1/clusters/cluster-gray"
}
```

Retrieve a list of all clusters by sending a GET request to the `/clusters` endpoint.

Example

This example retrieves a list of all clusters.

```
$ curl -k -X GET "https://<hostname>/redapi/v1/auth_user" -H "accept: */*" \
-H "user_id: realm_admin" -H "password: <password>"
```

Retrieve details for a single cluster by sending a GET request to the `/clusters/{cluster_name}` endpoint.

Example

Continuing the previous examples, the following response example shows that there is no configuration data in the new cluster.

```
$ curl https://edge02:8499/redapi/v1/clusters/cluster-gray -H "Authorization: Bearer $TOKEN"
{
  "desired_config": ""
}
```

C.3.3 Show Details of a Cluster

Retrieve details for a single cluster by sending a GET request to the `/clusters/{cluster_name}` endpoint.

Example

The following response example shows that there is no configuration data in the `cluster-gray` cluster.

```
$ curl https://edge02:8499/redapi/v1/clusters/cluster-gray -H "Authorization: Bearer $TOKEN"
{
  "desired_config": ""
}
```

C.3.4 Start a Cluster

Start all instances for a cluster by sending a POST request to the `/clusters/{cluster_name}/start` endpoint.

Example

This request example starts all instance for the `cluster-gray` cluster.

```
$ curl -X POST https://edge02:8499/redapi/v1/clusters/cluster-gray/start \
-H "Authorization: Bearer $TOKEN"
[
  {
    "id": 1001,
    "jsonrpc": "2.0",
    "result": true,
    "timestamp": 1596831724673207172,
    "version": "1.1"
  }
]
```

C.3.5 Stop a Cluster

Stop all instances for a cluster by sending a POST request to the `/clusters/{cluster_name}/stop` endpoint.

Example

This request example stops all instance for the `cluster-gray` cluster.

```
$ curl -X POST https://edge02:8499/redapi/v1/clusters/cluster-gray/stop \
-H "Authorization: Bearer $TOKEN"
[
  {
    "id": 1009,
    "jsonrpc": "2.0",
    "result": true,
    "timestamp": 1596833275536923321,
    "version": "1.1"
  }
]
```

C.3.6 Query Cluster Status

Retrieve the current cluster status by sending a GET request to the `/clusters/{cluster_name}/status` endpoint.

Example

This response example shows the status for the `cluster-gray` cluster. The output indicates that the devices (CATS) are enabled for read and write, the cluster state is running, and the instances are joined. Further, the value of cce and sce are the same, indicating the cluster is up and ready for use.

```
$ curl https://edge02:8499/redapi/v1/clusters/cluster-gray/status \
-H "Authorization: Bearer $TOKEN"
{
  "cats": {
    "1": {
      "can_read": true, "can_write": true, "evicted": false, "eviction_catchup": false, "joined":
true, "owner": 1, "uuid": "32e0b453-3a1b-42ab-8430-f0cc5e4e8d6c"
    },
    ...
  },
  "cce": 7, "cluster_state": "running", "cluster_state_epoch": 7, "instances": {
    "1": {
      "evicted": false,
      "joined": true,
      "replay": false,
      "uuid": "c5823f22-f294-41f1-a9d1-6db17e4545d1"
    }
  },
  "rde": 6,
  "sce": 7
}
```

C.3.7 Delete a Cluster

Prior to deleting a cluster, ensure that you stop all instances for the cluster by sending a POST request to the `/clusters/{cluster_name}/stop` endpoint.

Delete the cluster by sending a DELETE request to the `/clusters/{cluster_name}` endpoint.

Example

These request examples stop all instance for the `cluster-gray` cluster and then delete the cluster.

```
$ curl -X POST https://edge02:8499/redapi/v1/clusters/cluster-gray/stop \
-H "Authorization: Bearer $TOKEN"
[
  {
    "id": 1009,
    "jsonrpc": "2.0",
    "result": true,
    "timestamp": 1596833275536923321,
    "version": "1.1"
  }
]

$ curl -X DELETE https://edge02:8499/redapi/v1/clusters/cluster-gray \
-H "Authorization: Bearer $TOKEN"
```

Continuing with this example, retrieving a list of all clusters returns an empty set.

```
$ curl https://edge02:8499/redapi/v1/clusters -H "Authorization: Bearer $TOKEN"
{}
```

C.3.8 Show Inventory Details

Retrieve the current hardware inventory data by sending a GET request to the `/inventory` endpoint.

Example

The following response example shows the inventory data for the current hardware.

```
$ curl https://edge02:8499/redapi/v1/inventory -H "Authorization: Bearer $TOKEN"
{
  "path": "inventory/",
  "tree": {
    "devices": {
      "Samsung SSD 970 EVO Plus 500GB      S4P2NG0M108176K      ": {
        ...
      }
    }
  }
}
```

C.3.9 Run Discovery

Run discovery to update the current hardware inventory by sending a PUT request to the `/inventory` endpoint.

Example

This request example runs discovery to update the current hardware inventory.

```
$ curl -X PUT https://edge02:8499/redapi/v1/inventory -H "Authorization: Bearer $TOKEN"
{
  "detail": "Request fulfilled, document follows",
  "instance": "/redapi/v1/inventory",
  "status": 200,
  "timestamp": "2020-08-07T19:18:50 +0000",
  "title": "Request successful: OK"
}
```

Retrieve the current hardware inventory data by sending a GET request to the `/inventory` endpoint.

Example

Continuing the previous example, the following response example shows the inventory data for the current hardware.

```
$ curl https://edge02:8499/redapi/v1/inventory -H "Authorization: Bearer $TOKEN"
{
  "path": "inventory/",
  "tree": {
    "devices": {
      "Samsung SSD 970 EVO Plus 500GB      S4P2NG0M108176K      ": {
        ...
      }
    }
  }
}
```

Filter the output JSON data using the `jq` tool, as shown in the following example.

Example

The following response example shows only the device identifiers (udids) that are in the current hardware inventory.

```
$ curl https://edge02:8499/redapi/v1/inventory \
-H "Authorization: Bearer $TOKEN" | jq -r '.tree.devices | keys[] '

Samsung SSD 970 EVO Plus 500GB      S4P2NG0M108176K
Samsung SSD 970 EVO Plus 500GB      S4P2NG0M108366R
Samsung SSD 970 EVO Plus 500GB      S4P2NG0M108367M
Samsung SSD 970 EVO Plus 500GB      S4P2NG0M108372E
```

A more complicated example lists the device identifier, the device location, and device name.

```
$ curl http://edge02:8099/redapi/v1/inventory \
-H "Authorization: Bearer $TOKEN" | jq \
-r '.tree.devices | keys[] as $k | "\($k), \(.[$k] | .nodes | keys[] as $n \
| "\(.[$n] | [.location , .device_name])")"'

Samsung SSD 970 EVO Plus 500GB      S4P2NG0M108176K      , ["edge01, addr: 0000:44:00.0","vfio44"]
Samsung SSD 970 EVO Plus 500GB      S4P2NG0M108366R      , ["edge01, addr: 0000:45:00.0","vfio45"]
Samsung SSD 970 EVO Plus 500GB      S4P2NG0M108367M      , ["edge01, addr: 0000:46:00.0","vfio46"]
Samsung SSD 970 EVO Plus 500GB      S4P2NG0M108372E      , ["edge01, addr: 0000:43:00.0","vfio43"]
```

C.3.10 Auto-create a Configuration

Auto-create a configuration in your cluster by posting a JSON object with the name of the new configuration to the `/clusters/{cluster_name}/configs` endpoint and passing the `autocreate=true` query parameter.

Example

This request example creates a new configuration named `newcfg` for the `cluster-gray` cluster.

```
$ curl -X POST https://edge02:8499/redapi/v1/clusters/cluster-gray/configs?autocreate=true \
--data '{"path":"clusters/cluster-gray/configs/newcfg"}' -H "Authorization: Bearer $TOKEN" \
-H "Content-Type: application/json"
{
  "newcfg": "https://edge02:8499/redapi/v1/clusters/cluster-gray/newcfg"
}
```

Retrieve a list of all configurations in the cluster by sending a GET request to the `/clusters/{cluster_name}/configs` endpoint.

Example

Continuing the previous example, the following response example shows the new configuration named `newcfg`.

```
$ curl https://edge02:8499/redapi/v1/clusters/cluster-gray/configs \
-H "Authorization: Bearer $TOKEN"
{
  "newcfg": "https://edge02:8499/redapi/v1/clusters/cluster-gray/configs/newcfg"
}
```

Retrieve details for a single configuration by sending a GET request to the `/clusters/{cluster_name}/configs/{config_name}` endpoint.

Example

Continuing the previous examples, the following response example shows the auto-generated configuration settings.

```
$ curl https://edge02:8499/redapi/v1/clusters/cluster-gray/configs/newcfg \
-H "Authorization: Bearer $TOKEN"
{
  "path": "clusters/cluster-gray/configs/newcfg/",
  "tree": {
    "cluster_tunables": {
      ...
    }
  }
}
```

Filter the output JSON data using the `grep` tool, as shown in the following example.

Example

The following response example shows only the device identifiers (udids), which is the same set of devices that were shown in the previous discovery example ().

```
$ curl -s https://edge02:8499/redapi/v1/clusters/cluster-gray/configs/newcfg \
-H "Authorization: Bearer $TOKEN" | grep udid
  "udid": "Samsung SSD 970 EVO Plus 500GB"          S4P2NG0M108372E    "
  "udid": "Samsung SSD 970 EVO Plus 500GB"          S4P2NG0M108367M    "
  "udid": "Samsung SSD 970 EVO Plus 500GB"          S4P2NG0M108176K    "
  "udid": "Samsung SSD 970 EVO Plus 500GB"          S4P2NG0M108366R    "
```

C.3.11 Propose a New Configuration

Initialize the runtime of the cluster by sending a PUT request to the `/clusters/{cluster_name}/desired_config/{config_name}` endpoint and passing the `initinstances=true` query parameter.

Example

This request example updates the runtime to a newly created configuration by setting the `desired_config` of the `cluster-gray` cluster to `newcfg`.

```
$ curl -X PUT \
https://edge02:8499/redapi/v1/clusters/cluster-gray/desired_config/newcfg?initinstances=true \
-H "Authorization: Bearer $TOKEN"
{
  "cluster-gray": "https://edge02:8499/redapi/v1/clusters/cluster-gray/desired_config/newcfg"
}
```

Retrieve details for the runtime configuration by sending a GET request to the `/clusters/{cluster_name}/configs/runtime` endpoint.

Example

Continuing the previous examples, the following response example shows the runtime configuration, which is now the same devices that were in `newcfg`.

```
$ curl -s https://edge02:8499/redapi/v1/clusters/cluster-gray/configs/runtime \
-H "Authorization: Bearer $TOKEN" | grep udid
      "udid": "Samsung SSD 970 EVO Plus 500GB           S4P2NG0M108176K   "
      "udid": "Samsung SSD 970 EVO Plus 500GB           S4P2NG0M108366R   "
      "udid": "Samsung SSD 970 EVO Plus 500GB           S4P2NG0M108367M   "
      "udid": "Samsung SSD 970 EVO Plus 500GB           S4P2NG0M108372E   "
```


Appendix D `nettest` Utility

NOTE: This section applies to realm admins only.

This appendix provides details about the following:

- [Prerequisites to Execute `nettest`](#) (Appendix D.1 on page 258)
- [Execute the `nettest` Test Cases](#) (Appendix D.2 on page 259)
- [Update the `nettest` Configuration and Tests](#) (Appendix D.3 on page 262)
- [Audit the Network Connections and Workloads using `nettest`](#) (Appendix D.4 on page 263)
- [Add Client Nodes to the `nettest` Configuration](#) (Appendix D.5 on page 264)

Run the `nettest` utility to validate and characterize the network connectivity and performance between DDN Infinia instances, and optionally, client nodes.

D.1 Prerequisites to Execute nettest

Before running **nettest**, ensure the following prerequisites are met:

- If you are required to execute tests that include client IO, setup the DDN Infinia Client following the instructions in [Setup DDN Infinia Client](#) (Section 2.6 on page 69).

- Log into DDN Infinia:

```
redcli user login <username> -p <password>
```

- Establish the DDN Infinia client environment:

```
redcli client setup  
source $HOME/.config/red/redrc
```

- Verify that you have a running DDN Infinia cluster:

```
redcli cluster show -s
```

- If you are running **nettest** using the nettest-agent (daemon), ensure that the node on which **nettest** is to be executed, has network access to the IPs of the DDN Infinia instances. If you are running **nettest** using the redcli, this prerequisite is not required.

The **nettest** utility requires two configuration files:

- **\$HOME/.config/red/nettest/{cluster_name}_config.json** — Specifies the collection of DDN Infinia storage nodes and DDN Infinia client nodes.
- **\$HOME/.config/red/nettest/{cluster_name}_test.json** — Specifies the collection of test scenarios that **nettest** may execute.

Both files are generated the first time you run **nettest run exec** and will be automatically updated if a change in the DDN Infinia configuration is detected.

D.2 Execute the nettest Test Cases

Run one or more specific tests using **nettest run exec** and passing the test numbers with the **-f** argument. Note that if the nettest configuration and test files are not found, they will be generated. The default recommended tests are **0,26,27**. These tests perform a basic all-all performance test on the cluster.

Example

The following is an example output for **nettest run exec -f 0,26,27**.

| Test: WARMUP: all -> all (parallel) multi-threaded 0k RPC | | | | |
|---|-------------------|---------|-----------------------------|-------------------------------|
| ENDPOINT | HOST | IOPS | IOPS RANGE [min/avg/max] | LATENCY (µs) [min/avg/max] |
| al-ddn-infinia-01_client_48c11cd8 | al-ddn-infinia-01 | 580608 | [580608/580608/580608] | [19/19/19] |
| al-ddn-infinia-01_instance_1 | al-ddn-infinia-01 | 235324 | [235324/235324/235324] | [39/39/39] |
| al-ddn-infinia-02_client_43b42e5f | al-ddn-infinia-02 | 491142 | [491142/491142/491142] | [22/22/22] |
| al-ddn-infinia-02_instance_2 | al-ddn-infinia-02 | 172576 | [172576/172576/172576] | [57/57/57] |
| al-ddn-infinia-03_client_9a3ba323 | al-ddn-infinia-03 | 451487 | [451487/451487/451487] | [24/24/24] |
| al-ddn-infinia-03_instance_3 | al-ddn-infinia-03 | 276670 | [276670/276670/276670] | [34/34/34] |
| al-ddn-infinia-04_client_6e9d4de3 | al-ddn-infinia-04 | 457470 | [457470/457470/457470] | [24/24/24] |
| al-ddn-infinia-04_instance_4 | al-ddn-infinia-04 | 296175 | [296175/296175/296175] | [34/34/34] |
| al-ddn-infinia-05_client_de96d541 | al-ddn-infinia-05 | 361717 | [361717/361717/361717] | [33/33/33] |
| al-ddn-infinia-05_instance_5 | al-ddn-infinia-05 | 220660 | [220660/220660/220660] | [41/41/41] |
| al-ddn-infinia-06_client_4c28841e | al-ddn-infinia-06 | 352679 | [352679/352679/352679] | [33/33/33] |
| al-ddn-infinia-06_instance_6 | al-ddn-infinia-06 | 318516 | [318516/318516/318516] | [30/30/30] |
| al-ddn-infinia-07_client_9b4a9938 | al-ddn-infinia-07 | 339494 | [339494/339494/339494] | [35/35/35] |
| al-ddn-infinia-07_instance_7 | al-ddn-infinia-07 | 249337 | [249337/249337/249337] | [36/36/36] |
| al-ddn-infinia-08_client_bb2583a1 | al-ddn-infinia-08 | 356370 | [356370/356370/356370] | [29/29/29] |
| al-ddn-infinia-08_instance_8 | al-ddn-infinia-08 | 215430 | [215430/215430/215430] | [42/42/42] |
| al-ddn-infinia-09_instance_9 | al-ddn-infinia-09 | 283739 | [283739/283739/283739] | [32/32/32] |
| al-ddn-infinia-10_client_fd02c691 | al-ddn-infinia-10 | 399429 | [399429/399429/399429] | [27/27/27] |
| al-ddn-infinia-10_instance_10 | al-ddn-infinia-10 | 283053 | [283053/283053/283053] | [32/32/32] |
| TOTAL | | 6341887 | [172576/333783/580608] | |

| | ENDPOINT | HOST | IOPS |
|------------|------------------------------|-------------------|--------|
| Top 15% | al-ddn-infinia-04_instance_4 | al-ddn-infinia-04 | 296175 |
| | al-ddn-infinia-06_instance_6 | al-ddn-infinia-06 | 318516 |
| Bottom 15% | al-ddn-infinia-02_instance_2 | al-ddn-infinia-02 | 172576 |

Test: all -> all (parallel) multi-threaded 1k RPC

| ENDPOINT | HOST | IOPS | IOPS RANGE [min/avg/max] | LATENCY (µs) [min/avg/max] |
|-----------------------------------|-------------------|---------|-----------------------------|-------------------------------|
| al-ddn-infinia-01_client_48c11cd8 | al-ddn-infinia-01 | 472219 | [472219/472219/472219] | [198/198/198] |
| al-ddn-infinia-01_instance_1 | al-ddn-infinia-01 | 227847 | [227847/227847/227847] | [205/205/205] |
| al-ddn-infinia-02_client_43b42e5f | al-ddn-infinia-02 | 442425 | [442425/442425/442425] | [212/212/212] |
| al-ddn-infinia-02_instance_2 | al-ddn-infinia-02 | 186288 | [186288/186288/186288] | [258/258/258] |
| al-ddn-infinia-03_client_9a3ba323 | al-ddn-infinia-03 | 452376 | [452376/452376/452376] | [208/208/208] |
| al-ddn-infinia-03_instance_3 | al-ddn-infinia-03 | 232745 | [232745/232745/232745] | [200/200/200] |
| al-ddn-infinia-04_client_6e9d4de3 | al-ddn-infinia-04 | 461590 | [461590/461590/461590] | [204/204/204] |
| al-ddn-infinia-04_instance_4 | al-ddn-infinia-04 | 232490 | [232490/232490/232490] | [202/202/202] |
| al-ddn-infinia-05_client_de96d541 | al-ddn-infinia-05 | 459623 | [459623/459623/459623] | [204/204/204] |
| al-ddn-infinia-05_instance_5 | al-ddn-infinia-05 | 218250 | [218250/218250/218250] | [215/215/215] |
| al-ddn-infinia-06_client_4c28841e | al-ddn-infinia-06 | 449819 | [449819/449819/449819] | [208/208/208] |
| al-ddn-infinia-06_instance_6 | al-ddn-infinia-06 | 233461 | [233461/233461/233461] | [200/200/200] |
| al-ddn-infinia-07_client_9b4a9938 | al-ddn-infinia-07 | 467495 | [467495/467495/467495] | [203/203/203] |
| al-ddn-infinia-07_instance_7 | al-ddn-infinia-07 | 203585 | [203585/203585/203585] | [232/232/232] |
| al-ddn-infinia-08_client_bb2583a1 | al-ddn-infinia-08 | 438227 | [438227/438227/438227] | [214/214/214] |
| al-ddn-infinia-08_instance_8 | al-ddn-infinia-08 | 230236 | [230236/230236/230236] | [202/202/202] |
| al-ddn-infinia-09_instance_9 | al-ddn-infinia-09 | 221131 | [221131/221131/221131] | [215/215/215] |
| al-ddn-infinia-10_client_fd02c691 | al-ddn-infinia-10 | 448476 | [448476/448476/448476] | [209/209/209] |
| al-ddn-infinia-10_instance_10 | al-ddn-infinia-10 | 193475 | [193475/193475/193475] | [245/245/245] |
| TOTAL | | 6271768 | [186288/330093/472219] | |

| | ENDPOINT | HOST | IOPS |
|------------|------------------------------|-------------------|--------|
| Top 15% | al-ddn-infinia-03_instance_3 | al-ddn-infinia-03 | 232745 |
| | al-ddn-infinia-06_instance_6 | al-ddn-infinia-06 | 233461 |
| Bottom 15% | al-ddn-infinia-02_instance_2 | al-ddn-infinia-02 | 186288 |

Test: all -> all (parallel) multi-threaded 1m RPC (rdma)

| ENDPOINT | HOST | IOPS | IOPS RANGE [min/avg/max] | LATENCY (µs) [min/avg/max] |
|-----------------------------------|-------------------|-------|-----------------------------|-------------------------------|
| al-ddn-infinia-01_client_48c11cd8 | al-ddn-infinia-01 | 34246 | [34246/34246/34246] | [7467/7467/7467] |
| al-ddn-infinia-01_instance_1 | al-ddn-infinia-01 | 15958 | [15958/15958/15958] | [2875/2875/2875] |
| al-ddn-infinia-02_client_43b42e5f | al-ddn-infinia-02 | 25358 | [25358/25358/25358] | [8873/8873/8873] |
| al-ddn-infinia-02_instance_2 | al-ddn-infinia-02 | 13648 | [13648/13648/13648] | [3342/3342/3342] |

| | | | | |
|-----------------------------------|-------------------|--------|---------------------|------------------|
| a1-ddn-infinia-03_client_9a3ba323 | a1-ddn-infinia-03 | 27689 | [27689/27689/27689] | [6837/6837/6837] |
| a1-ddn-infinia-03_instance_3 | a1-ddn-infinia-03 | 19223 | [19223/19223/19223] | [2373/2373/2373] |
| a1-ddn-infinia-04_client_6e9d4de3 | a1-ddn-infinia-04 | 25987 | [25987/25987/25987] | [7464/7464/7464] |
| a1-ddn-infinia-04_instance_4 | a1-ddn-infinia-04 | 16076 | [16076/16076/16076] | [2897/2897/2897] |
| a1-ddn-infinia-05_client_de96d541 | a1-ddn-infinia-05 | 35145 | [35145/35145/35145] | [7692/7692/7692] |
| a1-ddn-infinia-05_instance_5 | a1-ddn-infinia-05 | 15043 | [15043/15043/15043] | [3068/3068/3068] |
| a1-ddn-infinia-06_client_4c28841e | a1-ddn-infinia-06 | 33914 | [33914/33914/33914] | [8004/8004/8004] |
| a1-ddn-infinia-06_instance_6 | a1-ddn-infinia-06 | 15069 | [15069/15069/15069] | [3073/3073/3073] |
| a1-ddn-infinia-07_client_9b4a9938 | a1-ddn-infinia-07 | 32141 | [32141/32141/32141] | [8003/8003/8003] |
| a1-ddn-infinia-07_instance_7 | a1-ddn-infinia-07 | 14392 | [14392/14392/14392] | [3186/3186/3186] |
| a1-ddn-infinia-08_client_bb2583a1 | a1-ddn-infinia-08 | 28655 | [28655/28655/28655] | [8072/8072/8072] |
| a1-ddn-infinia-08_instance_8 | a1-ddn-infinia-08 | 15450 | [15450/15450/15450] | [3001/3001/3001] |
| a1-ddn-infinia-09_instance_9 | a1-ddn-infinia-09 | 21062 | [21062/21062/21062] | [2179/2179/2179] |
| a1-ddn-infinia-10_client_fd02c691 | a1-ddn-infinia-10 | 21195 | [21195/21195/21195] | [8203/8203/8203] |
| a1-ddn-infinia-10_instance_10 | a1-ddn-infinia-10 | 15221 | [15221/15221/15221] | [3034/3034/3034] |
| TOTAL | | 425481 | [13648/22393/35145] | |

| | ENDPOINT | HOST | IOPS |
|------------|------------------------------|-------------------|-------|
| Top 15% | a1-ddn-infinia-03_instance_3 | a1-ddn-infinia-03 | 19223 |
| | a1-ddn-infinia-09_instance_9 | a1-ddn-infinia-09 | 21062 |
| Bottom 15% | a1-ddn-infinia-02_instance_2 | a1-ddn-infinia-02 | 13648 |

If required, run the full suite of standard tests using **nettest run exec**.

NOTE: Running all the tests may take an excessive amount of time.

Example

```
$ nettest run exec
6:49PM INF Cluster Configuration: cluster-gray_config.json, Size: 2455, Modified: 2023-02-14
18:48:17.082309486 +0000 UTC
6:49PM INF Cluster: cluster-gray
6:49PM INF 4 instances discovered
6:49PM INF 0 clients discovered
6:49PM INF Tests Configuration: cluster-gray_test.json, Size: 15556, Modified: 2023-02-14
18:48:17.102311352 +0000 UTC
6:49PM INF Running 41 tests.
Test 1/41: WARMUP: all -> all (parallel) multi-threaded 0k RPC 100% | (1/1, 59 it/min)
<output omitted>
```

D.3 Update the nettest Configuration and Tests

The nettest configuration files are automatically generated but you can regenerate them when required. For example, when client nodes are added, you need to regenerate both the configuration and the test case files:

```
nettest config generate
nettest test generate
```

View the nettest configuration using **nettest config show**.

Example

```
$ nettest config show
```

| CLUSTER | UUID | GROUP |
|--------------|--------------------------------------|-------|
| cluster-gray | 2f1c8155-8ccc-48a9-8465-36cbb8a4b9ed | |

| HOST | ID | TYPE | UUID | TLS | IPS | PORT |
|---------|----|----------|--------------------------------------|-------|-------------|------|
| server1 | 1 | instance | 4e0e06e3-852c-4ae2-b7b5-3723adf379ab | false | 10.138.0.4 | 3160 |
| server2 | 2 | instance | 10724c09-5e97-4875-b55b-ba38ffa993a1 | false | 10.138.0.6 | 3160 |
| server3 | 3 | instance | a6b95286-8e02-4e52-9d72-53ae16719ee4 | false | 10.138.0.8 | 3160 |
| server4 | 4 | instance | 0256c1c4-bea0-465a-ae89-32e459d06a07 | false | 10.138.0.31 | 3105 |

List the collection of tests using **nettest test list**.

Show the details of a specific test case using **nettest test show <testcase_number>**.

Example

The following example shows the details of the test case number ten for **cluster-gray**.

```
$ nettest test show 10
6:51PM INF Cluster: cluster-gray
```

| | |
|----------------|---|
| TEST 10 | server -> server single-threaded 1k RPC |
| SERIAL | true |
| RUNTIME (secs) | 10 |
| MULTITHREADED | false |
| INFLIGHT COUNT | 32 |
| SRC | 1_instance |
| DST | 2_instance |
| # TEST SPECS | 1 |

| TEST SPECS | | | | | | | |
|------------|---------|----------|----------|------------|------|-------|------------|
| # | REQ LEN | RESP LEN | RDMA LEN | RDMA FRAGS | RPC | TYPE | VERIFY CRC |
| 1 | 1024 | 0 | 0 | 0 | ping | false | 0 |

D.4 Audit the Network Connections and Workloads using nettest

Verify connectivity using **nettest run audit connections**. Note that no performance tests are executed by this command.

Example

```
$ nettest run audit connections
6:53PM INF Cluster: cluster-gray
6:53PM INF 4 instances discovered
6:53PM INF 0 clients discovered
6:53PM INF Connections audit:
6:53PM INF inst-4 (server4): connected to all peers
6:53PM INF inst-2 (server2): connected to all peers
6:53PM INF inst-1 (server1): connected to all peers
6:53PM INF inst-3 (server3): connected to all peers
```

Verify the status of tests executed on each node using **nettest run audit workloads**.

Example

```
$ nettest run audit workloads
15:36PM INF Cluster: cluster-gray
5:36PM INF 4 instances discovered
5:36PM INF 0 clients discovered
5:36PM INF Workloads audit:
5:36PM INF inst-2 (server2): {"completed":201996,"errors":0,"running":1}
5:36PM INF inst-3 (server3): {"completed":279976,"errors":0,"running":1}
5:36PM INF inst-1 (server1): {"completed":238594,"errors":0,"running":1}
5:36PM INF inst-4 (server4): {"completed":227704,"errors":0,"running":1}
```

D.5 Add Client Nodes to the nettest Configuration

To add client nodes to the nettest configuration using the nettest-agent (daemon), complete the following steps:

Procedure

1. Ensure that you meet the prerequisites for running **nettest**, as listed in [Prerequisites to Execute nettest](#) (Appendix D.1 on page 258).
2. Execute **nettest-agent** on one or more client nodes:

```
nettest-agent
```

3. On a DDN Infinia client or server node, regenerate the nettest config and testcase files:

```
nettest config generate && nettest test generate
```

Example

The following is an example output:

```
10:57PM INF Configuration for cluster cluster-gray written
10:57PM INF 41 tests, and 8 PRD tests generated.
10:57PM INF Tests for cluster cluster-gray written
```

4. Show the updated nettest configuration and note the additional entry of type **nettest** in the output.

```
nettest config show
```

Example

The following is example output:

| CLUSTER | UUID | GROUP |
|--------------|--------------------------------------|-------|
| cluster-gray | 158500a6-c063-4258-84c5-c0a7fa6d24be | |

| HOST | ID | TYPE | UUID | TLS | IPS | PORT |
|---------|----|----------|--------------------------------------|-------|--|------|
| server1 | 1 | instance | 087d11c7-47ba-44c0-8c95-d891a7ecacb1 | false | 10.138.0.4 | 3181 |
| user1 | 1 | nettest | 441bcb95-b52b-4399-b080-d33219067b0a | false | 10.138.0.2 172.17.0.1 172.19.0.1 | 4017 |
| server2 | 2 | instance | cd44611d-8574-497e-a3de-5726eef5e55d | false | 10.138.0.6 | 3143 |
| server3 | 3 | instance | f8ad1435-59ea-465e-89cb-79bbc76c334e | false | 10.138.0.8 | 3181 |
| server4 | 4 | instance | 84527239-c018-498b-b3a5-94b93f40266a | false | 10.138.0.31 | 3181 |

5. Execute tests that include client IO.

Example

The following example runs nettest with test case number 31.

```
$ nettest run exec -f 31
11:00PM INF Cluster Configuration: cluster-gray_config.json, Size: 3152, Modified: 2023-02-14
22:57:31.477223122 +0000 UTC
11:00PM INF Cluster: cluster-gray
11:00PM INF 4 instances discovered
11:00PM INF 1 clients discovered
11:00PM INF Tests Configuration: cluster-gray_test.json, Size: 20677, Modified: 2023-02-14
22:57:31.573232083 +0000 UTC
11:00PM INF Running 1 tests.
Test 1/1: all clients -> all servers (serial) single-threaded 1m RPC (rdma) 100% | (10/10, 60 it/min)
Test 1/1: all clients -> all servers (serial) single-threaded 1m RPC (rdma) 100% | (10/10, 60 it/min)
Test 1/1: all clients -> all servers (serial) single-threaded 1m RPC (rdma) 100% | (10/10, 60 it/min)
Test 1/1: all clients -> all servers (serial) single-threaded 1m RPC (rdma) 100% | (10/10, 60 it/min)
```

| ENDPOINT | HOST | IOPS TOTAL | IOPS RANGE [min/avg/max] | B/W (MB/s) | B/W RANGE [min/avg/max] | LATENCY (µs) [min/avg/max] |
|-----------|-------|---------------|-----------------------------|---------------|----------------------------|-------------------------------|
| 1_nettest | user1 | 2677 | [524/669/869] | 2807 | [549/701/911] | [1150/1544/1907] |

Appendix E Using the DDN Infinia User Interface

As described in [Manage and Monitor DDN Infinia from the Dashboard](#) (Section 2.2.2.3 on page 54), the DDN Infinia UI provides the capability to perform various functions for storage management and monitoring from the **Dashboard** page. Note that the menu commands, tabs, and widgets that appear on the UI varies as per the type of user and the scopes and capabilities assigned to the user:

- For an example of a realm admin **Dashboard** page, see [Figure 8](#) in Section 2.2.2.3 on page 54.
- For an example of a tenant admin **Dashboard** page, see [Figure 10](#) in Section 2.2.2.3 on page 54.

This appendix provides detailed instructions for some commonly performed administration tasks:

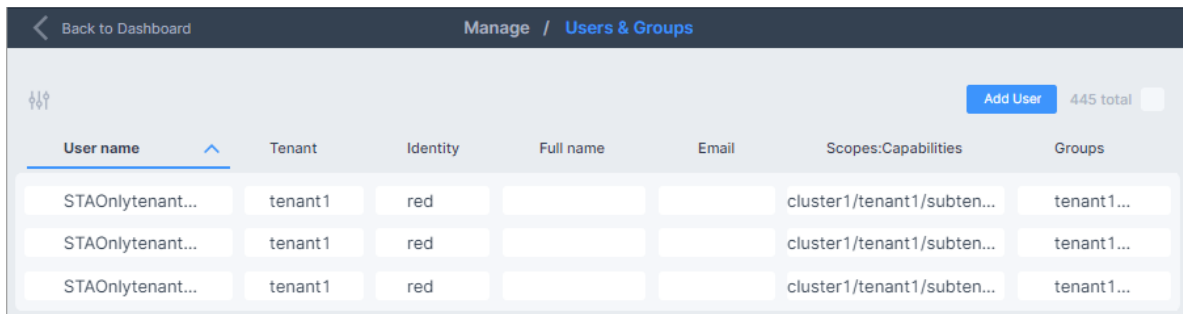
- [Add an User](#) (Appendix E.1 on page 267)
- [Delete an User](#) (Appendix E.2 on page 269)
- [Add a Group](#) (Appendix E.3 on page 270)
- [Add an Existing User to a Group](#) (Appendix E.4 on page 271)
- [Add a New User to a Group](#) (Appendix E.5 on page 271)
- [Delete a Group](#) (Appendix E.6 on page 272)
- [Add a Tenant](#) (Appendix E.7 on page 273)
- [Add a Sub Tenant](#) (Appendix E.8 on page 274)
- [Add an Object Data Service](#) (Appendix E.9 on page 276)
- [Add a Block Data Service](#) (Appendix E.10 on page 277)
- [Enable Data Access](#) (Appendix E.11 on page 279)
- [Add a Bucket](#) (Appendix E.12 on page 281)
- [View Events](#) (Appendix E.13 on page 282)
- [Manage Servers](#) (Appendix E.14 on page 283)
- [Change the Physical Details of a Server](#) (Appendix E.15 on page 283)
- [Perform Various Operations On the Server](#) (Appendix E.16 on page 284)
- [View Tasks](#) (Appendix E.17 on page 285)

For definitions of commonly used terminology in DDN Infinia, including realm administrator ([realm admin](#)) and tenant administrator ([tenant admin](#)), see [Appendix F](#) on page 286.

E.1 Add an User

As a realm administrator or a tenant administrator, add an user by using the following steps:

1. On the **Manage** menu, click **Users & Groups**.
2. On the **Manage Users & Groups** page, click **Add User**.

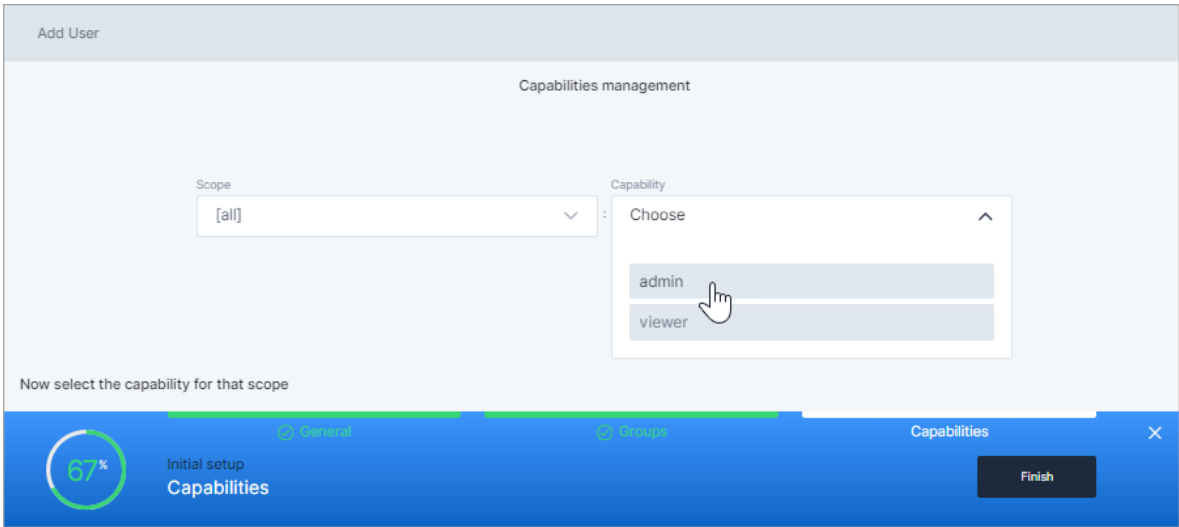


3. In the **Add User** wizard, enter the following details in the respective tabs. Click **Next** to move to next tab.

Note that mandatory field is indicated by a red asterisk.

| Tab | Field | Description |
|---------|-------------------|---|
| General | Tenants | For adding a realm user, skip this field. For adding a tenant user, enter the required tenant name. |
| | Name* | Enter a unique username for the new user |
| | Full Name | Enter the full user name |
| | Email | Enter the user email ID |
| | Password* | Copy the system generated new-user-password to be used for first login |
| | Identity Provider | If required, select the identity provider that will be used for the user authentication |

| Tab | Field | Description |
|--------|-------|---|
| Groups | | If required, add the new user to a group. To add the new user to a group, select the check box for the required group and then click Add the user to the selected groups . For UI screenshot, see Groups tab information in Add a New User to a Group , step 4. |



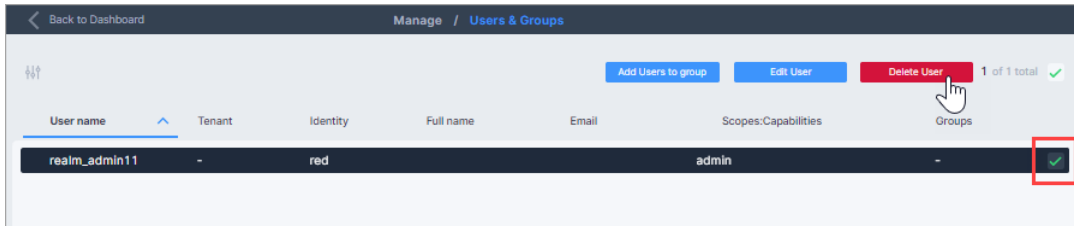
| Tab | Field | Description |
|--------------|------------|---|
| Capabilities | Scope | Assign the scope for the user by selecting from the following options: <ul style="list-style-type: none">• [realm] : Realm access scope• [all] : All tenants access scope• [all]/[all] : All tenants and all sub tenants access scope• Any tenant or tenants available |
| | Capability | Assign Admin or Viewer capability to the user |

4. Click **Finish**.

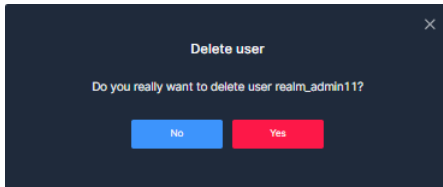
E.2 Delete an User

As a realm administrator or a tenant administrator, if required, delete an existing user by using the following steps:

1. On the **Manage** menu, click **Users & Groups**.
2. On the **Manage Users & Groups** page, select the check box for the user to be deleted and then click **Delete User**.



3. In the **Delete user** confirmation dialog box, click **Yes**.

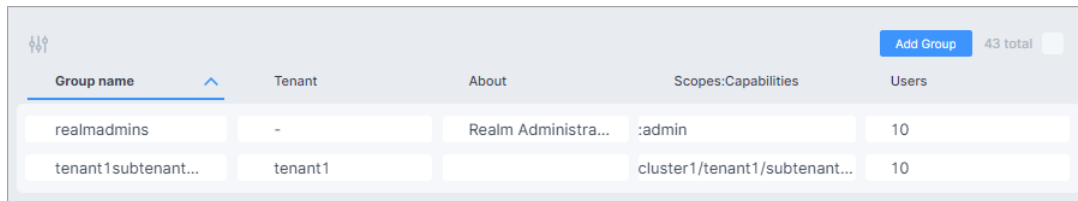


E.3 Add a Group

A group is an entity to collectively define and apply scope and capability (**scope:capability**) to users. You need to first add a group in DDN Infinia and then add users to the group (see [Add an Existing User to a Group](#)) to apply scopes and capabilities of the group to the users.

As a realm administrator or a tenant administrator, add a group using the following steps:

1. On the **Manage** menu, click **Users & Groups**.
2. On the **Manage Users & Groups** page, click **Add Group**.



3. In the **Add Group** dialog box, enter the following details.

Note that mandatory field is indicated by a red asterisk.

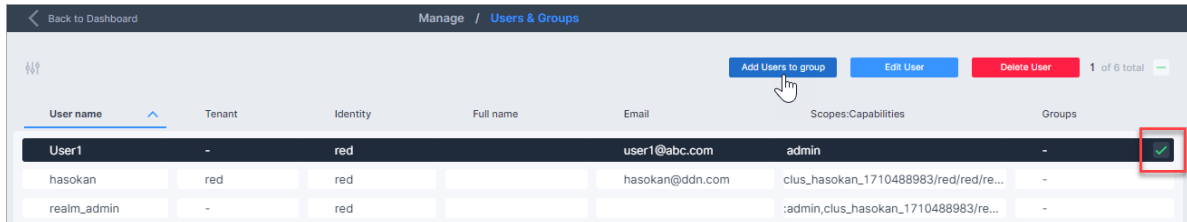
| Field | Description |
|-------------|---|
| Tenants | If applicable, enter the tenant name to which the group should belong |
| Name* | Enter a name for the group |
| About | Enter a short description about the group |
| Scope* | Assign scope for the group from the available options |
| Capability* | Assign Admin or Viewer capability to the group |

4. Click **Add**.

E.4 Add an Existing User to a Group

As a realm administrator or a tenant administrator, add an existing user to an existing group using the following steps:

1. On the **Manage** menu, click **Users & Groups**.
2. On the **Manage Users & Groups** page, select the check box next to the user to be added to a group.

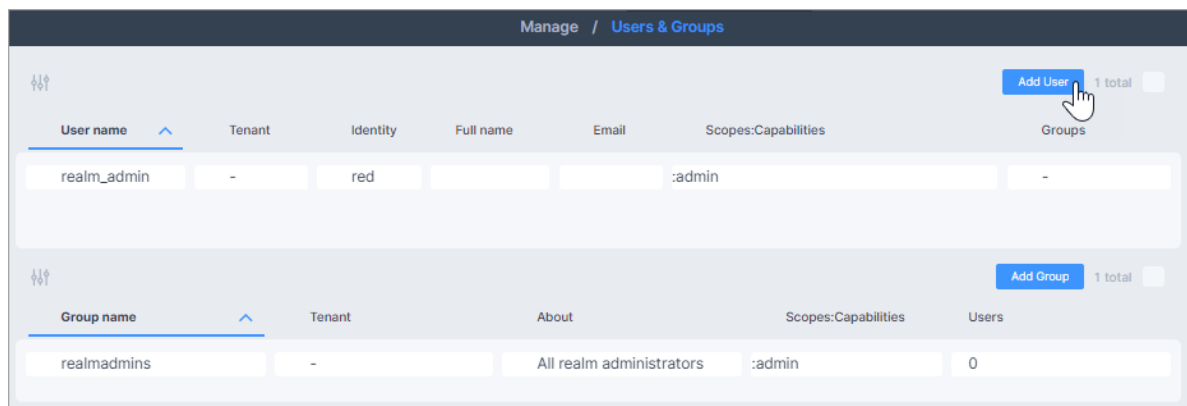


3. Click **Add Users to group**.
4. In the **Add Users To groups** dialog box, select the group and then click **Add**.

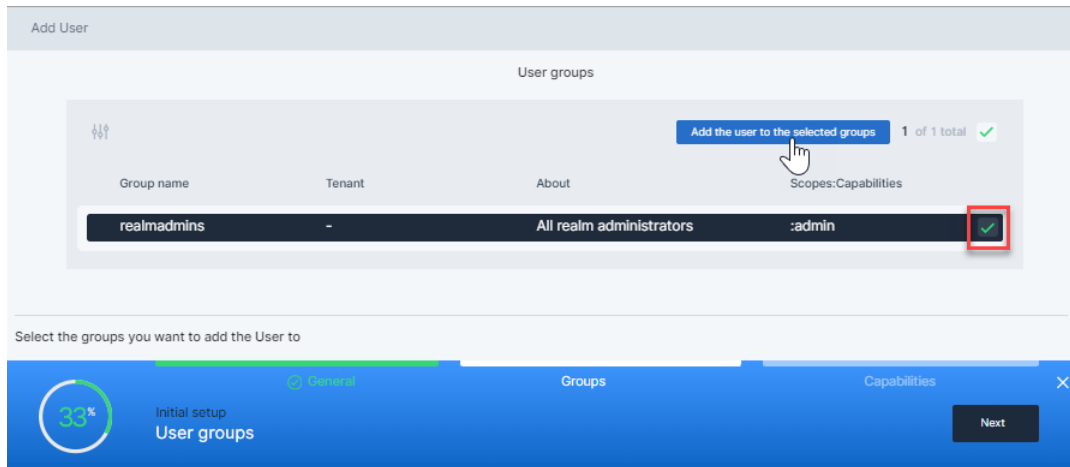
E.5 Add a New User to a Group

As a realm administrator or a tenant administrator, add a new user to an existing group using the following steps:

1. On the **Manage** menu, click **Users & Groups**.
2. On the **Manage Users & Groups** page, click **Add User**.



3. In the **Add User** wizard, enter the details in the **General** and **Capabilities** tabs as shown in [Add an User](#), step 3.
4. In the **Groups** tab, to add new user to a group, select the check box for the required group and then click **Add the user to the selected groups** as shown in the following image.



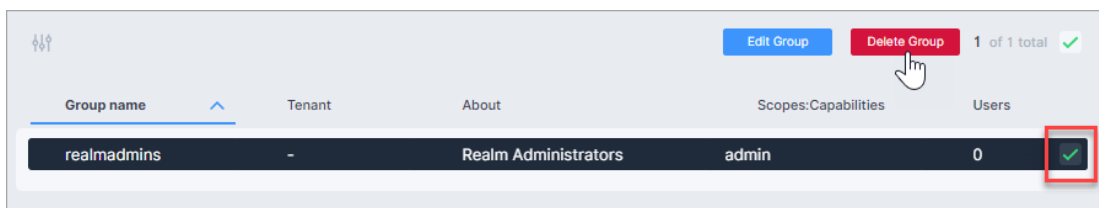
5. After completion, click **Finish**.

E.6 Delete a Group

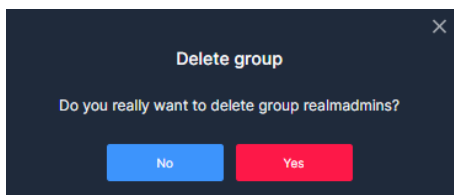
NOTE: You cannot delete a group that has users assigned to it.

As a realm administrator or a tenant administrator, delete a group using the following steps:

1. On the **Manage** menu, click **Users & Groups**.
2. On the **Manage Users & Groups** page, select the group that you want to delete and then click **Delete Group**.



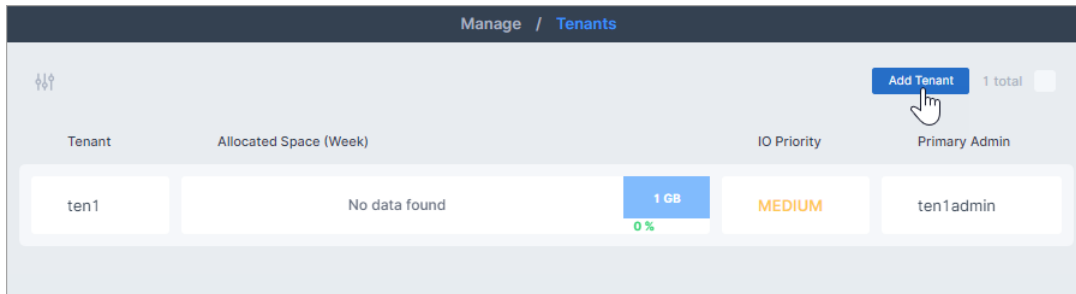
3. In the **Delete group** confirmation dialog box, click **Yes**.



E.7 Add a Tenant

As a realm administrator, add a tenant using the following steps:

1. On the **Manage** menu, click **Tenants**.
2. On the **Manage Tenants** page, click **Add Tenant**.



3. In the **Add Tenant** dialog box enter the following details:

Note that mandatory field is indicated by a red asterisk.

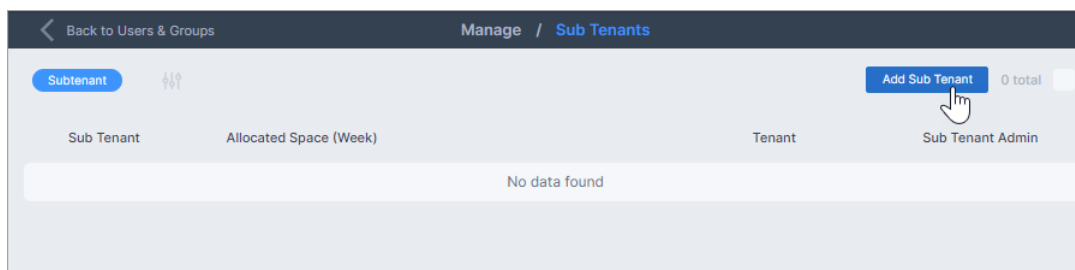
| Field | Description |
|-----------------------|--|
| Name* | Enter name for new tenant |
| Primary admin* | Choose an existing realm user as a tenant primary admin or Enter a name for new tenant primary admin and then press Enter . <ul style="list-style-type: none"> In the Add User Wizard that appears, copy the system generated new-user-password to be used for first login. Click Finish. |
| IO Priority | Set the IO relative priority of tenants from the following options: <ul style="list-style-type: none"> low medium high |
| Admins | Enter the realm users and groups that should have administrator capabilities for the tenant |
| Viewers | Enter the realm users and groups that should have viewer capabilities for the tenant |
| tenantname_Allocation | Enter the pool space to be allocated for the tenant |

4. Click **Add**.

E.8 Add a Sub Tenant

As a tenant administrator, add a sub tenant using the following steps:

1. On the **Manage** menu, click **Sub Tenants**.
2. On the **Manage Sub Tenants** page, click **Add Sub Tenant**.



3. In the **Add Sub Tenant** dialog box enter the following details:

The screenshot shows the 'Add Sub Tenant' dialog box with the following details filled in:

- Tenant ***: ten2
- Name ***: ten2subten1
- Primary admin ***: ten2subten1_admin
- IO Priority**: medium
- Admins**: Enter or select an Admin
- Viewers**: Enter or select an Viewer
- ten2subten1 Allocation**: 100 GB
- ten2subten1 Over provisioning ratio**: normal (x 0.2)
- ten2subten1 Space Preview**: 100 GB

Note that mandatory field is indicated by a red asterisk.

| Field | Description |
|-----------------|---|
| Tenant* | Enter the tenant name |
| Name* | Enter the sub tenant name |
| Primary admin* | Choose an existing tenant user as a tenant primary admin or Enter a name for new sub tenant primary admin and then press Enter . <ul style="list-style-type: none"> In the Add User Wizard that appears, copy the system generated primary-admin-password to be used for first login. Click Finish. |
| IO Priority | Set the IO relative priority of sub tenants from the following options: <ul style="list-style-type: none"> low medium high |
| Admins | Enter the users and groups that should have administrator capabilities for the sub tenant |
| Viewers | Enter the users and groups that should have viewer capabilities for the sub tenant |
| Pool Allocation | Enter the space to be allocated for the sub tenant |

4. Click **Add**.

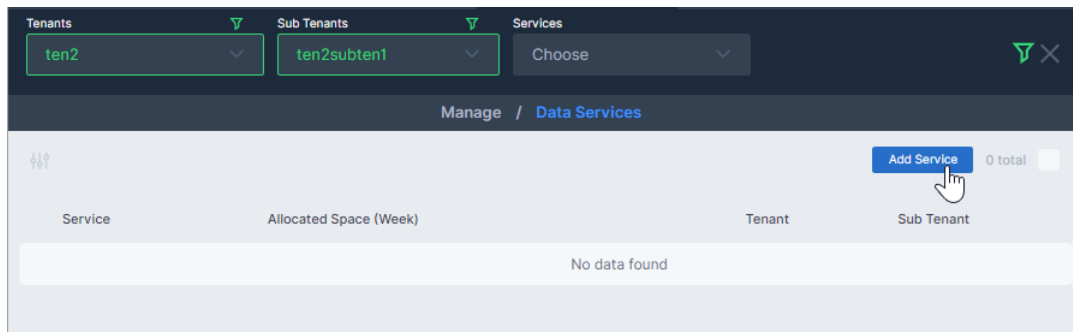
Note that after you add a sub tenant the respective tenant admin is by-default assigned the administration capability for the sub tenant.

E.9 Add an Object Data Service

As a subtenant admin, add an object data service using the following steps:

NOTE: Before adding a object data service, you need to add the vhost information in the etchost. For more information, see [Define Virtual Hosts on Nodes](#) (Section 2.4.1 on page 60).

1. On the **Manage** menu, click **Data Services**.
2. On the **Manage Data Services** page, select the required tenant and sub tenant using the **Tenants** and **Sub Tenants** filters.
3. Click **Add Service**.



4. In the **Add Data Service** dialog box enter the following details:

Note that mandatory field is indicated by a red asterisk.

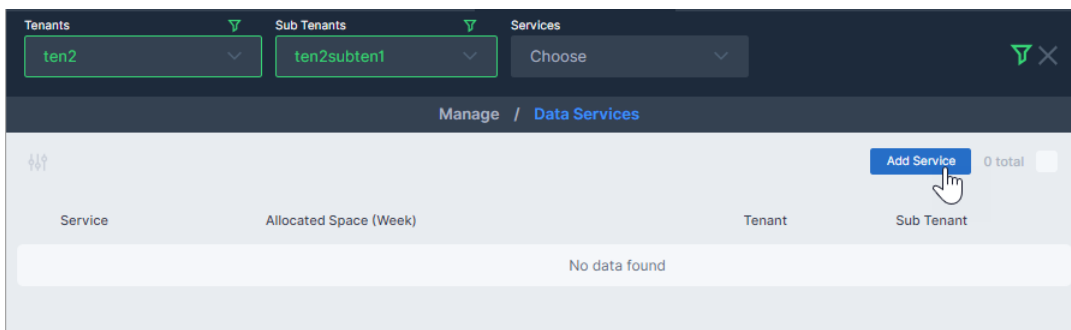
| Field | | Description |
|-------------------------|---------|---|
| Sub Tenant* | | Enter the required sub tenant name |
| Service Type | | Select Object service. |
| Service Name* | | Enter a suitable name for the service |
| Protocol | | Enter the applicable protocol |
| Vhost* | | Enter virtual host name |
| Capabilities Management | Admins | Enter the users and groups that should have administrator capabilities for the sub tenant |
| | Viewers | Enter the users and groups that should have viewer capabilities for the sub tenant |
| Data Access | | Assign a user to enable the data service |

- Click **Add**.

E.10 Add a Block Data Service

As a subtenant admin, add a block data service using the following steps:

- On the **Manage** menu, click **Data Services**.
- On the **Manage Data Services** page, select the required tenant and sub tenant using the **Tenants** and **Sub Tenants** filters.
- Click **Add Service**.



- In the **Add Data Service** dialog box enter the following details:

Add Data Service

Sub Tenant *

ten1sub1

Service Name *

t1s1block

Service Type *

Block

Protocol *

CSI

Network

Enter a Network

DataSet *

New

Name *

t1s1ds1

Capabilities Management

Admins

ten1sub1_admin

ten1sub1_ad...

Viewers

Choose

Service Accounts

Enter or select a Service Account

Service Allocation

1

GB

Service Over provisioning ratio is normal (x 1.0)

Service Space Preview

1 GB

Cancel

Add

Note that mandatory field is indicated by a red asterisk.

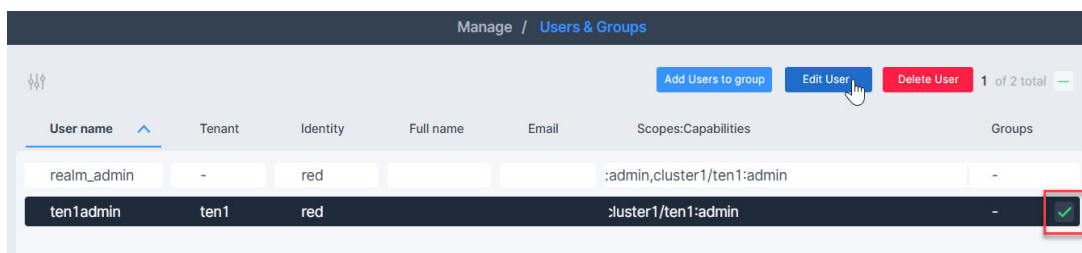
| Field | | Description |
|-------------------------|------------------|---|
| Sub Tenant* | | Enter the required sub tenant name |
| Service Type* | | Select Block service. |
| Service Name* | | Enter a suitable name for the service |
| Protocol* | | Enter the applicable protocol |
| Network | | If required, enter network details |
| DataSet* | | Select an existing dataset or select <i>New</i> for a new dataset |
| Name* | | Enter the name for New dataset. |
| Capabilities Management | Admins | Enter the users and groups that should have administrator capabilities for the sub tenant |
| | Viewers | Enter the users and groups that should have viewer capabilities for the sub tenant |
| | Service Accounts | If required, select a service account user |
| Servicename_Allocation | | Enter the pool space to be allocated for the service |

- Click **Add**.

E.11 Enable Data Access

To use a data service, the user assigned with the data access enabling capability must enable the data access. To enable the data access, as a sub tenant admin, complete the following steps:

- On the **Manage** menu, click **Users & Groups**.
- On the **Manage Users & Groups** page, select the check box for the user assigned with the data access enabling capability and then click **Edit User**.



3. In the **Modify User** *ten1admin* dialog box, on the **Object Data Access** tab, set the toggle switch to **Enabled** state.

Modify User ten1admin

Object Data Access

Enabled

Generate S3 Key and Secret

S3-KEY * S3-SECRET * S3-EXPIRATION

1 Years

Click on the icon in order to generate pair of S3 Key and Secret

100% Initial setup Object Data Access Finish

4. Choose the S3 expiration period and then click *Generate S3 Key and Secret* icon.
5. Copy the S3 Key and S3 Secret code to be used to run I/O from client login.

S3-KEY * S3-SECRET * S3-EXPIRATION

S91CS3VN4KEWFY27TCM5

.....

Show password

1 Years

Click to add another pair

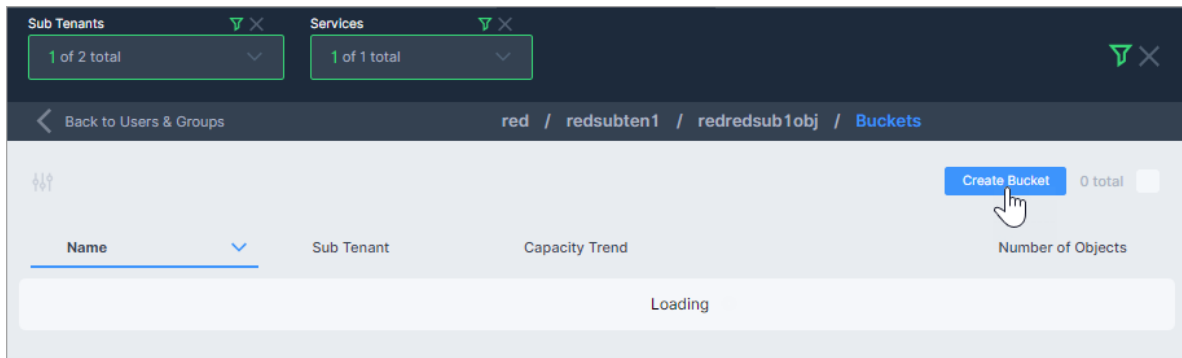
6. Click **Finish**.

E.12 Add a Bucket

As a sub tenant admin with admin capability on a data service, add a bucket using the following steps:

NOTE: To add a bucket, the data access must be enabled. For more details, see [Enable Data Access](#) (Appendix E.11 on page 279).

1. On the **Manage** menu, click **Buckets**.
2. On the **Manage Buckets** page, select the required subtenant and data services using the **Sub Tenants** and **Services** filters.
3. Click **Create Bucket**.



4. In the **Create Bucket** dialog box enter the following details:

Note that mandatory field is indicated by a red asterisk.

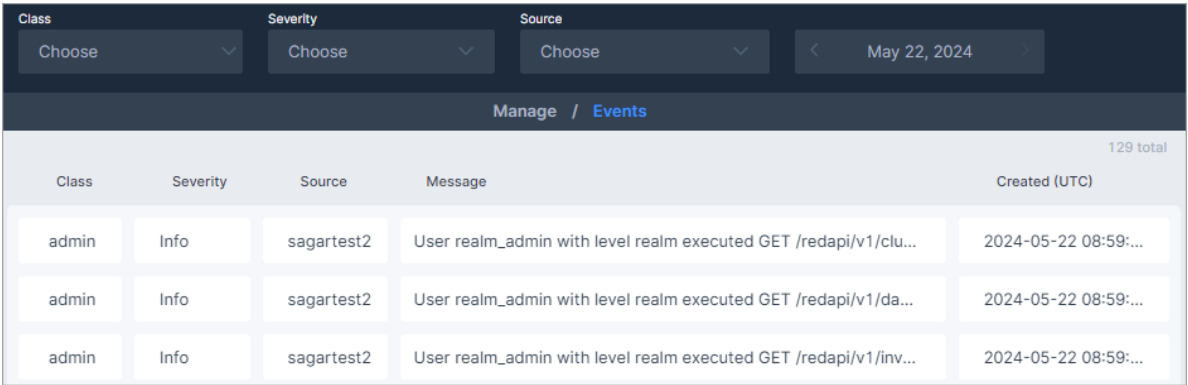
| Field | Description |
|-------------------|--|
| Bucket Name* | Enter a suitable name for the bucket |
| Bucket Allocation | Enter the storage space to be allocated for the bucket |

5. Click **Create**.

E.13 View Events

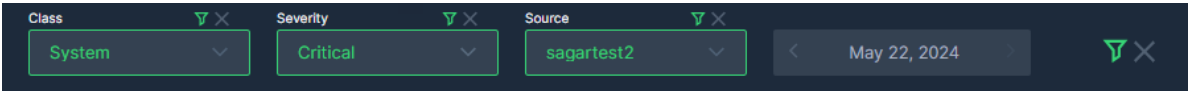
As a realm admin, view various events in DDN Infinia as follows:

1. On the **Manage** menu, click **Events**.



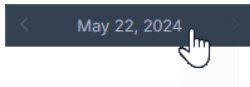
| Class | Severity | Source | Message | Created (UTC) |
|-------|----------|------------|--|----------------------|
| admin | Info | sagartest2 | User realm_admin with level realm executed GET /redapi/v1/clu... | 2024-05-22 08:59:... |
| admin | Info | sagartest2 | User realm_admin with level realm executed GET /redapi/v1/da... | 2024-05-22 08:59:... |
| admin | Info | sagartest2 | User realm_admin with level realm executed GET /redapi/v1/inv... | 2024-05-22 08:59:... |

2. On the **Manage Events** page, select the required filters to sort the events from the event list.

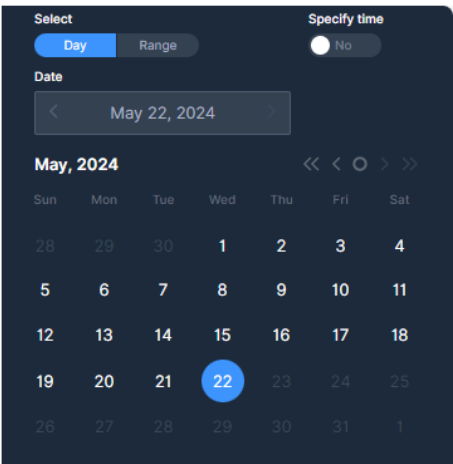


Class: System, Severity: Critical, Source: sagartest2, Date: May 22, 2024

3. To sort the events based on a specific day, range of dates, or a specific time, click the date filter and then select the required options.



< May 22, 2024



Select: Day, Range, Specify time: No

Date: May 22, 2024

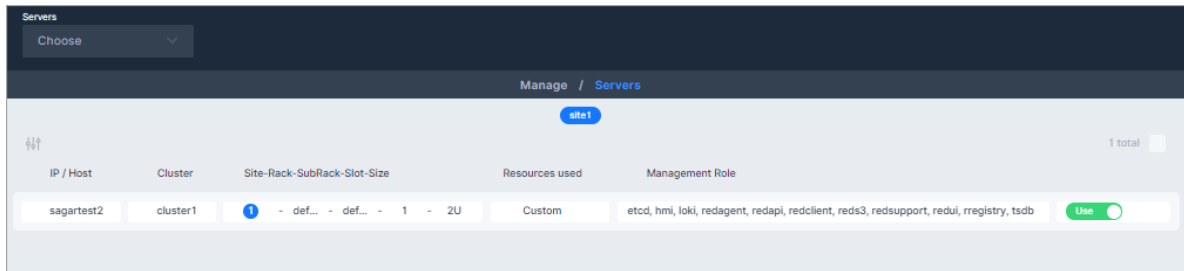
May, 2024

| | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|
| Sun | Mon | Tue | Wed | Thu | Fri | Sat |
| 28 | 29 | 30 | 1 | 2 | 3 | 4 |
| 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 12 | 13 | 14 | 15 | 16 | 17 | 18 |
| 19 | 20 | 21 | 22 | 23 | 24 | 25 |
| 26 | 27 | 28 | 29 | 30 | 31 | 1 |

E.14 Manage Servers

As a realm admin, view the details of servers using the following steps:

1. On the **Manage** menu, click **Servers**.

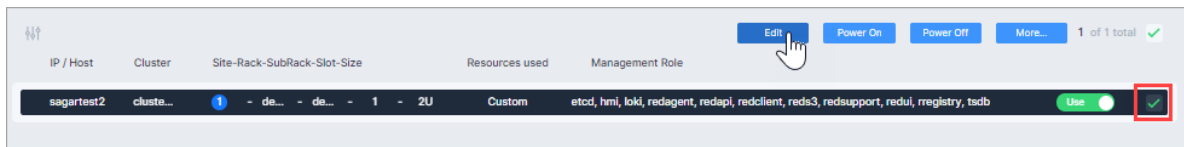


2. On the **Manage Servers** page, view the list of servers in DDN Infinia.
3. To sort the server list and select a required server, use the **Servers** filter.

E.15 Change the Physical Details of a Server

As a realm admin, change the physical details of a server using the following steps:

1. On the **Manage** menu, click **Servers**.
2. On the **Manage Servers** page, select the check box for the server that need the change in physical details and then click **Edit**.



3. In the **Modify_servername** dialog box, change the required details from the following items.

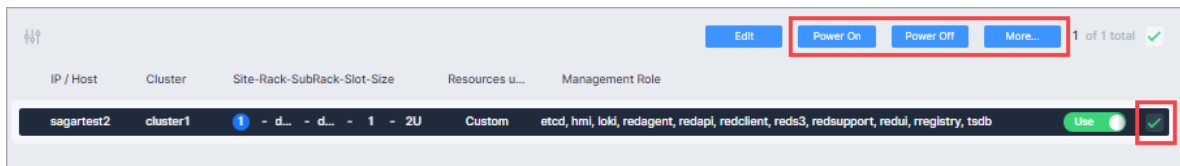
| Field | Description |
|-----------------|--------------------------------|
| Rack | Enter the new rack location |
| SubRack Section | Enter the new subrack location |
| Slot | Enter the new slot number |
| Size | Enter the new rack unit size |

4. Click **Modify**.

E.16 Perform Various Operations On the Server

As a realm admin, remotely manage Power On, Power Off, Restart, or Reset operations on the server using the following steps:

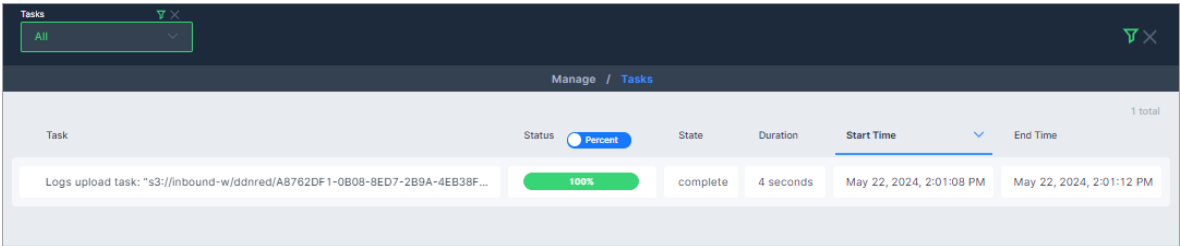
1. On the **Manage** menu, click **Servers**.
2. On the **Manage Servers** page, select the check box for the server on which you want to perform an operation (Power On, Power Off, Restart, or Reset) and then click the button for the operation.



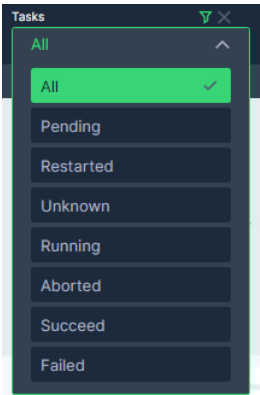
E.17 View Tasks


As a realm admin, view status of various tasks in DDN Infinia as follows:

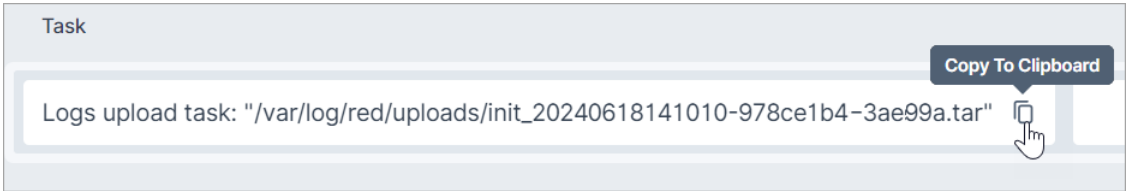
- 1. On the **Manage** menu, click **Tasks**.



- 2. On the **Manage Tasks** page, use the **Tasks** filter to sort the tasks in the task list to view the required task.



Note that for the logs upload task, the UI provides path of the uploaded logs file. If you need the path, hover over the upload task and then click **Copy To Clipboard** icon  .



Appendix F Terminology

The following define commonly used terminology in DDN Infinia.

B

beptree

B-epsilon tree

bucket

A *bucket* is a container for objects stored in S3.

C

capability

Capability defines the permissions assigned for a particular scope, for example, admin or viewer.

cluster

A *cluster* is a collection of DDN Infinia instances grouped mostly for resiliency management of data store and other services (infrastructure and data services).

Core Affine storage Targets (CATs)

CAT is a single/multiple storage devices, affine to single/multiple cores. Each CAT contains an Intent log (OPREC journal), copy-on-write B-epsilon tree, and Bulk data.

Container Storage Interface (CSI)

CSI is an abstraction of various container-orchestration volume interfaces. It can be used with DDN Infinia volumes.

D

Dataset

The *dataset* is the unit of virtual storage management. It defines the *dataservices* that are provided, the security and resilience under which it operates, and the SLAs that determine performance and capacity.

dataservices

Data access is provided using *dataservices*, which is software component that shares and/or processes the data and metadata and makes it available to users over the network.

I

identity provider

Identity provider (also known as *identity backend*) is the component responsible for authenticating users. It can be local, where user information is stored entirely in the DDN Infinia environment, or external, for example a tenant's Active Directory or LDAP system. Identity providers can be assigned to the realm, as well as to the tenants level.

Instance

An *Instance* is a logical term that refers to the collection of services (CATs, DataServices, infrastructure services), running on a single node.

N

node

A *node* is an OS instance running either on a single physical server (on-premise) or an OS instance running on a Virtual Machine in the Cloud.

P

physical server

A *physical server* is a physical machine (on premise) or Virtual Machine (in the Cloud) running one supported OS Instance. The server will have various resources (disks, CPU Cores, Memory and network adapters) that can be used by DDN Infinia software.

physical site

A *physical site* usually defines a geographical location (such as “New York”, “Los Angeles”, “Tokyo” etc.); but might also be used to define a finer granularity, in case of large locations. A physical site can be used to introduce a sync or async replication for backup and/or disaster recovery use cases or to define where to send Support personnel to fix broken hardware.

R

realm

A *realm* is a set of connected, mutually-aware nodes, in the same geographic location, which can be used to create a DDN Infinia cluster.

realm admin

A *realm administrator* (realm admin) is a user that is defined at the realm domain and has admin capabilities on the realm. A realm admin manages tenants and hardware resources, which include acquiring servers, disks, and network infrastructure; allocating them to tenants; and monitoring hardware health, as well as using resources to fix and adding more resource when required.

realm configuration

Realm configuration specifies the collection of containers (and their resource limits) that are managed by DDN Infinia.

realm-entry

The *realm-entry* is the first Instance installed in a DDN Infinia Realm. All other Instances obtain certain configuration from the realm-entry node.

S

scope

Scope (User Model) defines the breadth of control assigned to a user or a group. It can be a combination of realm, one or more tenants, one or more subtenants, and one or more data services. When assigning a user with a scope of tenant, subtenant, or dataservice, it is possible to specify [a11] or a list of entries to define the extent of the management scope.

site

Site is the physical location of a device or node.

subtenant

A *subtenant* is a subset of a tenant. It is used with customers that require more than one level of service management. A subtenant is managed by a subtenant administrator (subtenant admin).

subtenant admin

A *subtenant admin* is a user that is defined at the realm domain or the tenant domain and has admin capabilities on a subtenant. A subtenant admin manages services, which include defining data services (which implies datasets) based on resources received from a tenant admin and monitoring those services for health and capacity to request more resources, if required.

T

tenant

The *tenant/subtenant* is a DDN Infinia cluster abstraction, which allows multiple organizations to share single physical DDN Infinia storage securely and efficiently. The inventory and configuration objects of a cluster define physical resources that provide storage (servers, network, ssd) and tenants/subtenants/datasets (as well as volume/objects in a dataset) allocate that storage to consumers/clients. A tenant is managed by a tenant administrator (tenant admin).

tenant admin

A *tenant admin* is a user that is defined at the realm domain or the tenant domain and has admin capabilities on a tenant. A tenant admin manages subtenants, which include requesting resources from a realm admin, allocating them to sub tenants, and monitoring resource usage to request additional resources from a realm admin if required.

U

user

User is a single user instance with a universally unique identifier in DDN Infinia. A user must authenticate with one of the identity backends before any access to the DDN Infinia system is granted.

users group

A *users group* is a collection of users that are assigned a common set of scope and capability definitions.

V

volume

See CSI definition.

Appendix G redsupport

redsupport is a secured web-based program for shared terminal access by authorized DDN support personnel.

To use **redsupport**, DDN Infinia must be connected to the Internet with access to both **redsupport.ddn.com** and **redsupport-terminal.ddn.com**. If remote access is required to further investigate a problem on your DDN Infinia, initiate a support session via challenge/response.

NOTE: The support session is created one node at a time. If access is required to multiple nodes in a cluster, for each node, start its own session.

All communication to the central server is secured with SSL/TLS. Login to the central server is limited to a small group of DDN engineers and requires two-factor authentication. The alternative login method requires a 32-byte random token, accessible only to realm admin users.

To start the support session,

1. On the command prompt of the node requiring remote support, enter:

```
source /opt/ddn/red/red_aliases
```

2. Run **redsupport-start**

On success, the last line of output provides the URL for the browser session for the DDN support engineer. The session will by default last for 4 hours. If required, adjust the session lifetime as described in a following section.

At most only one support session should be run per node. The browser UI allows multiple terminal windows to the same node within a single support session.

To view token for browser Login,

1. Run **redsupport-token**
2. Copy and paste the token on the browser login page.

To reset token for browser Login,

- Run **redsupport-newtoken**

This action invalidates the old token, generates and shows new token. DDN recommends this command with redirection to `> /dev/null` if you find anything insecure about the terminal in which the command is running.

To view the support session status and details,

- Run **redsupport-status**

If no support session is running, you will get a notification about no container. If a session is running, you will see the details such as the session ID used in the URL.

To stop the support session,

- Run **redsupport-stop**

This command stops the support session regardless of remaining session lifetime. All browser logins will be terminated. If the session lifetime is expired, this command is useful to perform the cleanup of resources.

To adjust the lifetime of support session,

- Run `redsupport-lifetime <HOURS>`

where **HOURS** (a mandatory argument) is number of hours to be set as the lifetime of the remaining session, for example 2.75 or 3.

Note that this command does not extend the lifetime of the session by the given amount, rather it replaces the value for remaining lifetime. For example, if 45 minutes are remaining and you set the lifetime to 30 minutes, the session will expire in 30 minutes.

redsupport Limitations

Following are the **redsupport** limitations and the suggested workarounds:

- When using Chrome browser on macOS, on the login page the Login button for customer login is missing or non-functional.
Use Firefox browser or another browser on macOS.
- When using Safari browser on macOS, text pasting functionality does not work in the terminal window.
Use Firefox browser or another browser on macOS.

- In the terminal window, by default certain programs such as **systemctl**, **redsetup**, and **docker**—which depend on **dbus**—fail to run, because the default container environment is not a process descendant of **init** command.

To overcome this limitation, type `/host/bin/ssh <USER>@localhost` in the terminal window. Here **<USER>** is a valid username on the node for which you know the password. This action launches a new shell outside the container that has the required process ancestry.

- In the terminal window, by default some commands do not work as expected when interacting with system files in `/etc` and other standard locations. This is because the full filesystem of the node is mapped into `/host` in the **redsupport** container, therefore the paths are different.

To overcome this limitation, type `/host/bin/ssh <USER>@localhost` in the terminal window. Here **<USER>** is a valid username on the node for which you know the password. This action launches a new shell outside the container that has the full filesystem rooted at `/` as all system utilities expect.

- The terminal window is of fixed size which does not allow monitoring of all required processes using commands such as **top**.

To overcome this limitation, create a custom script that filters the output to what is necessary and run the script with the **watch** utility. For example, the following sequence of commands create and run a script that shows only the processes with names containing the character-string **red**.

```
i. echo -e "#\!/bin/bash\nuptime\nps -o pid,user,pcpu,pmem,args |grep red" >topr.sh
ii. chmod +x topr.sh
iii. watch -n 1 ./topr.sh
```

Appendix H redsetup

The **redsetup** executable performs all bootstrapping required to make a storage node ready for use in a DDN Infinia cluster, as described in [Run Setup](#) (Section 2.1.3 on page 36). This appendix lists available **redsetup** options.

Usage

redsetup [options]

Options:

--admin-password PASSWORD

Define password for the auto-created realm administrator.

--admin-user USER

Define user ID for the auto-created realm administrator. Default is **realm_admin**.

--api-server-port PORT

REST API Server listening port. Default is 8080.

--bluefield-gateway STRING

Gateway that will be assigned to the network ports of the bluefield adapter.

--bluefield-ips VALUE

Comma-separated list of IP addresses that will be assigned to the network ports of the bluefield adapter. If no IP addresses are provided, DHCP will be used by default.

--ctrl-plane-ip IP

IP address that will be used to communicate to other realm nodes. This option is required to bootstrap realm entry node.

--d

Print debug messages to the console and log **redsetup** requests, implies **verbose**.

--dont-save-config

Do not update configuration file on start.

--etcd-partition KEY

etcd partition key.

--etcd-unsecured

etcd service is unsecured.

--external-etcd LIST

Comma-separated list of external etcd node addresses. Empty by default.

--jrpc-unsecured

JRPC calls are unsecured.

--keep-logs

Do not delete any related logs folders.

--management-ip LIST

Comma-separated list of IP addresses that will be used to communicate with the node from outside of the realm. By default, the IP is resolved from `<hostname>`.

--multi-arch

Pull images for all architectures into the realm entry registry, enabling multi-arch clusters.

--ntp-servers LIST

Comma-separated list of NTP Servers. Default is `pool.ntp.org`.

--prepare-only

Prepare node for DDN Infinia, install needed packages and containers, but do not run containers or start `redsetup` service.

--realm-entry

Use the node as a realm deployment entry point.

--realm-entry-address IP

Control plane IP address of a node in the realm to be joined where `redapi` service is running.

--realm-entry-secret SECRET

Shared secret password to join the given realm.

--registries LIST

Comma-separated list of container images registries from which to pull DDN Infinia images. The registries will be used in the order listed. Default is `https://quay.io:443`.

The general form representing a registry is `[http[s]//[login:password@]host:port]`.

The algorithm used is to try a registry, and if the registry cannot be used, try the next, until out of registries.

--release-metadata-file FILE

Path to a file with release-metadata that contains release-version.

--release-version VERSION

Release version of DDN Infinia images to use (realm-entry/single-node bootstrap parameter).

--reset

Stop all services on the node and clean up configuration and configuration's folders.

--reset-with-images

Stop all services on the node, clean up configuration and configuration's folders, and remove DDN Infinia container images.

--rest-server

Execute REST server.

--root-log-path PATH

Root log path. Default is `/var/log/red`.

--skip-bluefield

Skip bluefield adapter configuration.

--skip-hardware-check

Skip hardware minimal requirements checking.

--skip-image-pull

Skip pull of container images onto this node in bootstrap phase. Using this flag may result in unpredictable behavior if latest DDN Infinia images are not pulled onto this node.

--skip-kernel-check

Skip check and update of kernel boot parameters in bootstrap phase. Using this flag may result in unpredictable behavior if kernel boot parameters are not updated as required on this node.

--skip-package-check

Skip check and install of prerequisite packages onto this node in bootstrap phase. Using this flag may result in unpredictable behavior if prerequisite packages are not installed on this node.

--skip-reboot

Skip reboot in bootstrap phase. Using this flag may result in unpredictable behavior if bootstrap phase makes changes to system parameters that require a reboot.

--skip-sol-setup

Skip setting up serial-over-lan connection. Using this flag will take away the capability to remotely access DDN Infinia appliance's console over lan.

--use-asan-image

Use DDN Infinia core container image with the address sanitizer enabled.

--v

Show **redsetup** version and exit.

--verbose

Include additional details in output.

--version

Show **redsetup** version and exit.

Appendix I Manage BIOS Configuration using SUM

NOTE: DDN Infinia 1.1 does not support BIOS configuration on the AI2000. For a list of platforms that support BIOS configuration, contact DDN Support. For improved performance on these platforms, DDN recommends setting **NUMA Nodes Per Socket = NPS4** in the BIOS configuration. This recommend setting is available under `/opt/ddn/red/mounts/` on these nodes.

In addition to the `redcli` commands described in [Manage BIOS Configuration](#) (Section 6.2 on page 146), administrators can manage the BIOS configuration on Supermicro platforms using the [Supermicro Update Manager \(SUM\)](#). This tool provides an option to save the BIOS configuration to a file and to update the BIOS configuration from a file.

This appendix includes an example workflow. Note that all SUM commands in the following workflow are executed in remote OOB context. In other words, they are run from a jump host pointing at one or more remote BMC IPs.

Workflow

1. If possible, save the BIOS configuration from a node that includes the recommended **NUMA Nodes Per Socket = NPS4** setting to a file.

For example, the following command saves the node at BMC IP `192.168.40.115` with credentials `admin/ddn@2024` to the file `SMC-BIOS-NUMA4.cfg`.

```
[SUM_HOME]$ ./sum -i 192.168.40.115 -u admin -p ddn@2024 -c GetCurrentBiosCfg --file ~/SMC-BIOS-NUMA4.cfg
```

Verify that the setting **NUMA Nodes Per Socket = NPS4** is captured in the file by looking for "NPS4" in the `grep` output.

Continuing the previous example, the following command shows this setting in `SMC-BIOS-NUMA4.cfg`.

```
$ grep "NUMA Nodes Per Socket" ~/SMC-BIOS-NUMA4.cfg
<Setting name="NUMA Nodes Per Socket" selectedOption="NPS4" type="Option">
```

If the node's NUMA setting was incorrectly reset after a BIOS upgrade, the setting would be displayed as **"Auto"**, as shown in the following example.

```
[SUM_HOME]$ ./sum -i 192.168.40.115 -u admin -p ddn@2024 -c GetCurrentBiosCfg | grep "NUMA Nodes Per Socket"
<Setting name="NUMA Nodes Per Socket" selectedOption="Auto" type="Option">
```

2. If saving the desired configuration from an existing node is not an option, create a BIOS configuration file with the desired setting values.

For example, the following is a configuration file with the **NUMA Nodes Per Socket = NPS4** setting.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="yes"?>
<BiosCfg>
  <Menu name="Advanced">
    <Menu name="ACPI Settings">
      <Setting name="NUMA Nodes Per Socket" selectedOption="NPS4" type="Option">
        <Information>
          <AvailableOptions>
            <Option value="0">NPS0</Option>
            <Option value="1">NPS1</Option>
            <Option value="2">NPS2</Option>
            <Option value="3">NPS4</Option>
            <Option value="7">Auto</Option>
          </AvailableOptions>
          <DefaultOption>Auto</DefaultOption>
          <Help><![CDATA[Specifies the number of desired NUMA nodes per socket. Zero will
attempt to interleave the two sockets together.]]></Help>
        </Information>
      </Setting>
    </Menu>
  </Menu>
</BiosCfg>
```

3. Update the BIOS configuration using the previously saved file from step 1 or a newly created file with the desired setting values from step 3. Note that a system list file containing BMC IPs per line can be passed to apply it on multiple systems.

For example, the following commands use the system list file **SList.txt**.

In this example, the **--reboot** and **--individually** flags are optional. The **--reboot** flag forces the managed system to reboot or power up after operation; while the **--individually** flag updates each BIOS with the corresponding configuration file individually.


```

$ cat ~/SList.txt
192.168.40.115

$ ./sum -l ~/SList.txt -u admin -p ddn@2024 -c ChangeBiosCfg --file ~/SMC-BIOS-NUMA4only.cfg --reboot
Supermicro Update Manager (for UEFI BIOS) 2.13.0-p8 (2023/12/29) (x86_64)
Copyright(C) 2013-2023 Super Micro Computer, Inc. All rights reserved.
The average upload speed per thread is limited to 1000 mbps on higher bandwidth system,
while the speed is limited to 2 mbps on lower bandwidth system.
Start to do ChangeBiosCfg for systems listed in /home/ddn/SList.txt

Multi system log file created:
/home/ddn/SList.txt.log_2024-09-25_04-51-36_1517941
Press ENTER to see the current execution (Index: 1 ~ 1) status:

-----Final Results-----
Executed Command:
./sum -u ***** -p ***** -l /home/ddn/SList.txt -c ChangeBiosCfg --file
/home/ddn/SMC-BIOS-NUMA4.cfg --reboot
Summary:
1 EXECUTIONS ( WAITING: 0 RUNNING: 0 SUCCESS: 1 FAILED: 0 INCOMPLETE: 0 RETRY: 0 IGNORED: 0 CANCEL: 0 )
Status List (Index: 1 ~ 1, Total: 1):
  Index |      System Name | Elapsed |      Status | Exit Code
    1 | 192.168.40.115 | 00:04:27 |     SUCCESS |         0
Summary:
1 EXECUTIONS ( WAITING: 0 RUNNING: 0 SUCCESS: 1 FAILED: 0 INCOMPLETE: 0 RETRY: 0 IGNORED: 0 CANCEL: 0 )
-----

Please check output message:
/home/ddn/SList.txt.log_2024-09-25_04-51-36_1517941

$ cat ~/SList.txt.log_2024-09-25_04-51-36_1517941
-----Last Update Time-----
2024-09-25_04:56:04
Process finished.
-----Execution parameters-----
Executed Command:
./sum -u ***** -p ***** -l /home/ddn/SList.txt -c ChangeBiosCfg --file
/home/ddn/SMC-BIOS-NUMA4.cfg --reboot
-----Summary-----
1 EXECUTIONS ( WAITING: 0 RUNNING: 0 SUCCESS: 1 FAILED: 0 INCOMPLETE: 0 RETRY: 0 IGNORED: 0 CANCEL: 0 )
-----Status List-----
Index |System Name      |Start Time      |End Time        |Elapsed|Status  |Exit Code
  1    |192.168.40.115  |09-25_11:51:36 |09-25_11:56:03 |00:04:27|SUCCESS |0
-----Execution Message-----
System Name
192.168.40.115
Message
Supermicro Update Manager (for UEFI BIOS) 2.13.0-p8 (2023/12/29) (x86_64)
Copyright(C) 2013-2023 Super Micro Computer, Inc. All rights reserved.

Notice:
The mlist format
IP_or_HostName
.....
Status: The managed system 192.168.40.115 is rebooting.

.....Done
.....
.....
.....

Status: The BIOS configuration is updated for 192.168.40.115

WARNING: Without option --post_complete, please manually confirm the managed system is POST complete before
executing next action.

```

4. Once the node reboots either manually or as part of the **ChangeBiosCfg** command, verify that the change is effective.

Continuing the previous example, the following command shows "**NUMA Nodes Per Socket**" is set to "**NPS4**".

```
[SUM_HOME]$ ./sum -i 192.168.40.115 -u admin -p ddn@2024 -c GetCurrentBiosCfg | grep "NUMA Nodes  
Per Socket"  
    <Setting name="NUMA Nodes Per Socket" selectedOption="NPS4" type="Option">
```

Contacting DDN Support

If you have questions or require assistance, contact DDN Support:

Web

| | |
|-----------------------------|---|
| DDN Customer Support Portal | support.ddn.io |
| Portal Registration | support.ddn.io/DDNUserRegistration |
| Portal Assistance | portalsupport@ddn.com |

Email

| | |
|-------------------|----------------|
| DDN Support Email | support@ddn.io |
|-------------------|----------------|

Telephone

| | |
|---------------------------------|--|
| DDN Support Worldwide Directory | www.ddn.com/support/global-services-overview |
|---------------------------------|--|

Bulletins & Notices

| | |
|-------------------------------|--|
| Support Bulletins | www.ddn.com/support/technical-support-bulletins |
| End-of-Life Notices | www.ddn.com/support/end-of-life-notices |
| Bulletin Subscription Request | support-tsb@ddn.com |

