Oracle® Exadata

Exascale User's Guide

25.1
F17209-08
February 2025

Oracle Exadata Exascale User's Guide, 25.1

F17209-08

Primary Author: Peter Fusek

Contributing Authors: Akshay Shah, Alex Blyth, Alex Tsukerman, Allan Graves, Arun Venkataraman, Barb Glover, Boris Erlikhman, Chandra Pabba, Christian Craft, Elisabeth Thibault, Evangelos Vlachos, Frank Kobylanski, Francois Marcoux, Gaurav Chadha, Gautam Bhatt, Jaime Figueroa, Joshua Smith, Kai Zhang, Krishnadev Telikicherla, Kuan-Ju Chen, Maruti Sharma, Min Yun Law, Nilesh Choudhury, Pedro Gonzalez, Ruggero Citton, Saeed Abedigozalabad, Seth Miller, Siddhartha Datta, Soma Prasad, Tony Dziedzic, Vijay Sridharan, Vyomkesh Tripathi

Contents

Preface

This guide describes how to use and administer Oracle Exadata Exascale.

This Preface contains the following topics:

Audience

The audience for this book includes system administrators, database administrators, and storage administrators. This book is intended for database and storage administrators who configure, administer, manage, and consume storage services using Oracle Exadata Exascale.

To use this document, you should be familiar with basic Oracle Exadata and Oracle Database concepts and administrative procedures. Also, you might want to review documentation about related products, such as Oracle Advanced Cluster File System (ACFS), Oracle Grid Infrastructure, and Oracle Real Application Clusters (Oracle RAC). For a list of related books, refer to Related Documents.

Documentation Accessibility

For information about Oracle's commitment to accessibility, visit the Oracle Accessibility Program website at http://www.oracle.com/pls/topic/lookup?ctx=acc&id=docacc.

Access to Oracle Support

Oracle customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Diversity and Inclusion

Oracle is fully committed to diversity and inclusion. Oracle respects and values having a diverse workforce that increases thought leadership and innovation. As part of our initiative to build a more inclusive culture that positively impacts our employees, customers, and partners, we are working to remove insensitive terms from our products and documentation. We are also mindful of the necessity to maintain compatibility with our customers' existing technologies and the need to ensure continuity of service as Oracle's offerings and industry standards evolve. Because of these technical constraints, our effort to remove insensitive terms is ongoing and will take time and external cooperation.

Related Documents

For more information, refer to the following Oracle resources:

- *Oracle Exadata Database Machine System Overview*
- *Oracle Exadata Database Machine Installation and Configuration Guide*
- *Oracle Exadata System Software User's Guide*
- *Oracle Exadata Database Machine Maintenance Guide*
- *Oracle Database Administrator's Guide*
- *Oracle Database Concepts*
- *Oracle Database Reference*
- *Oracle Database SQL Language Reference*
- *Oracle Clusterware Administration and Deployment Guide*
- *Oracle Real Application Clusters Administration and Deployment Guide*

Conventions

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

1

Oracle Exadata Exascale Overview

1.1 What is Oracle Exadata Exascale?

Before Exascale, Exadata administrators were required to allocate dedicated storage resources for every Oracle Grid Infrastructure (GI) cluster. Furthermore, administrators had to decide how much storage to dedicate to data files and how much to apportion for recovery files. Once set, these allocations were not easily changeable. Consequently, multiple GI clusters and databases could not share storage easily and dynamically, resulting in fragmentation and under-utilization of storage resources.

Oracle Exadata Exascale further empowers Exadata to meet the most demanding corporate and cloud computing requirements by decoupling Oracle Database and GI clusters from the underlying Exadata storage servers. Exascale software services can manage a large fleet of Exadata storage servers connected by the Exadata RDMA Network Fabric, providing storage services to multiple GI clusters and databases while enabling:

- Secure sharing of storage resources with strict data isolation, allowing different users and databases to share a large pool of storage while ensuring that data is inaccessible to users without the appropriate privileges
- Flexible and dynamic storage provisioning for many users and databases
- Increased storage utilization and efficiency while reducing storage costs
- Sharing of otherwise idle storage processing resources to improve performance

Furthermore, Exascale introduces advanced snapshot and cloning capabilities that are tightly integrated with Oracle Database and eliminate the requirement for a test master database to support snapshots and clones on Exadata. For example, Oracle Database provides native snapshot and cloning functionality for pluggable databases (PDBs) through the `CREATE PLUGGABLE DATABASE` and `ALTER PLUGGABLE DATABASE` SQL commands. When Oracle Database uses Exascale storage, the PDB snapshot and snapshot copy (cloning) functions automatically use native Exascale snapshots and clones, which are space-efficient file copies based directly on the underlying Oracle Database files.

In addition to unparalleled support for Oracle Database, Exascale provides block storage services, which deliver sophisticated capabilities to create and manage arbitrary-sized raw block volumes based on Exascale storage.

While end users can create and use Exascale block volumes for numerous applications, Exadata also leverages Exascale block volumes internally to store Exadata database server virtual machine (VM) images. Placing VM images in Exascale removes the dependency on local storage inside the Exadata compute nodes, which enables the creation of more VMs and provides the infrastructure to support seamless VM migration between different Exadata compute nodes.

Despite the fact that Exascale transforms Exadata storage, Exascale also preserves the proven strengths and benefits of Exadata:

- Scalability - including efficient support for hundreds of Exadata storage servers in an Exascale cluster
- High availability - based on a clustered architecture with built-in redundancy and dynamic fail-over of software services
- High performance - utilizing Exadata RDMA memory and Exadata Smart Flash Cache
- Reliability - using proven Exadata storage server technologies
- Security - employing advanced security protocols and automatic encryption

Exascale requires Oracle Exadata system hardware with RoCE Network Fabric (X8M or later). For full-featured native Oracle Database file storage in Exascale, you must use Oracle Database 23ai release 23.5.0 or later. You can also employ Exascale block volumes to support databases using older Oracle Database software releases back to Oracle Database 19c.

1.2 Exascale Components and Concepts

This section describes key Exascale components and concepts:

1.2.1 Exascale Services

Exascale system components are implemented using a series of clustered software services. The core Exascale storage services predominantly run on the Exadata storage servers.

Exascale contains the following storage services:

- **Cluster Services**

  Exascale cluster services, also known as Exascale global services (EGS), provide the core foundation for the Exascale system. EGS primarily manages the storage allocated to Exascale storage pools. It also manages storage cluster membership, provides security and identity services for storage servers and Exascale clients, and monitors the other Exascale services. Exascale cluster services use the Raft consensus algorithm.

  For high availability, every Exascale cluster contains five EGS instances.

  For Exascale clusters with five or more Exadata storage servers, one EGS instance runs on each of the first five storage servers.

  For Exascale configurations with fewer than five storage servers, one EGS instance runs on each Exadata storage server, and the remaining EGS instances run on the Exadata compute nodes to make up the required total of five. In a bare-metal configuration, EGS compute node instances run on the server operating system. In a configuration with compute nodes running on virtual machines (VMs), EGS compute node instances run in the hypervisor.

- **Control Services**

  Exascale control services, also known as Exascale RESTful Services (ERS), provide a management endpoint for Exascale management operations. All Exascale management operations come through ERS. But, no file I/O operations come through ERS.

  ERS service instances are deployed using front-end and back-end server processes. The front-end ERS processes provide a highly available client endpoint with load-balancing capabilities. The back-end ERS processes work with other software services to process requests and reply back to the client.

  Multiple ERS instances provide high availability and share the Exascale management workload. Typically, five ERS instances are distributed across the Exadata storage servers. However, for configurations with fewer than five storage servers, one ERS instance usually runs on each Exadata storage server.

  The Exascale Command Line (ESCLI) utility provides a simple command-line interface to perform Exascale monitoring and management functions. ESCLI commands are translated into ERS calls and run through ERS. ESCLI works in conjunction with the Exadata cell command-line interface (CellCLI) and does not replace it.

- **Exascale Vault Manager Services**

  Exascale vault manager, also known as Exascale data services (EDS), is the collective name for the Exascale software services that manage file and vault metadata:

  – **System Vault Manager**

    The system vault manager service (SYSEDS) serves and manages the metadata for Exascale vaults. This metadata includes vault-level access control lists (ACLs) and attributes.

    SYSEDS is a lightweight process, so one instance can usually service the load for an entire Exascale cluster. However, to ensure high availability, five SYSEDS instances are typically distributed across the Exadata storage servers. For configurations with fewer than five storage servers, one SYSEDS instance usually runs on each Exadata storage server.

  – **User Vault Manager**

    The user vault manager service (USREDS) serves and manages the metadata for files inside the Exascale vaults. This metadata includes file-level access control lists (ACLs)

and attributes, along with metadata that defines clones and snapshots. All file control operations, such as open and close, are serviced by the user vault manager service.

Multiple USREDS instances provide high availability and share the user workload. Typically, five USREDS instances are distributed across the Exadata storage servers. However, for configurations with fewer than five storage servers, one USREDS instance usually runs on each Exadata storage server.

- **Block Store Manager**

  The block store manager service (BSM) serves and manages the metadata for Exascale block storage. All block store management operations are serviced by BSM. These block store management operations include creating a volume, attaching a volume to an iSCSI initiator, creating a volume snapshot, and so on. BSM maintains the availability of the block store virtual IP (VIP) addresses that are used by iSCSI targets. It also coordinates the block store worker processes.

  To provide high availability, five BSM instances are typically distributed across the Exadata storage servers. However, for configurations with fewer than five storage servers, one BSM instance usually runs on each Exadata storage server.

- **Block Store Worker**

  The block store worker service (BSW) primarily services requests from block store clients and performs the resulting storage server I/O. It plays a role in clone and snapshot creation operations and is also responsible for performing volume backup and restore operations.

  To provide high availability and share the user workload, five BSW instances are typically distributed across the Exadata storage servers. However, for configurations with fewer than five storage servers, one BSW instance usually runs on each Exadata storage server.

  Optionally, BSW can also run on the Exadata compute nodes. This location enables the BSW instance to access the Exadata client network, which may be used to run volume backups to Oracle Cloud Infrastructure (OCI) object storage and to service iSCSI initiators external to Exadata.

  In a bare-metal configuration, BSW compute node instances run on the server operating system. In a configuration with compute nodes running on virtual machines (VMs), BSW compute node instances are typically run inside a dedicated guest VM.

- **Instance Failure Detection**

  The instance failure detection (IFD) service is a dedicated lightweight service that quickly detects and responds to any storage server failure. IFD automatically runs on every storage server that is associated with the Exascale cluster.

- **Exadata Cell Services**

  Exascale works in conjunction with, and relies on, the core Exadata cell services. Specifically, Exascale requires running instances of Cell Server (CELLSRV), Management Server (MS), and Restart Server (RS) on every storage server. Also, Exascale requires running instances of Management Server (MS) and Restart Server (RS) on every compute server that is associated with the Exascale cluster.

In addition to the Exascale storage services, the following client-side services provide specific support for various Exascale functions on the Exadata compute nodes:

- **Exascale Node Proxy**

  The Exascale node proxy (ESNP) service maintains information about the current state of the Exascale cluster, which it provides to local Oracle Grid Infrastructure and Oracle Database processes.

ESNP is a background process that runs on each Exadata compute node. In a bare-metal configuration, ESNP runs on the server operating system. In a configuration with compute nodes running on virtual machines (VMs), ESNP runs inside the guest VMs.

- **Exascale Direct Volume**

  Exascale Direct Volume (EDV) is the default and recommended volume attachment mechanism within the Exadata RDMA Network Fabric.

  The EDV service exposes Exascale volumes as EDV devices on Exadata compute nodes and services the I/O on each EDV device. EDV-managed storage can be used as raw block devices or to support various file systems, including Oracle Advanced Cluster File System (ACFS).

  The EDV service is required on all Exadata compute nodes where you want to use EDV devices. In a bare-metal configuration, the EDV service runs on the server operating system. In a configuration with compute nodes running on virtual machines (VMs), the EDV service runs in the KVM host to provide the hypervisor access to VM image files based on EDV-managed storage. The EDV service also runs in each guest VM to enable direct access to EDV-managed storage from inside the VM.

Related Topics

- Administer the Exascale Cluster

1.2.2 Storage Media Types

Exascale storage is provided by Exadata storage servers, and each storage device is categorized according to its media type. The supported media types are:

- `HC`: Identifies high capacity storage media, which uses hard disk drives (HDDs).
- `EF`: Identifies extreme flash storage media, which uses flash devices.

1.2.3 Pool Disks

Exascale uses pool disks and storage pools to organize the physical storage provided by Exadata storage servers.

A pool disk is an Exascale-specific Exadata grid disk that reserves a portion of an Exadata storage device (HDD or flash device) for use by Exascale. Like any other Exadata grid disk, a pool disk is created by allocating space from an Exadata cell disk.

Related Topics

- Administer Exascale Pool Disks

1.2.4 Storage Pools

Exascale uses pool disks and storage pools to organize the physical storage provided by Exadata storage servers.

An Exascale storage pool is a collection of pool disks that provides persistent physical storage for Exascale vaults and files. Each storage pool is a collection of pool disks using only one type of storage media (for example, HC). You cannot define a storage pool with pool disks having a mixture of media types (for example, HC and EF).

A working Exascale system requires at least one storage pool. However, there is no need to define multiple storage pools for data separation because Exascale vaults provide strong data isolation.

A storage pool can contain pool disks that reside on different generations of Exadata storage server hardware, which allows for easy migration to new generations of Exadata hardware.

You can dynamically reconfigure a storage pool by changing the size of the pool disks or by adding or removing pool disks (or Exadata storage servers). However, you should pay attention to these recommendations:

- For each storage pool, use pool disks that are spread across all of the available storage devices in each storage server.
- On each storage server, maintain consistent sizing for all of the pool disks belonging to a storage pool. However, within a storage pool, pool disks on different storage servers can have different sizes.

Internally, a storage pool contains separate areas known as storage pool rings. Furthermore, the physical storage is arranged into smaller groups of pool disks and their associated storage servers, known as disk partner groups and cell partner groups.

The disk and cell partner groups maximize availability by limiting the potential effects of multiple simultaneous failures. This is because the probability of multiple simultaneous failures within a small group is much lower than the probability of multiple simultaneous failures anywhere across a large system. Also, simultaneous failures affecting different groups can be handled independently within each group.

Furthermore, Exascale uses separate storage pool rings for data and recovery files. The organization of the storage pool rings guarantees that data file extents and related recovery files automatically use different disk partner groups. So, in the unlikely event that multiple failures affect a data file, its corresponding recovery files remain available.

You cannot specify or change the internal organization of the storage pool rings or the cell and disk partner groups. However, Exascale provides information about these constructs, which is helpful for monitoring the system and understanding the impact of any storage failure scenario.

Related Topics

- Administer Exascale Storage Pools

1.2.5 Vaults

An Exascale vault is a logical storage container that uses the physical resources provided by Exascale storage pools. Each vault is associated with at least one storage pool. At most, a vault may be associated with one storage pool for each type of storage media (one HC storage pool and one EF storage pool).

By default, a vault can use all the underlying storage pool resources. However, an Exascale administrator can limit the amount of space, I/O resources (I/Os per second, or IOPS), and cache resources associated with each vault.

To an end-user and Oracle Database, a vault appears like a top-level directory that contains files. Exascale uses the convention of beginning vault names with the at sign (`@`) character. For example, `@MYVAULT`. So, a fully qualified Exascale file path always starts with the at sign (`@`) and vault name, followed by the rest of the file path. For example, `@MYVAULT/myexample/myfilename`.

Exascale vaults facilitate strict data separation, ensuring that data is isolated to specific users and separated from other data and users. A vault, and its contents, are invisible to users without the appropriate privileges. Without the correct entitlements, users of one vault cannot see another vault, even though data from both vaults may be striped across the same underlying storage pools.

Related Topics

- Administer Exascale Vaults

1.2.6 Files

Exascale is optimized to directly store and manage files associated with Oracle Database and Oracle Grid Infrastructure.

When Oracle Database or Oracle Grid Infrastructure creates a file inside a vault, Exascale automatically understands the file type. By using a template that is associated with the file type, Exascale automatically applies the appropriate file storage attributes. This process indirectly determines the storage pool that houses the file.

You can also use the Exascale Command Line (ESCLI) mkfile command to create a file. By this means, you can explicitly set the file storage attributes, or you can influence the file storage attributes by specifying the file type or applying a specific template. You can then use the ESCLI putfile command to copy the contents of a file into Exascale.

To optimize storage efficiency, physical space is consumed only when data is written to the file. For example, when you use the ESCLI mkfile command to create a file, the specified file size is reserved in the vault, but the file does not use any physical storage at that point. Physical storage is only allocated when data is written to the file.

Technically, you can store any type of files inside an Exascale vault. However, Exascale is optimized to manage Oracle Database files, which are typically much larger than regular files. To optimally store and manage regular files, you can define a block volume on Exascale, attach the volume to a server, and then use the volume attachment to support a file system. See Exascale Block Store.

Related Topics

- Administer Exascale Files

1.2.7 File Storage Attributes

During file creation, every file in Exascale is associated with file storage attributes, which govern how the file is stored and managed. The attributes are:

- `mediaType`: Specifies the physical media type that is used to store the file. Exascale uses this attribute to place the file in a storage pool that uses the specified media type. Possible values are:
  - `HC`: Identifies high capacity storage media, which uses hard disk drives (HDDs).
  - `EF`: Identifies extreme flash storage media, which uses flash devices.
- `redundancy`: Specifies the number of data copies that are maintained. Possible values are:
  - `normal`: Indicates 2 mirrored copies of the file data.
  - `high`: Indicates 3 mirrored copies of the file data.
- `contentType`: Specifies the type of content in the file. Exascale internally uses this attribute to place file extents on physically separate devices in a way that maximizes availability if failures occurs. Possible values are:
  - `DATA`
  - `RECO`

File storage attributes are assigned to all files when they are created. Typically, the file storage attributes are implicitly assigned using templates, which assign attribute settings based on the file type. File storage attributes can be assigned explicitly when a file is created by using the ESCLI mkfile command.

You cannot change the file storage attributes after the file is created.

1.2.8 Templates

During file creation, every file in Exascale is associated with attributes that govern how the file is stored and managed.

An Exascale template is a named collection of file storage attribute settings. For example:

```
name: DATAFILE
mediaType: HC
redundancy: high
contentType: DATA
```

With templates, you can set file storage attributes automatically and consistently. When creating a file, Exascale automatically uses the template that matches the file type by default. For example, when Oracle Database creates a data file, the `DATAFILE` template is automatically used by default.

Note that templates only govern the assignment of file storage attributes when creating a file. You cannot change the file storage attributes after the file is created. Any change to a template applies only to files created after the change.

Within Exascale, templates are defined at two levels:

- **Cluster Templates** are defined at the cluster level. A cluster template is used if no vault-level template exists to override it.
- **Vault Templates** are defined at the vault level. A vault template defines vault-specific attributes that override the corresponding cluster template. For example, your cluster-level `DATAFILE` template may specify the use of `HC` media, but you may override that setting for a vault by having a vault-level `DATAFILE` template that specifies the use of `EF` media.

Additionally, **User Templates** are templates with user-specified names that do not correspond to a specific file type. User templates effectively override file-type templates. However, to use a user template, you must explicitly specify it during file creation. User templates can be defined at the cluster level or at the vault level. A vault-level user template will override a cluster-level user template having the same name.

Related Topics

- File Storage Attributes
- Administer Cluster Templates
- Administer Vault Templates

1.2.9 Extended Attributes

A user-defined extended attribute is non-standard metadata that is associated with an Exascale vault or file. An extended attribute consists of an extended attribute name and associated value. The extended attribute name is a string value that the user chooses. The extended attribute value can be a string value, or a binary value that is read from a file.

Multiple extended attributes are allowed for each Exascale vault or file. For each file or vault, the total available space for all extended attributes is 16 kilobytes, including the extended attribute names and values, along with an allowance for internal metadata.

Extended attributes with names that start with a dollar sign (`$`) are reserved for internal use by Exascale services and tools. You cannot create, modify, or delete these extended attributes.

Related Topics

- Administer Extended Attributes

1.2.10 Clones

An Exascale clone is a thinly-provisioned point-in-time copy of a file that is readable and writable. The source file for a clone can be a regular file, a snapshot, or another clone. Exascale uses redirect-on-write techniques to create and maintain clones very quickly and space-efficiently.

The clone creation process can work on an individual file or a groups of files. All clones created in the same operation are point-in-time consistent, and all files in a clone operation must be in the same vault.

A clone inherits the ACL and file storage attribute settings from the original file. After creation, you can modify the clone ACL and any modifiable attributes independently from the original file. You can also modify, or even delete, the original file without affecting the clone.

Typically, you will automatically use Exascale clones through the Oracle Database pluggable database (PDB) snapshot and cloning capabilities. For example, the Oracle Database `CREATE PLUGGABLE DATABASE ... FROM ...` command internally uses Exascale clones when the `FROM` database is on Exascale.

You can also use the ESCLI clonefile command to manually create clones, and you can use lssnapshots to view the association between clones and their sources.

Related Topics

- Administer Exascale Files
- Clone Files

1.2.11 Snapshots

An Exascale snapshot is a thinly-provisioned read-only point-in-time copy of a file. The source file for a snapshot can be a regular file, a clone, or another snapshot. Exascale uses redirect-on-write techniques to create and maintain snapshots very quickly and space-efficiently.

The snapshot creation process can work on an individual file or a groups of files. All snapshots created in the same operation are point-in-time consistent, and all files in a snapshot operation must be in the same vault.

A snapshot inherits the ACL and file storage attribute settings from the original file. After creation, you can modify the snapshot ACL and any modifiable attributes independently from the original file. You can also modify, or even delete, the original file without affecting the snapshot.

You can use the ESCLI snapshotfile command to create snapshots, and you can use lssnapshots to view the association between snapshots and their sources.

While an Exascale snapshot is read-only, you can effectively manufacture a writable snapshot by creating a snapshot and then making a clone of the snapshot. In this case, the snapshot

provides a durable point-in-time copy of the file, and the clone enables writing to a thinly-provisioned copy of the snapshot. At any point, you can effectively roll back to the snapshot creation time by removing and re-creating the clone.

Related Topics

- Administer Exascale Files
- Snapshot Files

1.2.12 Datasets

An Exascale dataset is a logical grouping of files within a vault. The primary purpose of a dataset is to enable tracking and management of storage utilization for files associated with Oracle Database and Oracle Grid Infrastructure.

Exascale automatically creates and maintains system-defined datasets, which are conceptually organized in a hierarchical tree. Each dataset directly contains some number of files and indirectly contains child datasets that correspond with other entities in the hierarchy. The hierarchy of system-defined datasets contains the following levels:

1. Exascale vault
2. Oracle Grid Infrastructure (GI) cluster
3. Oracle multitenant container database (CDB)
4. Oracle pluggable database (PDB)

Starting at the bottom, the files belonging to each PDB are contained in a separate dataset. The parent CDB dataset contains the files belonging to the CDB and all of the associated PDB datasets. All of the CDBs in a GI cluster as grouped in a GI-level dataset, which also contains GI-specific files such as the Oracle Cluster Registry (OCR) and voting files. At the top of the hierarchy, the vault-level dataset contains all of the GI-level datasets that consume storage in the vault. The vault-level dataset is also the default container for any files that don't belong to Oracle Database and Oracle Grid Infrastructure.

Each system-defined dataset is identified by a composite ID, which contains unique identifiers for the associated entities in the hierarchy. For example, each vault-level dataset is simply identified by the vault name (for example, `@MYVAULT`). But, a PDB dataset ID has the following format:

```
@Vault-name:GI-cluster-ID:CDB-ID:PDB-ID
```

Each system-defined dataset also has a name. The dataset name is also a composite of the relevant entities, but it contains human-readable names for the GI cluster, CDB and PDB components (instead of the system-generated unique identifiers). For example, a PDB dataset name has the following format:

```
@Vault-name/GI-cluster-name:CDB-name.PDB-name
```

Related Topics

- Administer Exascale Datasets

1.2.13 Exascale Users

Exascale has a system of user accounts enabling different users to perform actions and access data according to their assigned privileges. Though you can create a single Exascale

user with privileges to do everything, a typical configuration contains cluster administration users and storage users:

- Cluster administration users are typically provisioned with privileges to administer the Exascale cluster. Cluster administration users typically administer the physical storage objects; namely storage servers, storage pools and pool disks. They also administer Exascale software services, vaults, cluster templates and user accounts. See Oracle Exadata Exascale System Administration.

  By default, each Exascale cluster contains one superuser account. The user identifier (ID) for the superuser account is `admin`. The `admin` user can implicitly perform any system operation and effectively holds all system privileges.

  While you can use the `admin` user to perform cluster administration tasks, Oracle recommends that you create your own cluster administration users with specific privileges. For example, rather than having one cluster administrator that does everything, you may choose to have dedicated user accounts for security administration, storage administration, and so on.

- Storage users are typically provisioned with privileges to use storage within Exascale vaults. Storage users often administer the vaults they use, and sometimes even create new vaults. Storage users typically manage their own files and the access control lists (ACLs) that govern file access. They also administer vault-level templates, extended file attributes, and their own user credentials. See Oracle Exadata Exascale User-Specific Administration.

Additionally, Exascale contains one node administration account for every node (storage server or compute node) that runs Exascale software services. Each node administration account inherits its user ID from the server hostname and each account contains the privileges required to run the Exascale software services on the node. Do not directly use or modify these accounts.

Related Topics

- Administer Exascale Users

1.2.14 Exascale User Credentials

Exascale user authentication uses public and private key pairs, with user credentials stored in a digital key store.

Each Exascale user is associated with a public key. To prove their identity and connect to Exascale, a user must supply the matching private key.

Exascale user credentials are contained in a digital key store, also known as a wallet. To use Exascale, a user's wallet must contain their private key. It must also contain a copy of the Exascale trust store. As a matter of convenience, a wallet can also store the default endpoint for Exascale control services.

To facilitate flexible key management, each Exascale user can be associated with up to three public and private key pairs. However, each wallet should contain only one Exascale user name and one private key.

For maximum security, an Exascale user should create their own public and private key pairs and manage their own wallet. This is recommended to ensure the integrity of the private keys, since the user should never share a private key, not even with the Exascale administrator.

Related Topics

- Administer Exascale User Credentials

1.2.15 User Privileges

User privileges control the actions performed by Exascale users.

Each Exascale user is subject to a set of user privileges, which govern the actions that the user is allowed to perform.

User privileges are assigned to users by using the ESCLI mkuser or chuser commands.

There are four types of Exascale user privileges, and any user may hold privileges across multiple privilege types. The following list describes the privilege types and the available user privileges:

- **Cluster Level Storage Privileges** primarily govern the administration actions that the receiving user is allowed to perform on storage resources in the Exascale cluster. Typically, cluster level storage privileges are only assigned to users that administer the Exascale cluster. A user may hold zero or one of the following cluster level storage privileges:
  - `cl_monitor`: Enables the receiving user to monitor the Exascale cluster by performing list operations using ESCLI and CellCLI.
  - `cl_operator`: Enables the receiving user to:
    - Monitor the Exascale cluster by performing list operations using ESCLI and CellCLI.
    - Manage pool disks (create, drop, online, offline).
    - Manage software services (list, startup, shutdown, restart, delete).
    - Manage the trust store.
  - `cl_admin`: A set of system administrator privileges that includes all the `cl_monitor` and `cl_operator` privileges, along with all of the privileges from the other privilege types; namely:
    - All the cluster level user privileges: `user_create`, `system_restore`, and `on_behalf_of`.
    - All of the vault top-level privileges specified in `vlt_manage`.
    - All the service privileges: `cellsrv`, `egs`, `ers`, `syseds`, `usreds`, `bsm`, and `bsw`.

    This privilege also enables the receiving user to:
    - Grant any privilege to any user.
    - Reset a key for any user.
    - Create and delete storage pools.
    - View extent map information.
- **Cluster Level User Privileges** govern the administration actions that the receiving user is allowed to perform on the Exascale cluster. Typically, cluster level user privileges are only assigned to users that administer the Exascale cluster. A user may hold zero or more of the following cluster level privileges:
  - `user_create`: Enables the receiving user to create new users in the cluster.
  - `system_restore`: Enables the receiving user to restore an Exascale backup.
  - `on_behalf_of`: A special privilege that enables the receiving user to send a request to Exascale control services (ERS) on behalf of another user.

For example, consider a user that sends a request to ERS, which involves an action that must be performed by another Exascale service. In this case, ERS uses this privilege to forward the action to the other Exascale service on behalf of the original end user.

Typically, this privilege is only assigned to the internal administration accounts that reside on each Exascale node.

- **Vault Top-Level Privileges** govern the actions that the receiving user is allowed to perform on all vaults and files. Typically, vault top-level privileges are assigned to users that use and manage files in Exascale vaults. A user may hold zero or one of the following vault top-level privileges:
  - `vlt_inspect`: Enables the receiving user to create new vaults. The receiving user also gets complete control over files created in those vaults. This privilege is assigned to new users by default.
  - `vlt_read`: Includes the `vlt_inspect` privileges and also enables the receiving user to list all existing vaults, display attributes for any vault, create files in any vault, list files in any vault, and display attributes for any file.
  - `vlt_use`: Includes the `vlt_read` privileges and also enables the receiving user to open any file for reading.
  - `vlt_manage`: Includes the `vlt_use` privileges and also enables the receiving user to open any file for read and write, alter any vault or file, and drop vaults and files.

  Vault top-level privileges work in addition to access control lists (ACLs). To perform an action on a vault or file, a user requires the appropriate vault top-level privilege or the appropriate ACL privilege. See Vault and File Access Control.
- **Service Privileges** govern the Exascale software services that the receiving user is allowed to run. Typically, service privileges are only assigned to the internal node-specific administration accounts that reside on each Exascale node. A user may hold zero or more of the following service privileges:
  - `cellsrv`: Enables the receiving user to run the core Exadata cell services.
  - `egs`: Enables the receiving user to run Exascale cluster services (also known as Exascale Global Services).
  - `ers`: Enables the receiving user to run Exascale control services (also known as Exascale RESTful Services).
  - `syseds`: Enables the receiving user to run the system vault manager service.
  - `usreds`: Enables the receiving user to run the user vault manager service.
  - `bsm`: Enables the receiving user to run the block storage manager service.
  - `bsw`: Enables the receiving user to run the block storage worker service.
  - `edv`: Enables the receiving user to run the Exascale Direct Volume service.

Additionally, `no_privilege` is a special privilege that removes all privileges from the receiving user. When it is assigned to a user, `no_privilege` cannot be combined with any other privilege.

Related Topics

- Modify User Privileges

1.2.16 Access Control Lists

Access control lists (ACLs) govern the operations that users can perform on Exascale vaults and files.

Each Exascale vault or file has an ACL. A vault ACL enables users to perform actions on the vault and on the files that it contains. A file ACL only controls the file that it is associated with.

The following table lists the ACL privileges and the actions that they enable users to perform:

ACL Privilege	In a vault ACL, the ACL privilege enables the user to:	In a file ACL, the ACL privilege enables the user to:
`inspect`	• Create a file in the vault. • View attributes of the vault, but not the vault contents.	• View attributes of the file, but not the file contents.
`read`	• View attributes of all files in the vault, but not their contents. • Perform all `inspect` actions.	• Read the file contents. • Perform all `inspect` actions.
`use`	• Read the contents of all files in the vault. • Perform all `inspect` and `read` actions.	• Read and write the file contents. • Alter attributes of the file. • Perform all `inspect` and `read` actions.
`manage`	• Read and write the contents of any file in the vault. • Alter the attributes and ACL for the vault and any file in the vault. • Drop the vault and any file in the vault. • Perform all `inspect`, `read` and `use` actions.	• Alter the file ACL. • Drop the file. • Perform all `inspect`, `read` and `use` actions.

Note that the same ACL privilege enables different actions in a vault ACL or a file ACL. For example, in a file ACL the `read` privilege enables the user to read the contents of the file. However, to read file contents using a vault ACL requires the `use` privilege.

Every ACL is a list of user IDs and privilege pairs. Depending on the user creation method, the user ID may be a system-generated value or a user-specified value. For example:

```
96a68014-5762-4579-86ee-29eb743decbd:manage;scott:use;sue:inspect;dd7c8e35-3c8
d-4441-a9b0-f58e959b84ba:read
```

A user is added to an ACL when they are assigned one of the ACL privileges. A user is removed from an ACL when they are assigned the `none` privilege. It is possible for a vault or file to have an empty list of user and privilege pairs, which is also known as a null ACL.

ACLs work in conjunction with user privileges, in particular vault top-level privileges. To perform an action on a vault or file, a user requires the appropriate ACL privilege or the appropriate vault top-level privilege. See Vault and File Access Control.

Related Topics

- Administer Access Control Lists

1.2.17 Vault and File Access Control

Access control lists (ACLs) work together with user privileges, in particular vault top-level privileges, to control access to Exascale vaults and files. To perform an action on a vault or file, a user requires the appropriate ACL privilege or the appropriate vault top-level user privilege. Because Exascale has no formal concept of vault or file ownership, all operations are governed by the combination of user privileges and ACLs.

The following table lists the minimum vault top-level user privilege, vault ACL privilege, or file ACL privilege that is required to perform various operations on Exascale vaults and files. Where relevant, associated ESCLI commands are listed along with each operation.

Operation	Required Vault Top-Level User Privilege	Required Vault ACL Privilege	Required File ACL Privilege
Create vault (mkvault)	`vlt_inspect`	Not applicable.	Not applicable.
List vaults (ls)	`vlt_read`	`inspect`	Not applicable.
List files in a vault (ls)	`vlt_read`	`read`	Not applicable.
Drop vault (rmvault)	`vlt_manage`	`manage`	Not applicable.
View vault attributes (lsacl, lsxattr, lstemplate)	`vlt_read`	`inspect`	Not applicable.
Alter vault attributes (chxattr, mktemplate, rmtemplate, rmxattr)	`vlt_manage`	`manage`	Not applicable.
Alter vault ACL (chacl)	`vlt_manage`	`manage`	Not applicable.
Create file (mkfile)	`vlt_read`	`inspect`	Not applicable.
Drop file (rmfile)	`vlt_manage`	`manage`	`manage`
Read and write file contents (putfile)	`vlt_manage`	`manage`	`use`
Read file contents (getfile)	`vlt_use`	`use`	`read`
View file attributes (lsacl, lsxattr)	`vlt_read`	`read`	`inspect`
Alter file attributes (chxattr, rmxattr)	`vlt_manage`	`manage`	`use`
Alter file ACL (chacl)	`vlt_manage`	`manage`	`manage`

To perform an operation, a user requires at least one of the privileges that is listed beside the operation. For example, to open a file for read-only access the requesting user must have at least one of the following:

- The `vlt_use` vault top-level user privilege.
- The `use` vault ACL privilege for the vault containing the file.
- The `read` file ACL privilege for the file being opened.

> **Note:**
>
> - To create a snapshot or a clone, the user requires the privileges for the 'read file contents' operation to read the source file, and they also require the privileges for the 'create file' operation to create a file for the snapshot or clone. After creation, operations on snapshots and clones require the same privileges as for any other file.
> - Exascale ensures that users can manage the vaults and files that they create. During vault creation, if the creating user does not have the `vlt_manage` vault top-level user privilege, then Exascale adds the creating user to the vault ACL with the `manage` privilege. During file creation, if the creating user does not have the `vlt_manage` vault top-level user privilege and the user does not have the `manage` privilege in the vault ACL, then Exascale adds the creating user to the file ACL with the `manage` privilege.

Related Topics

- Modify User Privileges
- Administer Access Control Lists

1.2.18 Trust Store

The Exascale trust store is a group of digital certificates that facilitates trusted communication between servers running Exascale cluster services (EGS). The digital certificates ensure the identity of the EGS servers, which enables clients to trust the legitimacy of the Exascale cluster.

During the initial configuration of Exascale, the trust store is automatically created and distributed across the cluster.

The trust store is modified when EGS servers are added to, or deleted from, the Exascale cluster.

While most operations involving the trust store occur automatically, some manual operations are occasionally required. For example, the trust store must be manually fetched into a new user wallet.

Related Topics

- Fetch the Trust Store

1.2.19 Resource Management

Resource management governs how various storage resources are consumed. Exascale resource management works in conjunction with existing Exadata I/O Resource Management (IORM) and Oracle Database Resource Manager (DBRM) capabilities.

Exascale resource management is implemented in a hierarchical manner. Firstly, inter-vault resource management governs how resources are allocated to different vaults using the IORM vault plan. The resulting resource allocation for each vault is further divided among the different databases and block volumes using intra-vault resource management with Exascale resource profiles. Finally, the resulting resource allocation for each database is divided among PDBs and consumer groups using intra-database resource management with DBRM definitions.

- Inter-Vault Resource Management Using Exadata IORM
- Intra-Vault Resource Management Using Exascale Resource Profiles
- Intra-Database Resource Management Using DBRM

1.2.19.1 Inter-Vault Resource Management Using Exadata IORM

Inter-vault resource management defines how resources are allocated to different vaults. Inter-vault resource management uses Exadata IORM in conjunction with a vault plan, which controls resource allocation between Exascale vaults.

By default, each vault has access to all of the resources in the Exascale cluster. In this case, the vault plan shares resources equally across all vaults.

However, an Exascale administrator can set various vault-specific resource provisioning attributes, which may limit the space, I/O bandwidth (IOPS), and caching resources available to the vault. In this case, the vault plan is automatically derived from the vault definitions and propagated to every storage server in the Exascale cluster. Furthermore, the vault plan automatically adjusts with changes to the resource provisioning attributes. No additional administrator intervention is required.

On a system that uses Exascale and traditional Exadata storage based on Oracle Automatic Storage Management (Oracle ASM), the vault plan coexists with other parts of the IORM plan. For example, an Exadata administrator can still define an inter-database plan (`dbplan`) to manage resource allocation across databases using Oracle ASM. In this case, resources are shared equally between Exascale and Oracle ASM.

1.2.19.2 Intra-Vault Resource Management Using Exascale Resource Profiles

Intra-vault resource management defines how resources are shared by the various Oracle databases and block store volumes that use a vault. Within Exascale, intra-vault resource management is governed by resource profiles.

By default, every Exascale client (Oracle database or block store volume) has access to all of the resources in their associated vault. Furthermore, I/O resources are shared equally when the system is under load.

To enable more granular I/O resource management, you can associate each Exascale client with a resource profile. You can define any number of resource profiles, but you can only associate an Exascale client with one resource profile at a time.

Each resource profile contains the following resource limits and settings:

- I/O Bandwidth (IOPS) — For each Exascale media type (HC and EF), you can define a limit value and a share value. While I/O bandwidth utilization is less than the vault capacity, the limit controls each client. However, when I/O bandwidth utilization reaches capacity, resource allocation across all clients is further governed by the share values.

  The limit value specifies the absolute limit of I/O bandwidth available to each client associated with the resource profile. The value represents a fraction out of 10000. For example, a value of 1 represents a limit of 1/10000 (0.01%), 5000 represents 5000/10000 (50%), 10000 represents 10000/10000 (100%, or effectively unlimited), and so on. If not specified, the default limit value is 10000 (effectively unlimited).

  The share value defines the proportional share of I/O bandwidth available to each client associated with the resource profile. Each client's share is relative to all other client shares. A higher share value implies higher priority. For example, consider a system with 2 clients, where client A has a share value of 2 and client B has a share value of 1. In this case, when I/O bandwidth utilization reaches capacity, client A gets 2/3 (66.67%) of the I/O bandwidth, which is twice as much as client B (1/3, or 33.33%). Now, consider adding client C with a share value of 7. After the addition of client C, when I/O bandwidth utilization reaches capacity, client A gets 2/10 (20%) of the I/O bandwidth, which is still twice as much as client B (1/10, or 10%), but client C gets 7/10 (70%) of the I/O bandwidth. The range of valid values is 1-100. If not specified, the default share value is 1.

- Flash Cache and Exadata RDMA Memory (XRMEM) Cache — For each type of cache, you can enable or disable use of the cache by clients associated with the resource profile. Then, for each enabled cache type, you can specify a minimum and maximum usage value in the range of 0 to 10000.

  The minimum value guarantees a portion of the cache for each client associated with the resource profile. Nominally, the value represents a fraction out of 10000. For example, a value of 1 represents a limit of 1/10000 (0.01%), 5000 represents 5000/10000 (50%), 10000 represents 10000/10000 (100%, or effectively unlimited), and so on. However, if the sum of all minimum values exceeds 10000 across all clients and resource profiles, then all values are scaled down proportionally to ensure that minimum guarantees can be honored. If not specified, the default minimum value is 0 (no guaranteed minimum).

  The maximum value specifies the absolute limit of cache space available to each client associated with the resource profile. The value represents a fraction out of 10000. For example, a value of 1 represents a limit of 1/10000 (0.01%), 5000 represents 5000/10000 (50%), 10000 represents 10000/10000 (100%, or effectively unlimited), and so on. If not specified, the default value is 10000 (effectively unlimited).

- Flash Log — You can enable or disable use of the flash log accelerator for clients associated with the resource profile. If not specified, the default setting enables the use of flash log.

You can also create a system-reserved resource profile named `$UNASSIGNED`. All Exascale clients not explicitly associated with a resource profile are automatically governed by the `$UNASSIGNED` profile. The `$UNASSIGNED` resource profile contains only two modifiable attributes, which specify the maximum fraction (out of 10000) of flash cache space and XRMEM cache space assigned to the profile. All other attributes of the `$UNASSIGNED` resource profile use the previously described default values.

All Exascale clients governed by the `$UNASSIGNED` profile share the specified cache resources. The behavior differs from regular resource profiles, where each application of the resource profile defines the resource allocation for one associated client.

If you do not create the `$UNASSIGNED` resource profile, all unassigned Exascale clients share any unassigned flash cache space and XRMEM cache space. If there is no unassigned space to share, the system automatically reserves 5% of the cache space for unassigned Exascale clients.

Resource over-provisioning across multiple clients and resource profiles is allowed. For over-provisioned resources, the system automatically adjusts the resource shares to maintain the relative proportions for each client.

For example, consider a system with the following resource profiles:

- GOLD resource profile:
  - EF IOPS: share=11, limit=5100
  - HC IOPS: share=40, limit=5200
  - Flash Cache: enabled=TRUE, minimum=500, maximum=1000
  - XRMEM Cache: enabled=TRUE, minimum=400, maximum=800
  - Flash Log: enabled=TRUE
- SILVER resource profile:
  - EF IOPS: share=6, limit=2500
  - HC IOPS: share=20, limit=2500
  - Flash Cache: enabled=TRUE, minimum=300, maximum=600
  - XRMEM Cache: enabled=TRUE, minimum=200, maximum=400
  - Flash Log: enabled=TRUE
- BRONZE resource profile:
  - EF IOPS: share=3, limit=900
  - HC IOPS: share=10, limit=1500
  - Flash Cache: enabled=TRUE, minimum=200, maximum=500
  - XRMEM Cache: enabled=TRUE, minimum=100, maximum=200
  - Flash Log: enabled=FALSE
- `$UNASSIGNED` resource profile:
  - Flash Cache: maximum=1000
  - XRMEM Cache: maximum=500

Also, imagine that the system hosts 8 databases, which are associated with the resource profiles as follows:

- DB1 and DB2 are associated with the GOLD resource profile.
- DB3 is associated with the SILVER resource profile.
- DB4 and DB5 are associated with the BRONZE resource profile.
- DB6, DB7, and DB8 are not explicitly associated with any resource profile. Therefore, they are implicitly associated with the `$UNASSIGNED` resource profile.

Now consider how the resource profiles are used to manage a specific resource, such as I/O bandwidth on high capacity storage media (HC IOPS):

- While the resource utilization is less than 100% of the vault capacity, each database is capped using the limit value. For example:
  - DB1: 5200 (52%)
  - DB2: 5200 (52%)
  - DB3: 2500 (25%)

  - DB4: 1500 (15%)
  - DB5: 1500 (15%)
  - DB6: 10000 (100%, default value, effectively unlimited)
  - DB7: 10000 (100%, default value, effectively unlimited)
  - DB8: 10000 (100%, default value, effectively unlimited)

  If the limits keep utilization to less than 100% of the vault capacity, then no further intervention is required.

- While the resource utilization is 100%, the resource is allocated proportionally using the share value. For example:
  - DB1: 40
  - DB2: 40
  - DB3: 20
  - DB4: 10
  - DB5: 10
  - DB6: 1 (default value)
  - DB7: 1 (default value)
  - DB8: 1 (default value)

  In this example, the sum of the shares is 123, resulting in the following resource allocations:
  - DB1: 40/123 (35.52%)
  - DB2: 40/123 (35.52%)
  - DB3: 20/123 (16.26%)
  - DB4: 10/123 (8.13%)
  - DB5: 10/123 (8.13%)
  - DB6: 1/123 (0.81%)
  - DB7: 1/123 (0.81%)
  - DB8: 1/123 (0.81%)

Related Topics

- Administer Resource Profiles

1.2.19.3 Intra-Database Resource Management Using DBRM

Intra-database resource management defines how resources are shared within an Oracle database, which may include multiple pluggable databases (PDBs). Intra-database resource management uses existing Oracle Database Resource Manager (DBRM) definitions, which are transparently propagated to Exadata storage server.

1.2.20 Exascale Block Store

The Exascale block store provides capabilities to create and manage arbitrary-sized raw block volumes based on Exascale storage. Each Exascale volume can be used as an Exascale Direct Volume (EDV) attachment or iSCSI target.

The Exascale block store features:

- High availability - based on a clustered architecture with built-in redundancy and dynamic fail-over of software services
- High performance - utilizing Exadata RDMA memory and flash cache
- Reliability - using proven Exadata storage server technologies
- Security - employing standard security protocols and automatic volume encryption
- Rich functionality - including instantaneous volume snapshots and backups

The following topics introduce various Exascale block store concepts and features:

- Exascale Volumes
- Volume Attachments
- Volume Snapshots
- Volume Clones
- Volume Backups
- Block Store VIPs
- Oracle Advanced Cluster File System (ACFS) on Exascale
- Virtual Machine Images

1.2.20.1 Exascale Volumes

An Exascale block volume is an arbitrary-sized allocation of storage space, which can be used as an Exascale Direct Volume (EDV) attachment or iSCSI target.

Internally, each volume is an Exascale file with special properties that identify it as block storage space. Like other Exascale files, physical space for the volume is only materialized when data is written to the volume.

Each volume is created in a user-specified vault. You can also optionally specify the physical media type used to store the volume. By default, volumes are stored on hard disk drives (`HC` media type).

Each volume can be associated with a series of optional attributes, which define the detailed characteristics of the volume. For example, the `redundancyType` attribute specifies the number of data copies (mirrors) that are maintained. Attributes like `maxReadIops` and `maxWriteIops` constrain the system resources that a volume is allowed to consume.

To use a volume, you must create a volume attachment. Depending on the type of attachment that you create, the volume is marked for use as an EDV attachment or iSCSI target.

Unlike regular Exascale files, volumes are not subject to access controls defined by user privileges and ACLs. Consequently, volume administration tasks (attachment, detachment, modification, and so on) must be performed by the volume owners or an Exascale cluster administrator (having the `cl_admin` privilege).

Related Topics

- Administer Volumes

1.2.20.2 Volume Attachments

To use a block volume, you must create a volume attachment. There are two types of volume attachment:

- An Exascale Direct Volume (EDV) attachment creates a association between the volume and an EDV device file on the Exadata compute node or Oracle Grid Infrastructure (GI) cluster hosting the attachment.

  If you create a cluster-wide attachment, the EDV device file is created on every node in the GI cluster. If you create a node-specific attachment, the corresponding EDV device is only created on that node.

  After attachment, the volume can be used as a raw block storage device or to support a file system. To implement Oracle Advanced Cluster File System (ACFS) on Exascale block volume storage, you must use an EDV attachment.

  The I/O for each attachment is serviced by an EDV server process. Automatic data encryption and decryption occurs within the EDV process.

  EDV is the default and recommended volume attachment type for cases where clients want to use volumes within the Exadata RDMA Network Fabric. In addition to general file storage, an EDV attachment may be used to support Oracle Database data files for versions before Oracle Database 23ai. However, an EDV attachment cannot be used for a bootable volume.
- An iSCSI attachment enables the volume to be used as an iSCSI target.

  Attachments of this kind create a association between the volume (iSCSI target) and an iSCSI initiator. After attachment, the iSCSI initiator can use standard iSCSI protocols and commands to interact with the target (volume).

  The I/O for each attachment is serviced by a block store worker (BSW) process that provides a reliable network endpoint by using a block store virtual IP address (VIP).

  Exascale supports mutual CHAP (Challenge Handshake Authentication Protocol) for authentication of iSCSI initiators. Data passing through an iSCSI volume attachment is automatically encrypted and decrypted inside the block store worker process.

  An Exascale iSCSI attachment may be used to support a bootable volume.

After you create the first attachment for a volume, the volume is marked and can only be attached using the same type of volume attachment.

Regardless of attachment type, Exascale supports the simultaneous use of multiple attachments for each volume, which can be used to support various types of clustered file systems.

Related Topics

- Administer Volume Attachments

1.2.20.3 Volume Snapshots

A volume snapshot is a thinly-provisioned read-only point-in-time copy of a volume.

Volume snapshots have the following uses:

- You can create attachments to a volume snapshot and use it as a read-only volume.
- You can use a volume snapshot as the source for a volume clone.
- You can use a volume snapshot as the source for a volume backup.

Related Topics

- Administer Volume Snapshots

1.2.20.4 Volume Clones

A volume clone is a thinly-provisioned read-write point-in-time copy of a volume snapshot.

A volume clone is functionally equivalent to a standard (non-cloned) volume.

Volume clones have the following uses:

- You can create attachments to a volume clone and use it like any other writable volume.
- You can create volume snapshots that are based on the volume clone.

Related Topics

- Administer Volume Clones

1.2.20.5 Volume Backups

A volume backup is a backup of an Exascale volume snapshot, which provides a consistent point-in-time copy of the volume.

Volume backups are stored in Oracle Cloud Infrastructure (OCI) object storage. To use volume backups you must have access to OCI Object Storage Service, and then specify the service details as attributes of the Exascale cluster.

You can restore a volume backup into a new volume. When you restore a volume backup, the restored volume is separate from the volume backup and the original backup source volume. There is no on-going association between a restored volume and the volume backup or the original backup source volume.

Related Topics

- Administer Exascale Block Store System Objects
- Administer Volume Backups

1.2.20.6 Block Store VIPs

To facilitate high availability and load balancing, iSCSI initiators communicate with Exascale iSCSI targets by using virtual IP addresses (VIPs).

Each VIP is associated with an Exascale block store worker (BSW) and provides a reliable network endpoint for iSCSI initiators. If the BSW instance fails, then the VIP, and the workload it supports, dynamically relocates to a working BSW instance.

Each BSW instance can support multiple networks. If the iSCSI initiators reside outside the Exadata environment, then they must connect using the Exadata client network. However, if the iSCSI initiators are located on the Exadata compute nodes, then you can leverage the high-speed low-latency storage network.

For the sake of load balancing, Oracle recommends using four VIPs for each network on every BSW instance. Using this arrangement, if the BSW instance fails, then the VIPs, and their workload, are redistributed across the working instances.

For example, consider the situation where the iSCSI initiators exclusively reside outside the Exadata environment and you have an Exascale cluster with three BSW instances. In that case, Oracle recommends that you configure 12 block store VIPs located on the Exadata client network.

The Exascale system administrator usually configures all of the block store VIPs. After configuration, the VIPs are dynamically allocated to the BSW instances. There are no ongoing VIP management tasks. If required, an administrator can delete a VIP.

Related Topics

- Administer Exascale Block Store System Objects

1.2.20.7 Oracle Advanced Cluster File System (ACFS) on Exascale

Exascale contains integrated support for Oracle Advanced Cluster File System (ACFS) on Exascale block storage using Exascale Direct Volumes (EDV).

By using this facility, an Exascale administrator can quickly and easily create and manage ACFS files systems on Exascale block storage.

For example, using one simple command an Exascale administrator can:

1. Create an ACFS file system on the specified EDV block storage volume.
2. Register and mount the file system on all servers associated with the EDV attachment.

   If the EDV attachment is a cluster-wide attachment, the ACFS file system is mounted on every node in the Oracle Grid Infrastructure (GI) cluster. If the EDV attachment is a node-specific attachment, the file system is mounted only on that node. In all cases, the ACFS details are registered with the GI cluster, and the file system is automatically mounted (or remounted) by Oracle Clusterware as required.

Related Topics

- Administer an Advanced Cluster File System (ACFS) on Exascale
- About Oracle ACFS

1.2.20.8 Virtual Machine Images

Exadata systems configured with Exascale can use Exascale Direct Volumes (EDV) to store virtual machine (VM) guest image files for Exadata database server VMs.

This capability removes previous constraints imposed by the limited amount of local storage space available on each Exadata database server.

Furthermore, VM images on Exascale are easily accessible from any VM host. This capability effectively decouples VM guests and hosts, providing the infrastructure to enable quick and easy migration of a VM guest to another host.

Decoupling VM guests and hosts also opens up additional possibilities for:

- Balancing database server VM workloads by separating busy VMs onto different hosts.
- Reducing the impact of scheduled VM host maintenance by proactively moving VMs to another host.
- Reducing the impact of unscheduled VM host downtime by reactively moving VMs to another host.

1.2.21 Features

Exascale contains infrastructure to track and manage software features throughout the Exascale cluster, which is used automatically by Exascale software to ensure compatibility between all services across the Exascale cluster.

For example, consider the possible introduction of a new security protocol for communications between Exascale software services across different servers in an Exascale cluster. Clearly, communications would break if one server started using the new communications protocol before all servers were updated with the capability to understand it. By using the infrastructure to track and manage software features, the Exascale software can be updated to facilitate the new protocol, but the feature can remain disabled until the entire cluster is ready to support it.

The infrastructure to track and manage Exascale software features is mostly internal, but Exascale administrators can view the metadata associated with software features. Additionally, administrators have some manual controls, such as the ability to manually enable a disabled feature, but these controls are typically reserved for use under the guidance of Oracle Support.

2

Oracle Exadata Exascale System Administration

This chapter covers administration tasks that define and manage an Exascale system.

Administration activities for Oracle Exadata Exascale are logically divided into two groups. System administration activities define and manage the Exascale system components that are shared across the system. User-specific administration activities define and manage Exascale storage objects and metadata that belong to a specific Exascale user.

This chapter contains the following system administration topics:

2.1 Configure Exascale

2.1.1 Configure Exascale Using the OEDA Web User Interface

You can use the Oracle Exadata Deployment Assistant (OEDA) Web user interface to initially configure Exascale on a fresh Exadata deployment, or to add Exascale alongside an existing Exadata deployment.

To initially configure Exascale on a fresh Exadata deployment:

1. Use the OEDA Web user interface to create a new Exadata XML configuration file (`es.xml`) that includes configuration details for an Exascale cluster and Exascale-enabled Oracle Database and Oracle Grid Infrastructure installations.

   See OEDA Web User Interface Settings and Options for Exascale.

2. Use the OEDA deployment utility (`install.sh`) to configure the fresh Exadata deployment as specified in the Exadata XML configuration file (`es.xml`).

To add Exascale alongside an existing Exadata deployment:

1. Ensure that the existing Exadata system configuration has free space in the storage servers to accommodate Exascale. The total amount of required free space is governed by your projected use of Exascale.

If required, reconfigure the existing ASM disk groups and Exadata grid disks to ensure that every Exadata cell disk contains the same amount of space available for Exascale to use. Also, ensure that any adjustments to the storage configuration are reflected in the Exadata XML configuration file (`es.xml`).

2. Start the OEDA Web user interface and import the Exadata XML configuration file (`es.xml`) from your existing Exadata deployment.
3. Use the OEDA Web user interface to modify the Exadata configuration as follows:

   See also OEDA Web User Interface Settings and Options for Exascale.

   a. On the OEDA Select Hardware page, click the check box to **Enable Exascale**.
   b. Use the Exascale page to define an Exascale cluster.
   c. Use the Define Clusters page to add new Exascale-enabled VM clusters alongside your existing ASM-based VM clusters.
   d. Use the Create Database page to add new databases that use Exascale storage.
   e. Generate the updated Exadata XML configuration files.
4. Use the OEDA deployment utility (`install.sh`) in conjunction with the cluster-specific XML configuration files for the new Exascale-enabled VM clusters.

- OEDA Web User Interface Settings and Options for Exascale
- Exascale Post Deployment Settings

2.1.1.1 OEDA Web User Interface Settings and Options for Exascale

The following list describes the Exascale-specific settings and options, which are spread across different pages in the OEDA Web user interface.

- **Select Hardware page**

  Exascale configuration commences on the OEDA Select Hardware page. For each individual Exadata Database Machine, you must click the check box to **Enable Exascale**. Otherwise, the following configuration options are not available.

- **Exascale page**

  After you select the option to **Enable Exascale** and apply the settings in the Select Hardware page, OEDA includes the Exascale page, which you use to define the Exascale clusters.

  Use the Exascale page to specify the following settings and options:

  – **Cluster Name**: Specify the name used to identify the Exascale cluster.
  – **ERS IP Address**: Specify an IP address for Exascale control services (also known as Exascale RESTful Services or ERS).

    The specified IP address should reside in the Exadata administration network (also known as the management network).

    The specified IP address is associated with a highly-available virtual IP (VIP) network interface, which provides a consistent network end point for Exascale control services. The VIP is hosted by one of the Exadata storage servers, which also runs front-end ERS processes. If the storage server or ERS instance becomes unavailable, then the VIP moves to another server containing front-end ERS processes.
  – **ERS Host Name**: Specify an host name for Exascale control services (also known as Exascale RESTful Services or ERS).

The specified host name is associated with the **ERS IP Address**.

- **Available nodes/Selected nodes**: Specify the Exadata storage servers that become members of the Exascale cluster.

  The specified cells must all be of the same Exadata storage server type. That is, they must all be High Capacity (HC) or Extreme Flash (EF).

- **Storage Pool Name**: Specify the name for the storage pool.

  You can only define one storage pool for each Exascale cluster using the OEDA Web user interface.

> **Note:**
>
> The media type for the storage pool matches the underlying Exadata storage server type. For example, if the Exadata system uses High Capacity (HC) storage cells, then the storage pool media type will be `HC`.

- **Storage Pool Size (%/GB/TB)**: Specify the amount of storage space to provision in the storage pool.

  Specify a percentage of the available space or an amount of storage space in gigabytes (GB) or terabytes (TB).

On the Exascale page, the **Advanced** button displays a window containing:

- **Enable ERS Network ID**: This option adds a field to specify the ERS Network ID value associated with the Exascale cluster. The ERS Network ID is used to maintain the highly-available virtual IP (VIP) network interface that provides a consistent network end point for Exascale control services (ERS).

> **Note:**
>
> This option is not required when deploying only one Exascale cluster in the subnet supporting the ERS IP address.
>
> However, when multiple Exascale clusters have ERS IP addresses that share the same subnet, each Exascale cluster requires a unique ERS Network ID.
>
> By default, OEDA attempts to allocate a unique ERS Network ID to each Exascale cluster. However, this option enables users to provide a specific ERS Network ID value.

- **Define Clusters page**

  The Define Clusters page configures the Oracle Grid Infrastructure (GI) clusters on the Exadata Database Machine.

  To enable Exascale storage on a GI cluster, you must use the **Exascale clusters** list to select an Exascale cluster that you defined previously on the Exascale page. If you do not select an Exascale cluster, then Oracle Automatic Storage Management (ASM) is used for the GI cluster and all of the databases that it contains.

  Enabling Exascale storage on a GI cluster implicitly creates an Exascale vault, which is used to store the Oracle Cluster Registry (OCR) and voting files. Consequently, you must also specify:

  - **Vault Name**: Specify a name to identify the Exascale vault.
  - **Vault Size**: Specify the amount of storage to provision for the vault.

  Additionally, if the system is configured to use Virtual Machines (VMs) for the database servers, you are presented with the option to **Use Exascale for VM file storage**. If you select the check box to enable this option, Exascale is used to store the guest image files for Exadata database server VMs.

  Using Exascale to store VM images removes previous constraints imposed by the limited amount of local storage space available on each physical Exadata database server. Furthermore, VM images on Exascale are easily accessible from any VM host. This capability effectively decouples VM guests and hosts, providing the infrastructure to enable quick and easy migration of a VM guest to another host.

  All other settings and options on the Define Clusters page are unaffected by Exascale.

- **Create Database page**

  The Create Database page is used to create Oracle databases on the Exadata Database Machine.

  To enable Exascale storage for a database, you can:

  - Use the **Available vaults for this database** list to select a previously defined Exascale vault.
  - Click **Add Vault**, and specify the **Vault Name** and **Vault Size**.

  All other settings and options on the Create Database page are unaffected by Exascale.

2.1.1.2 Exascale Post Deployment Settings

This section describes the Exascale objects and settings left after deployment of an Exascale configuration created with the OEDA Web user interface.

- Exascale Storage Configuration
- Exascale User Accounts and Wallets

2.1.1.2.1 Exascale Storage Configuration

This topic describes the Exascale storage objects created during system deployment.

During system deployment, one Exascale storage pool is created for each Exascale cluster defined on the OEDA Web UI Exascale page:

- The storage pool uses the **Storage Pool Name** and sizing details (**Storage Pool Size** and **Storage Pool Size Type**) specified on the Exascale page.
- The storage pool spans the Exadata storage servers specified in the **Selected Cells** list.
- The storage pool media type is derived from the type of Exadata storage servers in use. For example, an `HC` storage pool is created on High Capacity (HC) Exadata storage servers.
- To support the storage pool, an Exascale pool disk is created on each primary storage device in the Exadata storage servers. Each pool disk is uniformly sized as a fraction of the overall storage pool size specified on the Exascale page.

Additionally, Exascale vaults are created according to definitions specified in the OEDA Web UI on the Define Clusters page and the Create Database page.

To use Exascale storage, each compute cluster (database server cluster or VM cluster) in the Define Clusters page is associated with an Exascale cluster. An Exascale cluster can support multiple compute clusters. However, each compute cluster requires a separate vault to store the Oracle Grid Infrastructure shared clusterware files (Oracle Cluster Registry and voting disks). Consequently, one vault is created for each Exascale-enabled compute cluster.

Then, for each database that you define on an Exascale-enabled compute cluster, you can use the **Available Vaults** list on the Create Database page to place the Oracle Database files (data files, control files, log files, and so on) on a previously defined vault. Or, you can use the **Add new Vault** button to define a new vault to house the Oracle Database files.

During deployment, each vault that is created uses the **Vault Name** specified in the OEDA Web UI. The corresponding **Vault Size** is applied to the vault space provisioning attribute associated with the underlying storage pool media type. For example, if the vault is created on an Exascale cluster containing an `HC` storage pool, the `spaceProvHC` vault attribute is set to **Vault Size**.

2.1.1.2.2 Exascale User Accounts and Wallets

Exascale has a system of user accounts enabling different users to perform actions and access data according to their assigned privileges. This topic describes the Exascale user accounts and associated key stores (wallets) created during system deployment.

After system deployment, the number of Exascale user accounts, and their settings, depends on the configuration details specified on the OEDA Web UI User(s) and Groups page and the clusters defined on the Define Clusters page.

The following occurs for each virtual machine (VM) cluster or bare-metal database server cluster on the Define Clusters page:

- If the cluster is associated with a default (non-role-separated) user configuration containing one Oracle OS user account, then one matching Exascale user account is created.

  In this case, the Exascale user manages the Oracle Grid Infrastructure shared clusterware files (Oracle Cluster Registry and voting disks) and the Oracle Database files (data files, control files, log files, and so on) for all databases associated with the cluster. Consequently, the Exascale user is the owner and manager of all Exascale vaults associated with the cluster.

  The Exascale user identifier (ID) is set to a concatenation of:

  – The **User Name** (typically `oracle`) as specified on the User(s) and Groups page.

  – The value of the **Cluster Name** as specified on the Define Clusters page.

  For example, if the **User Name** is `oracle` and the **Cluster Name** is `Escluster1`, then the Exascale user ID is `oracleEscluster1`.

  The Exascale user account uses a system-generated public and private key pair for authentication. The Exascale user account definition contains the public key, and each compute node (database server or VM) contains a system-generated key store (wallet), which contains the corresponding private key. The wallet is created at `/etc/oracle/cell/network-config/eswallet`.

- If the cluster is associated with a role-separated user configuration containing one Grid OS user account and one Oracle OS user account, then two matching Exascale user accounts are created.

  The Exascale user that matches the Grid OS user manages the Oracle Grid Infrastructure shared clusterware files (Oracle Cluster Registry and voting disks). Also known as the grid Exascale user, this Exascale user account has the following characteristics:

  - The user ID is set to a concatenation of:
    * The **User Name** for the Grid OS user (typically `grid`) as specified on the User(s) and Groups page.
    * The value of the **Cluster Name** as specified on the Define Clusters page.

    For example, if the **User Name** is `grid` and the **Cluster Name** is `Escluster2`, then the Exascale user ID is `gridEscluster2`.
  - The Exascale user account uses a system-generated public and private key pair for authentication. The Exascale user account definition contains the public key, and each compute node (database server or VM) contains a system-generated key store (wallet), which contains the corresponding private key. The wallet for the grid Exascale user account is created at `/etc/oracle/cell/network-config/eswallet`.
  - The Exascale grid user is the owner and manager of the Exascale vault specified on the Define Clusters page.

  The Exascale user that matches the Oracle OS user manages the Oracle Database files (data files, control files, log files, and so on) for all databases associated with the cluster. Also known as the oracle Exascale user, this Exascale user account has the following characteristics:

  - The user ID is set to a concatenation of:
    * The **User Name** for the Oracle OS user (typically `oracle`) as specified on the User(s) and Groups page.
    * The value of the **Cluster Name** as specified on the Define Clusters page.

    For example, if the **User Name** is `oracle` and the **Cluster Name** is `Escluster2`, then the Exascale user ID is `oracleEscluster2`.
  - The Exascale user account uses a system-generated public and private key pair for authentication. The Exascale user account definition contains the public key, and each compute node (database server or VM) contains a system-generated key store (wallet), which contains the corresponding private key. The wallet for the Exascale user account that corresponds with the Oracle OS user is created at `$ORACLE_BASE/admin/eswallet` (typically `/u01/app/oracle/admin/eswallet`).
  - The Exascale oracle user is the owner and manager of the database-specific Exascale vaults specified on the Create Database page.

    However, if any database uses the vault that contains the Oracle Grid Infrastructure shared clusterware files, then the Exascale oracle user is added as a manager of that vault.

In addition to the aforementioned Exascale user accounts, each Exascale cluster contains:

- One superuser account. The user ID for the superuser account is `admin`. The `admin` user can implicitly perform any system operation and effectively holds all system privileges.

  During system deployment, the `admin` user wallet is created on every storage server at `/opt/oracle/cell/cellsrv/deploy/config/security/admwallet` and every wallet contains the same system-generated private key.
- One node administration account for every node (storage server or compute node) that runs Exascale software services. Each node administration account inherits its user ID from the server hostname and each account contains the privileges required to run the Exascale software services on the node. Do not directly use or modify these accounts.

  During system deployment, the node administration user wallet is created on every node, and every wallet contains a system-generated private key. On a storage server, the wallet

is located at `/opt/oracle/cell/cellsrv/deploy/config/eswallet`. On a compute node, the wallet is located at `/opt/oracle/dbserver/dbms/deploy/config/eswallet`.

2.2 Administer the Exascale Cluster

- Administer Exascale Services
- Modify the Exascale Cluster
- Stop the Exascale Cluster
- Add a Storage Server

2.2.1 Administer Exascale Services

- About Automatic Service Placement
- List Services
- Alter the Number of Service Instances
- Alter the Placement of Services
- Restart a Service
- Stop a Service
- Start a Service
- Disable a Service
- Administer Client Services

Related Topics

- Exascale Services

2.2.1.1 About Automatic Service Placement

Exascale system components are implemented using a series of clustered software services. This topic describes the default automatic placement of the Exascale storage services.

By default, Exascale uses automatic service placement to ensure that the required Exascale storage services are available across the Exascale cluster. If Exascale detects that a storage service is not running, the service instance is automatically restarted on another node.

The following outlines the default automatic service placement methodology for Exascale storage services:

- **Cluster Services (EGS)**

  Every Exascale cluster contains five EGS instances.

  For Exascale clusters with five or more Exadata storage servers, one EGS instance runs on each of the first five storage servers.

  For Exascale configurations with fewer than five storage servers, one EGS instance runs on each Exadata storage server, and the remaining EGS instances run on the Exadata compute nodes to make up the required total of five. In a bare-metal configuration, EGS compute node instances run on the server operating system. In a configuration with compute nodes running on virtual machines (VMs), EGS compute node instances run in the hypervisor.

- **Control Services (ERS)**

For Exascale clusters with five or more Exadata storage servers, a total of five ERS instances are distributed across the Exadata storage servers. The first three ERS instances contain front-end and back-end processes, and the final two instances only contain back-end processes.

For Exascale configurations with fewer than five storage servers, one ERS instance runs on each storage server. The first three ERS instances contain front-end and back-end processes, and the remaining instances only contain back-end processes.

- **System Vault Manager (SYSEDS) and User Vault Manager (USREDS)**

  For Exascale clusters with five or more Exadata storage servers, a total of five SYSEDS instances and five USREDS instances are distributed across the Exadata storage servers.

  For Exascale configurations with fewer than five storage servers, one SYSEDS instance and one USREDS instance runs on each storage server.

- **Block Store Manager (BSM) and Block Store Worker (BSW)**

  By default, a total of three BSM instances and three BSW instances are distributed across the Exadata storage servers.

In all cases, the Exascale storage service instances are automatically distributed across the available storage servers with the aim of evenly distributing the overall workload.

In addition to the Exascale storage services governed by automatic service placement, every storage server that is associated with the Exascale cluster also contains the Instance Failure Detection (IFD) service and the core Exadata cell services: Cell Server (CELLSRV), Management Server (MS), and Restart Server (RS).

2.2.1.2 List Services

Exascale system components are implemented using a series of software services.

- To display information about Exascale storage services running across the Exascale cluster, including Exascale cluster services (EGS), control services (ERS), vault manager services (SYSEDS and USREDS), and block store services (BSM and BSW), use the ESCLI lsservice command. For example:

  ```
  @> lsservice --detail
  ```

  Using the various command options available with the lsservice command, you can view output with different levels of detail and specify the services you want to display.

- To display information about client-side Exascale services, including Exascale Node Proxy (ESNP) and Exascale Direct Volume (EDV), use the DBMCLI `LIST DBSERVER` command, as follows:

  ```
  DBMCLI> list dbserver detail
  ```

  The command output displays status information about all of the client-side services running on the Exadata compute node. You must run the command separately on each compute node.

2.2.1.3 Alter the Number of Service Instances

Exascale system components are implemented using a series of software services.

By default, each Exascale cluster is configured with a number of storage service instances that is usually sufficient for most workloads. However, as an Exascale administrator, you can alter the number of service instances that are deployed.

To alter the number of storage service instances deployed in the Exascale cluster, use the ESCLI chcluster command as follows:

```
@> chcluster --attributes attribute_name=value
```

In the command, specify one of the following as the *attribute_name*:

- `numControlServicesFrontends`: The number of Control Service (ERS) instances that contain front-end and back-end processes.
- `numControlServicesBackends`: The number of Control Service (ERS) instances that contain only back-end processes.
- `numSystemVaultManagers`: The number of System Vault Manager (SYSEDS) instances.
- `numUserVaultManagers`: The number of User Vault Manager (USREDS) instances.
- `numVolumeManagers`: The number of Block Store Manager (BSM) instances.
- `numStorageVolumeWorkers`: The number of Block Store Worker (BSW) instances.

> **Note:**
>
> This setting controls only the number of BSW instances deployed across the Exadata storage servers.
>
> Additional BSW instances can be manually deployed on the Exadata compute nodes. This location enables the BSW instance to access the Exadata client network, which may be used to run volume backups to Oracle Cloud Infrastructure (OCI) object storage and to service iSCSI initiators external to Exadata.

> **Note:**
>
> You cannot alter the number of Exascale cluster services (EGS). EGS always runs on the first five nodes in the Exascale cluster.

Related Topics

- About Automatic Service Placement

2.2.1.4 Alter the Placement of Services

Exascale system components are implemented using a series of software services.

By default, the storage service instances are distributed across the Exascale cluster with the aim of evenly distributing the overall workload. However, as an Exascale administrator, you can alter the placement of the services to optimize the workload balance.

To alter the placement of Exascale storage services, you can start a service on a node where it is not running and then disable the same service on another node where it is running.

For example, you can use the following ESCLI command sequence to alter the placement of block store worker service (BSW) by starting the service on `cell09` and disabling it on `cell03`.

```
@> chservice --attributes name=bsw_cell09 --start
@> chservice --attributes name=bsw_cell03 --disable
```

> **Note:**
>
> You cannot alter the placement of Exascale cluster services (EGS). EGS always runs on the first five nodes in the Exascale cluster.

Related Topics

- About Automatic Service Placement
- Start a Service
- Disable a Service

2.2.1.5 Restart a Service

Exascale system components are implemented using a series of software services.

Typically, the Exascale cluster automatically contains the required Exascale storage services. However, as an Exascale administrator, you may occasionally need to restart a software service instance.

To restart an Exascale storage service, use the ESCLI chservice command with the `--restart` option.

If you issue a command to restart a stopped or disabled service, then the command starts the service and completes successfully.

- To restart an Exascale storage service on an Exadata storage server, the command syntax is:

  ```
  @> chservice --attributes name=service_cell[,frontend={true|false}] --
  restart
  ```

  To identify the service instance you want to restart on an Exadata storage server, specify the `name` attribute using the format: `name=service_cell`.

  In the attribute value:

  – *service*: Specifies the service that you want to act on, which is one of the following:

    * `cellsrv`: Specifies the core Exadata cell services (CELLSRV, MS, and RS).
    * `egs`: Specifies the Exascale cluster service, also known as Exascale global service (EGS).
    * `ers`: Specifies the Exascale control service, also known as Exascale RESTful service (ERS).
    * `syseds`: Specifies the system vault manager service.
    * `usreds`: Specifies the user vault manager service.

    - `bsm`: Specifies the block store manager service.
    - `bsw`: Specifies the block store worker service.
  - *cell*: Specifies the name of the storage server (`cell`) that hosts the service.

    The cell must be identified by its name. Use the lscell command to find all cell names.

  Additionally, when you restart ERS only, you can optionally set the `frontend` attribute to control the front-end server processes on the ERS instance. To enable the front-end server processes on the ERS instance, specify `frontend=true`. To disable the front-end server processes on the ERS instance, specify `frontend=false`. The `frontend` attribute only applies to ERS.

- To restart an Exascale storage service on an Exadata compute server, the command syntax is:

  ```
  @> chservice --attributes name=service_compute --restart
  ```

  To identify the service instance you want to restart on an Exadata compute server, specify the `name` attribute using the format: `name=service_compute`.

  In the attribute value:

  - *service*: Specifies the service that you want to act on, which is one of the following:
    - `egs`: Specifies the Exascale cluster service, also known as Exascale global service (EGS).
    - `bsw`: Specifies the block store worker service.
  - *compute*: Specifies the name of the compute server (`dbserver`) that hosts the service.

    Depending on the required service location, specify either the name of a specific compute node virtual machine (VM) or the Exadata compute node host domain (KVM host or Dom0).

2.2.1.6 Stop a Service

Exascale system components are implemented using a series of software services.

Typically, the Exascale cluster automatically contains the required Exascale storage services. However, as an Exascale administrator, you may occasionally need to stop a software service instance.

> **Note:**
>
> By default, Exascale uses automatic service placement to ensure that the required storage services are available across the Exascale cluster. Consequently, if you manually stop a software service instance (as described below), a replacement service instance is automatically started on another node (if possible).

To stop an Exascale storage service, use the ESCLI chservice command with the `--stop` option.

If you issue a command to stop service that is not running, then the command completes successfully but it has no effect.

- To stop an Exascale storage service on an Exadata storage server, the command syntax is:

  ```
  @> chservice --attributes name=service_cell --stop
  ```

  To identify the service instance you want to stop on an Exadata storage server, specify the `name` attribute using the format: `name=service_cell`.

  In the attribute value:

  – *service*: Specifies the service that you want to act on, which is one of the following:

    * `cellsrv`: Specifies the core Exadata cell services (CELLSRV, MS, and RS).
    * `egs`: Specifies the Exascale cluster service, also known as Exascale global service (EGS).
    * `ers`: Specifies the Exascale control service, also known as Exascale RESTful service (ERS).
    * `syseds`: Specifies the system vault manager service.
    * `usreds`: Specifies the user vault manager service.
    * `bsm`: Specifies the block store manager service.
    * `bsw`: Specifies the block store worker service.

  – *cell*: Specifies the name of the storage server (`cell`) that hosts the service.

    The cell must be identified by its name. Use the lscell command to find all cell names.

- To stop an Exascale storage service on an Exadata compute server, the command syntax is:

  ```
  @> chservice --attributes name=service_compute --stop
  ```

  To identify the service instance you want to stop on an Exadata compute server, specify the `name` attribute using the format: `name=service_compute`.

  In the attribute value:

  – *service*: Specifies the service that you want to act on, which is one of the following:

    * `egs`: Specifies the Exascale cluster service, also known as Exascale global service (EGS).
    * `bsw`: Specifies the block store worker service.

  – *compute*: Specifies the name of the compute server (`dbserver`) that hosts the service.

    Depending on the required service location, specify either the name of a specific compute node virtual machine (VM) or the Exadata compute node host domain (KVM host or Dom0).

2.2.1.7 Start a Service

Exascale system components are implemented using a series of software services.

Typically, the Exascale cluster automatically contains the required Exascale storage services. However, as an Exascale administrator, you may occasionally need to start a software service instance.

To start an Exascale storage service, use the ESCLI chservice command with the `--start` option.

If you issue a command to start an already running service, then the command completes successfully but it has no effect.

- To start an Exascale storage service on an Exadata storage server, the command syntax is:

  ```
  @> chservice --attributes name=service_cell --start
  ```

  To identify the service instance you want to start on an Exadata storage server, specify the `name` attribute using the format: `name=service_cell`.

  In the attribute value:

  – *service*: Specifies the service that you want to act on, which is one of the following:

    * `cellsrv`: Specifies the core Exadata cell services (CELLSRV, MS, and RS).
    * `egs`: Specifies the Exascale cluster service, also known as Exascale global service (EGS).
    * `ers`: Specifies the Exascale control service, also known as Exascale RESTful service (ERS).
    * `syseds`: Specifies the system vault manager service.
    * `usreds`: Specifies the user vault manager service.
    * `bsm`: Specifies the block store manager service.
    * `bsw`: Specifies the block store worker service.

  – *cell*: Specifies the name of the storage server (`cell`) that hosts the service.

    The cell must be identified by its name. Use the lscell command to find all cell names.

- To start an Exascale storage service on an Exadata compute server, the command syntax is:

  ```
  @> chservice --attributes name=service_compute --start
  ```

  To identify the service instance you want to start on an Exadata compute server, specify the `name` attribute using the format: `name=service_compute`.

  In the attribute value:

  – *service*: Specifies the service that you want to act on, which is one of the following:

    * `egs`: Specifies the Exascale cluster service, also known as Exascale global service (EGS).
    * `bsw`: Specifies the block store worker service.

  – *compute*: Specifies the name of the compute server (`dbserver`) that hosts the service.

    Depending on the required service location, specify either the name of a specific compute node virtual machine (VM) or the Exadata compute node host domain (KVM host or Dom0).

2.2.1.8 Disable a Service

Exascale system components are implemented using a series of software services.

Typically, the Exascale cluster automatically contains the required Exascale storage services. However, as an Exascale administrator, you may occasionally need to disable a software service instance.

To disable an Exascale storage service, use the ESCLI chservice command with the `--disable` option.

A disabled service instance is stopped and prevented from automatically restarting on the associated server. To enable a disabled service instance, you must start it manually.

- To disable an Exascale storage service on an Exadata storage server, the command syntax is:

  ```
  @> chservice --attributes name=service_cell --disable
  ```

  To identify the service instance you want to disable on an Exadata storage server, specify the `name` attribute using the format: `name=service_cell`.

  In the attribute value:

  – *service*: Specifies the service that you want to act on, which is one of the following:

    * `egs`: Specifies the Exascale cluster service, also known as Exascale global service (EGS).
    * `ers`: Specifies the Exascale control service, also known as Exascale RESTful service (ERS).
    * `syseds`: Specifies the system vault manager service.
    * `usreds`: Specifies the user vault manager service.
    * `bsm`: Specifies the block store manager service.
    * `bsw`: Specifies the block store worker service.

  – *cell*: Specifies the name of the storage server (`cell`) that hosts the service.

    The cell must be identified by its name. Use the lscell command to find all cell names.

- To disable an Exascale storage service on an Exadata compute server, the command syntax is:

  ```
  @> chservice --attributes name=service_compute --disable
  ```

  To identify the service instance you want to disable on an Exadata compute server, specify the `name` attribute using the format: `name=service_compute`.

  In the attribute value:

  – *service*: Specifies the service that you want to act on, which is one of the following:

    * `egs`: Specifies the Exascale cluster service, also known as Exascale global service (EGS).
    * `bsw`: Specifies the block store worker service.

  – *compute*: Specifies the name of the compute server (`dbserver`) that hosts the service.

Depending on the required service location, specify either the name of a specific compute node virtual machine (VM) or the Exadata compute node host domain (KVM host or Dom0).

You can also add the `--force` option to forcibly disable the service, even if the server hosting the service is unavailable.

2.2.1.9 Administer Client Services

In addition to the Exascale storage services, further client-side services provide support for specific Exascale functions on the Exadata compute nodes.

By default, the following client-side service instances are configured on each Exascale compute node:

- **Exascale Node Proxy (ESNP)**

  In a bare-metal configuration, ESNP runs on the server operating system. In a configuration with compute nodes running on virtual machines (VMs), ESNP runs inside the guest VMs.

- **Exascale Direct Volume (EDV)**

  This service is required on all Exadata compute nodes where you want to use EDV devices. In a bare-metal configuration, the EDV service runs on the server operating system. In a configuration with compute nodes running on virtual machines (VMs), the EDV service runs in the KVM host to provide the hypervisor access to VM image files based on EDV-managed storage. The EDV service also runs in each guest VM to enable direct access to EDV-managed storage from inside the VM.

To administer the client-side services, use the `ALTER DBSERVER` command in the Oracle Exadata Database Machine Command-Line Interface (DBMCLI).

The relevant command syntax is:

```
DBMCLI> ALTER DBSERVER { STARTUP | SHUTDOWN | RESTART } SERVICES { ESNP |
EDV }
```

To display the status of the client-side services, use the DBMCLI `LIST DBSERVER` command.

Related Topics

- List Services

2.2.2 Modify the Exascale Cluster

Exascale system components are implemented using a series of clustered software services. You can control various aspects of the Exascale cluster by modifying the cluster attributes.

To modify attributes of the Exascale cluster, use the ESCLI chcluster command and specify one or more attribute settings. For example:

```
@> chcluster --attributes alterServiceTimeoutInSecs=60
```

2.2.3 Stop the Exascale Cluster

Exascale system components are implemented using a series of clustered software services.

To stop all of the Exascale software services in the Exascale cluster, use the ESCLI chcluster command with the `--shutdown` option.

However, to shut down the Exascale cluster, note that you must:

- Run ESCLI as the Exascale `admin` user or as another user with the `cl_admin` cluster level storage privilege.
- Connect ESCLI directly to an online Exascale control services (ERS) back-end server process. To do this you must start ESCLI and specify the `--ctrl` option as follows:

  ```
  $ escli --wallet admin-wallet-location --ctrl ERS-server-IP:8080
  ```

  To find the IP address for an online ERS server (*ERS-server-IP*), start a regular ESCLI session and use the following command:

  ```
  @> lsservice --filter serviceType=controlServices,status=ONLINE --
  attributes url
  ```

  The command displays the private IP addresses associated with all of the online ERS servers. You can use any of the reported IP addresses.

For example:

```
$ escli --wallet admin-wallet-location --ctrl ERS-server-IP:8080 chcluster --
shutdown
```

2.2.4 Add a Storage Server

Adding an Exadata storage server to an existing Exascale cluster enables you to:

- Add storage space to the Exascale storage pools.
- Run Exascale cluster services on the Exadata storage server.

To add a storage server to an existing cluster use the Oracle Exadata Deployment Assistant (OEDA) command line interface (OEDACLI):

1. Load the OEDA XML system configuration file.

   ```
   oedacli> LOAD FILE NAME=es.xml
   ```

2. Clone an existing storage server to add a storage server to the Exascale cluster definition.

   For example:

   ```
   oedacli> CLONE NEWCELL SRCNAME=cell07.example.com
   TGTNAME=cell10.example.com
   oedacli> SET ADMINNET NAME=cell10.example.com IP=192.0.2.21
   NETMASK=255.255.248.0 GATEWAY=192.0.2.1
   oedacli> SET PRIVNET NAME1=cell10-priv1.example.com IP1=192.168.0.27
   NAME2=cell10-priv2.example.com IP2=192.168.0.28
   oedacli> SET ILOMNET NAME=cell10-ilom.example.com IP=198.51.100.110
   oedacli> SET RACK NUM=1, ULOC=8
   oedacli> SAVE ACTION FORCE
   ```

In the example:

- The name of the existing storage server being cloned is *cell07.example.com*.
- The name of the new storage server being added is *cell10.example.com*.
- The host names, network addresses, and rack details are all examples and you must substitute your own valid values.

3. Save the OEDA XML system configuration file.

```
oedacli> SAVE FILE NAME=es.xml
```

4. Alter the cluster containing the storage servers and deploy the changes.

   For example:

```
oedacli> ALTER CLUSTER ADDCELLS='cell10.example.com' WHERE CLUSTERNUMBER=1
oedacli> SAVE ACTION
oedacli> MERGE ACTIONS
oedacli> DEPLOY ACTIONS
```

   In the example, the example cluster number is *1* and you must substitute your own valid value.

2.3 Administer Exascale Users

- Create a User
- List User Information
- Modify a User
- Remove a User
- Administer the Internal User Accounts

Related Topics

- Exascale Users
- Exascale User Credentials

2.3.1 Create a User

By using different users, you can enforce data separation within Exascale.

Prior to performing the following procedure, the prospective new Exascale user should create their own public/private key pair and supply the public key. See Create User Keys. This is recommended to ensure the integrity of the private key, since the user should never share the private key, not even with the Exascale administrator.

To provision a new Exascale user:

1. Create the Exascale user.

   Use the ESCLI mkuser command and specify the name of the new Exascale user and a unique user ID.

For example:

```
@> mkuser theusername --attributes id=theuserID
User created with ID: theuserID
```

If you do not specify the user ID, then the user is assigned as system-generated unique user ID. For example:

```
@> mkuser theusername
User created with ID: 96a68014-5762-4579-86ee-29eb743decbd
```

> **Note:**
>
> The user ID is fixed at user creation time. There is no way to modify the user ID afterward.

Take note of the user ID for the newly created user. You must supply the user ID value to the actual user so that they can configure their credential store (wallet).

2. Associate the new Exascale user with the user's supplied public key.

   Use the ESCLI chuser command and specify:

   - The ID for the user that is being modified in the `chuser` command.
   - The location of the file that contains the user's public key in PEM format.

   For example:

```
@> chuser 96a68014-5762-4579-86ee-29eb743decbd --public-key-file1 pub.pem
```

2.3.2 List User Information

This topic describes how to display information about Exascale users.

- To display a complete list of Exascale user names and corresponding user IDs, use the ESCLI lsuser command with no options. For example:

```
@> lsuser
```

- To display detailed information about a specific Exascale user, use the lsuser command with the `--detail` option and specify the ID for the user that is being queried. For example:

```
@> lsuser theuserID --detail
```

Using the command options available with the lsuser command, you can specify the user attributes you want to display and display the output in a long, tabular form.

2.3.3 Modify a User

You can modify attributes for an existing Exascale user by using the ESCLI chuser command. You can:

- Modify a User Name
- Modify User Privileges
- Manage a User's Public Keys

2.3.3.1 Modify a User Name

To change the name of a user, use the ESCLI chuser command with the `--name` option and specify:

- The unique user ID for the user that is being modified. You can use the lsuser command to find the ID for each user.
- The new name to apply to the user.

For example:

```
@> chuser theuserID --attributes name=theusername
```

2.3.3.2 Modify User Privileges

To change a user's privileges, use the ESCLI chuser command with the `--privilege` option and specify:

- The unique user ID for the user that is being modified. You can use the lsuser command to find the ID for each user.
- The list of privileges to apply to the user.

For example:

```
@> chuser theuserID --privilege vlt_read|egs
```

The `--privilege` option specifies a list of one or more privileges of the form *privilege-1|privilege-2|...*.

A privilege is one of the following: `no_privilege`, `cellsrv`, `egs`, `ers`, `syseds`, `usreds`, `bsm`, `bsw`, `ms`, `vlt_manage`, `vlt_use`, `vlt_read`, `vlt_inspect`, `cl_admin`, `cl_operator`, `cl_monitor`, `on_behalf_of`, `user_create`, `system_restore`. For descriptions, see User Privileges.

`no_privilege` cannot be combined with any other type of privileges, otherwise an error is returned.

Vault top-level privileges (`vlt_manage`, `vlt_use`, `vlt_read`, and `vlt_inspect`) are mutually exclusive. If any two or more are combined, an error is returned.

Cluster privileges (`cl_admin`, `cl_operator`, and `cl_monitor`) are mutually exclusive. If any two or more are combined, an error is returned.

2.3.3.3 Manage a User's Public Keys

This topic describes how to manage the public keys that are associated with an existing Exascale user.

- To associate a public key with a user, use the ESCLI chuser command with the `--public-key-file1`, `--public-key-file2`, or `--public-key-file3` option and specify:

  - The unique user ID for the user that is being modified. You can use the lsuser command to find the ID for each user.
  - The location of a file that contains the user's public key in PEM format.

  For example:

  ```
  @> chuser theuserID --public-key-file1 pub.pem
  ```

  By using the `--public-key-file1`, `--public-key-file2`, and `--public-key-file3` options, Exascale provides three public key 'slots' for each Exascale user.

  If you supply a public key for a previously unused slot, then the specified key is inserted into the slot. Otherwise, the existing key in the slot is overwritten with the specified public key.

- To remove a public key from a user, use the chuser command with the `--rm-public-key1`, `--rm-public-key2`, or `--rm-public-key3` option and specify the unique user ID for the user that is being modified.

  For example:

  ```
  @> chuser theuserID --rm-public-key1
  ```

  By using the `--rm-public-key1`, `--rm-public-key2`, and `--rm-public-key3` options, you can selectively remove the public key in any of the three public key 'slots' for the Exascale user.

2.3.4 Remove a User

To remove an existing Exascale user, use the ESCLI rmuser command and specify the ID for the user that is being removed. For example:

```
@> rmuser theuserID
```

You can use the lsuser command to find the ID for each user.

2.3.5 Administer the Internal User Accounts

- Administer the ADMIN User
- Administer the Node Administration Users

2.3.5.1 Administer the ADMIN User

By default, each Exascale cluster contains one superuser account. The user identifier (ID) for the superuser account is `admin`. The `admin` user can implicitly perform any system operation and effectively holds all system privileges.

During system deployment, the `admin` user wallet is created on every storage server at `/opt/oracle/cell/cellsrv/deploy/config/security/admwallet` and every wallet contains the same system-generated private key.

If you choose to use the `admin` user for ongoing system administration, then you must manage access to the `admin` user wallet.

Alternatively, Oracle recommends the following approach:

1. Use the `admin` user to create your own dedicated Exascale administrator account or accounts.
2. Extract the `admin` user private key from the wallet and store it in a secure off-site key store. You can extract the private key from a wallet by using the ESCLI lswallet command.
3. Remove all copies of the `admin` user wallet.

By using this approach, you effectively disable the `admin` user and you must recreate the wallet if you require future `admin` access.

2.3.5.2 Administer the Node Administration Users

Exascale contains one node administration account for every node (storage server or compute node) that runs Exascale software services. Each node administration account inherits its user identifier (ID) from the server hostname and each account contains the privileges required to run the Exascale software services on the node. Do not directly use or modify these accounts.

During system deployment, the node administration user wallet is created on every node, and every wallet contains a system-generated private key. On a storage server, the wallet is located at `/opt/oracle/cell/cellsrv/deploy/config/eswallet`. On a compute node, the wallet is located at `/opt/oracle/dbserver/dbms/deploy/config/eswallet`.

By default, each node is configured with appropriate operating system privileges to limit wallet access.

Additionally, Oracle recommends that you extract the private key from each wallet and store it in a secure off-site key store. By doing this, you have a backup of the private key, which you can use if you need to rebuild the wallet. You can extract the private key from a wallet by using the ESCLI lswallet command.

2.4 Administer Exascale Pool Disks

- Create Pool Disks
- List Pool Disks
- Modify the State of a Pool Disk
- Resize Pool Disks
- Drop Pool Disks

Related Topics

- Pool Disks

2.4.1 Create Pool Disks

A pool disk is a portion of an Exadata storage device that is designated for use by Exascale. A pool disk is created by allocating space from an Exadata cell disk.

To create pool disks, use the CellCLI `CREATE GRIDDISK` command and specify the name of the storage pool that you want to associate with the newly created pool disks.

For example:

```
CellCLI> create griddisk all harddisk prefix=pool1, storagepool=POOL1
```

You must specify the `storagepool` attribute to create Exascale pool disks. Otherwise, you will create Exadata grid disks, which Exascale cannot use.

When you specify the `storagepool` attribute, the Exadata grid disks are marked as Exascale pool disks, and the storage pool name is written into the pool disk metadata.

The `create griddisk` command does not immediately add the pool disks to the specified storage pool, and at this point, the storage pool does not need to exist. The pool disks are only available for data storage after they are included in the storage pool, either during storage pool creation or after the storage pool is modified by using the chstoragepool command with the `--reconfig` option.

Related Topics

- Create a Storage Pool
- Modify a Storage Pool

2.4.2 List Pool Disks

A pool disk is a portion of an Exadata storage device that is designated for use by Exascale.

To display information about pool disks, use the ESCLI lspooldisk command. For example:

```
@> lspooldisk
```

By default, without any optional arguments, the command displays basic information about all of the pool disks associated with the Exascale cluster.

Using the command options available with the ESCLI lspooldisk command, you can:

- Specify the pool disks that you want to display.
- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

2.4.3 Modify the State of a Pool Disk

This topic describes how to modify a pool disk and change its state; either online or offline.

A pool disk is a portion of an Exadata storage device that is designated for use by Exascale. You can modify a pool disk by setting its state to online or offline.

To modify the state of a pool disk, use the ESCLI chpooldisk command and specify:

- The identifier for the pool disk that you want to modify.

  You can use the ESCLI lspooldisk command to display the pool disk identifier for any Exascale pool disk.
- The desired pool disk state. Either `--offline` or `--online`.

For example:

```
@> chpooldisk 15 --offline
```

2.4.4 Resize Pool Disks

A pool disk is a portion of an Exadata storage device that is designated for use by Exascale.

When you resize pool disks that are allocated to a storage pool, you indirectly change the size of the storage pool. Depending on the situation, the change may trigger a rebalance operation within the storage pool.

To resize pool disks:

1. Modify the size attribute for the pool disks.

   Use the CellCLI `ALTER GRIDDISK` command and specify the pool disks that you want to resize along with a setting for the size attribute. For example:

   ```
   CellCLI> alter griddisk size=2000G where storagepool=POOL1
   ```

   You can increase or reduce the size of existing pool disks.

   Increasing the size of a pool disk is an almost instantaneous operation. But, you cannot increase the size of a pool disk beyond the available free space in the underlying cell disk.

   When reducing the size of a pool disk, existing allocations beyond the new pool disk boundary move into available locations inside the pool disk. The time required to complete the operation depends on the amount of data being moved. The operation fails if you attempt to reduce the size to the point where the pool disk cannot accommodate the existing data.

2. Reconfigure the storage pool to accommodate the resized pool disks.

   Use the ESCLI chstoragepool command. For example:

   ```
   @> chstoragepool POOL1 --reconfig
   ```

   If required, reconfiguring the storage pool triggers a rebalance operation.

2.4.5 Drop Pool Disks

A pool disk is a portion of an Exadata storage device that is designated for use by Exascale.

Exascale only supports dropping of pool disks for the purpose of shrinking a storage pool by removing all of the pool disks provided by a storage server. You cannot drop individual pool disks.

To drop the pool disks associated with a storage server:

1. Prepare the pool disks for removal.

   Use the CellCLI `DROP GRIDDISK` command with the `prepare` option. For example:

   ```
   CellCLI> drop griddisk all harddisk prefix=pool1 prepare
   ```

   This command marks the pool disks for removal and sets the pool disk status to `waiting for storage pool reconfig`.

> **Note:**
>
> If required, you can undo this step before you reconfigure the storage pool by using CellCLI `DROP GRIDDISK` command with the `cancel` option. For example:
>
> ```
> CellCLI> drop griddisk all harddisk prefix=pool1 cancel
> ```

2. Reconfigure the storage pool.

   Use the ESCLI chstoragepool command with the `--reconfig` option. For example:

   ```
   @> chstoragepool POOL1 --reconfig
   ```

   This command permanently removes the pool disks and rebalances the storage pool.

2.5 Administer Exascale Storage Pools

- Create a Storage Pool
- List Storage Pools
- Maintain Free Space in a Storage Pool to Protect Against Disk Failure
- Modify a Storage Pool
- Drop a Storage Pool

Related Topics

- Storage Pools

2.5.1 Create a Storage Pool

This topic describes how to create a new Exascale storage pool.

An Exascale storage pool is a collection of pool disks that provides persistent physical storage for Exascale vaults and files. Each storage pool is a collection of pool disks using only one type of storage media (for example, HC). You cannot define a storage pool with pool disks having a mixture of media types (for example, HC and EF).

Before you create a storage pool, you need some pool disks that are already associated with the storage pool name.

To create a storage pool, use the ESCLI mkstoragepool command and specify the name for the new storage pool.

The essential command syntax is:

```
@> mkstoragepool storagepool-name
```

Additionally, you can specify the `--nowait` option to instruct the command to return immediately instead of waiting until the storage pool is created.

Related Topics

- Create Pool Disks

2.5.2 List Storage Pools

An Exascale storage pool is a collection of pool disks that provides the persistent physical storage for Exascale vaults and files.

To display information about storage pools, use the ESCLI lsstoragepool command. For example:

```
@> lsstoragepool
```

By default, without any optional arguments, the command displays basic information about all of the storage pools associated with the Exascale cluster.

Using the command options available with the lsstoragepool command, you can:

- Specify the storage pools that you want to display.
- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

2.5.3 Maintain Free Space in a Storage Pool to Protect Against Disk Failure

An Exascale storage pool is a collection of pool disks that provides the persistent physical storage for Exascale vaults and files. If the storage device hosting one pool disk fails, then the storage pool can continue to function with reduced redundancy. However, the storage pool may go offline if multiple pool disks fail at the same time.

To maintain maximum protection against storage failures, you should maintain sufficient free space in each storage pool to facilitate an automatic data rebalance operation after the failure of a main storage device.

The amount of free space in a storage pool is the difference between the following storage pool attributes:

- `spaceRaw`: Specifies the total amount of storage space associated with the storage pool.
- `spaceUsed`: Specifies the total amount of storage space allocated (used) in the storage pool.

You can use the ESCLI lsstoragepool command to examine these attributes. For example, the following command shows the `spaceRaw` and `spaceUsed` values associated with the storage pool named `POOL1`:

```
@> lsstoragepool POOL1 --attributes spaceRaw,spaceUsed
```

The amount of free space required to rebalance a storage pool after losing a storage device largely depends on the number and type of storage servers associated with the storage pool.

Another consideration is the redundancy level used to protect against data loss. However, because high redundancy (triple mirroring) is generally advised, the following recommendations assume that the storage pool only contains high redundancy files. Less free space may be sufficient if the storage pool includes a significant proportion of normal redundancy (double-mirrored) files.

When a storage pool occupies four or more storage servers, the lost data can be reconstructed on any partner storage server. In this case, the amount of free space that should be maintained to successfully rebalance a storage pool after losing one main storage device is whichever is greater out of the following:

- 3% of the total space in the storage pool.
- Three times the size of the lost pool disk(s).

> **Note:**
>
> On an EF storage server with 4 capacity-optimized flash devices, each device contains two pool disks.

When the storage pool occupies only three storage servers, high redundancy data must be reconstructed on the same server. In this case, the required free space to reconstruct data after losing a storage device is linked to the number of physical storage devices in the storage server.

The following formula describes the free space requirement to reconstruct data after losing one main storage device for a storage pool using only three storage servers. The free space requirement is expressed as a percentage of the total amount of storage space associated with the storage pool. In the formula, *DeviceCount* represents the number of main storage devices contained in each storage server.

```
Free Space Percentage = ceiling(100/DeviceCount)+3
```

For example, on a High Capacity (HC) storage server with 12 disks, one disk represents 8.33% (`100/12`) of the total storage capacity. So, for a storage pool using three 12-disk HC servers, you should maintain 12% (`ceiling(8.33)+3`) of the storage pool as free space to reconstruct data after losing a storage device.

Based on the preceding formula, the following table summarizes the amount of free space that should be maintained to successfully rebalance a storage pool spanning only three storage servers after losing one main storage device.

Storage Server Type	Free Space Requirement
High Capacity (HC) storage server with 12 disks	12% of the total space in the storage pool.
HC storage server with 6 disks	20% of the total space in the storage pool.
Extreme Flash (EF) storage server with 8 performance-optimized flash devices	16% of the total space in the storage pool
EF storage server with 4 capacity-optimized flash devices	28% of the total space in the storage pool

2.5.4 Modify a Storage Pool

An Exascale storage pool is a collection of pool disks that provides the persistent physical storage for Exascale vaults and files.

Fundamentally, you can modify a storage pool in two ways:

- You can modify the storage configuration associated with the storage pool.
- You can modify other attributes that are associated with the storage pool.

To modify a storage pool, use the ESCLI chstoragepool command:

- To modify the storage configuration associated with the storage pool, use the chstoragepool command in conjunction with the following options:
  - `--reconfig`: Reconfigures the storage pool and makes permanent any changes to the number or size of the underlying pool disks. Reconfiguration may trigger a rebalance operation to ensure that data is spread evenly across the pool disks in the storage pool. The time required for reconfiguration depends on the amount of data movement required to rebalance the storage pool.
  - `--force`: Optionally forces a reconfiguration operation even if the system detects no apparent change to the underlying pool disks.

  For example:

  ```
  @> chstoragepool POOL1 --reconfig --force
  ```

  Implicitly, you also modify the storage configuration associated with a storage pool when you perform any of the following tasks:
  - Add a Storage Server.
  - Create Pool Disks and allocate them to an existing storage pool.
  - Resize Pool Disks.
  - Drop Pool Disks.
- To modify the attributes that are associated with the storage pool, use the chstoragepool command, and specify the attribute that you want to modify. For example:

  ```
  @> chstoragepool mypoolname --attributes diskOfflineTimerInMins=60
  ```

  Use the `describe chstoragepool` command to view details about the storage pool attributes you can modify with `chstoragepool`.

2.5.5 Drop a Storage Pool

An Exascale storage pool is a collection of pool disks that provides the persistent physical storage for Exascale vaults and files.

To remove a storage pool, use the ESCLI rmstoragepool command and specify the storage pool that you want to remove. For example:

```
@> rmstoragepool storagepool-name
```

2.6 Administer the Storage Devices in an Exascale Storage Pool

Use the following procedures to administer the storage devices in an Exascale storage pool. If your system is configured to use Exascale and Oracle ASM (also known as a hybrid storage configuration), use these procedures in conjunction with the procedures described in Maintaining Oracle Exadata Storage Servers.

- Replacing a Failed Storage Device
  Failure of a storage device can affect performance and data redundancy. Consequently, a failed storage device should be replaced as soon as possible.

- Proactively Replacing a Storage Device
  You may need to proactively replace a storage device that has not failed.

Related Topics

- Pool Disks
- Storage Pools
- Maintaining Oracle Exadata Storage Servers

2.6.1 Replacing a Failed Storage Device

Failure of a storage device can affect performance and data redundancy. Consequently, a failed storage device should be replaced as soon as possible.

A storage device is considered to have failed in the following circumstances:

- A hardware or firmware fault causes the device to stop functioning.
- The device enters a predictive failure state.

  In this case, the device is still usable, but there is an indication that the device may soon stop functioning. For example, a hard disk drive (HDD) could be short of spare sectors, or a flash device could be approaching wear limits.
- Exadata software confines the device, and the device fails the post-confinement checks.

  Exadata automatically confines a storage device after detecting a significant performance problem or functional anomaly. After confinement, Exadata attempts to resolve the issue and recheck the device. However, if the post-confinement checks fail, the device is considered to have failed.

If a storage device fails, Exadata automatically drops all of the grid disks contained on the storage device. If any grid disk is used as an Exascale pool disk, Exascale automatically removes it and rebalances the storage pool to restore data redundancy.

Exadata also generates an alert when a storage device fails. The alert message includes specific instructions for replacing the device. If alert notifications are configured on the storage server, then the alert notification is automatically sent using email and SNMP.

After the failed storage device is replaced, Exadata automatically creates the cell and grid disks on the new device. If any grid disk is configured as an Exascale pool disk, it is automatically added to the storage pool, and the storage pool is rebalanced.

The following steps outline the procedure for replacing a failed storage device:

1. Confirm the location of the failed device.

   Use the following CellCLI `LIST PHYSICALDISK` command:

   ```
   CellCLI> list physicaldisk where status!=normal detail
   ```

   In the output, the `slotNumber` value describes the physical location of the device.

   You can also classify the failure type by examining the `status` value:

   - `failed` or `failed - dropped for replacement` - Indicates that the device stopped functioning due to a hardware or firmware failure.
   - `warning - predictive failure` - Indicates that the device entered a predictive failure state.

- `warning - poor performance` - Indicates that Exadata software confined the device, and the device failed the post-confinement checks.

For example, the following output indicates that the hard disk drive (HDD) in slot 5 stopped functioning due to a hardware or firmware failure:

```
CellCLI> list physicaldisk where status!=normal detail
         name:                   0:5
         deviceName:             /dev/sdi
         diskType:               HardDisk
         enclosureDeviceId:      0
         luns:                   0_5
         makeModel:              "WDC W7222A520ORA022T"
         physicalFirmware:       A7B0
         physicalInsertTime:     2023-07-07T17:20:44-07:00
         physicalInterface:      sas
         physicalSerial:         70SP8E
         physicalSize:           20.009765625T
         slotNumber:             5
         status:                 failed
```

2. Ensure that the storage server Do Not Service LED is not lit.
3. Ensure that the failed device is ready for removal.
   - If the failed device is a HDD or flash drive located in one of the hot-swappable drive bays in the front of the server, ensure that the blue OK to Remove LED on the device is lit before removing the device.
   - If the failed device is a hot-swappable flash card contained inside the server, ensure that the power LED on the flash card is not lit before removing the device. Starting with Exadata Storage Server X7-2, all storage server models contain hot-swappable flash cards.
4. Remove the failed storage device and install the replacement.

   See the associated server hardware guide for additional details about physical hardware replacement.
5. Wait for the server to recognize the replaced device.

   When you physically replace a hot-swappable storage device, it may take a few minutes for the server to recognize the new device.
6. Confirm the status of the replacement device.

   Use the CellCLI `LIST PHYSICALDISK` command to confirm that the status of the replacement device is `normal`.

   For example:

```
CellCLI> list physicaldisk 0:5 detail
         name:                   0:5
         deviceName:             /dev/sdi
         diskType:               HardDisk
         enclosureDeviceId:      0
         luns:                   0_5
         makeModel:              "WDC W7222A520ORA022T"
         physicalFirmware:       A7B0
         physicalInsertTime:     2023-09-01T12:00:25-07:00
         physicalInterface:      sas
```

```
physicalSerial:          75X8RD
physicalSize:            20.009765625T
slotNumber:              5
status:                  normal
```

7. Monitor the storage pool rebalance operation.

   As part of reintegrating any Exascale pool disk on the replacement storage device, the affected storage pool undergoes a rebalance operation.

   Use the ESCLI lsstoragepooloperation command to monitor the storage pool rebalance operation.

8. Confirm that Exascale is using the replacement device.

   Use the ESCLI lspooldisk command and examine the `status` attribute.

   Initially, as the replacement device comes online, the pool disk status is briefly set to `BEING ADDED`. However, the status value transitions to `ONLINE` as Exascale reintegrates the replacement device.

2.6.2 Proactively Replacing a Storage Device

You may need to proactively replace a storage device that has not failed.

For example, you may choose to replace a device after Exadata detects a temporary performance anomaly (temporary confinement), or you may need to replace a functioning device for any other reason.

The following steps outline the procedure to proactively replace a storage device:

1. Prepare the storage device for replacement.

   Use the CellCLI `ALTER PHYSICALDISK` command to prepare the system for replacement of the specified storage device. You can choose whether or not to maintain redundancy throughout the device replacement process.

   - If you maintain redundancy throughout the device replacement process, Exascale moves data off any affected pool disk and rebalances the associated storage pool. This option is generally recommended and provides the best protection from any other failure that could occur while replacing the device. However, it comes with the cost of moving the affected data and rebalancing the storage pool.

     To choose this option, add the `MAINTAIN REDUNDANCY` clause to the `ALTER PHYSICALDISK ... DROP FOR REPLACEMENT` command. For example, you can use the following command to prepare storage device `0:5` for replacement while maintaining redundancy:

     ```
     CellCLI> alter physicaldisk 0:5 drop for replacement maintain redundancy
     ```

   - If you choose not to maintain redundancy throughout the device replacement process, Exascale only performs a quick redundancy check before the device is taken offline. The redundancy check ensures that another copy of data is available for any affected pool disk. This option allows for quicker device replacement using reduced redundancy. However, it comes with an increased risk of data loss or a storage pool outage if another failure occurs while replacing the device. Consequently, this option is only recommended where data loss can be tolerated (for example, a non-critical test system) or where the increased risk associated with reduced redundancy is mitigated by some other means (for example, the data is replicated using Oracle Data Guard).

To choose this option, use the `ALTER PHYSICALDISK ... DROP FOR REPLACEMENT` command without the `MAINTAIN REDUNDANCY` clause. For example, you can use the following command to prepare storage device `0:5` for replacement without maintaining redundancy:

```
CellCLI> alter physicaldisk 0:5 drop for replacement
```

2. Ensure that the storage server Do Not Service LED is not lit.
3. Ensure that the device is ready for removal.
   - If the failed device is a hard disk drive (HDD) or flash drive located in one of the hot-swappable drive bays in the front of the server, ensure that the blue OK to Remove LED on the device is lit before removing the device.
   - If the failed device is a hot-swappable flash card contained inside the server, ensure that the power LED on the flash card is not lit before removing the device. Starting with Exadata Storage Server X7-2, all storage server models contain hot-swappable flash cards.
4. Remove the failed storage device and install the replacement.

   See the associated server hardware guide for additional details about physical hardware replacement.
5. Wait for the server to recognize the replaced device.

   When you physically replace a hot-swappable storage device, it may take a few minutes for the server to recognize the new device.
6. Confirm the status of the replacement device.

   Use the CellCLI `LIST PHYSICALDISK` command to confirm that the status of the replacement device is `normal`.

   For example:

```
CellCLI> list physicaldisk 0:5 detail
         name:                   0:5
         deviceName:             /dev/sdi
         diskType:               HardDisk
         enclosureDeviceId:      0
         luns:                   0_5
         makeModel:              "WDC W7222A520ORA022T"
         physicalFirmware:       A7B0
         physicalInsertTime:     2023-09-01T12:00:25-07:00
         physicalInterface:      sas
         physicalSerial:         75X8RD
         physicalSize:           20.009765625T
         slotNumber:             5
         status:                 normal
```

7. Monitor the storage pool rebalance operation.

   After the storage device is replaced, the Exadata cell disks and grid disks that existed on the original device are re-created on the new device. If any grid disk is designated as an Exascale pool disk, it is added to the storage pool and the storage pool is automatically rebalanced.

   Use the ESCLI lsstoragepooloperation command to monitor the storage pool rebalance operation.

8. Confirm that Exascale is using the replacement device.

   Use the ESCLI lspooldisk command and examine the `status` attribute.

   Initially, as the replacement device comes online, the pool disk status is briefly set to `BEING ADDED`. However, the status value transitions to `ONLINE` as Exascale reintegrates the replacement device.

2.7 Administer Cluster Templates

- Create a Cluster Template
- List Cluster Templates
- Modify a Cluster Template
- Remove a Cluster Template

Related Topics

- Templates

2.7.1 Create a Cluster Template

This topic describes how to create a user-defined cluster template.

A cluster template is a named collection of file storage attribute settings defined at the cluster level. By using cluster templates, you can manage file storage attributes automatically and consistently across the Exascale cluster.

Exascale is automatically preconfigured with cluster templates for each system-defined file type; for example, `DATAFILE`, `ONLINELOG`, `CONTROLFILE`, and so on. Therefore, you can only create new user templates at the cluster level.

To create a user-defined cluster template, use the ESCLI mktemplate command with the `--cluster` and `--name` options and specify the file storage attribute settings:

- `--media-type`: Specifies the physical media type that is used to store the file. Exascale uses this attribute to place the file in a storage pool that uses the specified media type. Possible values are:
  - `HC`: Identifies high capacity storage media, which uses hard disk drives (HDDs).
  - `EF`: Identifies extreme flash storage media, which uses flash devices.
- `--redundancy`: Specifies the number of data copies that are maintained. Possible values are:
  - `normal`: Maintains 2 mirrored copies of the file data.
  - `high`: Maintains 3 mirrored copies of the file data.
- `--content-type`: Specifies the type of content in the file. Exascale internally uses this attribute to place file extents on physically separate devices in a way that maximizes availability if failures occurs. Possible values are:
  - `DATA`
  - `RECO`

The command syntax is:

```
@> mktemplate --cluster --name template-name --content-type content-type --
media-type media-type --redundancy redundancy
```

2.7.2 List Cluster Templates

A cluster template is a named collection of file storage attribute settings defined at the cluster level. By using cluster templates, you can manage file storage attributes automatically and consistently across the Exascale cluster.

To display information about Exascale cluster templates, use the ESCLI lstemplate command with the `--cluster` option. For example:

```
@> lstemplate --cluster
```

This simple form of the command displays basic information about all of the cluster templates.

Using the command options available with the lstemplate command, you can specify the templates you want to display and display output with different levels of detail.

2.7.3 Modify a Cluster Template

A cluster template is a named collection of file storage attribute settings defined at the cluster level. By using cluster templates, you can manage file storage attributes automatically and consistently across the Exascale cluster.

- To modify an existing cluster template that is associated with a file type, use the ESCLI chtemplate command with the `--cluster` and `--file-type` options and specify the file storage attribute settings:
  - `--media-type`: Specifies the physical media type that is used to store the file. Exascale uses this attribute to place the file in a storage pool that uses the specified media type. Possible values are:
    - `HC`: Identifies high capacity storage media, which uses hard disk drives (HDDs).
    - `EF`: Identifies extreme flash storage media, which uses flash devices.
  - `--redundancy`: Specifies the number of data copies that are maintained. Possible values are:
    - `normal`: Maintains 2 mirrored copies of the file data.
    - `high`: Maintains 3 mirrored copies of the file data.
  - `--content-type`: Specifies the type of content in the file. Exascale internally uses this attribute to place file extents on physically separate devices in a way that maximizes availability if failures occurs. Possible values are:
    - `DATA`
    - `RECO`

  The command syntax is:

  ```
  @> chtemplate --cluster --file-type file-type --content-type content-type
  --media-type media-type --redundancy redundancy
  ```

- To modify a user-defined cluster template, use the chtemplate command with the `--cluster` and `--name` options and specify the file storage attribute settings. The command syntax is:

```
@> chtemplate --cluster --name template-name --content-type content-type --
media-type media-type --redundancy redundancy
```

2.7.4 Remove a Cluster Template

A cluster template is a named collection of file storage attribute settings defined at the cluster level. By using cluster templates, you can manage file storage attributes automatically and consistently across the Exascale cluster.

You can only remove user templates at the cluster level. You cannot remove a cluster template that is associated with a file type.

To remove a user-defined cluster template, use the ESCLI rmtemplate command with the `--cluster` and `--name` options. The command syntax is:

```
@> rmtemplate --cluster --name template-name
```

2.8 Administer Exascale Block Store System Objects

- Set the Volume Backup Location
- List the Volume Backup Location
- Create a Block Store VIP
- List Block Store VIPs
- Remove a Block Store VIP

Related Topics

- Block Store VIPs
- Volume Backups

2.8.1 Set the Volume Backup Location

This topic describes how to set the location where volume backups are stored.

Volume backups are stored in Oracle Cloud Infrastructure (OCI) object storage. To use volume backups you must have access to OCI Object Storage Service, and then specify the service details as attributes of the Exascale cluster.

To set the location where volume backups are stored, use the ESCLI chcluster command and specify the following attributes:

- `backupObjectStoreNamespace`: Specifies the OCI object store namespace that contains the backup resources, which is the top-level container for all storage buckets and objects.
- `backupCompartmentName`: Specifies the name of the OCI compartment that contains the backup resources.
- `backupCompartmentOcid`: Specifies the OCI unique identifier (OCID) for the compartment that contains the backup resources.

- `backupObjectStoreBucketName`: Specifies the name of the OCI object store bucket that contains the backup resources.
- `backupUserOcid`: Specifies the OCID for the backup user. The backup user is the OCI user that provides access to Object Storage Service for the Exascale volume backup service.
- `backupUserPubKeyFprint`: Specifies the public key fingerprint in base64 format for the backup user.
- `backupUserPrivKey`: Specifies the private key in PEM format for authenticating the backup user.
- `regionIdentifier`: Specifies the OCI region that contains the Exascale cluster resources.
- `tenancyOcid`: Specifies the OCID for the tenancy that contains the Exascale cluster resources.
- `backupEndpoint`: Optionally specifies the API endpoint URL for Object Storage Service. If specified, this value overrides the `regionIdentifier` value for volume backup purposes.

Because of the complex parameter values that are involved, consider using a file to contain the command. For example:

```
$ cat command.txt
chcluster --attributes
backupObjectStoreNamespace=exampledbaas,
backupCompartmentName=bkupComp,
backupCompartmentOcid=ocid1.compartment.oc1..aaaaaaaa********************a2h4p
6upiwtwtg6ehl7lo6mjqlj6zdvq,
backupObjectStoreBucketName=bkupBucket,
backupUserOcid=ocid1.user.oc1..aaaaaaaa********************qfacrpincf433r2koe3
7u5qgsidl6tra,
backupUserPrivKey=PEM-format-private-key-string,
backupUserPubKeyFprint=1e:1f:da:ee:**:**:**:**:**:**:**:e6:0c:1a:25,
regionIdentifier=us-phoenix-1,
tenancyOcid=ocid1.tenancy.oc1..aaaaaaaa********************7lb5gbw5icbm5lwwncx
s6uhjqhzexg2a

$ escli --wallet admin-user-wallet-location `cat command.txt`
```

2.8.2 List the Volume Backup Location

Volume backups are stored in Oracle Cloud Infrastructure (OCI) object storage. To use volume backups you must have access to OCI Object Storage Service, and then specify the service details as attributes of the Exascale cluster.

To display the backup storage location configuration details, use the ESCLI lscluster command and specify the `--backup` option. For example:

```
@> lscluster --backup --detail
id                                  1
backupCompartmentName               bkupComp
backupCompartmentOcid
ocid1.compartment.oc1..aaaaaaaa********************a2h4p6upiwtwtg6ehl7lo6mjqlj
6zdvq
backupUserOcid
ocid1.user.oc1..aaaaaaaa********************qfacrpincf433r2koe37u5qgsidl6tra
backupUserPubKeyFprint
```

```
1e:1f:da:ee:**:**:**:**:**:**:**:e6:0c:1a:25
backupObjectStoreNamespace           exampledbaas
tenancyOcid
ocid1.tenancy.oc1..aaaaaaaa********************7lb5gbw5icbm5lwwncxs6uhjqhzexg2
a
```

2.8.3 Create a Block Store VIP

Exascale uses block store virtual IP addresses (VIPs) to enable high availability and load balancing for iSCSI targets.

Each VIP is associated with an Exascale block store worker (BSW) and provides a reliable network endpoint for iSCSI initiators. Oracle recommends using four VIPs for each network on every BSW instance.

Typically, a pool of block store VIPs is created as part of the initial configuration procedure for Exascale. However, as an Exascale administrator, you may need to create other block store VIPs, most likely in response to a change in your Exascale system or network settings.

To create a block store VIP, use the ESCLI mkvolumeHAVIP command. For example:

```
@> mkvolumeHAVIP --attributes networkIPaddress=192.0.2.11
```

As shown in the example, the simplest form of the command specifies the IP address that is assigned to the block store VIP.

Using other options available with the mkvolumeHAVIP command, you can also specify the subnet mask, gateway address, and broadcast address that are associated with the VIP network.

2.8.4 List Block Store VIPs

Exascale uses block store high-availability virtual IP addresses (VIPs) to enable high availability and load balancing for iSCSI targets.

To display information about block store VIPs, use the ESCLI lsvolumeHAVIP command. For example:

```
@> lsvolumeHAVIP
```

By default, without any optional arguments, the command displays basic information about all of the block store VIPs associated with the Exascale cluster.

Using the command options available with the lsvolumeHAVIP command, you can:

- Specify the block store VIPs that you want to display.
- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

2.8.5 Remove a Block Store VIP

Exascale uses block store high-availability virtual IP addresses (VIPs) to enable high availability and load balancing for iSCSI targets.

As an Exascale administrator, you may need to delete a block store VIP, most likely in response to a change in your Exascale system or network settings.

To remove an existing block store VIP, use the ESCLI rmvolumeHAVIP command and specify the identifier for the block store VIP that is being removed. For example:

```
@> rmvolumeHAVIP  block-store-vip-id
```

You can use the lsvolumeHAVIP command to find the identifier for each block store VIP.

3

Oracle Exadata Exascale User-Specific Administration

This chapter covers administration tasks that define and manage Exascale storage objects and metadata that belong to a specific Exascale user.

Administration activities for Oracle Exadata Exascale are logically divided into two groups. System administration activities define and manage the Exascale system components that are shared across the system. User-specific administration activities define and manage Exascale storage objects and metadata that belong to a specific Exascale user.

This chapter contains the following user-specific administration topics:

3.1 Administer Exascale User Credentials

Related Topics

- Exascale User Credentials

3.1.1 Create User Keys

This topic describes how to create a public/private key pair for an Exascale user.

Exascale authentication uses a system of public and private key pairs. Each Exascale user is associated with a public key. To prove their identity, an Exascale user must supply the matching private key.

For maximum security, an Exascale user should create their own public/private key pair. This is recommended to ensure the integrity of the private key, since the user should never share the private key, not even with the Exascale administrator.

For use with Exascale, the public and private keys must be in PEM format.

To create a key pair, you can use standard utilities, such as `openssl`, or you can use the ESCLI mkkey command. For example:

```
@> mkkey --private-key-file priv.pem --public-key-file pub.pem
```

After you create a key pair, you will need to send the public key to the Exascale administrator so that they can associate it with your Exascale user account.

Never share the private key, not even with the Exascale administrator.

3.1.2 Create a Wallet

Exascale user credentials are contained in a digital key store, also known as a wallet.

For maximum security, an Exascale user should create and manage their own wallet.

Before you can associate a wallet with an Exascale user, you must know the user identifier for the Exascale user. Consult with your Exascale administrator to determine the required value.

To create a wallet for an Exascale user:

1. Create an empty wallet.

   Use the ESCLI mkwallet command and specify the directory location for the new wallet. For example:

   ```
   @> mkwallet --wallet /home/user/user.wallet
   ```

2. Associate the wallet with the Exascale user.

   Use the ESCLI chwallet command and specify:

   - The user ID for the user that is being associated with the wallet.
   - The wallet directory location.

   For example:

   ```
   @> chwallet --wallet /home/user/user.wallet --attributes user=theuserID
   ```

3.1.3 Store a Private Key in a Wallet

Exascale authentication uses a system of public and private key pairs. Each Exascale user is associated with a public key. To prove their identity, an Exascale user must supply the matching private key, which is stored inside a digital wallet.

Before you can store a private key in your wallet, you must already have the key and the wallet.

To store a private key in a wallet, use the ESCLI chwallet command and specify:

- The wallet location.
- The location of the file that contains the user's private key in PEM format.

For example:

```
@> chwallet --wallet /home/user/user.wallet --private-key-file
     priv.pem
```

If the wallet did not contain a private key before the chwallet command, then the specified private key is inserted into the wallet. Otherwise, the existing private key is overwritten with the specified private key.

3.1.4 Fetch the Trust Store

The Exascale trust store is a group of digital certificates that facilitates trusted communication between servers in the Exascale cluster. To use Exascale, users need to copy the trust store into their wallet.

To copy the trust store into your wallet, use the ESCLI chwallet command and specify the wallet location. For example:

```
@> chwallet --wallet /home/user/user.wallet --fetch-trust-store
```

3.1.5 Store the URL for Exascale Cluster Services

If you want to use a wallet to access data files on Exascale storage directly, the wallet must contain the URL for Exascale cluster services (EGS).

This requirement does not apply if the wallet does not directly access data files on Exascale storage. For example, a wallet need not contain the URL for Exascale cluster services (EGS) if it is used only for Exascale administration operations through the ESCLI utility.

You must already have a wallet before you can store the URL for Exascale cluster services (EGS) in it.

The URL for Exascale cluster services (EGS) is contained in the `exaRootUrl` attribute associated with the Exascale cluster. You can also get the required value by copying the `exaRootUrl` value from another previously configured wallet.

To store the URL for Exascale cluster services (EGS) in your wallet, use the ESCLI chwallet command and specify:

- The wallet location.
- The URL for Exascale cluster services (EGS).

For example:

```
@> chwallet --wallet /home/user/user.wallet --attributes
exaRootUrl="egs=egsexc4:192.0.2.217:5045 egs=egsexc4:192.0.2.218:5045
egs=egsexc4:192.0.2.219:5045"
```

3.1.6 Store the Exascale Control Services Endpoint

As a matter of convenience, you can store the default endpoint for Exascale control services in your wallet. Unless another value is explicitly provided in the command line, the ESCLI utility automatically uses the Exascale control services endpoint that is contained in your wallet.

You must already have a wallet before you can store the Exascale control services endpoint in it.

To store the Exascale control services endpoint in your wallet, use the ESCLI chwallet command and specify:

- The wallet location.
- The server and port that host the Exascale control services endpoint.

For example:

```
@> chwallet --wallet /home/user/user.wallet --attributes
restEndPoint=exa01:61926
```

3.1.7 Distribute a Wallet

You may need to move or copy your Exascale user wallet to connect using various different Exascale clients. However, for security reasons the Exascale user wallet is tied to the host where it is created. So, you cannot simply copy the wallet directory and its contents.

To copy your Exascale user wallet, you must create a new wallet on the desired host and populate it with the required user credentials. To move an Exascale user wallet, you must create a copy of the wallet and then delete the original.

If you have the required user credentials, then you can use the procedures described in:

- Create a Wallet
- Store a Private Key in a Wallet
- Fetch the Trust Store
- Store the Exascale Control Services Endpoint

If you do not have the required user credentials, then you can get them from the existing wallet by using the ESCLI lswallet command.

For example, the following command displays the Exascale user identifier and the Exascale control services endpoint that are stored in the wallet at */home/user/user.wallet*:

```
@> lswallet --wallet /home/user/user.wallet --attributes user,restEndPoint
```

In the following example, the private key is extracted from the wallet and stored in a regular PEM file located at */home/user/priv.pem*:

```
@> lswallet --wallet /home/user/user.wallet --private-key-file /home/user/
priv.pem
```

3.1.8 Change User Keys

This topic describes how to change a public/private key pair for an Exascale user.

Exascale authentication uses a system of public and private key pairs. Each Exascale user is associated with a public key. To prove their identity, an Exascale user must supply the matching private key, which is stored inside a digital wallet.

If your private key is ever compromised, you will need to generate a new key pair and change to using it. It is also good practice to periodically change, or rotate, your keys.

To change a public/private key pair:

1. Generate a new key pair:

   To create a key pair, you can use standard utilities, such as `openssl`, or you can use the ESCLI mkkey command. For example:

   ```
   @> mkkey --private-key-file newpriv.pem --public-key-file newpub.pem
   ```

2. Associate the new public key with the Exascale user:

   Use the ESCLI chuser command and specify:

   - The unique user ID for the user that is being modified in the `chuser` command. By default, an Exascale user can change their own public key.
   - The location of the file that contains the user's new public key in PEM format.

   For example:

   ```
   @> chuser theuserID --public-key-file1 newpub.pem
   ```

   If you want to ensure that a previously associated public key is no longer usable, then use the public key slot (`--public-key-file1`, `--public-key-file2`, or `--public-key-file3`) that contains the key that you want to overwrite.

3. Store the new private key in the user's wallet:

   Repeat this step for every copy of the user's wallet, which may reside on different Exascale servers.

   Use the ESCLI chwallet command and specify:

   - The wallet location.
   - The location of the file that contains the user's new private key in PEM format.

   For example:

   ```
   @> chwallet --wallet /home/user/user.wallet --private-key-file
         newpriv.pem
   ```

   When prompted, specify that you want to overwrite the existing private key in the wallet.

3.2 Administer Exascale Vaults

- Create a Vault

- List Vaults
- Modify a Vault
- Remove a Vault

Related Topics

- Vaults

3.2.1 Create a Vault

An Exascale vault is a logical file container.

To create a vault, use the ESCLI mkvault command. For example:

```
@> mkvault @myvaultname
```

As shown in the example, the simplest form of the command specifies only the name of the new vault. In this case, the resulting vault has unlimited access to all of the resources in the Exascale cluster.

Alternatively, when you create a new vault you can specify one or more resource provisioning attributes. For example, the following command creates a new vault that is provisioned with 10 TB of high capacity storage (HC) space:

```
@> mkvault @myvaultname --attributes spaceProvHC=10T
```

Following is the list of vault-specific resource provisioning attributes that may be set:

- `spaceProvEF`: Provisions the vault with the specified amount of `EF` storage space.
- `spaceProvHC`: Provisions the vault with the specified amount of `HC` storage space.
- `iopsProvEF`: Provisions the vault with the specified number of IOPS from `EF` storage.
- `iopsProvHC`: Provisions the vault with the specified number of IOPS from `HC` storage.
- `flashCacheProv`: Provisions the vault with the specified amount of flash cache space.
- `flashLogProv`: Boolean value (`true` or `false`) indicating whether the vault is provisioned with access to Exadata Smart Flash Log. The default value is `true`.
- `xrmemCacheProv`: Provisions the vault with the specified amount of Exadata RDMA Memory Cache (XRMEM cache) space.

  Use this attribute only on systems with `HC` or `EF` storage, but not both. For systems with `HC` and `EF` storage, use the following media-specific XRMEM cache provisioning attributes.
- `xrmemCacheProvEF`: Provisions the vault with the specified amount of XRMEM cache space associated with `EF` media.
- `xrmemCacheProvHC`: Provisions the vault with the specified amount of XRMEM cache space associated with `HC` media.

3.2.2 List Vaults

An Exascale vault is a logical file container.

To display information about vaults, use the ESCLI ls command while the root directory is the current working directory in the ESCLI session. For example:

```
@> ls
```

By default, without any optional arguments, the command displays basic information about all of the vaults in the Exascale cluster.

Using the command options available with the ls command, you can:

- Specify the vault(s) that you want to display.
- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

3.2.3 Modify a Vault

An Exascale vault is a logical file container.

To modify an existing vault, use the ESCLI chvault command and specify:

- The vault name.
- The attributes that you want to change.

For example, the following command modifies the specified vault to enforce a 20 TB limit for the high capacity (HC) storage space:

```
@> chvault @myvaultname --attributes spaceProvHC=20T
```

Use the `describe chvault` command to view details about all the vault attributes you can modify with `chvault`.

Following is the list of vault-specific resource provisioning attributes that may be changed:

- `spaceProvEF`: Provisions the vault with the specified amount of `EF` storage space.
- `spaceProvHC`: Provisions the vault with the specified amount of `HC` storage space.
- `iopsProvEF`: Provisions the vault with the specified number of IOPS from `EF` storage.
- `iopsProvHC`: Provisions the vault with the specified number of IOPS from `HC` storage.
- `flashCacheProv`: Provisions the vault with the specified amount of flash cache space.
- `flashLogProv`: Boolean value (`true` or `false`) indicating whether the vault is provisioned with access to Exadata Smart Flash Log. The default value is `true`.
- `xrmemCacheProv`: Provisions the vault with the specified amount of Exadata RDMA Memory Cache (XRMEM cache) space.

  Use this attribute only on systems with `HC` or `EF` storage, but not both. For systems with `HC` and `EF` storage, use the following media-specific XRMEM cache provisioning attributes.
- `xrmemCacheProvEF`: Provisions the vault with the specified amount of XRMEM cache space associated with `EF` media.
- `xrmemCacheProvHC`: Provisions the vault with the specified amount of XRMEM cache space associated with `HC` media.

3.2.4 Remove a Vault

An Exascale vault is a logical file container.

To delete a vault, use the ESCLI rmvault command. For example:

```
@> rmvault @myvaultname
```

As shown in the example, the simplest form of the command deletes the specified vault.

You can add the `--nowait` option to return immediately and run the operation in the background instead of waiting for the command to finish.

You can add the `--force` option to delete the vault and all of the files that it contains. Take extreme care with this option!

If you attempt to delete a vault that is not empty without the `--force` option, then the operation fails and an error is displayed.

3.3 Administer Exascale Files

- Create a File
- Copy a File
- List Files
- Clone Files
- Snapshot Files
- List Clones and Snapshots
- Remove a File

Related Topics

- Files

3.3.1 Create a File

This topic describes how to create a file in Exascale using ESCLI.

Typically, you implicitly create files in Exascale as a byproduct of performing some action with an Exascale client, such as Oracle Database. For example, when you create a database or add a file to a tablespace.

However, you can also explicitly create a file inside an Exascale vault by using ESCLI.

To create a file using ESCLI, use the mkfile command. For example:

```
@> mkfile @myvaultname/myfilename
```

As shown in the example, the simplest form of the command creates an empty file in the specified vault location.

Using other command options available with the mkfile command, you can:

- Specify the file size.

- Specify the file type.
- Specify a template to apply.
- Explicitly specify the file storage attributes. That is, the physical media type, the redundancy (mirroring) level, and the content type.

Related Topics

- File Storage Attributes
- Templates
- Using Oracle Database with Exascale

3.3.2 Copy a File

You can copy a file from your local filesystem into Exascale, or you can copy a file from Exascale into your local filesystem.

Before you copy a file into Exascale, you might want to first create an empty copy of the file so that you can specify the file storage attributes.

To copy a file between the local filesystem and Exascale, use the XSH cp command:

- In the following example, XSH copies the local file located at */home/myfile* into an Exascale file at *@myvaultname/myfilecopy*.

```
$ xsh cp /home/myfile @myvaultname/myfilecopy
```

- In the following example, XSH copies the Exascale file located at *@myvaultname/myfile* to */home/myfilecopy* on the system running XSH.

```
@> xsh cp @myvaultname/myfile /home/myfilecopy
```

You can also use the ESCLI putfile and getfile commands to copy files to and from Exascale. However, the XSH cp command supports the transfer of much larger files and provides superior transfer performance.

3.3.3 List Files

To display information about files inside an Exascale vault, use the ESCLI ls command and specify the vault name followed by the slash character (/). For example:

```
@> ls @myvaultname/
```

As shown in the example, a simple form of the command displays basic information about all of the files inside the specified vault.

Alternatively, you can change the working directory in the ESCLI session by using the cd command, and then run the ls command within the context of the specified vault. For example:

```
@> cd @myvaultname
@myvaultname/> ls
```

Using the command options available with the ls command, you can also:

- Specify the files that you want to display.

- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

3.3.4 Clone Files

An Exascale clone is a thinly-provisioned point-in-time copy of a file that is readable and writable. The source file for a clone can be a regular file, a snapshot, or another clone. Exascale uses redirect-on-write techniques to create and maintain clones very quickly and space-efficiently.

To clone a file or group of files, use the ESCLI clonefile command.

All clones created in the same operation are point-in-time consistent, and all files in a clone operation must be in the same vault.

- To clone a single file, use the clonefile command and specify the source file and the clone destination. For example:

  ```
  @> clonefile @myvaultname/myfilename @myvaultname/myclonefile
  ```

  In the previous example, the clone (*@myvaultname/myclonefile*) is created in the same location as the original file (*@myvaultname/myfilename*). In the following example, the clone is created under *@myvaultname/clones*.

  ```
  @> clonefile @myvaultname/myfilename @myvaultname/clones/myotherclonefile
  ```

- To clone a group of files, use the clonefile command and specify the source files and the clone destinations by including the wildcard character (*). For example:

  ```
  @> clonefile @myvaultname/myfiles* @myvaultname/myclonefiles*
  ```

  In the previous example, the clones are created in the same location as the original files. In the following example, the clones are created in a different location.

  ```
  @> clonefile @myvaultname/myfiles* @myvaultname/moreclones/moreclonefiles*
  ```

- To clone multiple files or groups of files, use the clonefile command and specify multiple source and destination pairs. For example:

  ```
  @> clonefile @myvaultname/file1 @myvaultname/file1clone @myvaultname/
  myfiles* @myvaultname/myclonefiles*
  ```

  In the previous example, an individual file (*@myvaultname/file1*) and a group of files (*@myvaultname/myfiles**) are cloned in the same operation, and the clones (*@myvaultname/file1clone* and *@myvaultname/myclonefiles**) are created in the same location as the original files.

  In the following example, the clones are created in multiple different locations.

  ```
  @> clonefile @myvaultname/file1 @myvaultname/location1/file1clone
  @myvaultname/myfiles* @myvaultname/location2/myclonefiles*
  ```

When you specify multiple source and destination pairs, the source file specifications are considered in order, and only the first match is used.

For example, the following command creates clones under *@myvaultname/clone/a* for files matching *@myvaultname/a**, while the clones for the other files are created under *@myvaultname/clone/other*.

```
@> clonefile @myvaultname/a* @myvaultname/clone/a/a* @myvaultname/*
@myvaultname/clone/other/*
```

However, the following command creates all of the clones under *@myvaultname/clone/other* because all of the files match *@myvaultname/**. So, in this case, the second argument pair is never used.

```
@> clonefile @myvaultname/* @myvaultname/clone/other/* @myvaultname/a*
@myvaultname/clone/a/a*
```

- To clone some files while excluding others, use the clonefile command with one of the following exclusion methods:
  - You can use the `--exclude` option to specify files that are excluded from the clone operation.

    For example, the following command clones files in *@myvaultname*, except for files matching *@myvaultname/a** or *@myvaultname/b**:

    ```
    @> clonefile @myvaultname/* @myvaultname/clone/* --exclude
    @myvaultname/a* --exclude @myvaultname/b*
    ```

  - If you specify a null (empty) string as a clone destination value, no clones are created for the corresponding source files. The null destination and corresponding source is considered in order along with all other source and destination pairs, and only the first match is used.

    For example, the following command also clones files in *@myvaultname*, except for those matching *@myvaultname/a** or *@myvaultname/b**.

    ```
    @> clonefile @myvaultname/a* "" @myvaultname/b* "" @myvaultname/*
    @myvaultname/clone/*
    ```

After creation, clones are managed using the same commands and procedures that are used for other files. For example, you must use the rmfile command to delete a clone.

Related Topics

- Clones

3.3.5 Snapshot Files

An Exascale snapshot is a thinly-provisioned read-only point-in-time copy of a file. The source file for a snapshot can be a regular file, a file clone, or another snapshot. Exascale uses redirect-on-write techniques to create and maintain snapshots very quickly and space-efficiently.

To snapshot a file or group of files, use the ESCLI snapshotfile command.

All snapshots created in the same operation are point-in-time consistent, and all files in a snapshot operation must be in the same vault.

- To snapshot a single file, use the snapshotfile command and specify the source file and the snapshot destination. For example:

  ```
  @> snapshotfile @myvaultname/myfilename @myvaultname/mysnapshotfile
  ```

  In the previous example, the snapshot (*@myvaultname/mysnapfile*) is created in the same location as the original file (*@myvaultname/myfilename*). In the following example, the snapshot is created under *@myvaultname/snapshots*.

  ```
  @> snapshotfile @myvaultname/myfilename @myvaultname/snapshots/
  myothersnapshotfile
  ```

- To snapshot a group of files, use the snapshotfile command and specify the source files and the snapshot destinations by including the wildcard character (*). For example:

  ```
  @> snapshotfile @myvaultname/myfiles* @myvaultname/mysnapshotfiles*
  ```

  In the previous example, the snapshots are created in the same location as the original files. In the following example, the snapshots are created in a different location.

  ```
  @> snapshotfile @myvaultname/myfiles* @myvaultname/moresnapshots/
  moresnapshotfiles*
  ```

- To snapshot multiple files or groups of files, use the snapshotfile command and specify multiple source and destination pairs. For example:

  ```
  @> snapshotfile @myvaultname/file1 @myvaultname/file1snapshot @myvaultname/
  myfiles* @myvaultname/mysnapshotfiles*
  ```

  In the previous example, the snapshot operation processes an individual file (*@myvaultname/file1*) and a group of files (*@myvaultname/myfiles**), and the snapshots (*@myvaultname/file1snapshot* and *@myvaultname/mysnapshotfiles**) are created in the same location as the original files.

  In the following example, the snapshots are created in multiple different locations:

  ```
  @> snapshotfile @myvaultname/file1 @myvaultname/location1/file1snapshot
  @myvaultname/myfiles* @myvaultname/location2/mysnapshotfiles*
  ```

  When you specify multiple source and destination pairs, the source file specifications are considered in order, and only the first match is used.

  For example, the following command creates snapshots under *@myvaultname/snap/a* for files matching *@myvaultname/a**, while the snapshots for the other files are created under *@myvaultname/snap/other*.

  ```
  @> snapshotfile @myvaultname/a* @myvaultname/snap/a/a* @myvaultname/*
  @myvaultname/snap/other/*
  ```

However, the following command creates all of the snapshots under *@myvaultname/snap/other* because all of the files match *@myvaultname/**. So, in this case, the second argument pair is never used.

```
@> snapshotfile @myvaultname/* @myvaultname/snap/other/* @myvaultname/a*
@myvaultname/snap/a/a*
```

- To snapshot some files while excluding others, use the snapshotfile command with one of the following exclusion methods:
  - You can use the `--exclude` option to specify files that are excluded from the snapshot operation.

    For example, the following command snapshots files in *@myvaultname*, except for files matching *@myvaultname/a** or *@myvaultname/b**:

    ```
    @> snapshotfile @myvaultname/* @myvaultname/snap/* --exclude
    @myvaultname/a* --exclude @myvaultname/b*
    ```

  - If you specify a null (empty) string as a snapshot destination value, no snapshots are created for the corresponding source files. The null destination and corresponding source is considered in order along with all other source and destination pairs, and only the first match is used.

    For example, the following command also snapshots files in *@myvaultname*, except for those matching *@myvaultname/a** or *@myvaultname/b**.

    ```
    @> snapshotfile @myvaultname/a* "" @myvaultname/b* "" @myvaultname/*
    @myvaultname/snap/*
    ```

After creation, snapshots are generally managed using the same commands and procedures that are used for other files. For example, you can use the rmfile command to delete a snapshot.

Related Topics

- Snapshots

3.3.6 List Clones and Snapshots

To display information about related files, clones, and snapshots, use the ESCLI lssnapshots command. This command has two principal modes of operation:

- To display information about the snapshots that are based on a specific file, use the lssnapshots command and specify the file that is the subject of the command. The specified file can be a snapshot, clone, or regular file. For example:

  ```
  @> lssnapshots @myvaultname/file1
  name
  @myvaultname/file1
  @myvaultname/snap2_of_file1
  @myvaultname/snap1_of_file1
  ```

  As shown in the example, a simple form of the command displays the snapshots that are directly based on the specified file.

  Using the command options available with the lssnapshots command, you can also:

  - Filter the output according to your specified conditions.
  - Sort the output using specific attributes.
  - Choose to display output with different levels of detail.

- To display information about related files, clones, and snapshots, use the lssnapshots command and specify a file name in conjunction with the `--all` option or the `--tree` option.

  Both options display information about all files, clones, and snapshots belonging to the snapshot tree that contains the specified file. The specified file can be a snapshot, clone, or regular file at any level in the snapshot tree. No other command options are permitted in conjunction with the `--all` option or the `--tree` option.

  For example, consider a group of related files that is based on the following command sequence:

```
@myvaultname/> putfile somelocalfile file1
Putting file somelocalfile to @myvaultname/file1
Success.

@myvaultname/> snapshotfile file1 snap1_of_file1
Success.

@myvaultname/> clonefile file1 clone1
Success.

@myvaultname/> snapshotfile file1 snap2_of_file1
Success.

@myvaultname/> snapshotfile clone1 snap1_of_clone1
Success.
```

  The following example shows how to display related files using the `--all` command option. The output is presented in a series of lists. The first list contains the files and clones in the snapshot tree. The remaining output lists snapshots that are grouped by their source file.

```
@> lssnapshots @myvaultname/clone1 --all

---FILE/CLONES:---

2022-05-02 06:14:41 UTC @myvaultname/file1
2022-05-02 06:15:15 UTC @myvaultname/clone1

---SNAPSHOTS:---

@myvaultname/file1
  2022-05-02 06:15:06 UTC @myvaultname/snap1_of_file1
  2022-05-02 06:15:15 UTC @myvaultname/?8000_0000_000a:00000001
  2022-05-02 06:15:24 UTC @myvaultname/snap2_of_file1

@myvaultname/clone1
  2022-05-02 06:15:45 UTC @myvaultname/snap1_of_clone1
```

Note that the command output includes an internal snapshot (`@myvaultname/?8000_0000_000a:00000001`), which was implicitly created as part of the cloning operation to create `@myvaultname/clone1`.

The following example shows how to display related files in a snapshot tree using the `--tree` command option.

```
@> lssnapshots @myvaultname/clone1 --tree

+---- [1] @myvaultname/snap1_of_file1
+---- [2] @myvaultname/?8000_0000_000a:00000001
|
|`---+---- [4] @myvaultname/snap1_of_clone1
|    +---- [6] @myvaultname/clone1
|
+---- [3] @myvaultname/snap2_of_file1
+---- [5] @myvaultname/file1
```

3.3.7 Remove a File

To delete a file in an Exascale vault, use the ESCLI rmfile command. For example:

```
@> rm @myvaultname/myfilename
```

As shown in the example, the simplest form of the command deletes the specified file.

You can add the `--force` option to delete the file even if it is in use.

You can add the `--nowait` option to return immediately and run the operation in the background instead of waiting for the command to finish.

You can add the `--with-snapshots` option to delete the file and any associated snapshots and clones.

3.4 Administer Exascale Datasets

Exascale automatically creates and maintains system-defined datasets, which you can use to track storage utilization for groups of related files, such as all of the files belonging to a specific database.

- List Datasets

Related Topics

- Datasets

3.4.1 List Datasets

To display information about an Exascale dataset, use the ESCLI lsdataset command.

- To display a simple list of all datasets in all vaults, use the lsdataset command with no arguments or options. For example:

  ```
  @> lsdataset
  ```

- To display detailed information about a specific dataset, use the lsdataset command with the `--detail` option and specify the dataset identifier.

  Each system-defined dataset has a unique composite identifier, which contains unique identifiers for the associated entities. The dataset identifier has one of the following formats:

  - *`@Vault-name`*: Identifies the vault-level dataset for a specific named vault.
  - *`@Vault-name`*`:`*`GI-cluster-ID`*: Identifies the dataset for a specific Oracle Grid Infrastructure (GI) cluster, which consumes storage space in the specified vault.
  - *`@Vault-name`*`:`*`GI-cluster-ID`*`:`*`CDB-ID`*: Identifies the dataset for a specific Oracle multitenant container database (CDB) that belongs to the specified GI cluster.
  - *`@Vault-name`*`:`*`GI-cluster-ID`*`:`*`CDB-ID`*`:`*`PDB-ID`*: Identifies the dataset for an Oracle pluggable database (PDB) that is associated with the specified CDB, GI cluster, and vault.

  For example, the following command displays detailed information about a specific PDB dataset.

  ```
  @> lsdataset --details
  @MYDATA:a5b4997a027d6f91ffd9729702ff6ec5:1427076301:2164757665
  ```

- To display information about the files in a specific dataset, use the lsdataset command with the `--files` option and specify the dataset identifier.

  For example, the following command lists the files that belong to the specified GI-level dataset.

  ```
  @> lsdataset --files @MYDATA:a5b4997a027d6f91ffd9729702ff6ec5
  ```

  If you add the `--recursive` option, then the command output also includes the files in all of the descendant datasets.

  For example, the following command shows the files belonging to the specified GI cluster and all of the associated CDB and PDB datasets.

  ```
  @> lsdataset --files --recursive @MYDATA:a5b4997a027d6f91ffd9729702ff6ec5
  ```

- When specifying a dataset identifier, the asterisk (`*`) can be used for wildcard searches.

  For example, the following command lists all CDB datasets associated with the specified GI cluster and vault.

  ```
  @> lsdataset @MYDATA:a5b4997a027d6f91ffd9729702ff6ec5:*
  ```

Using other command options available with the lsdataset command, you can also:

- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose the dataset attributes that are displayed.

3.5 Administer Access Control Lists

- List ACLs

- Modify an ACL

Related Topics

- Access Control Lists

3.5.1 List ACLs

Access control lists (ACLs) govern the operations that users can perform on Exascale vaults and files.

To display information about ACLs, use the ESCLI lsacl command:

- To display the ACLs for all of the vaults, use the lsacl command with no arguments while the root directory is the current working directory in the ESCLI session. For example:

  ```
  @> lsacl
  ```

- To display the ACLs for all files in a vault, use the lsacl command and specify the vault name along with a wildcard (*) for the files. For example:

  ```
  @> lsacl @myvaultname/*
  ```

  Alternatively, you can use the lsacl command with no arguments while the current working directory is in the desired vault. For example:

  ```
  @> cd @myvaultname
  @myvaultname/> lsacl
  ```

- To display the ACLs for a specific vault or file, use the lsacl command and specify the desired vault or file. For example:

  ```
  @> lsacl @myvaultname/somefile
  ```

- To display the ACLs for multiple vaults and files, use the lsacl command and specify multiple vault and file specification strings. For example:

  ```
  @> lsacl @myvaultname @myvaultname/* @anothervaultname @anothervaultname/
  somefile
  ```

3.5.2 Modify an ACL

To modify an access control list (ACL) for an Exascale vault or file, use the ESCLI chacl command and specify:

1. The vault or file for which you are modifying the ACL.
2. The ACL string, which defines the modification to the ACL.

   The ACL string is a list of user IDs and ACL privilege pairs. Depending on the user creation method, the user ID may be a system-generated value or a user-specified value. For example:

   ```
   dd7c8e35-3c8d-4441-a9b0-f58e959b84ba:read;scott:inspect
   ```

If the ACL string begins with a plus sign (+), then the rest of the ACL string is added to the existing ACL for the file or vault. Without the +, the existing vault or file ACL is overwritten.

- To set the ACL for a vault, use the chacl command and specify the vault and the new ACL string. For example, to make *jenny* the sole manager of *@vault1* use:

```
@> chacl @vault1 jenny:manage
```

- To set the ACL for a file, use the chacl command and specify the file and the new ACL string. For example, to make *jenny* the manager and *jill* a reader of *@vault1/file1* use:

```
@> chacl @vault1/file1 jenny:manage;jill:read
```

- To add to an existing ACL, use the chacl command, specify the vault or file name, and specify an ACL string that begins with +. For example, to add *peter* as a user of *@vault1* use:

```
@> chacl @vault1 +peter:use
```

- To remove a user from an ACL, use the chacl command, specify the vault or file name, and specify an ACL string that begins with + and uses the ACL privilege `none`. For example, to remove *jill* from the ACL for *@vault1/file1* use:

```
@> chacl @vault1/file1 +jill:none
```

3.6 Administer Extended Attributes

- List Extended Attributes
- Set an Extended Attribute
- Remove an Extended Attribute

Related Topics

- Extended Attributes

3.6.1 List Extended Attributes

An extended attribute is non-standard user-defined metadata that is associated with an Exascale vault or file.

To display information about extended attributes, use the ESCLI lsxattr command:

- To display all of the extended attributes for a specific vault or file, use the lsxattr command and specify the desired vault or file name. For example:

```
@> lsxattr @myvaultname/myfile
```

- To display a specific extended attribute for a vault or file, use the lsxattr command and specify:
  - The name of the vault or file that is associated with the extended attribute.
  - The extended attribute name.

For example:

```
@> lsxattr @myvaultname/myfile --name myattribute
```

- To write a binary extended attribute value to a local file, use the lsxattr command and specify:
  - The name of the vault or file that is associated with the extended attribute.
  - The extended attribute name.
  - The name of the file where you want to write the value of the extended attribute.

  For example:

```
@> lsxattr @myvaultname/myfile --name mybinaryattribute --bin-value-file /
home/myoutputfile
```

3.6.2 Set an Extended Attribute

A user-defined extended attribute is non-standard metadata that is associated with an Exascale vault or file.

You can set new extended attributes for a vault or a file, or you can overwrite the value for existing user-defined extended attributes.

> **Note:**
>
> Extended attributes with names that start with a dollar sign ($) are reserved for internal use by Exascale services and tools. You cannot create, modify, or delete these extended attributes.

To set a user-defined extended attribute and associate it with an Exascale vault or file, use the ESCLI chxattr command:

- To set an extended attribute with a string value, use the chxattr command and specify:
  - The name of the vault or file that is associated with the extended attribute.
  - The extended attribute name.
  - The extended attribute value.

  For example:

```
@> chxattr @myvaultname/myfile --name myattribute --value myattributevalue
```

- To set an extended attribute with a binary value, use the chxattr command and specify:
  - The name of the vault or file that is associated with the extended attribute.
  - The extended attribute name.
  - The name of the file that contains the binary extended attribute value.

For example:

```
@> chxattr @myvaultname/myfile --name mybinaryattribute --bin-value-file /
home/mybinaryfile
```

3.6.3 Remove an Extended Attribute

A user-defined extended attribute is non-standard metadata that is associated with an Exascale vault or file.

To delete an extended attribute that is associated with an Exascale vault or file, use the ESCLI rmxattr command and specify:

- The name of the vault or file that is associated with the extended attribute.
- The extended attribute name.

For example:

```
@> rmxattr @myvaultname/myfile --name myattribute
```

> **Note:**
>
> Extended attributes with names that start with a dollar sign (`$`) are reserved for internal use by Exascale services and tools. You cannot create, modify, or delete these extended attributes.

3.7 Administer Vault Templates

- Create a Vault Template
- List Vault Templates
- Modify a Vault Template
- Remove a Vault Template

Related Topics

- Templates

3.7.1 Create a Vault Template

A vault template is a named collection of file storage attribute settings defined at the vault level and used when creating new files in the vault. By using vault templates, you can manage file storage attributes automatically and consistently across the vault. A vault template overrides the settings in the corresponding cluster template.

- To create a vault template that is associated with a file type, use the ESCLI mktemplate command with the `--vault` and `--file-type` options and specify the file storage attribute settings:
  - `--media-type`: Specifies the physical media type that is used to store the file. Exascale uses this attribute to place the file in a storage pool that uses the specified media type. Possible values are:

    - `HC`: Identifies high capacity storage media, which uses hard disk drives (HDDs).
    - `EF`: Identifies extreme flash storage media, which uses flash devices.
  - `--redundancy`: Specifies the number of data copies that are maintained. Possible values are:
    - `normal`: Maintains 2 mirrored copies of the file data.
    - `high`: Maintains 3 mirrored copies of the file data.
  - `--content-type`: Specifies the type of content in the file. Exascale internally uses this attribute to place file extents on physically separate devices in a way that maximizes availability if failures occurs. Possible values are:
    - `DATA`
    - `RECO`

  The command syntax is:

  ```
  @> mktemplate --vault vault-name --file-type file-type --content-type
  content-type --media-type media-type --redundancy redundancy
  ```

- To create a user-defined vault template, use the mktemplate command with the `--vault` and `--name` options and specify the file storage attribute settings. The command syntax is:

  ```
  @> mktemplate --vault vault-name --name template-name --content-type
  content-type --media-type media-type --redundancy redundancy
  ```

3.7.2 List Vault Templates

A vault template is a named collection of file storage attribute settings defined at the vault level and used when creating new files in the vault. By using vault templates, you can manage file storage attributes automatically and consistently across the vault. A vault template overrides the settings in the corresponding cluster template.

To display information about Exascale vault templates, use the ESCLI lstemplate command with the `--vault` option. For example:

```
@> lstemplate --vault vault-name
```

This simple form of the command displays basic information about all templates that are associated with the specified vault (*vault-name*), including the vault-specific templates and any cluster-level templates not overridden by vault-specific templates.

Using the command options available with the lstemplate command, you can specify the templates you want to display and display output with different levels of detail.

3.7.3 Modify a Vault Template

A vault template is a named collection of file storage attribute settings defined at the vault level and used when creating new files in the vault. By using vault templates, you can manage file storage attributes automatically and consistently across the vault. A vault template overrides the settings in the corresponding cluster template.

- To modify an existing vault template that is associated with a file type, use the ESCLI chtemplate command with the `--vault` and `--file-type` options and specify the file storage attribute settings:
  - `--media-type`: Specifies the physical media type that is used to store the file. Exascale uses this attribute to place the file in a storage pool that uses the specified media type. Possible values are:
    - `HC`: Identifies high capacity storage media, which uses hard disk drives (HDDs).
    - `EF`: Identifies extreme flash storage media, which uses flash devices.
  - `--redundancy`: Specifies the number of data copies that are maintained. Possible values are:
    - `normal`: Maintains 2 mirrored copies of the file data.
    - `high`: Maintains 3 mirrored copies of the file data.
  - `--content-type`: Specifies the type of content in the file. Exascale internally uses this attribute to place file extents on physically separate devices in a way that maximizes availability if failures occurs. Possible values are:
    - `DATA`
    - `RECO`

  The command syntax is:

  ```
  @> chtemplate --vault vault-name --file-type file-type --content-type
  content-type --media-type media-type --redundancy redundancy
  ```

- To modify a user-defined vault template, use the chtemplate command with the `--vault` and `--name` options and specify the file storage attribute settings. The command syntax is:

  ```
  @> chtemplate --vault vault-name --name template-name --content-type
  content-type --media-type media-type --redundancy redundancy
  ```

3.7.4 Remove a Vault Template

A vault template is a named collection of file storage attribute settings defined at the vault level and used when creating new files in the vault. By using vault templates, you can manage file storage attributes automatically and consistently across the vault. A vault template overrides the settings in the corresponding cluster template.

To remove a vault template, use the ESCLI rmtemplate command with the `--vault` option as follows:

- To remove a template that is associated with a file type, use the `--file-type` option. For example:

  ```
  @> rmtemplate --vault vault-name --file-type file-type
  ```

- To remove a user-defined vault template, use the `--name` option. For example:

  ```
  @> rmtemplate --vault vault-name --name template-name
  ```

3.8 Administer Resource Profiles

- Create a Resource Profile
- List Resource Profiles
- Modify a Resource Profile
- Remove a Resource Profile

Related Topics

- Resource Management

3.8.1 Create a Resource Profile

Resource profiles govern how resources are managed within a vault. Each resource profile defines a set of resource limits and settings, which can be associated with various Exascale clients (Oracle databases and block store volumes).

To create a regular resource profile, use the ESCLI mkresourceprofile command and specify:

- The name of the resource profile.
- The name of the vault that the resource profile governs.
- An attribute list that defines the resource limits and settings. The following list outlines all of the available attributes:
  - `iopsShareEF`: Specifies the relative share of I/O bandwidth (IOPS) from extreme flash (EF) storage media that is available to each client associated with the resource profile. Each client's share is relative to all other client shares. A higher share value implies higher priority. The range of valid values is `1-100`, and the default is `1`.
  - `iopsLimitEF`: Specifies the upper limit of the I/O bandwidth (IOPS) from extreme flash (EF) storage media that is available to each client associated with the resource profile. The value represents a fraction out of 10000. The range of valid values is `1-10000`. The default value is `10000` (effectively unlimited).
  - `iopsShareHC`: Specifies the relative share of I/O bandwidth (IOPS) from high capacity (HC) storage media that is available to each client associated with the resource profile. Each client's share is relative to all other client shares. A higher share value implies higher priority. The range of valid values is `1-100`, and the default is `1`.
  - `iopsLimitHC`: Specifies the upper limit of the I/O bandwidth (IOPS) from high capacity (HC) storage media that is available to each client associated with the resource profile. The value represents a fraction out of 10000. The range of valid values is `1-10000`. The default value is `10000` (effectively unlimited).
  - `enableFlashCache`: Enables or disables use of flash cache for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).
  - `flashCacheMin`: Specifies the guaranteed minimum fraction of flash cache available to each client associated with the resource profile. The range of valid values is `0-10000`. The default value is `0` (no set minimum). If the sum of all values across all resource profiles exceeds 10000, then all values are proportionally scaled down. This option is valid only when `enableFlashCache=true`.
  - `flashCacheMax`: Specifies the maximum fraction of flash cache available to each client associated with the resource profile. This option is valid only when

`enableFlashCache=true`. The range of valid values is `0-10000`. The default value is `10000`. This option is valid only when `enableFlashCache=true`.

- `enableXrmemCache`: Enables or disables use of Exadata RDMA Memory Cache (XRMEM cache) for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).
- `xrmemCacheMin`: Specifies the guaranteed minimum fraction of XRMEM cache available to each client associated with the resource profile. The range of valid values is `0-10000`. The default value is `0` (no set minimum). If the sum of all values across all resource profiles exceeds 10000, then all values are proportionally scaled down. This option is valid only when `enableXrmemCache=true`.
- `xrmemCacheMax`: Specifies the maximum fraction of XRMEM cache available to each client associated with the resource profile. This option is valid only when `enableXrmemCache=true`. The range of valid values is `0-10000`. The default value is `10000`. This option is valid only when `enableXrmemCache=true`.
- `enableFlashLog`: Enables or disables use of flash log for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).

For example:

```
@> mkresourceprofile myvaultname/myresourceprofilename --attributes
iopsShareHC=40,iopsLimitHC=6000,enableXrmemCache=false
```

In addition to regular user-defined resource profiles, you can also create a system-reserved resource profile named `$UNASSIGNED`. All Exascale clients not explicitly associated with a resource profile are automatically governed by the `$UNASSIGNED` profile. The `$UNASSIGNED` resource profile contains only two modifiable attributes:

- `flashCacheMax`: Specifies the maximum fraction of flash cache shared by clients associated with the `$UNASSIGNED` profile. The range of valid values is `0-10000`. The default value is `10000`.
- `xrmemCacheMax`: Specifies the maximum fraction of XRMEM cache shared by clients associated with the `$UNASSIGNED` profile. The range of valid values is `0-10000`. The default value is `10000`.

For example:

```
@> mkresourceprofile myvaultname/$UNASSIGNED --attributes
flashCacheMax=3000,xrmemCacheMax=2000
```

All Exascale clients governed by the `$UNASSIGNED` profile share the corresponding cache resources. The behavior differs from regular resource profiles, where each application of the resource profile defines the resource allocation for one associated client.

If the `$UNASSIGNED` resource profile does not exist, all unassigned Exascale clients share any unassigned flash cache space and XRMEM cache space. If there is no unassigned space to share, the system automatically reserves 5% of the cache space for unassigned Exascale clients.

3.8.2 List Resource Profiles

Resource profiles govern how resources are managed within a vault. Each resource profile defines a set of resource limits and settings, which can be associated with various Exascale clients (Oracle databases and block store volumes).

To display information about resource profiles, use the ESCLI lsresourceprofile command.

For example, the following command displays basic information about all resource profiles associated with the specified vault (*myvault*):

```
@> lsresourceprofile --filter vault=myvault
```

And the following command displays detailed information about the specified resource profile (*myvault*/*myresourceprofile*):

```
@> lsresourceprofile myvault/myresourceprofile --detail
```

Using the command options available with the lsresourceprofile command, you can also:

- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

3.8.3 Modify a Resource Profile

Resource profiles govern how resources are managed within a vault. Each resource profile defines a set of resource limits and settings, which can be associated with various Exascale clients (Oracle databases and block store volumes).

To modify a resource profile, use the ESCLI chresourceprofile command and specify:

- The name of the resource profile.
- The name of the vault that the resource profile governs.
- An attribute list that defines the resource limits and settings that you want to modify. Unspecified attributes are not modified. The following list outlines all of the available attributes:
  - `iopsShareEF`: Specifies the relative share of I/O bandwidth (IOPS) from extreme flash (EF) storage media that is available to each client associated with the resource profile. Each client's share is relative to all other client shares. A higher share value implies higher priority. The range of valid values is `1-100`, and the default is `1`.
  - `iopsLimitEF`: Specifies the upper limit of the I/O bandwidth (IOPS) from extreme flash (EF) storage media that is available to each client associated with the resource profile. The value represents a fraction out of 10000. The range of valid values is `1-10000`. The default value is `10000` (effectively unlimited).
  - `iopsShareHC`: Specifies the relative share of I/O bandwidth (IOPS) from high capacity (HC) storage media that is available to each client associated with the resource profile. Each client's share is relative to all other client shares. A higher share value implies higher priority. The range of valid values is `1-100`, and the default is `1`.

- `iopsLimitHC`: Specifies the upper limit of the I/O bandwidth (IOPS) from high capacity (HC) storage media that is available to each client associated with the resource profile. The value represents a fraction out of 10000. The range of valid values is `1-10000`. The default value is `10000` (effectively unlimited).
- `enableFlashCache`: Enables or disables use of flash cache for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).
- `flashCacheMin`: Specifies the guaranteed minimum fraction of flash cache available to each client associated with the resource profile. The range of valid values is `0-10000`. The default value is `0` (no set minimum). If the sum of all values across all resource profiles exceeds 10000, then all values are proportionally scaled down. This option is valid only when `enableFlashCache=true`.
- `flashCacheMax`: Specifies the maximum fraction of flash cache available to each client associated with the resource profile. This option is valid only when `enableFlashCache=true`. The range of valid values is `0-10000`. The default value is `10000`. This option is valid only when `enableFlashCache=true`.
- `enableXrmemCache`: Enables or disables use of Exadata RDMA Memory Cache (XRMEM cache) for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).
- `xrmemCacheMin`: Specifies the guaranteed minimum fraction of XRMEM cache available to each client associated with the resource profile. The range of valid values is `0-10000`. The default value is `0` (no set minimum). If the sum of all values across all resource profiles exceeds 10000, then all values are proportionally scaled down. This option is valid only when `enableXrmemCache=true`.
- `xrmemCacheMax`: Specifies the maximum fraction of XRMEM cache available to each client associated with the resource profile. This option is valid only when `enableXrmemCache=true`. The range of valid values is `0-10000`. The default value is `10000`. This option is valid only when `enableXrmemCache=true`.
- `enableFlashLog`: Enables or disables use of flash log for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).

For example:

```
@> chresourceprofile myvaultname/myresourceprofilename --attributes
iopsShareHC=30,enableXrmemCache=true
```

In addition to regular user-defined resource profiles, you can also modify the system-reserved resource profile named `$UNASSIGNED`. All Exascale clients not explicitly associated with a resource profile are automatically governed by the `$UNASSIGNED` profile. The `$UNASSIGNED` resource profile contains only two modifiable attributes:

- `flashCacheMax`: Specifies the maximum fraction of flash cache shared by clients associated with the `$UNASSIGNED` profile. The range of valid values is `0-10000`. The default value is `10000`.
- `xrmemCacheMax`: Specifies the maximum fraction of XRMEM cache shared by clients associated with the `$UNASSIGNED` profile. The range of valid values is `0-10000`. The default value is `10000`.

For example:

```
@> chresourceprofile myvaultname/$UNASSIGNED --attributes
flashCacheMax=4000,xrmemCacheMax=1000
```

All Exascale clients governed by the `$UNASSIGNED` profile share the corresponding cache resources. The behavior differs from regular resource profiles, where each application of the resource profile defines the resource allocation for one associated client.

You must create the `$UNASSIGNED` resource profile before you can modify it. If the `$UNASSIGNED` resource profile does not exist, all unassigned Exascale clients share any unassigned flash cache space and XRMEM cache space. If there is no unassigned space to share, the system automatically reserves 5% of the cache space for unassigned Exascale clients.

3.8.4 Remove a Resource Profile

Resource profiles govern how resources are managed within a vault. Each resource profile defines a set of resource limits and settings, which can be associated with various Exascale clients (Oracle databases and block store volumes).

To delete a resource profile, use the ESCLI rmresourceprofile command and specify the resource profile name and the associated vault name.

For example:

```
@> rmresourceprofile myvaultname/myresourceprofile
```

3.9 Administer Exascale Block Store Data Objects

- Administer Volumes
- Administer Volume Attachments
- Administer Volume Snapshots
- Administer Volume Clones
- Administer Volume Backups
- Administer an Advanced Cluster File System (ACFS) on Exascale

Related Topics

- Exascale Block Store

3.9.1 Administer Volumes

- Create a Volume
- List Volumes
- Modify a Volume
- Remove a Volume

Related Topics

- Exascale Volumes

3.9.1.1 Create a Volume

An Exascale block volume is an arbitrary-sized allocation of storage space, which can be used as an Exascale Direct Volume (EDV) attachment or iSCSI target.

To create an Exascale block volume, use the ESCLI mkvolume command. For example:

```
@> mkvolume  1TB --vault myvaultname --attributes name=myvolname
```

As shown in the example, a typical form of the command specifies:

- The amount of storage space that is allocated to the volume.

  > **Note:**
  >
  > If you intend to use the volume to support Oracle ACFS, note that ACFS requires a minimum volume size of 512 MB.

- The vault where the volume is stored.
- The name for the volume, which makes it easier for you to identify it later.

You can also specify a series of optional attributes, which define the detailed characteristics of the volume or constrain the system resources that the volume is allowed to consume. For instance, you can specify the physical media type used to store the volume. By default, volumes are stored on hard disk drives (`HC` media type).

The following example shows creating a volume that is 100 GB in size and located in the vault named `MYVOLS`. The example also includes the following specific attribute settings:

- The volume name is set to `vol9`.
- The volume is created on Extreme Flash (`EF`) storage media.
- The volume is provisioned with 1000 IOPS (IOs per second).
- The volume is created with two owners (Exascale users named `scott` and `dave`).

```
@> mkvolume 100g --vault MYVOLS --attributes
name=vol9,mediaType=EF,iopsProvisioned=1000,owners=scott,dave
```

After you create a volume, you must create a volume attachment to use the volume. An unattached volume is not available to any clients and is useless in isolation.

> **Note:**
>
> For simple cases, you can streamline the creation of an Exascale volume and associated Exascale Direct Volume (EDV) attachment by using the edvmkvol command.

3.9.1.2 List Volumes

An Exascale block volume is an arbitrary-sized allocation of storage space, which can be used as an Exascale Direct Volume (EDV) attachment or iSCSI target.

To display information about existing volumes, use the ESCLI lsvolume command. For example:

```
@> lsvolume
```

By default, without any optional arguments, the command displays basic information about all of the volumes associated with the Exascale cluster.

Using the command options available with the lsvolume command, you can:

- Specify the volumes that you want to display.
- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

3.9.1.3 Modify a Volume

An Exascale block volume is an arbitrary-sized allocation of storage space, which can be used as an Exascale Direct Volume (EDV) attachment or iSCSI target.

To modify an existing Exascale block volume, use the ESCLI chvolume command and specify:

- The volume identifier. You can use the lsvolume command to find the identifier for each volume.
- The attributes that you want to change.

For example:

- The following command sets the size of the specified volume to 200 GB:

  ```
  @> chvolume 2:50e52177583f4be4bad68ac20b65001e --attributes size=200g
  ```

- A volume can have up to two owners. The following command sets *scott* as the sole owner of the specified volume:

  ```
  @> chvolume 2:50e52177583f4be4bad68ac20b65001e --attributes owners=scott
  ```

- The following command adds *peter* as an owner of the specified volume:

  ```
  @> chvolume 2:50e52177583f4be4bad68ac20b65001e --attributes owners=+peter
  ```

- The following command removes *dave* as an owner of the specified volume:

  ```
  @> chvolume 2:50e52177583f4be4bad68ac20b65001e --attributes owners=-dave
  ```

3.9.1.4 Remove a Volume

An Exascale block volume is an arbitrary-sized allocation of storage space, which can be used as an Exascale Direct Volume (EDV) attachment or iSCSI target.

Removing a volume irreversibly deletes its contents. You cannot remove a volume with current attachments. You must remove all attachments to the volume before the volume can be removed.

To remove an existing volume, use the ESCLI rmvolume command and specify the identifier for the volume that is being removed. For example:

```
@> rmvolume volume-id
```

You can use the lsvolume command to find the identifier for each volume.

> **Note:**
>
> You can use the edvrmvol command to simplify the removal of an Exascale Direct Volume (EDV) attachment and associated Exascale volume that were created using the edvmkvol command.

Related Topics

- Remove an EDV Attachment
- Remove an iSCSI Attachment

3.9.2 Administer Volume Attachments

A volume attachment enables the use of an Exascale volume. There are two types of volume attachment.

- Administer EDV Attachments
- Administer iSCSI Attachments

Related Topics

- Volume Attachments

3.9.2.1 Administer EDV Attachments

An Exascale Direct Volume (EDV) attachment can be used as a raw block storage device or to support a file system. EDV is the default and recommended volume attachment type for cases where clients want to use volumes within the Exadata RDMA Network Fabric. To implement Oracle Advanced Cluster File System (ACFS) on Exascale block volume storage, you must use an EDV attachment.

- Create an EDV Attachment
- List EDV Attachments
- Remove an EDV Attachment

3.9.2.1.1 Create an EDV Attachment

To use a volume, you must create a volume attachment. An Exascale Direct Volume (EDV) attachment can be used as a raw block storage device or to support a file system. To implement Oracle Advanced Cluster File System (ACFS) on Exascale block volume storage, you must use an EDV attachment.

> **Note:**
>
> During initial system deployment with Oracle Exadata Deployment Assistant (OEDA), the Exascale Direct Volume (EDV) service is configured on each Exadata compute node (bare-metal or VM) and is related to the Exascale user that manages the Oracle Grid Infrastructure (GI) cluster. To create an EDV attachment, you must use the Exascale user linked with the EDV service.
>
> If the GI cluster uses a non-role-separated user configuration with one Oracle OS user account, then the associated Exascale user is related to the EDV service. If the GI cluster uses a role-separated configuration with a Grid OS user account and an Oracle OS user account, then the EDV service is linked to the Exascale user associated with the Grid OS account.
>
> To find the Exascale user linked with the EDV service, use the ESCLI lsinitiator command with the `--detail` option and examine the `user` attribute.

To create an EDV attachment, connect to ESCLI as the Exascale user linked with the EDV service. Then, run the mkvolumeattachment command and specify:

- The volume identifier. You can use the lsvolume command to find the identifier for each volume.
- The device name to use in conjunction with the attachment. This is a user-supplied name, which is applied to the device file that is associated with the attachment. After attachment, the corresponding device file is located under `/dev/exc/`.
- The cluster identifier (`giClusterId`) or specific EDV initiator identifier (`initiator`) where the attachment resides. You can use the lsinitiator command to find cluster and EDV initiator identifiers.

  If you create a cluster-wide attachment, the EDV device file is created on every node in the Oracle Grid Infrastructure (GI) cluster. If you create a node-specific attachment, the corresponding EDV device file is only created on the node associated with the specified EDV initiator.

For example, the following command creates a node-specific EDV attachment:

```
@> mkvolumeattachment 1:8e4fbaff261440b493e0a5e5e6808e66 myvol --attributes
initiator=1fd9b363-079a-84e2-1fd9-b363079a84e2
```

In the example:

- The volume identifier is *1:8e4fbaff261440b493e0a5e5e6808e66*.
- The device name to use in conjunction with the attachment is *myvol*, and the corresponding device file is located at `/dev/exc/myvol`.
- The EDV initiator identifier is *1fd9b363-079a-84e2-1fd9-b363079a84e2*. This value identifies a specific cluster node (server) that hosts the EDV attachment.

Note:

- Each EDV attachment also has a kernel device file at `/dev/exc-devN`, where *N* is the minor number of the device. The kernel device name is contained as an attribute of the EDV attachment and is visible using the ESCLI lsvolumeattachment command. The relationship between the kernel device file and the user-named device file (under `/dev/exc/`) is also recorded in the udev database and is visible using the following Linux command:

  ```
  # udevadm info device-file
  ```

  For the *device-file* value, you can specify either the kernel device file (`/dev/exc-devN`) or the user-named device file (under `/dev/exc/`).

- By default, read and write access to EDV device files is only available to the `root` operating system user and members of the `disk` group. Depending on your use case, you may need to modify the permissions on the EDV device files before using them.

  For example, to make the EDV device file at `/dev/exc/myvol` readable and writable by the `oracle` user and `dba` group, you could configure it using a udev rule similar to the following:

  ```
  # cat /etc/udev/rules.d/57-edv-user.rules
  KERNEL=="exc-*", ENV{EXC_ALIAS}=="myvol", OWNER="oracle",
  GROUP="dba", MODE="0660"
  ```

- To facilitate the management of udev rules related to EDV devices, each EDV client node is configured with a template udev rules file at `/etc/udev/rules.d/57-edv-user.rules`, which you can modify to fulfill your requirements. To maintain existing udev rules, `/etc/udev/rules.d/57-edv-user.rules` is preserved whenever the EDV client software is updated.
- Each EDV client node can support a maximum of 1024 attachments at the same time. This limit includes the total of all cluster attachments involving the server, as well as local attachments specific to the server.
- For simple cases, you can streamline the creation of an Exascale volume and associated Exascale Direct Volume (EDV) attachment by using the edvmkvol command.

3.9.2.1.2 List EDV Attachments

An Exascale Direct Volume (EDV) attachment can be used as a raw block storage device or to support a file system, such as Oracle Advanced Cluster File System (ACFS).

To display information about EDV attachments, use the ESCLI lsvolumeattachment command. For example:

```
@> lsvolumeattachment
```

By default, without any optional arguments, the command displays basic information about all of the EDV attachments associated with the Exascale cluster.

Using the command options available with the lsvolumeattachment command, you can:

- Specify the attachments that you want to display.
- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

3.9.2.1.3 Remove an EDV Attachment

An Exascale Direct Volume (EDV) attachment can be used as a raw block storage device or to support a file system, such as Oracle Advanced Cluster File System (ACFS).

Removing an EDV attachment removes only the ability for clients to access the volume. It does not modify or delete any of the volume contents.

To remove an existing EDV attachment, use the ESCLI rmvolumeattachment command and specify the EDV attachment identifier. For example:

```
@> rmvolumeattachment 1:90565644befb4c4abab1558b81b940eb
```

In the example, the EDV attachment identifier is *1:90565644befb4c4abab1558b81b940eb*. You can use the lsvolumeattachment command to find the identifier for each volume attachment.

> **Note:**
>
> You can use the edvrmvol command to simplify the removal of an Exascale Direct Volume (EDV) attachment and associated Exascale volume that were created using the edvmkvol command.

3.9.2.2 Administer iSCSI Attachments

An iSCSI attachment enables the volume to be used as an iSCSI target.

- Create an iSCSI Attachment
- List iSCSI Attachments
- Remove an iSCSI Attachment

3.9.2.2.1 Create an iSCSI Attachment

An iSCSI attachment enables the use of an Exascale volume as an iSCSI device.

To create an iSCSI attachment, use the ESCLI mkvolumeattachment command and specify:

- The volume identifier.
- The identifier used by iSCSI initiators to identify the attachment.
- The iSCSI protocol option (`--protocol iscsi`).
- The CHAP (Challenge Handshake Authentication Protocol) attributes for authentication of iSCSI initiators.

For example:

```
@> mkvolumeattachment 2:390535212ea7448299bcc7bc7f7653b2
iqn.1988-10.com.oracle.vm01 --protocol iscsi
     --attributes chapUserId=X1234567-user-1,chapPassword=X1234567-
pass-1,chapMutualUserId=X1234567-user-2,chapMutualPassword=X1234567-pass-2
```

In the example, the volume identifier is *2:390535212ea7448299bcc7bc7f7653b2*, and the identifier used by iSCSI initiators to identify the attachment is *iqn.1988-10.com.oracle.vm01*. The required CHAP attributes are specified following the `--attributes` command option.

3.9.2.2.2 List iSCSI Attachments

An iSCSI attachment enables the use of an Exascale volume as an iSCSI device.

To display information about existing iSCSI attachments, use the ESCLI lsvolumeattachment command and specify the command option to display iSCSI attachments. For example:

```
@> lsvolumeattachment --protocol iscsi
```

As shown in the example, the simplest form of the command displays basic information about all of the iSCSI attachments associated with the Exascale cluster.

Using the command options available with the lsvolumeattachment command, you can also:

- Specify the attachments that you want to display.
- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

3.9.2.2.3 Remove an iSCSI Attachment

An iSCSI attachment enables the use of an Exascale volume as an iSCSI device.

Removing an iSCSI attachment only removes the access path to the underlying volume. It does not modify or delete any of the volume contents.

To remove an existing iSCSI attachment, use the ESCLI rmvolumeattachment command and specify:

- The attachment identifier. You can use the lsvolumeattachment command to find the identifier for each volume attachment.
- The identifier for the iSCSI initiator.
- The iSCSI protocol option (`--protocol iscsi`).

For example:

```
@> rmvolumeattachment 2-2:9a7bced09e514bd9bb701d0556020c1c
iqn.1988-10.com.oracle.vm01 --protocol iscsi
```

In the example, the attachment identifier is *2-2:9a7bced09e514bd9bb701d0556020c1c*, and the identifier for the iSCSI initiator is *iqn.1988-10.com.oracle.vm01*.

3.9.3 Administer Volume Snapshots

- Create a Volume Snapshot
- List Volume Snapshots
- Remove a Volume Snapshot

Related Topics

- Volume Snapshots

3.9.3.1 Create a Volume Snapshot

A volume snapshot is a logical read-only point-in-time copy of a volume.

To create a volume snapshot, use the ESCLI mkvolumesnapshot command. For example:

```
@> mkvolumesnapshot 2:390535212ea7448299bcc7bc7f7653b2
```

As shown in the preceding example, the simplest form of the command specifies the identifier for the volume that you want to copy. You can use the lsvolume command to find the identifier for each volume.

You can also use the `--attributes` option to specify various attributes for the volume snapshot. For example:

```
@> mkvolumesnapshot 1:2ee25b4c77fa423a87a25bd2160154de --attributes name=snap1
```

> **Note:**
>
> The `mkvolumesnapshot` command fails if the specified volume supports Exascale Direct Volume (EDV) attachments and Oracle Advanced Cluster File System (ACFS) is currently mounted. To create a volume snapshot in this situation, you must either:
>
> - Unmount the affected file system on all servers before you use the `mkvolumesnapshot` command. Then, mount the file system again afterward.
> - Use the Oracle ACFS command-line utility (`acfsutil`) to synchronize the file system and create the volume snapshot. On an EDV client node, run the `acfsutil volsnap create` command as the `root` user and specify the ACFS mountpoint or EDV device.
>
>   For example:
>
>   ```
>   # acfsutil volsnap create /mnt/myacfs
>   ```

After you create a volume snapshot, you must create an attachment to read the contents. By itself, an unattached volume snapshot is not available to any clients.

3.9.3.2 List Volume Snapshots

A volume snapshot is a logical read-only point-in-time copy of a volume.

To display information about existing volume snapshots, use the ESCLI lsvolumesnapshot command. For example:

```
@> lsvolumesnapshot
```

By default, without any optional arguments, the command displays basic information about all of the volume snapshots associated with the Exascale cluster.

Using the command options available with the lsvolumesnapshot command, you can:

- Specify the volume snapshots that you want to display.
- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

3.9.3.3 Remove a Volume Snapshot

A volume snapshot is a logical read-only point-in-time copy of a volume.

Removing a volume snapshot irreversibly deletes it. Exascale provides no means to recover a deleted volume snapshot.

You cannot remove a volume snapshot with current attachments. You must remove all attachments to the volume snapshot before the snapshot can be removed.

To remove an existing volume snapshot, use the ESCLI rmvolumesnapshot command and specify the identifier for the volume snapshot that is being removed. For example:

```
@> rmvolumesnapshot 2.1:3c0e9866d66345afbb316472f0bce825
```

You can use the lsvolumesnapshot command to find the identifier for each volume snapshot.

Related Topics

- Remove an EDV Attachment
- Remove an iSCSI Attachment

3.9.4 Administer Volume Clones

- Create a Volume Clone
- List Volume Clones
- Modify a Volume Clone
- Remove a Volume Clone

Related Topics

- Volume Clones

3.9.4.1 Create a Volume Clone

An Exascale volume clone is a thinly-provisioned read-write point-in-time copy of a volume snapshot.

You can create an Exascale volume clone by cloning a volume snapshot or by directly cloning an existing volume.

- Create a Volume Clone from a Volume Snapshot
- Create a Volume Clone Directly from an Existing Volume

3.9.4.1.1 Create a Volume Clone from a Volume Snapshot

To create an Exascale volume clone from an Exascale volume snapshot, use the ESCLI mkvolume command and specify the attributes for the volume clone. The command syntax is:

```
@> mkvolume --attributes volumeSnapshot=parent_snapshot_id[,name=clone_name]
[,iopsProvisioned=integer_value][,iopsInherited={true|false}]
```

In the command:

- `volumeSnapshot=parent_snapshot_id`: Identifies the volume snapshot that you want to clone. You can use the lsvolumesnapshot command to find the identifier for each volume snapshot.
- `name=clone_name`: Optionally specifies the name of the volume clone, which makes it easier for you to identify later. If not specified, a system-generated name is assigned.
- `iopsProvisioned=integer_value`: Optionally specifies the I/O bandwidth provisioned for the volume clone. The I/O bandwidth is expressed as the number of I/Os per second (IOPS).
- `iopsInherited={true|false}`: Optionally specifies whether the volume clone inherits I/O bandwidth from the nearest ancestor in the volume hierarchy with provisioned (not inherited) I/O bandwidth.

For more details, see the mkvolume command reference.

After you create a volume clone, you must create an attachment to access the contents. By itself, an unattached volume clone is not available to any clients.

3.9.4.1.2 Create a Volume Clone Directly from an Existing Volume

> **Note:**
>
> To create an Exascale volume clone directly from an existing Exascale volume, you must use Oracle Exadata System Software release 25.1.0 or later.

To create a volume clone directly from an existing volume, use the ESCLI mkvolume command and specify the attributes for the volume clone. The command syntax is:

```
@> mkvolume --attributes volumeSource=parent_volume_id[,name=clone_name]
[,iopsProvisioned=integer_value][,iopsInherited={true|false}]
```

In the command:

- `volumeSource=parent_volume_id`: Identifies the existing volume that you want to clone. You can use the lsvolume command to find the identifier for each existing volume.

- `name=clone_name`: Optionally specifies the name of the volume clone, which makes it easier for you to identify later. If not specified, a system-generated name is assigned.
- `iopsProvisioned=integer_value`: Optionally specifies the I/O bandwidth provisioned for the volume clone. The I/O bandwidth is expressed as the number of I/Os per second (IOPS).
- `iopsInherited={true|false}`: Optionally specifies whether the volume clone inherits I/O bandwidth from the nearest ancestor in the volume hierarchy with provisioned (not inherited) I/O bandwidth.

For more details, see the mkvolume command reference.

After you create a volume clone, you must create an attachment to access the contents. By itself, an unattached volume clone is not available to any clients.

3.9.4.2 List Volume Clones

An Exascale volume clone is a thinly-provisioned read-write point-in-time copy of a volume snapshot.

To display information about existing volumes, including volume clones, use the ESCLI lsvolume command.

To focus on snapshot-based volume clones, look for volumes that contain a value for the `volumeSnapshot` attribute. For example:

```
@> lsvolume --detail --filter volumeSnapshot!=""
```

Without any other arguments or command options, the previous command displays detailed information about all of the snapshot-based volume clones associated with the Exascale cluster.

Likewise, to focus on clones that are based directly on other volumes, look for volumes that contain a value for the `volumeSource` attribute. For example:

```
@> lsvolume --detail --filter volumeSource!=""
```

Using the other command options available with the lsvolume command, you can:

- Display information about specific volumes or volume clones.
- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

3.9.4.3 Modify a Volume Clone

An Exascale volume clone is a thinly-provisioned read-write point-in-time copy of a volume snapshot.

To modify an existing Exascale volume clone, use the ESCLI chvolume command and specify:

- The volume identifier for the volume clone. You can use the lsvolume command to find the volume identifier for each volume clone.
- The attributes that you want to change.

For a volume clone, you can modify the following attributes:

- `name=clone_name`: Modifies the name of the volume clone.
- `iopsProvisioned=integer_value`: Modifies the I/O bandwidth provisioned for the volume clone. The I/O bandwidth is expressed as the number of I/Os per second (IOPS).
- `iopsInherited={true|false}`: Specifies whether the volume clone inherits I/O bandwidth from the nearest ancestor in the volume hierarchy with provisioned I/O bandwidth (not inherited).

  If you change `iopsInherited` from `true` to `false`, then the volume clone is governed by the `iopsProvisioned` setting. If `iopsProvisioned` is not set to a value, then the default value is `unlimited`.

  If you change `iopsInherited` from `false` to `true`, then the volume clone inherits I/O bandwidth from its nearest ancestor and any previous `iopsProvisioned` setting is nullified.

For example, the following command changes the name of the specified volume clone:

```
@> chvolume 2:50e52177583f4be4bad68ac20b65001e --attributes name=myNewName
```

3.9.4.4 Remove a Volume Clone

An Exascale volume clone is a thinly-provisioned read-write point-in-time copy of a volume snapshot.

Removing a volume clone irreversibly deletes its unique contents but does not affect the underlying volume snapshot. You cannot remove a volume clone with current attachments. You must remove all attachments to the volume clone before the clone can be removed.

To remove an existing volume clone, use the ESCLI rmvolume command and specify the volume identifier for the volume clone that is being removed. For example:

```
@> rmvolume volume-id
```

You can use the lsvolume command to find the volume identifier for each volume clone.

Related Topics

- Remove an EDV Attachment
- Remove an iSCSI Attachment

3.9.5 Administer Volume Backups

- Create a Volume Backup
- List Volume Backups
- Restore a Volume Backup
- Monitor the Progress of a Volume Backup or Restore
- Remove a Volume Backup

Related Topics

- Volume Backups

3.9.5.1 Create a Volume Backup

A volume backup is a backup of an Exascale volume snapshot, which provides a consistent point-in-time copy of the volume. Exascale volume backups are stored in an Oracle Cloud Infrastructure (OCI) object storage container that is associated with the Exascale cluster.

To create an Exascale volume backup, use the ESCLI mkvolumebackup command. For example:

```
@> mkvolumebackup --attributes
volumeSnapshot=2.1:3c0e9866d66345afbb316472f0bce825,name=myBackup1
```

As shown in the example, the command must specify the identifier for the volume snapshot that you want to back up. You can use the lsvolumesnapshot command to find the identifier for each volume snapshot. In the example, the volume backup is also named *myBackup1*. Setting the name of the volume backup is optional.

The command completes at the beginning of the backup operation. See also Monitor the Progress of a Volume Backup or Restore.

3.9.5.2 List Volume Backups

A volume backup is a backup of an Exascale volume snapshot, which provides a consistent point-in-time copy of the volume. Exascale volume backups are stored in an Oracle Cloud Infrastructure (OCI) object storage container that is associated with the Exascale cluster.

To display information about existing Exascale volume backups, use the ESCLI lsvolumebackup command. For example:

```
@> lsvolumebackup
```

By default, without any optional arguments, the command displays basic information about all of the volume backups associated with the Exascale cluster.

Using the command options available with the lsvolumebackup command, you can:

- Specify the volume backups that you want to display.
- Filter the output according to your specified conditions.
- Sort the output using specific attributes.
- Choose to display output with different levels of detail.

3.9.5.3 Restore a Volume Backup

A volume backup is a backup of an Exascale volume snapshot, which provides a consistent point-in-time copy of the volume. Exascale volume backups are stored in an Oracle Cloud Infrastructure (OCI) object storage container that is associated with the Exascale cluster.

You can restore a volume backup into a new volume. When you restore a volume backup, the restored volume is separate from the volume backup and the original backup source volume. There is no on-going association between a restored volume and the volume backup or the original backup source volume.

To restore a volume backup, use the ESCLI mkvolume command and specify:

- The volume backup identifier to use as the source for the new volume. You can use the lsvolumebackup command to find the identifier for each volume backup.
- The vault that hosts the new volume.

For example:

```
@> mkvolume --attributes
volumeBackup=2b1:7476692b2499488794951dfe59606e1b,vault=myvault
```

In the example, the volume backup identifier is *2b1:7476692b2499488794951dfe59606e1b* and the vault that hosts the restored volume is *myvault*.

The command completes at the beginning of the restore operation, which may take some time to complete. See Monitor the Progress of a Volume Backup or Restore.

3.9.5.4 Monitor the Progress of a Volume Backup or Restore

A volume backup is a backup of an Exascale volume snapshot, which provides a consistent point-in-time copy of the volume. Exascale volume backups are stored in an Oracle Cloud Infrastructure (OCI) object storage container.

The operation to create or restore a volume backup may take some time to complete. You can monitor the ongoing progress by viewing the completion percentage.

- To display the completion percentage for a volume backup operation, use the ESCLI lsvolumebackup command to display information about the desired volume backup operation and examine the `state` and `progress` attributes. For example:

  ```
  @> lsvolumebackup 2b1:7476692b2499488794951dfe59606e1b --attributes
  state,progress --detail
  state           CREATING
  progress        27%

  @>
  ```

  In the example, *2b1:7476692b2499488794951dfe59606e1b* is the volume backup identifier for the backup operation being monitored.

  You can also use the lsvolumebackup command to find the identifier for each volume backup.

- To display the completion percentage for a volume restore operation, use the ESCLI lsvolume command to display information about the new volume that is being created as part of the volume restore operation and examine the `state` and `progress` attributes. For example:

  ```
  @> lsvolume 3:4f71bc664cdb47e3928083505819aafa --attributes state,progress
  --detail
  state           RESTORING
  progress        59%

  @>
  ```

  In the example, the identifier for the new volume being created by the restore operation is *3:4f71bc664cdb47e3928083505819aafa*.

You can also use the lsvolume command to find the identifier for each volume being created as part of the volume restore operation.

During a backup operation, the `state` attribute contains the value `CREATING`. During a restore operation, the `state` attribute contains the value `RESTORING`. During both operations, the `progress` attribute contains the current completion percentage.

Upon successful completion, the `state` changes to `AVAILABLE`, and the `progress` attribute contains no value. If the operation fails, then the `state` attribute indicates the failure, and the `progress` attribute reflects the point of failure for a while before it is reset.

3.9.5.5 Remove a Volume Backup

A volume backup is a backup of an Exascale volume snapshot, which provides a consistent point-in-time copy of the volume. Exascale volume backups are stored in an Oracle Cloud Infrastructure (OCI) object storage container that is associated with the Exascale cluster.

To remove an existing volume backup, use the ESCLI rmvolumebackup command and specify the backup identifier. The command syntax is:

```
@> rmvolumebackup 2b1:7476692b2499488794951dfe59606e1b
```

You can use the lsvolumebackup command to find the identifier for each volume backup.

3.9.6 Administer an Advanced Cluster File System (ACFS) on Exascale

- Create an ACFS File System on Exascale
- List ACFS File Systems on Exascale
- Modify an ACFS File System on Exascale
- Remove an ACFS File System on Exascale

Related Topics

- Oracle Advanced Cluster File System (ACFS) on Exascale
- About Oracle ACFS

3.9.6.1 Create an ACFS File System on Exascale

Exascale contains integrated support for Oracle Advanced Cluster File System (ACFS) on Exascale block storage using Exascale Direct Volumes (EDV).

Before you create an Exascale-managed ACFS file system, you must create the Exascale volume to store the file system. The specified volume must have an associated EDV attachment, otherwise the operation cannot proceed.

The configuration of the EDV attachment governs how the file system is implemented. If the EDV attachment is a cluster-wide attachment, the ACFS file system is mounted on every node in the Oracle Grid Infrastructure (GI) cluster. If the EDV attachment is a node-specific attachment, the file system is mounted only on that node. In all cases, the ACFS details are registered with the GI cluster, and the file system is automatically mounted (or remounted) by Oracle Clusterware as required.

To create an Exascale-managed ACFS file system, use the ESCLI mkacfsfilesystem command and specify:

- The identifier of the Exascale volume that you want to store the file system. You can use the lsvolume command to find the identifier for each Exascale volume.
- The mount point location where you want to mount the file system.
- Optional attributes that define additional characteristics of the file system.

For example:

```
@> mkacfsfilesystem 1:bbd6fb4c75e2411b9bf366fe702eabaf /mnt/acfs1 --
attributes mountReadOnly=true
```

In the example:

- The volume identifier is *1:bbd6fb4c75e2411b9bf366fe702eabaf*.
- The file system mount point is */mnt/acfs1*.
- The file system is mounted in read-only mode by specifying the optional attribute setting: `mountReadOnly=true`.

Before using the `mkacfsfilesystem` command, ensure that ACFS is configured appropriately on the target system.

For instance, you can create a file system using ACFS encryption by specifying the `encryptionEnabled`, `encryptionAlgorithm`, and `encryptionKeyLength` attributes in the `mkacfsfilesystem` command. For example:

```
@> mkacfsfilesystem 1:bbd6fb4c75e2411b9bf366fe702eabaf /mnt/acfs1
    --attributes
encryptionEnabled=true,encryptionAlgorithm=AES,encryptionKeyLength=192
```

However, the command fails if ACFS encryption is not initialized on the target system. To initialize ACFS encryption, the system administrator must run the following command before the file system is created:

```
# acfsutil encr init
```

3.9.6.2 List ACFS File Systems on Exascale

Exascale contains integrated support for Oracle Advanced Cluster File System (ACFS) on Exascale block storage using Exascale Direct Volumes (EDV).

To display information about existing Exascale-managed ACFS file systems, use the ESCLI lsacfsfilesystem command. For example:

```
@> lsacfsfilesystem
```

By default, without any optional arguments, the command displays basic information about all of the Exascale-managed ACFS file systems associated with the Exascale cluster.

Using the command options available with the lsacfsfilesystem command, you can:

- Specify the file systems that you want to display.
- Filter the output according to your specified conditions.
- Sort the output using specific attributes.

- Choose to display output with different levels of detail.

3.9.6.3 Modify an ACFS File System on Exascale

Exascale contains integrated support for Oracle Advanced Cluster File System (ACFS) on Exascale block storage using Exascale Direct Volumes (EDV).

To modify an existing Exascale-managed ACFS file system:

1. Use the ESCLI acfsctl command to deregister the file system.

   Deregistering an Exascale-managed ACFS file system dismounts and removes the Oracle Grid Infrastructure (GI) registration for the host or cluster associated with the file system. The operation also removes the mount point directory from the host or GI cluster members. However, the file system contents remain in the underlying Exascale volume.

   To deregister an Exascale-managed ACFS file system, use the `acfsctl deregister` command and specify the file system identifier. For example:

   ```
   @> acfsctl deregister 1:2fb06f13cddd4a8d8d636d1f794046cb
   ```

   You can use the lsacfsfilesystem command to find the identifier for each Exascale-managed ACFS file system.

   You can also bypass errors in the deregistration process by adding the `--force` option.

2. After deregistering the file system, you can optionally remove and re-create the EDV attachment. See Administer EDV Attachments.

   Re-creating the EDV attachment enables you to move the attachment to a different host or GI cluster, which effectively moves the file system when it is registered again.

3. Use the ESCLI acfsctl command to register the file system.

   During file system registration, you must specify:

   - The identifier of the Exascale volume that stores the file system. You can use the lsacfsfilesystem command to find the volume identifier associated with each Exascale-managed ACFS file system.

     The specified volume must have an associated EDV attachment, otherwise the operation fails.

     Furthermore, the configuration of the EDV attachment governs how the file system is implemented. If the EDV attachment is a cluster-wide attachment, the ACFS file system is mounted on every node in the GI cluster. If the EDV attachment is a node-specific attachment, the file system is mounted only on that node. In all cases, the ACFS details are registered with the GI cluster, and the file system is automatically mounted (or remounted) by Oracle Clusterware as required.

   - The mount point location where you want to mount the file system.

     The specified mount point location may differ from the location used in previous registrations.

   During file system registration, you may specify optional attributes that govern how the file system is mounted, such as mounting it in read-only mode or read-write mode. However, you cannot change fundamental settings relating to the file system, such as changing the file system encryption setting or the metadata block size.

For example:

```
@> acfsctl register 1:bbd6fb4c75e2411b9bf366fe702eabaf /mnt/acfs1 --
attributes mountReadOnly=true
```

In the example:

- The volume identifier is *1:bbd6fb4c75e2411b9bf366fe702eabaf*.
- The file system mount point is */mnt/acfs1*.
- The file system is mounted in read-only mode by specifying the optional attribute setting: `mountReadOnly=true`.

3.9.6.4 Remove an ACFS File System on Exascale

Exascale contains integrated support for Oracle Advanced Cluster File System (ACFS) on Exascale block storage using Exascale Direct Volumes (EDV).

To delete and remove an existing Exascale-managed ACFS file system, use the ESCLI rmacfsfilesystem command and specify the identifier for the file system that is being removed. For example:

```
@> rmacfsfilesystem acfs-id
```

You can use the lsacfsfilesystem command to find the identifier for each Exascale-managed ACFS file system.

If the underlying EDV attachment is a node-specific attachment, the ACFS file system is dismounted and removed from the node containing the attachment. If the EDV attachment is a cluster-wide attachment, the file system is dismounted on every node in the Oracle Grid Infrastructure (GI) cluster. In all cases, the GI cluster registration is removed. Furthermore, the file system is removed from the underlying volume, effectively deleting the files in the file system. The command does not alter the underlying volume or volume attachment in any other way.

4

Using Exascale with Oracle Grid Infrastructure and Oracle Database

This chapter covers requirements and tasks for using Exascale storage to support Oracle Grid Infrastructure and Oracle Database.

This chapter contains the following topics:

4.1 Using Oracle Grid Infrastructure with Exascale

This topic outlines how Exascale storage natively supports Oracle Grid Infrastructure.

Oracle Grid Infrastructure (GI) includes two critical components that manage GI cluster configuration and node membership: Oracle Cluster Registry (OCR), and voting files. Starting with Oracle Grid Infrastructure 23ai release 23.5.0, you can use Exascale to natively store these critical files.

Following is a high-level overview of the procedure for configuring Oracle Grid Infrastructure in conjunction with Exascale. This procedure is automated for you when you use Oracle Exadata Deployment Assistant (OEDA) to create a VM cluster that uses Exascale storage.

1. Provision an Exascale user to manage the OCR and voting files.

   The same Exascale user must be used to access the OCR and voting files on all nodes in the GI cluster.

2. Create an Exascale vault for the OCR and voting files.

   You can dedicate a vault to store the OCR and voting files for a GI cluster, or you can use the vault to also store other files.

   To avoid potential confusion and compromised operations, you should not store the OCR and voting files for multiple GI clusters in one vault.

3. Install the Exascale user wallet into the Oracle Grid Infrastructure software on every node in the GI cluster.

4. Configure Oracle Grid Infrastructure to use the Exascale vault as the location for the OCR and voting files.

After configuration is completed, the OCR and voting files are located in the Exascale vault and can be accessed and managed like any Exascale file. Referencing the OCR and voting files on Exascale is essentially the same as using Oracle Automatic Storage Management (ASM). The fundamental difference is that you use an Exascale vault instead of an ASM disk group.

By default, the OCR and voting files are created with high redundancy (triple mirroring).

Related Topics

- Using Exascale with Other Oracle Releases

4.2 Using Oracle Database with Exascale

This topic outlines how Exascale storage natively supports Oracle Database.

Starting with Oracle Database 23ai release 23.5.0, you can use Exascale to natively store Oracle Database files such as data files, control files, log files, and so on.

Following is a high-level overview of the procedure for using Oracle Database in conjunction with Exascale. This procedure is automated for you when you use Oracle Exadata Deployment Assistant (OEDA) to create an Oracle database that uses Exascale storage.

1. Provision an Exascale user to manage the Oracle Database files.

   The same Exascale user must be used to access the database files on all nodes in the database cluster.

2. Create an Exascale vault for the Oracle Database files.

   You can dedicate a vault to store the files for one database, or you can use a vault to store the files for many databases.

3. Install the Exascale user wallet on every node in the database cluster.

4. Use the Exascale vault as the location for Oracle Database files.

   Referencing Oracle Database files on Exascale is essentially the same as using Oracle Automatic Storage Management (ASM). The fundamental difference is that you use an Exascale vault instead of an ASM disk group.

   In the following fully-qualified file name example, `@MYVAULT` specifies the vault that contains the data file.

   ```
   SQL> CREATE TABLESPACE mytbs1 DATAFILE '@MYVAULT/mydbfiles/data/
   mytbs1.dbf' SIZE 10G;
   ```

Related Topics

- Using Exascale with Other Oracle Releases

4.3 About Exascale User Credentials for Oracle Grid Infrastructure and Oracle Database

This topic describes how to identify the Exascale user credentials used in conjunction with Oracle Grid Infrastructure and Oracle Database software.

You can use Exascale to store the Oracle Grid Infrastructure shared clusterware files (Oracle Cluster Registry and voting disks) and Oracle Database files such as data files, control files, log files, and so on.

To natively use Exascale storage, Oracle Grid Infrastructure and Oracle Database must have access to an Exascale user account. The account access is provided using credentials located in a credential store file known as a wallet.

By default, Oracle Exadata Deployment Assistant (OEDA) automatically creates the required Exascale user accounts and wallets when you use it to deploy databases and VM clusters that use Exascale storage. See Exascale User Accounts and Wallets.

Otherwise, to create and administer an Exascale user, you can use the procedures described in Administer Exascale Users and Administer Exascale User Credentials.

To access Exascale storage, Oracle Grid Infrastructure and Oracle Database use the first Exascale user wallet available in the following search path:

1. `$ORACLE_BASE/admin/eswallet`
2. `/etc/oracle/cell/network-config/eswallet`

If the system uses the same operating system (OS) software owner (typically `oracle`) for both Oracle Grid Infrastructure and Oracle Database, then one Exascale user is typically used throughout the Oracle Grid Infrastructure cluster.

A role-separated software installation contains a separate Oracle Grid Infrastructure OS owner (typically `grid`) and Oracle Database OS owner (typically `oracle`). In this case, separate Exascale users are typically associated with the Oracle Database OS owner and the Oracle Grid Infrastructure OS owner.

4.4 Using Oracle Managed Files with Exascale

To store Oracle Database files on Exascale, Oracle recommends using the Oracle Managed Files (OMF) feature of Oracle Database.

Using OMF, you can simply define Oracle Database instance parameters (init.ora parameters) that specify the Exascale vault as the target destination. For example:

```
db_create_file_dest=@MYVAULT
db_create_online_log_dest_1=@MYVAULT
db_recovery_file_dest=@MYVAULT
```

You can also use OMF when specifying an Oracle Database file name in a SQL command. For example:

```
SQL> CREATE TABLESPACE mytbs DATAFILE '@MYVAULT' SIZE 10G;
```

If desired, you can specify OMF file locations that contain a more specific path inside an Exascale vault. For example:

```
db_create_file_dest=@MYVAULT/mylocation/
```

> **Note:**
>
> If you specify an OMF file location with a path inside an Exascale vault, you must include a forward slash character (/) at the end of the value.
>
> This requirement also applies when specifying a path inside an Exascale vault in a SQL command or to an Oracle Database client (such as RMAN). For example:
>
> ```
> RMAN> set archivelog destination to '@MYVAULT/mylogs/';
> ```

Using OMF, Oracle Database files are created and managed automatically using the following file naming convention:

OMF_dest/[*cluster_details*/]*db_unique_name*/[*pdb_guid_name*/]*file_type*/[*date_stamp*/]*name*.OMF.*random*

In the file naming convention:

- *OMF_dest*: Depending on the context, *OMF_dest* specifies the OMF target value specified directly in the SQL command or the value of the relevant OMF instance parameter. For example, if no value is explicitly specified in the SQL command, the value of `db_create_file_dest` is used for a data file, while the value of `db_recovery_file_dest` is used for an archived redo log file.
- *vault*: Specifies the Exascale vault name.
- *cluster_details*: Identifies the Oracle Grid Infrastructure (GI) cluster that contains the Oracle database. The value uses the format: *clustername-clusterGUID*.

  To find the *clusterGUID* for a GI cluster, you can use the following command:

  ```
  # crsctl get css clusterguid
  ```

  If the Oracle database is not associated with a GI cluster, then this element is not included in the file path.
- *db_unique_name*: Identifies the Oracle database by using the *db_unique_name* database parameter value.
- *pdb_guid_name*: Identifies the pluggable database (PDB). If the Oracle database is not a multitenant container database (CDB), then this element is not included in the file path.
- *file_type*: Identifies the Oracle Database file type; for example, `DATAFILE`, `ONLINELOG`, `CONTROLFILE`, and so on.
- *date_stamp*: Specifies the file creation date. This element is only included for specific Oracle Database file types where files are automatically generated over time, such as archive log files and automatically generated backup files.
- *name*: Specifies the OMF file name. The specific value depends on the Oracle Database file type. For example, a data file name contains the relevant tablespace name, while an archive log file name identifies the log thread and sequence number.
- *random*: Specifies a random alpha-numeric string to guarantee uniqueness.

4.5 Customizing the File Storage Attributes for Oracle Database Files

Every Exascale file is associated with file storage attributes that govern how the file is stored and managed.

By default, when a file is created, Exascale automatically applies the attributes from the template that matches the file type. For example, when Oracle Database creates a data file, the `DATAFILE` template is automatically used by default. You cannot change the file storage attributes after the file is created.

However, you can override the default template by specifying the template name in an Oracle Managed Files (OMF) specification or fully-qualified file name. In the following examples, *mytemplate* is used instead of the default template:

- OMF parameter example:

  ```
  db_create_file_dest=@MYVAULT(mytemplate)
  ```

- OMF data file example:

  ```
  SQL> CREATE TABLESPACE mytbs2 DATAFILE '@MYVAULT(mytemplate)' SIZE 10G;
  ```

- Fully-qualified data file name example:

  ```
  SQL> CREATE TABLESPACE mytbs3 DATAFILE '@MYVAULT(mytemplate)/mydbfiles/
  data/mytbs3.dbf' SIZE 10G;
  ```

By specifying a non-default template, you can customize the Exascale file storage attributes for specific databases or files within a database. For example, if the vault contains High Capacity (HC) and Extreme Flash (EF) storage media, you can choose EF storage instead of the default (HC) by using a template with `mediaType=EF`.

Related Topics

- File Storage Attributes
- Templates

4.6 Using Exascale Resource Profiles with Oracle Database

This topic describes how to manage Exascale resources by associating a database with a resource profile.

By default, an Oracle database has unlimited access to Exascale vault I/O resources and multiple databases share vault I/O resources equally when the system is under load.

To enable more granular I/O resource management, you can associate an Oracle database with an Exascale resource profile, which defines a collection of resource limits and settings. If you associate multiple databases with an Exascale resource profile, then the databases equally share the resources specified in the resource profile.

To associate an Oracle database with an Exascale resource profile, you must specify the resource profile name as the value for the `db_performance_profile` Oracle Database instance parameter (init.ora parameter). For example:

```
SQL> ALTER SYSTEM SET db_performance_profile=myresourceprofile SCOPE=spfile;
```

Related Topics

- Intra-Vault Resource Management Using Exascale Resource Profiles
- Administer Resource Profiles

4.7 Using Oracle Database with Exascale Snapshots and Clones

Exascale introduces advanced space-efficient snapshot and cloning capabilities that are tightly integrated with Oracle Database and eliminate the requirement for a test master database to support snapshots and clones on Exadata. The following topics describe typical use cases for Oracle Database in conjunction with Exascale snapshots and clones.

- Thin Cloning a Pluggable Database
- Thin Cloning a Pluggable Database in a Different Container Database
- Using a Carousel of Thinly Provisioned Pluggable Database Snapshots
- Cloning a Container Database

4.7.1 Thin Cloning a Pluggable Database

Within Oracle Database, you can use the `CREATE PLUGGABLE DATABASE` SQL command to create a writable snapshot copy (clone) of an existing pluggable database (PDB). For example, the following command creates a clone of *pdb1* named *pdb1c*.

```
SQL> CREATE PLUGGABLE DATABASE pdb1c
     FROM pdb1 SNAPSHOT COPY;
```

When Oracle Database uses Exascale storage, the PDB snapshot copy operation automatically uses native Exascale cloning functionality to thinly provision the underlying Oracle Database files.

The thin cloned PDB data files reside in the same Exascale vault as the underlying source PDB data files, and Exascale uses redirect-on-write techniques to create and maintain the cloned data files. As a result, the cloning operation is instantaneous, regardless of the size of the underlying data files. Also, the clone is extremely space-efficient. A cloned data file initially consumes no additional space, and physical storage space is only allocated when new data is written.

Within the Oracle Database, the cloned files function like regular data files and appear in the standard Oracle Database dictionary views, such as `DBA_DATA_FILES`. Additionally, you can examine various Exascale-specific file details in the V$EXA_FILE Oracle Database dictionary view.

Within Exascale, you can use the ESCLI lssnapshots command to view the association between the cloned data files and their underlying source files.

> **Note:**
>
> If you create a thinly provisioned PDB clone on an Oracle Data Guard primary database, the corresponding standby PDB data files contain a complete (not thinly provisioned) copy of the data, even if the standby database uses Exascale storage. To avoid this, you can prevent the PDB clone from propagating to the standby database by including `STANDBYS=NONE` in the `CREATE PLUGGABLE DATABASE` SQL command.

Related Topics

- Cloning a Local PDB

4.7.2 Thin Cloning a Pluggable Database in a Different Container Database

In conjunction with a database link, you can use the `CREATE PLUGGABLE DATABASE` SQL command to create a writable snapshot copy (clone) of an existing pluggable database (PDB) that resides in a different container database (CDB). For example, the following command creates a clone of *pdb1* named *pdb1c*.

```
SQL> CREATE PLUGGABLE DATABASE pdb1c
     FROM pdb1@CDB1-link SNAPSHOT COPY;
```

In this example, the cloned PDB (*pdb1c*) resides in the CDB processing the SQL command. The original (source) PDB (*pdb1*) resides in the CDB referenced in the database link (*CDB1-link*).

When both CDBs use the same Exascale vault, the PDB snapshot copy operation automatically uses native Exascale cloning functionality to thinly provision the underlying Oracle Database files.

The thin cloned PDB data files reside in the same Exascale vault as the underlying source PDB data files, and Exascale uses redirect-on-write techniques to create and maintain the cloned data files. As a result, the cloning operation is instantaneous, regardless of the size of the underlying data files. Also, the clone is extremely space-efficient. A cloned data file initially consumes no additional space, and physical storage space is only allocated when new data is written.

Within the Oracle Database, the cloned files function like regular data files and appear in the standard Oracle Database dictionary views, such as `DBA_DATA_FILES`. Additionally, you can examine various Exascale-specific file details in the V$EXA_FILE Oracle Database dictionary view.

Within Exascale, you can use the ESCLI lssnapshots command to view the association between the cloned data files and their underlying source files.

You can also use the technique introduced here to clone a PDB residing on an Oracle Data Guard standby database. This approach enables space-efficient test and development databases separate from the primary database.

To clone a PDB from a standby database, you must briefly stop the redo apply process on the standby database while creating the clone. For example

```
DGMGRL> edit database STANDBY set state='apply-off';


SQL> CREATE PLUGGABLE DATABASE pdb1test
     FROM pdb1@STANDBY-link SNAPSHOT COPY;


DGMGRL> edit database STANDBY set state='apply-on';
```

In this example, the cloned PDB (*pdb1test*) resides in the CDB processing the SQL command. The original (source) PDB (*pdb1*) resides in the standby database referenced in the database link (*STANDBY-link*). As in the earlier case, to use Exascale thin clones, the CDB processing the SQL command must use the same Exascale vault that houses the underlying source PDB data files.

> **Note:**
>
> Regardless of whether the source PDB is an Oracle Data Guard primary or standby database, if you create a thinly provisioned PDB clone on an primary database, the corresponding standby PDB data files contain a complete (not thinly provisioned) copy of the data, even if the standby database uses Exascale storage. To avoid this, you can prevent the PDB clone from propagating to the standby database by including `STANDBYS=NONE` in the `CREATE PLUGGABLE DATABASE` SQL command.

Related Topics

- Cloning a Remote PDB

4.7.3 Using a Carousel of Thinly Provisioned Pluggable Database Snapshots

Within Oracle Database, you can create a library of pluggable database (PDB) snapshots, known as a PDB snapshot carousel. A PDB snapshot is a point-in-time copy of a PDB. The source PDB can be open read-only or read/write while the snapshot is created. A PDB snapshot carousel is useful for maintaining a library of recent PDB copies for point-in-time recovery and cloning.

For example, the following command creates a snapshot carousel based on *pdb1* where a new snapshot is taken every 2 hours.

```
SQL> ALTER PLUGGABLE DATABASE pdb1
     SNAPSHOT MODE EVERY 2 hours;
```

When Oracle Database uses Exascale storage, the PDB snapshot operation automatically uses native Exascale functionality to snapshot the underlying Oracle Database files. A PDB snapshot carousel based on Exascale storage can contain up to 4096 PDB snapshots. Once the carousel reaches its limit, each new snapshot automatically replaces the oldest snapshot.

The snapshots reside in the same Exascale vault as the underlying source data files, and Exascale uses redirect-on-write techniques to create and maintain the thinly provisioned snapshot data files. As a result, the snapshot operation is instantaneous, regardless of the size of the underlying data files. Also, each snapshot consumes no additional space. Space is only consumed when new data is written to the source PDB or to a writable clone based on a snapshot.

Within the Oracle Database, the snapshot data files appear in the standard Oracle Database dictionary views, such as `DBA_PDB_SNAPSHOTFILE`. Additionally, you can examine various Exascale-specific file details in the V$EXA_FILE Oracle Database dictionary view.

Within Exascale, you can use the ESCLI lssnapshots command to view the association between the snapshots and their underlying source files.

> **Note:**
>
> If you create a thinly provisioned PDB snapshot on an Oracle Data Guard primary database, the snapshot metadata propagates to the standby. However, the snapshot is not accessible at the standby site, even after an Oracle Data Guard role transition (switchover or failover).

To utilize a snapshot from the PDB snapshot carousel as a writable PDB, you must create a clone (snapshot copy) based on the snapshot. For example:

```
SQL> CREATE PLUGGABLE DATABASE pdbCloneFromSnap FROM pdb1
     USING SNAPSHOT pdb1SnapshotN SNAPSHOT COPY;
```

In the example, if *pdb1SnapshotN* identifies a thinly provisioned PDB snapshot on Exascale storage, the new PDB (*pdbCloneFromSnap*) is also thinly provisioned.

Note that creating a clone (snapshot copy) based on the PDB snapshot requires access to archived redo log files from the time of the snapshot to ensure consistency across the PDB data files.

Related Topics

- Thin Cloning a Pluggable Database
- Administering a PDB Snapshot Carousel

4.7.4 Cloning a Container Database

Oracle Database contains integrated snapshot and cloning functionality for Oracle multitenant pluggable databases (PDBs), which is primarily available through the `CREATE PLUGGABLE DATABASE` and `ALTER PLUGGABLE DATABASE` SQL commands. When Oracle Database uses Exascale storage, the PDB snapshot and snapshot copy (cloning) functions automatically use native Exascale snapshots and clones to create space-efficient copies of the required files.

However, cloning an entire container database (CDB) requires more than just cloning the database files. The clone requires a unique database name and separate configuration settings for the database instances, requiring coordination of various operations apart from the original database.

To facilitate efficient CDB cloning, Oracle provides a generic database cloning utility called `gDBClone`. Historically, `gDBClone` provides support for various methods of CDB duplication. With

the introduction of Exascale, the `gDBClone` utility is enhanced to facilitate efficient CDB cloning in conjunction with Exascale storage.

Creating a CDB clone using the `gDBClone` utility and Exascale storage is:

- Easy. The utility requires only a few inputs.
- Reliable. The entire procedure is automated, reducing the risk of errors.
- Flexible. Using `gDBClone`, you can clone:
  - Single-instance and Oracle RAC databases.
  - Unencrypted databases and databases using Oracle Transparent Data Encryption (TDE).
  - Stand-alone databases, Oracle Data Guard primary databases, and Oracle Data Guard standby databases.

To use the `gDBClone` utility, you must first ensure that the source CDB is running in `ARCHIVELOG` mode.

Then, to create a CDB clone on Exascale storage, use the `gDBClone` utility with the following syntax and options:

```
# /opt/gDBClone/gDBClone.bin { clone | snap } -sdbname source_db_name -
tdbname target_db_name
     [ -tdbhome target_db_home ]
     [[[ -sga_max_size size_mb ] [ -sga_target size_mb ]] | -pfile file-name ]
     [ -sdbport source_listener_port ]
     [ -racmod { 0 | 1 | 2 }]
     [ -syspwf sys_password_file ]
     [ -walletpwf wallet_password_file ]
```

In the command:

- `clone`: Instructs `gDBClone` to create a CDB clone using a complete (not thinly provisioned) database copy. This option is recommended for copying a database into Exascale storage from elsewhere.
- `snap`: After the source database resides in an Exascale vault, the `snap` option instructs `gDBClone` to create a thinly provisioned space-efficient CDB clone in the same Exascale vault.

  Using this option, Exascale uses redirect-on-write techniques to efficiently create and maintain data file clones using minimal storage space. As a result, you can clone an entire CDB in minutes, with the underlying Exascale file cloning operation occurring instantaneously, regardless of the source database size.
- `-sdbname` *`source_db_name`*: Specifies the `db_unique_name` of the source CDB being cloned.
- `-tdbname` *`target_db_name`*: Specifies the `db_unique_name` to use for the new thinly provisioned target CDB.
- `-tdbhome` *`target_db_home`*: Optionally specifies the `ORACLE_HOME` location for the target CDB. If not specified, the target CDB automatically uses the same Oracle Database binaries as the source CDB.
- `-sga_max_size` *`size_mb`*: Optionally specifies the maximum size (in megabytes) of the target database system global area (SGA). If not specified, the target CDB automatically uses the same size as the source database.

- `-sga_target size_mb`: Optionally specifies the nominal total size (in megabytes) of the target database system global area (SGA). If not specified, the target CDB automatically uses the same size as the source database.
- `-pfile file-name`: Optionally specifies a file that contains one or more supported parameter settings for use by the target database. See the `gDBClone` reference (Oracle Support Document 2099214.1) for the list of supported parameters. You can use this option as an alternative to specifying `-sga_max_size` and `-sga_target` on the command line.
- `-sdbport source_listener_port`: Optionally specifies the Oracle Net listener port number for the source CDB. If not specified, the default value is 1521.
- `-racmod { 0 | 1 | 2 }`: Optionally specifies the target CDB configuration using one of the following values:
  - `0` - Single instance. This is the `gDBClone` default value.
  - `1` - Oracle RAC One Node.
  - `2` - Oracle RAC. This is the typical setting for databases on Oracle Exadata Exascale.
- `-syspwf sys_password_file`: Optionally specifies a file containing an encrypted version of the source database SYS user password, previously created using `gDBClone` with the `syspwf` option.

  If this option is not specified, `gDBClone` checks the default location (`/opt/gDBClone/SYSpasswd_file`) for a usable password file, and if that is not successful, `gDBClone` prompts to supply the password during command execution.
- `-walletpwf wallet_password_file`: Optionally specifies a file containing an encrypted version of the source database keystore (wallet) password. This option must be specified if the source database uses Oracle Transparent Data Encryption (TDE).

For example, the following command shows the simple case of thin cloning the `SOURCE` CDB to create the `THIN` CDB.

```
# /opt/gDBClone/gDBClone.bin snap -sdbname SOURCE -tdbname THIN -racmod 2
```

In the previous example:

- The `SOURCE` CDB was previously copied into an Exascale vault, and the `THIN` CDB resides in the same vault.
- The `SOURCE` CDB is an Oracle RAC database and the `THIN` CDB is also configured as an Oracle RAC database.
- Both CDBs share the same Oracle binaries and the `THIN` CDB inherits Oracle Database parameter settings from the `SOURCE` CDB.
- The `gDBClone` utility looks for an encrypted file containing the `SOURCE` CDB SYS user password at `/opt/gDBClone/SYSpasswd_file`. If the file doesn't exist or doesn't contain the required password, `gDBClone` prompts for the password at runtime.
- The `SOURCE` CDB is not encrypted using Oracle Transparent Data Encryption (TDE). If the source database contains TDE-encrypted tablespaces, then additional steps are required to configure the TDE keystore (wallet) before creation of the CDB thin clone.

The `gDBClone` utility provides many additional functions. For example, you can view the databases created with `gDBClone` and display the relationships between them by running:

```
# /opt/gDBClone/gDBClone.bin listdbs -tree
```

You can also remove a database quickly and easily by using the `deldb` option. For example:

```
# /opt/gDBClone/gDBClone.bin deldb -tdbname THIN
```

For a complete list of functions and further details, see the `gDBClone` reference at Oracle Support Document 2099214.1.

4.8 Using Exascale with Other Oracle Releases

This topic outlines options for using Exascale storage to support software releases before Oracle Database 23ai.

Starting with Oracle Database 23ai release 23.5.0, you can use Exascale to natively store and manage the Oracle Grid Infrastructure shared files (OCR and voting files), and Oracle Database files such as data files, control files, log files, and so on. Using Exascale in conjunction with natively supported Oracle software releases enables full functionality for optimal system performance.

However, you can use Exascale storage to support other Oracle Database software releases back to Oracle Database 19c. The following outlines the available options:

- **Option 1: Use Exascale Direct Volume (EDV).**

  This option uses Oracle Grid Infrastructure release 23.5.0 (or later) and native Exascale vault storage for the Oracle Grid Infrastructure shared files.

  You can use EDV to provide storage for databases that use Oracle Database software before release 23.5.0. You can use EDV in either of two ways:

  – You can use EDV attachments as ASM grid disks to define an ASM disk group based on EDV storage. The resulting disk group can store your Oracle Database files the same way as any other ASM disk group.

  – You can use Oracle Advanced Cluster File System (ACFS) to create a cluster-aware file system based on EDV storage. The resulting file system can store your Oracle Database files and other files, such as Oracle Database binaries, log files, and so on.

  > **Note:**
  >
  > By using either of the above approaches, the associated Oracle databases cannot use Exadata Storage Server offloading capabilities, such as Exadata Smart Scan, to offload database processing to the storage servers.

- **Option 2: Use Exascale Block Volumes and iSCSI.**

  This option uses standard iSCSI protocols as the means to expose Exascale block volumes to database servers or virtual machines (VMs).

  For example, you can use iSCSI attachments as ASM grid disks to define an ASM disk group based on Exascale block storage. The resulting disk group can store your Oracle Database files the same way as any other ASM disk group.

> **Note:**
>
> By using this option, the associated Oracle databases cannot use Exadata Storage Server offloading capabilities, such as Exadata Smart Scan, to offload database processing to the storage servers.

- **Option 3: Use a Hybrid Installation with Exascale and Oracle ASM.**

  In a hybrid installation, part of the available Exadata storage is allocated to Exascale, while the other part is allocated to Oracle ASM. Under this arrangement, you can use Exascale natively for supported Oracle Grid Infrastructure and Oracle Database releases. And, you can use Oracle ASM natively for older Oracle Grid Infrastructure and Oracle Database releases.

  By using this option, all of your Oracle databases can take full advantage of Exadata Smart Storage features, such as Exadata Smart Scan.

Related Topics

- Using Exascale Block Storage with EDV
- Using Exascale Block Storage with iSCSI

4.9 Copying a TDE Keystore Between a File System and Exascale Storage

You cannot use the XSH `cp` command or the ESCLI `putfile` or `getfile` commands to successfully copy an Oracle Transparent Database Encryption (TDE) keystore (wallet) file in either direction between a file system and Exascale storage. If you copy a TDE keystore between a file system and an Exascale vault using XSH or ESCLI, Oracle Database cannot open the resulting file.

The examples in the following procedure shows how to copy a TDE keystore from a file system to an Exascale vault. By appropriately modifying the keystore locations, you can also use the same approach to copy a TDE keystore from Exascale storage to a file system location.

1. Configure Oracle Database to use the desired target location for the TDE keystore.

   Set the `WALLET_ROOT` database parameter to the desired target keystore location. If you want to copy the TDE keystore into Exascale, specify a location in an Exascale vault.

   For example:

   ```
   SQL> ALTER SYSTEM SET WALLET_ROOT=@MYVAULT/mydb/TDEWALLET SCOPE=SPFILE;
   ```

2. Restart the database to use the new keystore location.
3. Create a new (empty) keystore for the database.

   For example:

   ```
   SQL> ADMINISTER KEY MANAGEMENT CREATE KEYSTORE IDENTIFIED BY
   'newKeystorePassword';
   ```

4. Note the location of the newly created keystore.

For example:

```
SQL> SELECT wrl_parameter FROM v$encryption_wallet;

WRL_PARAMETER
--------------------------------------------------------------------------------
-----
@MYVAULT/mydb/TDEWALLET/tde/
```

5. Merge the contents of the source keystore into the newly created (empty) keystore.

   In the following example, the source keystore is on file system storage (at `/u01/app/oracle/admin/mydb/wallet/`) and the newly created keystore is on Exascale storage at the location identified in the previous step.

```
SQL> ADMINISTER KEY MANAGEMENT MERGE KEYSTORE '/u01/app/oracle/admin/mydb/
wallet/'
         IDENTIFIED BY 'originalKeystorePassword'
       INTO EXISTING KEYSTORE '@MYVAULT/mydb/TDEWALLET/tde/'
         IDENTIFIED BY 'newKeystorePassword'
       WITH BACKUP;
```

6. Open the new keystore.

   For example:

```
SQL> ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY
'newKeystorePassword';
```

5
Using Exascale Block Storage

This chapter outlines how to use Exascale block storage by using Exascale Direct Volume (EDV) or through iSCSI.

EDV is the default and recommended volume attachment type for cases where clients want to use volumes within the Exadata RDMA Network Fabric. iSCSI provides a storage protocol that supports standard TCP/IP network clients, which may reside outside the Exadata RDMA Network Fabric.

This chapter contains the following topics:

5.1 Using Exascale Block Storage with EDV

This topic outlines the procedure for using Exascale block storage with Exascale Direct Volumes (EDV).

The Exascale block store provides capabilities to create and manage arbitrary-sized raw block volumes based on Exascale storage.

EDV is recommended for use-cases where clients want to use storage volumes inside the Exadata RDMA Network Fabric. For example:

- You can use EDV to support Oracle Advanced Cluster File System (ACFS) or local Linux file systems (such as XFS, EXT4, and so on) on the Exadata compute nodes.
- You can use EDV raw devices to support earlier Oracle Database versions, not natively supported on Exascale. For example, you can create an Oracle ASM disk group based on EDV raw devices and use the disk group to contain the Oracle Database data files. Or, you can create a file system on an EDV volume and use it to contain the Oracle Database software and data files.

Before you can use Exascale block storage with EDV, note the following:

- The block store manager (BSM) and block store worker (BSW) services must be running in the Exascale cluster.

  To display information about Exascale storage services running across the Exascale cluster, use the ESCLI lsservice command. For example:

  ```
  @> lsservice --detail
  ```

- The Exascale Direct Volume (EDV) service must be running on each Exadata compute node that you want to host an EDV attachment. If you are planning to use a cluster-wide attachment, then the EDV service must be running on every node in the Oracle Grid Infrastructure (GI) cluster.

To display information about client-side Exascale services, including Exascale Node Proxy (ESNP) and Exascale Direct Volume (EDV), use the DBMCLI `LIST DBSERVER` command, as follows:

```
DBMCLI> list dbserver detail
```

The command output displays status information for all of the client-side Exascale services running on the Exadata compute node. You must run the command separately on each compute node.

- During initial system deployment with Oracle Exadata Deployment Assistant (OEDA), the Exascale Direct Volume (EDV) service is configured on each Exadata compute node (bare-metal or VM) and is related to the Exascale user that manages the Oracle Grid Infrastructure (GI) cluster. To create an EDV attachment, you must use the Exascale user linked with the EDV service.

  If the GI cluster uses a non-role-separated user configuration with one Oracle OS user account, then the associated Exascale user is related to the EDV service. If the GI cluster uses a role-separated configuration with a Grid OS user account and an Oracle OS user account, then the EDV service is linked to the Exascale user associated with the Grid OS account.

  To find the Exascale user linked with the EDV service, use the ESCLI lsinitiator command with the `--detail` option and examine the `user` attribute.

To begin, you can create an Exascale block volume and EDV attachment.

For example:

```
@>mkvault myvault
Vault @myvault created.

@>mkvolume 1g --attributes name=edv1 --vault myvault
Created volume with id 2:72ebb8a56fba4e97b4716c25981702cb

@>lsvolume 2:72ebb8a56fba4e97b4716c25981702cb --detail
id                                  2:72ebb8a56fba4e97b4716c25981702cb
name                                edv1
blockSizeIOPS                       8192
contentType                         DATA
creationTime                        2024-12-06T04:46:07+00:00
filePath                            @myvault/
vol.5ceabbae36d04e31bc79f9c34658c22e
mediaType                           HC
numAttachments                      0
iopsProvisioned                     unlimited
bandwidthProvisioned                unlimited
redundancyType                      none
size                                1G
state                               AVAILABLE
owners                              exa01
vault                               @myvault
vipId                               111:02ae5dee-55a6-4dab-beb6-69a1b5525d60

@>lsinitiator --detail
id                                  e7e0db8c-9a2a-0279-e7e0-db8c9a2a0279
edvBaseDriverVersion                25.1.0.0.0.241130
```

```
edvEffectiveDriverVersion            25.1.0.0.0.241130
giClusterId                          deadbeef-badc-0fee-dead-beefbadc0fee
giClusterName                        edvTestCluster
hostName                             exa01
lastHeartbeat                        2024-12-06T04:45:33+00:00
registerTime                         2024-12-06T04:41:18+00:00
state                                ONLINE
user                                 exa01
version                              25.1.0.0.0.241130

@>mkvolumeattachment 2:72ebb8a56fba4e97b4716c25981702cb myedv1 --attributes
giClusterId=deadbeef-badc-0fee-dead-beefbadc0fee
Created edv attachment with id 1:cc9435ab951a49119dbb7e249a0b0219

@>lsvolumeattachment 1:cc9435ab951a49119dbb7e249a0b0219 --detail
id                                   1:cc9435ab951a49119dbb7e249a0b0219
attachTime                           2024-12-06T04:47:41+00:00
kernelDeviceName                     exc-dev1
deviceName                           myedv1
devicePath                           /dev/exc/myedv1
giClusterId                          deadbeef-badc-0fee-dead-beefbadc0fee
giClusterName                        edvTestCluster
hostName
initiator
logicalSectorSize                    512
volume                               2:72ebb8a56fba4e97b4716c25981702cb
volumeSnapshot

@>
```

The EDV attachment creates an association between the volume and an EDV device file, which resides on the Exadata compute nodes hosting the attachment. A node hosting an EDV attachment is also known as an EDV initiator.

If you create a cluster-wide attachment to support a cluster file system such as ACFS, then the EDV device file is created on every node in the Oracle Grid Infrastructure (GI) cluster. If you create a node-specific attachment, then the corresponding EDV device is only created on that node.

In the example, the EDV attachment is a cluster-wide attachment, and the EDV device name is `myedv1`, so the corresponding device file is located at `/dev/exc/myedv1` on each cluster node. The volume identifier (`2:72ebb8a56fba4e97b4716c25981702cb`) was reported to the user during volume creation. Volume identifiers can also be discovered using the ESCLI lsvolume command. The GI cluster identifier (`deadbeef-badc-0fee-dead-beefbadc0fee`) was found by using the ESCLI lsinitiator command.

Alternatively, for a node-specific attachment you must specify the node-specific EDV initiator identifier instead of the GI cluster identifier. For example:

```
@>lsinitiator --detail
id                                   e7e0db8c-9a2a-0279-e7e0-db8c9a2a0279
edvBaseDriverVersion                 25.1.0.0.0.241130
edvEffectiveDriverVersion            25.1.0.0.0.241130
giClusterId                          deadbeef-badc-0fee-dead-beefbadc0fee
giClusterName                        edvTestCluster
```

```
hostName                                exa01
lastHeartbeat                           2024-12-06T04:45:33+00:00
registerTime                            2024-12-06T04:41:18+00:00
state                                   ONLINE
user                                    exa01
version                                 25.1.0.0.0.241130

@>mkvolumeattachment 2:72ebb8a56fba4e97b4716c25981702cb myedv1 --attributes
initiator=e7e0db8c-9a2a-0279-e7e0-db8c9a2a0279
Created edv attachment with id 1:50e52177583f4be4bad68ac20b65001e

@>lsvolumeattachment 1:50e52177583f4be4bad68ac20b65001e --detail
id                                      1:50e52177583f4be4bad68ac20b65001e
attachTime                              2024-12-06T04:47:41+00:00
kernelDeviceName                        exc-dev1
deviceName                              myedv1
devicePath                              /dev/exc/myedv1
giClusterId
giClusterName
hostName                                exa01
initiator                               e7e0db8c-9a2a-0279-e7e0-db8c9a2a0279
logicalSectorSize                       512
volume                                  2:72ebb8a56fba4e97b4716c25981702cb
volumeSnapshot

@>
```

After attachment, the EDV device can be used on the Exadata compute node or GI cluster hosting the attachment.

For example, you could use the following command sequence to create and mount an ACFS file system using the EDV device defined in the previous examples (`/dev/exc/myedv1`):

```
# # Confirmation of the EDV device at /dev/exc/myedv1.
# lsblk /dev/exc/myedv1
NAME       MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
exc-dev1   251:1    0   1G  0 disk
# # Note the default ownership and permissions for the EDV device.
# ls -l /dev/exc/myedv1
brw-rw---- 1 root disk 251, 1 Oct  6 17:41 /dev/exc/myedv1
# # Now use the device to support ACFS.
# mkfs -t acfs /dev/exc/myedv1
mkfs.acfs: version                   = 23.0.0.0.0
mkfs.acfs: ACFS compatibility        = 23.0.0.0.0
mkfs.acfs: on-disk version           = 53.0
mkfs.acfs: volume                    = /dev/exc/myedv1
mkfs.acfs: volume size               = 1073741824  (   1.00 GB )
mkfs.acfs: file system size          = 1073741824  (   1.00 GB )
mkfs.acfs: Format complete.
# mkdir /mnt/myedv1
# mount /dev/exc/myedv1 /mnt/myedv1
# df /mnt/myedv1
Filesystem      1K-blocks   Used Available Use% Mounted on
/dev/exc/myedv1   1048576 318436    730140  31% /mnt/myedv1
```

```
#
# # Mount the file system on other cluster nodes as required.
```

Note:

- Each EDV attachment also has a kernel device file at `/dev/exc-devN`, where *N* is the minor number of the device. The kernel device name is contained as an attribute of the EDV attachment and is visible using the ESCLI lsvolumeattachment command. The relationship between the kernel device file and the user-named device file (under `/dev/exc/`) is also recorded in the udev database and is visible using the following Linux command:

  ```
  # udevadm info device-file
  ```

  For the *device-file* value, you can specify either the kernel device file (`/dev/exc-devN`) or the user-named device file (under `/dev/exc/`).

- By default, read and write access to EDV device files is only available to the `root` operating system user and members of the `disk` group. Depending on your use case, you may need to modify the permissions on the EDV device files before using them.

  For example, to make the EDV device file at `/dev/exc/myvol` readable and writable by the `oracle` user and `dba` group, you could configure it using a udev rule similar to the following:

  ```
  # cat /etc/udev/rules.d/57-edv-user.rules
  KERNEL=="exc-*", ENV{EXC_ALIAS}=="myvol", OWNER="oracle",
  GROUP="dba", MODE="0660"
  ```

- To facilitate the management of udev rules related to EDV devices, each EDV client node is configured with a template udev rules file at `/etc/udev/rules.d/57-edv-user.rules`, which you can modify to fulfill your requirements. To maintain existing udev rules, `/etc/udev/rules.d/57-edv-user.rules` is preserved whenever the EDV client software is updated.
- Each EDV client node can support a maximum of 1024 attachments at the same time. This limit includes the total of all cluster attachments involving the server, as well as local attachments specific to the server.

An ACFS file system on EDV can also be exported to clients outside the Exadata RDMA Network Fabric by using ACFS HANFS. For more information about ACFS, see Oracle Advanced Cluster File System Administrator's Guide.

EDV devices can also be used as raw devices or in conjunction with other Linux file systems, such as XFS, EXT4, and so on.

5.2 Using Exascale Block Storage with iSCSI

This topic outlines the procedure for using Exascale block storage in conjunction with iSCSI.

The Exascale block store provides capabilities to create and manage arbitrary-sized raw block volumes based on Exascale storage.

You can use standard iSCSI protocols and tools to utilize an Exascale volume as an iSCSI target. The iSCSI target is implemented on an Exadata compute node and supports iSCSI initiators that may be external to Exadata.

iSCSI is recommended for use-cases where a client wants to use the volume outside the Exadata RDMA Network Fabric or requires a bootable volume.

Before you can use Exascale block storage with iSCSI, note the following:

- The block store manager (BSM) service must be running in the Exascale cluster.
- The block store worker (BSW) service must be running on each Exadata compute node that you want act as an iSCSI initiator.

  To display information about Exascale storage services running across the Exascale cluster, use the ESCLI lsservice command. For example:

  ```
  @> lsservice --detail
  ```

- The block store virtual IP addresses (VIPs) must be configured by the Exascale administrator.

To begin, you can create an Exascale block volume and iSCSI volume attachment.

For example:

```
@>mkvault myvault
Vault @myvault created.

@>mkvolume 2g --attributes name=ivol1 --vault myvault
Created volume with id 2:390535212ea7448299bcc7bc7f7653b2

@>lsvolume 2:390535212ea7448299bcc7bc7f7653b2 --detail
id                                  2:390535212ea7448299bcc7bc7f7653b2
name                                ivol1
blockSizeIOPS                       8192
burstDurationSec                    120
burstIOPS                           2000
burstIOPSCreditPerSec               0
burstInitialDurationSec             60
burstInitialMBPS                    0
burstMBPS                           16
compartmentId
contentType                         DATA
creationTime                        2021-08-29 20:53:57 PDT
filePath                            @myvault/
vol.2ccedfbe32404fcca1859b4421aeb58a
mediaType                           HC
numAttachments                      0
iopsProvisioned                     10
bandwidthProvisioned                1
redundancyType                      normal
size                                2G
state                               AVAILABLE
owners                              exa01
vault                               @myvault
```

```
vipId                                   897
burstInitialIOPS                        2000


@>mkvolumeattachment 2:390535212ea7448299bcc7bc7f7653b2
iqn.1988-12.com.example.exa01 --protocol iscsi --attributes
chapUserId=X1234567-user-1,chapPassword=X1234567-
pass-1,chapMutualUserId=X1234567-user-2,chapMutualPassword=X1234567-pass-2
Created attachment with id 2-2:9a7bced09e514bd9bb701d0556020c1c

@>lsvolumeattachment --protocol iscsi 2-2:9a7bced09e514bd9bb701d0556020c1c --
detail
id                                      2-2:9a7bced09e514bd9bb701d0556020c1c
attachTime                              2021-08-29 21:00:13 PDT
initiatorIQN                            iqn.1988-12.com.example.exa01
state                                   AVAILABLE
targetIQN
iqn.2017-01.com.example.exa01.x8664:sn.dae3f3e1-ef8d-47f9-ae5b-3707b15b2072
targetPortal                            192.168.184.190:3260
volume                                  2:390535212ea7448299bcc7bc7f7653b2
user                                    exa01

@>
```

An iSCSI attachment creates a association between the Exascale block volume (iSCSI target) and an iSCSI initiator.

In the example, the volume ID is `2:390535212ea7448299bcc7bc7f7653b2`, which corresponds to the Exascale block volume that was created at the beginning of the example. The iSCSI initiator ID is `iqn.1988-12.com.example.exa01`. This value is an attribute of the iSCSI initiator.

After attachment, the iSCSI initiator can use standard iSCSI protocols and commands to interact with the target.

The following example shows the use of iSCSI client tools on Linux to access the attachment from the previous example. In the example, note the use of the attribute values that are derived directly from the volume attachment details in the `lsvolumeattachment` output.

```
# # Set variables for required attributes. Values come from Exascale volume
attachment attributes.
# export ip=192.168.184.190
# export port=3260
# export target=iqn.2017-01.com.example.exa01.x8664:sn.dae3f3e1-ef8d-47f9-
ae5b-3707b15b2072
# export username=X1234567-user-1
# export password=X1234567-pass-1
# export mutual_username=X1234567-user-2
# export mutual_password=X1234567-pass-2
# # Discover the iSCSI target.
# iscsiadm --mode discoverydb --type sendtargets --portal $ip:$port --discover
192.0.2.190:3260,1 iqn.2017-01.com.example.exa01.x8664:sn.dae3f3e1-ef8d-47f9-
ae5b-3707b15b2072
# # Configure CHAP.
# iscsiadm --mode node --portal $ip:$port --op=update --name
node.session.auth.authmethod --value=CHAP
# iscsiadm --mode node --portal $ip:$port --op=update --name
```

```
node.session.auth.username --value=$username
# iscsiadm --mode node --portal $ip:$port --op=update --name
node.session.auth.password --value=$password
# iscsiadm --mode node --portal $ip:$port --op=update --name
node.session.auth.username_in --value=$mutual_username
# iscsiadm --mode node --portal $ip:$port --op=update --name
node.session.auth.password_in --value=$mutual_password
# # Connect to the iSCSI target.
# iscsiadm --mode node -T $target --portal $ip:$port --login
Logging in to [iface: default, target:
iqn.2017-01.com.example.exa01.x8664:sn.dae3f3e1-ef8d-47f9-ae5b-3707b15b2072,
portal: 192.168.184.190,3260] (multiple)
Login to [iface: default, target:
iqn.2017-01.com.example.exa01.x8664:sn.dae3f3e1-ef8d-47f9-ae5b-3707b15b2072,
portal: 192.168.184.190,3260] successful.
# # The Exascale volume is now attached using iSCSI as /dev/sda.
# lsscsi
[2:0:0:0]    disk    ORACLE   ExascaleVolume   1.0   /dev/sda
# # Confirmation that the unit serial number for /dev/sda maps back to
Exascale volume ID 2:390535212ea7448299bcc7bc7f7653b2.
# sg_inq -p 0x80 /dev/sda
VPD INQUIRY: Unit serial number page
  Unit serial number: 2:390535212ea7448299bcc7bc7f7653b2
# # Now use the device to create and mount a file system.
# mkfs -t ext4 /dev/sda
mke2fs 1.42.9 (28-Dec-2013)
/dev/sda is entire device, not just one partition!
Proceed anyway? (y,n) y
Filesystem label=
OS type: Linux
Block size=4096 (log=2)
Fragment size=4096 (log=2)
Stride=16 blocks, Stripe width=16 blocks
131072 inodes, 524288 blocks
26214 blocks (5.00%) reserved for the super user
First data block=0
Maximum filesystem blocks=536870912
16 block groups
32768 blocks per group, 32768 fragments per group
8192 inodes per group
Superblock backups stored on blocks:
        32768, 98304, 163840, 229376, 294912

Allocating group tables: done
Writing inode tables: done
Creating journal (16384 blocks): done
Writing superblocks and filesystem accounting information: done

# mkdir /mnt/ivol1
# mount /dev/sda /mnt/ivol1
# df /mnt/ivol1
Filesystem     1K-blocks  Used Available Use% Mounted on
/dev/sda         1998672  6144   1871288   1% /mnt/ivol1
#
```

Note:

Depending on your use case, you may need to modify the permissions on the iSCSI device files before using them. For example, to use the iSCSI device files as Oracle ASM grid disks, you may need to configure them using udev rules similar to the following:

```
# cat /etc/udev/rules.d/99-oracle-asmdevices.rules
KERNEL=="sda", OWNER="oracle", GROUP="asmdba", MODE="0660"
```

6
Using ESCLI

This chapter describes how to use the Exascale command line interface (ESCLI):

- Start and Use ESCLI
- ESCLI Command Reference

6.1 Start and Use ESCLI

This topic describes how to start and use the Exascale command line interface (ESCLI).

ESCLI is a command-line administration tool that is located on Exadata compute nodes and storage servers. You can use ESCLI to perform Exascale monitoring and management functions. ESCLI works in conjunction with the Exadata cell command-line interface (CellCLI) and does not replace it.

To use ESCLI you must have access to an Exascale user account and the digital key store (wallet) for the Exascale user account.

To start ESCLI, use the following command line syntax:

```
$ escli [ --wallet wallet-location ] [ --ctrl server[:port] ] [ ESCLI-
command ]
```

In the command line:

- `--wallet` *`wallet-location`*: Identifies the wallet that you want to use for authentication to Exascale. If you do not specify the wallet location, then ESCLI checks the local server and uses the first Exascale user wallet available in the following search path:
  1. `$OSSCONF/eswallet`
  2. `$ORACLE_BASE/admin/eswallet`
  3. `/etc/oracle/cell/network-config/eswallet`
- `--ctrl` *`server`*`[:`*`port`*`]`: Specifies the endpoint address for connection to Exascale control services, also known as Exascale RESTful Services (ERS). If the ERS address is not specified on the command line, then ESCLI connects to the ERS address stored in the user's wallet.
- *ESCLI-command*: Specifies an ESCLI command to run immediately. For example:

```
$ escli --wallet /home/user/user.wallet ls -l @MYDATA
Total 5
  10.0M   05 Jan 12:59 x
  19.5k   05 Jan 13:08 y
   5.0k   05 Jan 13:09 z1
  10.0M   05 Jan 13:23 z2
  20.0G   05 Jan 13:14 z3
$
```

If an ESCLI command is not specified on the `escli` command line, ESCLI starts an interactive session and presents a command prompt. During an interactive session, you can run a series of ESCLI commands. The interactive session ends with the `exit` command. For example:

```
$ escli --wallet /home/user/user.wallet
@> ls
MYDATA
VAULT2
@> cd @MYDATA
@MYDATA/> ls -l
Total 5
  10.0M   05 Jan 12:59 x
  19.5k   05 Jan 13:08 y
   5.0k   05 Jan 13:09 z1
  10.0M   05 Jan 13:23 z2
  20.0G   05 Jan 13:14 z3
@MYDATA/> exit
$
```

During an interactive session, ESCLI supports command auto-completion by using the `<tab>` key. When you press the `<tab>` key, ESCLI attempts to complete the current command token. If multiple possibilities exist, ESCLI provides a list of alternatives. For example:

```
@> lsc<tab>
lscell                    (List all cells in the cluster)
lscelldisk             (Lists cell disks in this cluster)
lscluster                      (List cluster information)
lscomputeserver (List all compute servers in the cluster)
```

Command auto-completion also works within a command to suggest or complete command options. For example:

```
@> lscluster --<tab>
--attributes                        (Lists the specified attributes)
--backup                               (List only backup attributes)
--count                          (Maximum number of results to report)
--detail                    (Lists all attributes in a detailed form)
--filter (Used to specify conditions for filtering the list output)
--sort     (Used to sort the output using the specified attributes)
--volume                               (List only volume attributes)
```

6.2 ESCLI Command Reference

This section contains references for the Exascale command line interface (ESCLI) commands. Apart from command help and object description, which are universal, the commands are grouped by functional area:

- ESCLI Command Help
- Describing Resources and Attributes
- Service and Cluster Management
- Security and User Management

- Storage Pool and Pool Disk Management
- Vault Management
- Dataset Management
- File Management
- Template Management
- Resource Profile Management
- Extended Attribute Management
- Block Store Management

6.2.1 ESCLI Command Help

Purpose

The ESCLI `help` command displays help information for Exascale command line interface (ESCLI) commands.

Syntax

```
help ( command | list | all )
```

Command Options

The options for the `help` command are:

- *command*: Specifies the ESCLI command for which you want to print help information.
- `list`: Lists all of the available ESCLI commands.
- `all`: Prints help information for all of the available ESCLI commands.

Examples

Example 6-1 Print Help Information for a Specific Command

The following example prints help information for the `ls` command.

```
@> help ls
```

6.2.2 Describing Resources and Attributes

Purpose

The `describe` command displays information about Exascale resources and their attributes.

Syntax

```
describe { resource-spec | --all } [ --detail ]
```

Command Options

The options for the `describe` command are:

- *resource-spec*: Directs the command to display information about specific Exascale resources and their attributes as follows:
  - The *resource-spec* value is not case-sensitive.

    Consequently, the following commands are functionally equivalent:

    ```
    @> describe vaults
    ```

    ```
    @> describe VAULTS
    ```

    ```
    @> describe vAUltS
    ```

  - You can specify resource names in singular or plural form.

    Consequently, the following commands are functionally equivalent:

    ```
    @> describe vault
    ```

    ```
    @> describe vaults
    ```

  - If you specify a complete resource name (in singular or plural form), the output contains all attributes of the specified resource.
  - If you specify a resource name with a wildcard (`*`), the output contains all resources that match the pattern.

    For example, the following command describes all resources starting with the letter '`v`' (vaults, volumes, and so on):

    ```
    @> describe v*
    ```

  - If you specify an attribute name (using the notation: `<resource>.<attribute>`), the output describes the specified attribute.

    For example:

    ```
    @> describe vaults.name
    ```

  - If you specify an attribute name with a wildcard (`*`), the output contains all attributes that match the pattern.

    For example, the following command describes all attributes starting with '`n`' in all resources starting with '`v`':

    ```
    @> describe v*.n*
    ```

  - If *resource-spec* begins with '`mk`', the output contains only the attributes that are initializable during resource creation (`initializable != false`).

    For example, the following command describes all attributes that are initializable during file creation:

    ```
    @> describe mkfiles
    ```

You can further refine the output using the following options:

* `--mandatory`: Describes only the mandatory attributes that must be initialized during resource creation (`initializable = Mandatory`). For example:

```
@> describe mkfiles --mandatory
```

* `--optional`: Describes only the optional attributes that may be initialized during resource creation (`initializable = Optional`).

```
@> describe mkfiles --optional
```

– If *resource-spec* begins with '`ch`', the output contains only the attributes that are modifiable for an existing resource (`modifiable = true`).

For example, the following command describes the attributes that may be modified for existing files:

```
@> describe chfiles
```

- `--all`: Directs the command to display information for all Exascale resources.
- `--detail`: Optionally formats the command output so that values are displayed on separate lines, instead of using the default tabular output.

6.2.3 Service and Cluster Management

This section contains references for the Exascale command line interface (ESCLI) commands that are associated with service and cluster management:

- cellcli
  Execute a CellCLI command on a cell.
- chcluster
  Modify cluster attributes.
- chfeature
  Manage an Exascale feature.
- chfeatureupdate
  Change attributes for a deferred feature update.
- chservice
  Start, stop, restart, or disable a software service.
- dbmcli
  Execute a DBMCLI command on a compute server.
- lscell
  List storage cells.
- lscelldisk
  List cell disks.
- lscluster
  List cluster information.
- lscomputeserver
  List compute servers.

- lsfeature
  List features associated with an Exascale cluster.
- lsfeatureupdate
  List information about deferred feature updates.
- lsgriddisk
  List grid disks.
- lsservice
  List software services.
- mkfeature
  Create an Exascale feature definition.

6.2.3.1 cellcli

Execute a CellCLI command on a cell.

Purpose

The `cellcli` command enables you to run a CellCLI command on a specified Exadata storage server.

Syntax

```
cellcli --cell cell-name [ --xml ] [ -e ] CellCLI-command
```

Command Options

The options for the `cellcli` command are:

- *cell-name*: Specifies the name of cell that is the subject of the operation.

  The cell must be identified by its name. Use the lscell command to find all cell names.
- `--xml`: Generates XML-formatted command output.
- `-e` *`CellCLI-command`*: Specifies the CellCLI command. The `-e` argument is optional.

Usage Notes

To run the `cellcli` command in ESCLI, the Exascale user must have the `cl_admin` privilege. On the Exadata storage server, the command runs with the privilege of the `celladmin` user.

Examples

Example 6-2 Run a CellCLI command

In the following examples, the `list cell` command is run on `CELL1`. Both commands perform the same function. However, the second command specifies the optional `-e` argument.

```
@> cellcli --cell CELL1 list cell
```

```
@> cellcli --cell CELL1 -e list cell
```

Example 6-3 Run a CellCLI command and generate XML-formatted output

In this example, the `list cell` command is run on `CELL1` and the command output is rendered using XML.

```
@> cellcli --cell CELL1 --xml list cell
```

6.2.3.2 chcluster

Modify cluster attributes.

Purpose

The `chcluster` command allows you to shut down the Exascale cluster or to modify its attributes.

Syntax

```
chcluster [ --attributes attribute=value[,attribute=value] ... ] [ --
shutdown ]
```

Command Options

The options for the `chcluster` command are:

- `--attributes`: Specifies one or more attribute settings for the cluster.

  Use the `describe chcluster` command to view details about the cluster attributes you can modify with `chcluster`.

- `--shutdown`: Shuts down the Exascale cluster.

  To shut down the Exascale cluster, note that you must:

  – Run ESCLI as the Exascale `admin` user or as another user with the `cl_admin` cluster level storage privilege.

  – Connect ESCLI directly to an online Exascale control services (ERS) back-end server process. To do this you must start ESCLI and specify the `--ctrl` option as follows:

    ```
    $ escli --wallet admin-wallet-location --ctrl ERS-server-IP:8080
    ```

    To find the IP address for an online ERS server (*ERS-server-IP*), start a regular ESCLI session and use the following command:

    ```
    @> lsservice --filter serviceType=controlServices,status=ONLINE --
    attributes url
    ```

    The command displays the private IP addresses associated with all of the online ERS servers. You can use any of the reported IP addresses.

Examples

Example 6-4 Modify the Cluster

This example shows setting the compartment name for the Oracle Cloud Infrastructure (OCI) object storage container that is used for volume backups.

```
@> chcluster --attributes backupCompartmentName=myCompartment
```

Example 6-5 Shut Down the Cluster

This example shows how to shut down the Exascale cluster.

```
$ escli --wallet admin-wallet-location --ctrl ERS-server-IP:8080 chcluster --
shutdown
```

6.2.3.3 chfeature

Manage an Exascale feature.

Purpose

The `chfeature` command allows you to manage an Exascale software feature.

Syntax

```
chfeature feature-id [ --enable ]
```

Command Options

The options for the `chfeature` command are:

- *feature-id*: Identifies the feature that is the subject of the operation.
- `--enable`: Enables the specified feature.

6.2.3.4 chfeatureupdate

Change attributes for a deferred feature update.

Purpose

The `chfeatureupdate` command allows you to change attributes associated with a deferred feature update.

Syntax

```
chfeatureupdate name [ --attributes attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `chfeatureupdate` command are:

- *name*: Identifies the deferred feature update that is the subject of the operation.
- `--attributes`: Specifies one or more attribute settings for the deferred feature update.

Use the `describe chfeatureupdate` command to view details about the deferred feature update attributes you can modify with `chfeatureupdate`.

Examples

Example 6-6 Change the Time When a Deferred Feature Update Takes Effect

The following example shows how to change the time when a deferred feature update is scheduled to take effect.

```
@> chfeatureupdate deferred_update_1_1684214264 --attributes
expiryTime=2024-01-01T12:31:01
```

6.2.3.5 chservice

Start, stop, restart, or disable a software service.

Purpose

The `chservice` command allows you to start, stop, restart, or disable an Exascale software service. A `chservice` command can only perform one operation on one service instance.

Syntax

```
chservice --attributes name=service_cell[,frontend={true|false}] { --start |
--stop | --restart | --disable [ --force ]}


chservice --attributes name=service_compute { --start | --stop | --restart |
--disable [ --force ]}
```

Command Options

The options for the `chservice` command are:

- `--attributes`: Primarily identifies the server and service that is the subject of the operation. This option is also used to specify optional attribute settings.
  - `name`: Identifies the server and service that is the subject of the operation.

    For a service on an Exadata storage server, specify the `name` attribute using the format: `name=service_cell`. In the attribute value:
    - *service*: Specifies the service that you want to act on, which is one of the following:
      - `cellsrv`: Specifies the core Exadata cell services (CELLSRV, MS, and RS).
      - `egs`: Specifies the Exascale cluster service, also known as Exascale global service (EGS).
      - `ers`: Specifies the Exascale control service, also known as Exascale RESTful service (ERS).
      - `syseds`: Specifies the system vault manager service.
      - `usreds`: Specifies the user vault manager service.
      - `bsm`: Specifies the block store manager service.
      - `bsw`: Specifies the block store worker service.

    - *cell*: Specifies the name of the storage server (`cell`) that hosts the service.

      The cell must be identified by its name. Use the lscell command to find all cell names.

    For a service on an Exadata compute server, specify the `name` attribute using the format: `name=service_compute`. In the attribute value:

    - *service*: Specifies the service that you want to act on, which is one of the following:
      - `egs`: Specifies the Exascale cluster service, also known as Exascale global service (EGS).
      - `bsw`: Specifies the block store worker service.
    - *compute*: Specifies the name of the compute server (`dbserver`) that hosts the service.

      Depending on the required service location, specify either the name of a specific compute node virtual machine (VM) or the Exadata compute node host domain (KVM host or Dom0).
  - `frontend`: Optionally specifies whether to start ERS front-end server processes. This option is only valid when starting or restarting an ERS instance.

    To start the front-end server processes on the ERS instance, specify `frontend=true`. To not start the front-end server processes on the ERS instance, specify `frontend=false`.
- You must also specify one of the following operations:
  - `--start`: Starts a service. The service is also enabled if it was previously disabled.
  - `--stop`: Stops a service.
  - `--restart`: Restarts a service. The service is also enabled if it was previously disabled.
  - `--disable`: Disables a service. A disabled service is stopped and prevented from automatically restarting on the associated server.

    You can also add the `--force` option to forcibly disable the service, even if the server hosting the service is unavailable. This option only applies when disabling a service.

Examples

Example 6-7 Start the System Vault Manager Service (SYSEDS)

In this example, the system vault manager service is started on `CELL1`.

```
@> chservice --attributes name=syseds_CELL1 --start
```

Example 6-8 Restart the Exascale Control Service (ERS)

In this example, ERS is restarted on `CELL2`, along with the front-end ERS server processes.

```
@> chservice --attributes name=ers_CELL2,frontend=true --restart
```

6.2.3.6 dbmcli

Execute a DBMCLI command on a compute server.

Purpose

The `dbmcli` command enables you to run a DBMCLI command on a specified compute server.

Syntax

```
dbmcli --compute compute-name [ --xml ] [ -e ] dbmcli-command
```

Command Options

The options for the `dbmcli` command are:

- *compute-name*: Specifies the name of compute server that is the subject of the operation.

  The compute server must be identified by its name. Use the lscomputeserver command to find all compute server names.
- `--xml`: Generates XML-formatted command output.
- `-e` *dbmcli-command*: Specifies the DBMCLI command. The `-e` argument is optional.

Usage Notes

To run the `dbmcli` command in ESCLI, the Exascale user must have the `cl_admin` privilege. On the Exadata compute server, the command runs with the privilege of the `dbmadmin` user.

Examples

Example 6-9 Run a DBMCLI command

In the following examples, the `list dbserver` command is run on `DB01`. Both commands perform the same function. However, the second command specifies the optional `-e` argument.

```
@> dbmcli --compute DB01 list dbserver
```

```
@> dbmcli --compute DB01 -e list dbserver
```

Example 6-10 Run a DBMCLI command and generate XML-formatted output

In this example, the `list dbserver` command is run on `DB01` and the command output is rendered using XML.

```
@> dbmcli --compute DB01 --xml list dbserver
```

6.2.3.7 lscell

List storage cells.

Purpose

The `lscell` command displays information about Exadata storage servers in the Exascale cluster.

Syntax

```
lscell [ -l ] [ --detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lscell` command are:

- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: *<attribute><operator><value>*.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

- Sorting attributes are specified as: `[-]`*attribute*. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with –.

For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

```
--sort -name,createTime
```

Examples

Example 6-11 List Storage Servers

The following example shows how to list essential information about all of the storage servers in the Exascale cluster.

```
@> lscell
```

Example 6-12 List Storage Servers

The following example shows how to list detailed information about all of the storage servers in the Exascale cluster.

```
@> lscell --detail
```

6.2.3.8 lscelldisk

List cell disks.

Purpose

The `lscelldisk` command displays information about Exadata storage server cell disks in the Exascale cluster.

Syntax

```
lscelldisk [ celldisk [ celldisk ] ... ] [ -l ] [ --detail ] [ --attributes
attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lscelldisk` command are:

- *celldisk*: Identifies an Exadata cell disk that you want to list information about. If not specified, the command displays information about all cell disks.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-13 List Cell Disk Information

The following example shows how to list detailed information for all cell disks in the Exascale cluster.

```
@> lscelldisk --detail
```

Example 6-14 List Information for Specific Cell Disks

The following example shows how to list information about specific cell disks named `DISK1`.

```
@> lscelldisk --filter name=DISK1
```

6.2.3.9 lscluster

List cluster information.

Purpose

The `lscluster` command displays information about the Exascale cluster.

Syntax

```
lscluster [ -l ] [ --detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
```

```
[-]attribute] ... ]
     [ --count value ] [ --backup ] [ --volume ]
```

Command Options

The options for the `lscluster` command are:

- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.
- `--backup`: Limits the output to list only volume backup attributes.
- `--volume`: Limits the output to list only volume attributes, not including volume backup attributes.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-15 List Cluster Information

The following example shows how to list essential information about the Exascale cluster.

```
@> lscluster
```

Example 6-16 List Cluster Information

The following example shows how to list detailed information about the Exascale cluster.

```
@> lscluster --detail
```

Example 6-17 List Cluster Information

The following example shows how to list detailed information about volume attributes that are associated with the Exascale cluster.

```
@> lscluster --detail --volume
```

Example 6-18 List Cluster Information

The following example shows how to list detailed information about volume backup attributes that are associated with the Exascale cluster.

```
@> lscluster --detail --backup
```

6.2.3.10 lscomputeserver

List compute servers.

Purpose

The `lscomputeserver` command displays information about compute servers associated with the Exascale cluster.

Syntax

```
lscomputeserver [ -l ] [ --detail ] [ --attributes
attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lscomputeserver` command are:

- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: <*attribute*><*operator*><*value*>.

The allowed operators are =, !=, >=, <=, >, and <.

Multiple comma-separated filter conditions are combined using AND logic.

Dates can be specified using the following formats:

- `yyyy-MM-dd''T''HH:mm:ss`
- `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
- `HH:mm:ss` (Date is assumed to be today)

A date can also be followed by a timezone specification.

Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

- Sorting attributes are specified as: `[-]`*`attribute`*. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-19 List Compute Servers

The following example shows how to list essential information about all of the compute servers associated with the Exascale cluster.

```
@> lscomputeserver
```

Example 6-20 List Compute Servers

The following example shows how to list detailed information about all of the compute servers associated with the Exascale cluster.

```
@> lscomputeserver --detail
```

6.2.3.11 lsfeature

List features associated with an Exascale cluster.

Purpose

The `lsfeature` command displays information about software features in the Exascale cluster.

Syntax

```
lsfeature [ --detail ] [ --filter filter[,filter] ... ] [ -l ]
```

Command Options

The options for the `lsfeature` command are:

- `--detail`: Lists all attributes in a detailed form.

- `--filter`: Used to specify conditions for filtering the list output.
- `-l`: Returns output in a long, tabular form.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

6.2.3.12 lsfeatureupdate

List information about deferred feature updates.

Purpose

The `lsfeatureupdate` command displays information about all deferred feature updates associated with the Exascale cluster or details for a specific deferred feature update.

Syntax

```
lsfeatureupdate [ name ] [ --attributes attribute[,attribute] ... ]
     [ --detail ] [ --filter filter[,filter] ... ] [ -l ]
```

Command Options

The options for the `lsfeatureupdate` command are:

- *name*: Identifies the deferred feature update that you want to list information about. If not specified, the command displays information about all deferred feature updates associated with the Exascale cluster.
- `--attributes`: Lists the specific attributes to display.
- `--detail`: Lists all attributes in a detailed form.
- `--filter`: Used to specify conditions for filtering the list output.
- `-l`: Returns output in a long, tabular form.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

Multiple comma-separated filter conditions are combined using AND logic.

Dates can be specified using the following formats:

- `yyyy-MM-dd''T''HH:mm:ss`
- `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
- `HH:mm:ss` (Date is assumed to be today)

A date can also be followed by a timezone specification.

Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

Examples

Example 6-21 List All Deferred Feature Update Information

The following example shows how to list essential information about all of the deferred feature updates associated with the Exascale cluster.

```
@> lsfeatureupdate
```

Example 6-22 List Specific Deferred Feature Update Information

The following example shows how to list detailed information about a specific deferred feature update.

```
@> lsfeatureupdate deferred_update_1_1684214264 --detail
```

6.2.3.13 lsgriddisk

List grid disks.

Purpose

The `lsgriddisk` command displays information about Exadata storage server grid disks in the Exascale cluster.

Syntax

```
lsgriddisk [ griddisk [ griddisk ] ... ] [ -l ] [ --detail ] [ --attributes
attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lsgriddisk` command are:

- *griddisk*: Identifies an Exadata grid disk that you want to list information about. If not specified, the command displays information about all grid disks.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.

- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-23 List Grid Disk Information

The following example shows how to list detailed information for all grid disks in the Exascale cluster.

```
@> lsgriddisk --detail
```

Example 6-24 List Information for Specific Grid Disks

The following example shows how to list information about specific grid disks named `DISK2`.

```
@> lsgriddisk --filter name=DISK2
```

6.2.3.14 lsservice

List software services.

Purpose

The `lsservice` command displays information about software services in the Exascale cluster.

Syntax

```
lsservice [ -l ] [ --detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lsservice` command are:

- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with -.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-25 List Service Information

The following example shows how to list essential information about all of the services in the Exascale cluster.

```
@> lsservice
```

Example 6-26 List System Vault Manager Service Information

The following example shows how to list detailed information about system vault manager services in the Exascale cluster.

```
@> lsservice --detail --filter serviceType=systemVaultManagers
```

6.2.3.15 mkfeature

Create an Exascale feature definition.

Purpose

The `mkfeature` creates the definition for an Exascale software feature.

Syntax

```
mkfeature feature-id
```

Command Options

The options for the `mkfeature` command are:

- *feature-id*: Specifies the feature identifier.

6.2.4 Security and User Management

This section contains references for the Exascale command line interface (ESCLI) commands that are associated with security and user management:

- chacl
  Change an access control list (ACL).
- chuser
  Change attributes of a user, or upload API keys.
- chwallet
  Change authentication-related data in a local wallet.
- lsacl
  List ACLs for files and vaults.
- lskey
  Display key information.
- lsuser
  List user information.

- lswallet
  Display information stored in a wallet.
- mkkey
  Generate and write keys in PEM format.
- mkuser
  Create a new user.
- mkwallet
  Create a local wallet.
- rmuser
  Delete a user.

6.2.4.1 chacl

Change an access control list (ACL).

Purpose

The `chacl` command allows you to change the ACL for an Exascale vault or file.

Syntax

```
chacl  { file-name | vault-name } acl-string
```

Command Options

The options for the `chacl` command are:

- `{ file-name | vault-name }`: Specifies the name of the file or vault that is the subject of the operation.
- *acl-string*: Specifies an ACL string having the following format.

  ```
  [+]userID1:acl-priv[;userID2:acl-priv] ...
  ```

  In the ACL string:

  – The optional plus sign (+) at the beginning of the ACL string indicates that the specified ACL string is merged into the existing ACL for the file or vault. In this case, users previously listed in the ACL are updated, and new users are added. Without the optional plus sign, the previous ACL is overwritten.

  – *userIDn*: Specifies an Exascale user ID.

  Depending on the user creation method, the user ID may be a system-generated value (for example, `96a68014-5762-4579-86ee-29eb743decbd`) or a user-specified value (for example, `scott`).

  – *acl-priv*: Specifies an ACL privilege, which can be one of the following:

  * `I | inspect`: Specifies that the user can view attributes of the file or vault but not its contents.
  * `R | read`: Specifies that the user can read contents of the file, or list files in the vault. Also confers the `inspect` permission.
  * `U | use`: Specifies that the user can write to the file, and use the vault. Also confers all preceding permissions.

* `M | manage`: Specifies that the user can manage the file or vault. Also confers all preceding permissions.
* `0 | none`: Removes all existing permissions from the specified user.

Examples

Example 6-27 Replace a File ACL

In this example, the ACL string for the file is replaced with the new ACL string. Under the new ACL, `scott` is permitted to read and inspect the file. No other user can access this file unless permitted by the vault ACL.

```
@> chacl @VAULT/file scott:R
```

Example 6-28 Change a File ACL

In this example, the plus sign (`+`) at the beginning of the ACL string indicates that the specified ACL string is merged into the existing file ACL. In this case, any pre-existing permissions for `jason` are overwritten, and `jason` is now permitted to inspect, read, write, and manage the file. No other user permissions are changed.

```
@> chacl @VAULT/file +jason:M
```

Example 6-29 Replace a File ACL using an ACL String that Specifies Multiple Users

In this example, the ACL string for the file is replaced with the new ACL string that specifies permissions for multiple users. Under the new ACL, `scott` can inspect the file, and `jason` can read and inspect the file. No other user can access this file unless permitted by the vault ACL.

```
@> chacl @VAULT/file scott:inspect;jason:read
```

6.2.4.2 chuser

Change attributes of a user, or upload API keys.

Purpose

The `chuser` command allows you to change an attribute for an existing Exascale user, or manage API keys for a user.

Syntax

```
chuser user-ID [ --attributes attribute=value[,attribute=value] ... ] [ --
privilege privileges ] [ --public-key-file1 public-key ] [ --public-key-file2
public-key ] [ --public-key-file3 public-key ] [ --rm-public-key1 ] [ --rm-
public-key2 ] [ --rm-public-key3 ]
```

Command Options

The options for the `chuser` command are:

- *user-ID*: Identifies the unique user ID for the user that is the subject of the operation. You can use the lsuser command to find the ID for each user.
- `--attributes`: Optionally specifies values for attributes of the user.

- `--privilege`: Changes the user's privileges. The *privileges* value is a list of one or more privileges of the form `[+|-]privilege-1|privilege-2|...`.

  A privilege is one of the following: `no_privilege`, `cellsrv`, `egs`, `ers`, `syseds`, `usreds`, `bsm`, `bsw`, `ms`, `vlt_manage`, `vlt_use`, `vlt_read`, `vlt_inspect`, `cl_admin`, `cl_operator`, `cl_monitor`, `on_behalf_of`, `user_create`, `system_restore`. For descriptions, see User Privileges.

  `no_privilege` cannot be combined with any other type of privileges, otherwise an error is returned.

  Vault top-level privileges (`vlt_manage`, `vlt_use`, `vlt_read`, and `vlt_inspect`) are mutually exclusive. If any two or more are combined, an error is returned.

  Cluster privileges (`cl_admin`, `cl_operator`, and `cl_monitor`) are mutually exclusive. If any two or more are combined, an error is returned.

  An optional plus sign (`+`) at the beginning of the privilege string indicates that the specified privileges are added to the user's existing privileges. An optional minus sign (`-`) at the beginning of the privilege string indicates that the specified privileges are removed from the user's existing privileges. Otherwise, the user's existing privileges are overwritten by the specified privileges.
- `--public-key-file1`: Uploads a new public key in slot 1.
- `--public-key-file2`: Uploads a new public key in slot 2.
- `--public-key-file3`: Uploads a new public key in slot 3.
- `--rm-public-key1`: Deletes the public key in slot 1.
- `--rm-public-key2`: Deletes the public key in slot 2.
- `--rm-public-key3`: Deletes the public key in slot 3.

Examples

Example 6-30 Change the Name of a User

In this example, the name of the specified user is changed to `SCOTTY`.

```
@> chuser 59f6dce4-5687-4728-8751-acfe99089be2 --attributes name=SCOTTY
```

Example 6-31 Assign the Vault Top-Level Read Privilege to a User

In this example, the vault top-level read privilege is assigned to the specified user.

```
@> chuser 59f6dce4-5687-4728-8751-acfe99089be2 --privilege vlt_read
```

6.2.4.3 chwallet

Change authentication-related data in a local wallet.

Purpose

The `chwallet` command changes ESCLI authentication-related data like the user identifier and private key in a local wallet.

Syntax

```
chwallet  --wallet wallet-location [ --attributes
attribute=value[,attribute=value] ... ] [ --private-key-file pem-file [ --
force ]] [ --fetch-trust-store ] [ { --trusted-cert-file trust-file-name } ]
[ --clear-old-trusted-certs ] [ --private-key-remove ]
```

Command Options

The options for the `chwallet` command are:

- `--wallet`: Identifies the wallet that is the subject of the operation. *wallet-location* must be specified as a directory location or SSO file location. The wallet must already exist. A wallet can be created using the ESCLI mkwallet command.
- `--attributes`: Optionally specifies values for attributes of the wallet.
- `--private-key-file`: Specifies a regular file location that contains a private key in PEM format.
- `--force`: Optionally replaces the private key in the wallet without confirmation. This option works in conjunction with the `--private-key-file` option.
- `--fetch-trust-store`: Gets the trusted certificates from the Exascale storage cluster and stores them in the wallet.
- `--trusted-cert-file`: Reads trusted certificates from the specified file and stores them in the wallet. You can use this option to read a PEM format file containing one trusted certificate. Or, you can use a file containing multiple trusted certificates, which is a concatenation of multiple PEM format files, each containing one trusted certificate.
- `--clear-old-trusted-certs`: Removes all previous trusted certificates from the wallet.

  This option is useful for removing old and expired trusted certificates.

  > **Note:**
  >
  > This option also removes manually added trusted certificates and trusted certificates not issued by the trustStore.

- `--private-key-remove`: Removes the private key from the wallet.

Examples

Example 6-32 Set a Private Key and User Identifier in a Wallet

The following example shows setting the private key and user identifier inside the wallet at `/home/user/user.wallet`.

```
@> chwallet --wallet /home/user/user.wallet --private-key-file /home/user/
privatekey.pem --attributes user=0b9b8510-f88e-4f7b-ac57-10943c73dbe8
```

Example 6-33 Set the URL for the Exascale cluster services (EGS) in a Wallet

The following example shows setting the URL for the Exascale cluster services (EGS) inside the wallet at `/home/user/user.wallet`.

```
@> chwallet --wallet /home/user/user.wallet --attributes
exaRootUrl="egs=egsexc4:192.0.2.217:5045 egs=egsexc4:192.0.2.218:5045
egs=egsexc4:192.0.2.219:5045"
```

Example 6-34 Fetch Trust Store Certificates to a Wallet

The following example gets the trusted certificates from the Exascale storage cluster and stores them in the specified wallet. The command also removes the certificates that previously resided in the wallet.

```
@> chwallet --wallet /home/user/user.wallet --fetch-trust-store --clear-old-
trusted-certs
```

Example 6-35 Load Trust Store Certificates to a Wallet

The following examples read the trusted certificates from the specified PEM files and stores them in the specified wallet. Each command also removes the certificates that previously resided in the wallet.

Both examples are functionally equivalent assuming that `/tmp/concatenated-trust.pem` is a concatenation of `/tmp/trust1.pem`, `/tmp/trust2.pem`, and `/tmp/trust3.pem`.

```
@> chwallet --wallet /home/user/user.wallet --trusted-cert-file /tmp/
trust1.pem --trusted-cert-file /tmp/trust2.pem --trusted-cert-file /tmp/
trust3.pem --clear-old-trusted-certs
```

```
@> chwallet --wallet /root/eswallet/ --trusted-cert-file /tmp/concatenated-
trust.pem --clear-old-trusted-certs
```

Related Topics

- mkwallet
  Create a local wallet.

6.2.4.4 lsacl

List ACLs for files and vaults.

Purpose

The `lsacl` command allows you to list access control lists (ACLs) for Exascale files and vaults.

Syntax

```
lsacl  [ file-name | vault-name ] ...
```

Usage Notes

Note the following information when using this command:

- *file-name* and *vault-name* specify the names of files or vaults that are the subject of the operation. You can specify multiple files or vaults for which you want to display ACLs. The asterisk (`*`) can be used for wildcard searches.
- If you do not specify any *file-name* or *vault-name*, then the output depends on the current working directory in the ESCLI session. If the current working directory is the root directory, then the command returns the ACLs for all vaults. If the current working directory is not the root directory, then the command returns the ACLs for all files under the current working directory.

Examples

Example 6-36 List ACLs for all vaults

This example lists the ACLs for all vaults.

```
@> lsacl
```

Example 6-37 List ACLs for all files in a vault

This example lists the ACLs for all files in the `@MYDATA` vault.

```
@> cd @MYDATA
@MYDATA/> lsacl
```

Example 6-38 List ACLs for a Vault and Files

This example lists the ACLs for the `@MYDATA` vault and the files that it contains.

```
@> lsacl @MYDATA @MYDATA/*
```

6.2.4.5 lskey

Display key information.

Purpose

The `lskey` command displays information about the specificed key file.

Syntax

```
lskey pem-file [ --attributes attribute[,attribute] ... ] [ --detail ] [ -l ]
```

Command Options

The options for the `lskey` command are:

- *pem-file*: Identifies the key file that you want to display information about. The *pem-file* must specify a regular file location for a key file in PEM format.
- `--attributes`: Lists the specific attributes to display.
- `--detail`: Displays all of the key attributes.
- `-l`: Returns output in a long, tabular form.

Examples

Example 6-39 List Detailed Key Information

The following example displays detailed information from the key file named `pub.pem`.

```
@> lskey pub.pem --detail
```

Example 6-40 List Specific Key Information

The following example displays specific attributes (`id`, `privateKeyFile`, and `publicKeyFingerprint`) from the key file named `priv.pem`.

```
@> lskey priv.pem --attributes id,privateKeyFile,publicKeyFingerprint
```

6.2.4.6 lsuser

List user information.

Purpose

The `lsuser` command displays information about Exascale users.

Syntax

```
lsuser [ user-ID ] [ --attributes attribute[,attribute] ... ] [ --detail ] [ -
l ]
```

Command Options

The options for the `lsuser` command are:

- *user-ID*: Identifies the user that you want to list information about. If no user is identified, the command displays information about all users.
- `--attributes`: Lists the specific attributes to display.
- `--detail`: Lists all attributes in a detailed form.
- `-l`: Returns output in a long, tabular form.

Examples

Example 6-41 List Information for a Specific User

The following example shows how to list detailed information about the user ID: `SCOTT0123`.

```
@> lsuser SCOTT0123 --detail
```

6.2.4.7 lswallet

Display information stored in a wallet.

Purpose

The `lswallet` command displays ESCLI authentication data that is stored in a wallet.

Syntax

```
lswallet [ --wallet wallet-location ] [ --detail ] [ --private-key-file pem-
file ] [ --public-key-file pem-file ] [ --public-key-file8 pem-file ] [ --
attributes attribute[,attribute] ... ] [ --trusted-cert-file cert-file ]
```

Command Options

The options for the `lswallet` command are:

- `--wallet`: Identifies the wallet that you want to display information about. The *wallet-location* must be specified as a directory location or SSO file location. If not specified, the command displays information about the wallet used to authenticate the current ESCLI session.
- `--detail`: Displays the user ID and fingerprint of the public key from the wallet.
- `--private-key-file`: Extracts the private key from the wallet and stores it in the specified PEM file.
- `--public-key-file`: Extracts the public key from the wallet and stores it in the specified PEM file.
- `--public-key-file8`: Extracts the public key from the wallet in PKCS8 format and stores it in the specified PEM file.
- `--attributes`: Lists the specific attributes to display.
- `--trusted-cert-file`: Extracts the trusted certificates from the wallet and saves them in PEM format files.

  Files are created with numeric suffixes `1`, `2`, `3`, and so on. For example, if you specify `--trusted-cert-file cert_file`, then the certificates are stored in the files `cert_file1`, `cert_file2`, `cert_file3`, and so on.

Examples

Example 6-42 List Wallet Information

The following example displays essential information from the wallet used to authenticate the current ESCLI session.

```
@> lswallet
```

Example 6-43 List Detailed Wallet Information

The following example displays detailed information from the wallet used to authenticate the current ESCLI session.

```
@> lswallet --detail
```

Example 6-44 Extract a Private Key

The following example extracts the private key from the wallet used to authenticate the current ESCLI session. The private key is stored in `/home/user/privatekey.pem`

```
@> lswallet --private-key-file /home/user/privatekey.pem
```

Example 6-45 Extract a Public Key

The following example extracts the public key from the wallet used to authenticate the current ESCLI session. The public key is stored in `/home/user/publickey.pem`

```
@> lswallet --public-key-file /home/user/publickey.pem
```

Example 6-46 Extract Certificates to Files

The following example extracts the trusted certificates from the wallet used to authenticate the current ESCLI session. The certificates are stored in regular files `/home/user/cert1`, `/home/user/cert2`, `/home/user/cert3`, and so on.

```
@> lswallet --trusted-cert-file /home/user/cert
```

6.2.4.8 mkkey

Generate and write keys in PEM format.

Purpose

The `mkkey` command generates and writes a private key to a regular file in PEM format. You can also generate and write the corresponding public key to a regular file in PEM format.

Syntax

```
mkkey [ --private-key-file pem-file ] [ --public-key-file pem-file ] [ --
attributes attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `mkkey` command are:

- `--private-key-file`: Specifies a regular file location for the private key.
- `--public-key-file`: Optionally specifies a regular file location for the corresponding public key.
- `--attributes`: Optionally specifies attributes to change. This option provides an alternative method for specifying the key file names.

Examples

Example 6-47 Generate a Key Pair

The following examples generate a key pair and stores the private and public keys in `/home/user/privatekey.pem` and `/home/user/publickey.pem`, respectively.

```
@> mkkey --private-key-file /home/user/privatekey.pem --public-key-file /home/
user/publickey.pem
```

```
@> mkkey --attributes privateKeyFile=/home/user/privatekey.pem,publicKeyFile=/
home/user/publickey.pem
```

6.2.4.9 mkuser

Create a new user.

Purpose

The `mkuser` command allows you to create a new Exascale user with the specified user name.

Syntax

```
mkuser username [ --attributes attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `mkuser` command are:

- *username*: Specifies the name of the user. The user name must start with an alphanumeric character, and can only contain alphanumeric characters, hyphens (-) and periods (.).
- `--attributes`: Optionally specifies attributes to change.

  Use the `describe mkuser` command to view details about all the user attributes you can set with `mkuser`.

Usage Notes

Note the following information when using this command:

- You can set the user identifier by specifying the `id` attribute.

  If specified, the user identifier must start with an alphanumeric character, and can only contain alphanumeric characters, hyphens (-) and periods (.).

  If none is specified, then a unique user ID is generated automatically and returned by the command. The user ID value is required when managing users using chuser, rmuser, and lsuser.

  The user ID is fixed at user creation time. There is no way to modify the user ID afterward.
- You can set the user's privileges by specifying the `privilege` attribute.

  The `privilege` value is a list of one or more privileges of the form *`privilege-1|privilege-2|...`*.

  A privilege is one of the following: `no_privilege`, `cellsrv`, `egs`, `ers`, `syseds`, `usreds`, `bsm`, `bsw`, `ms`, `vlt_manage`, `vlt_use`, `vlt_read`, `vlt_inspect`, `cl_admin`, `cl_operator`, `cl_monitor`, `on_behalf_of`, `user_create`, `system_restore`. For descriptions, see User Privileges.

  `no_privilege` cannot be combined with any other type of privileges, otherwise an error is returned.

  Vault top-level privileges (`vlt_manage`, `vlt_use`, `vlt_read`, and `vlt_inspect`) are mutually exclusive. If any two or more are combined, an error is returned.

  Cluster privileges (`cl_admin`, `cl_operator`, and `cl_monitor`) are mutually exclusive. If any two or more are combined, an error is returned.

  If the `privilege` attribute is not specified, then the user is created and assigned only the `vlt_inspect` privilege by default.

Examples

Example 6-48 Create a User

This example shows creating a user named `CHERIE`.

```
@> mkuser CHERIE
```

Example 6-49 Create a User with a Specific User Identifier

This example shows creating a user named `SCOTT` using `SCOTT0123` as the user identifier.

```
@> mkuser SCOTT --attributes id=SCOTT0123
```

Example 6-50 Create a User with a Specific Privileges

This example shows creating a user named `PETER` with a specific set of privileges.

```
@> mkuser PETER --attributes privilege=vlt_read|ers
```

6.2.4.10 mkwallet

Create a local wallet.

Purpose

The `mkwallet` creates a new Oracle wallet at the specified location.

Syntax

```
mkwallet --wallet wallet-location
```

Command Options

The options for the `mkwallet` command are:

- `--wallet`: Specifies the location where the new wallet is created. *wallet-location* must be specified as a directory location or SSO file location.

Examples

Example 6-51 Create a Wallet

The following example shows creating a wallet at `/home/user/user.wallet`.

```
@> mkwallet --wallet /home/user/user.wallet
```

6.2.4.11 rmuser

Delete a user.

Purpose

The `rmuser` command deletes an existing Exascale user.

Syntax

```
rmuser user-ID
```

Command Options

The options for the `rmuser` command are:

- *user-ID*: Specifies the unique user ID for the user that is being deleted. You can use the lsuser command to find the ID for each user.

Examples

Example 6-52 Delete a User

This example shows deleting the user with the ID `0b9b8510-f88e-4f7b-ac57-10943c73dbe8`.

```
@> rmuser 0b9b8510-f88e-4f7b-ac57-10943c73dbe8
```

6.2.5 Storage Pool and Pool Disk Management

This section contains references for the Exascale command line interface (ESCLI) commands that are associated with storage pool and pool disk management:

- chpooldisk
  Modify a pool disk.
- chstoragepool
  Modify a storage pool.
- lspooldisk
  List pool disks.
- lsstoragepool
  List storage pools.
- lsstoragepooloperation
  List ongoing storage pool operations.
- mkstoragepool
  Create a storage pool.
- rmstoragepool
  Delete a storage pool.

6.2.5.1 chpooldisk

Modify a pool disk.

Purpose

The `chpooldisk` command enables you to modify an Exascale pool disk.

Syntax

```
chpooldisk pooldisk-id { --offline | --online }
```

Command Options

The options for the `chpooldisk` command are:

- *pooldisk-id*: Identifies the Exadata pool disk that you want to modify. You can use the lspooldisk command to find the ID for each pool disk.
- `--offline`: Takes the pool disk offline.
- `--online`: Bring the pool disk online.

Examples

Example 6-53 Bring a Pool Disk Online

The following example shows how to bring a pool disk online.

```
@> chpooldisk 10 --online
```

6.2.5.2 chstoragepool

Modify a storage pool.

Purpose

The `chstoragepool` command enables you to modify an Exascale storage pool.

Syntax

```
chstoragepool storagepool-name [ --attributes
attribute=value[,attribute=value] ... ] [ --reconfig [ --force ]]
```

Command Options

The options for the `chstoragepool` command are:

- *storagepool-name*: Identifies an Exascale storage pool that you want to modify.
- `--attributes`: Optionally specifies one or more attribute settings for the storage pool.

  Use the `describe chstoragepool` command to view details about the storage pool attributes you can modify with `chstoragepool`.
- `--reconfig`: Reconfigures the storage pool and makes permanent any changes to the number or size of the underlying pool disks. Reconfiguration may trigger a rebalance operation to ensure that data is spread evenly across the pool disks in the storage pool. The time required for reconfiguration depends on the amount of data movement required to rebalance the storage pool.
- `--force`: Optionally forces a reconfiguration operation even if the system detects no apparent change to the underlying pool disks.

Examples

Example 6-54 Set the Disk Offline Timer for a Storage Pool

The following example shows how to set the disk offline timer attribute associated with the Exascale storage pool named `SP1`.

```
@> chstoragepool SP1 --attributes diskOfflineTimerInMins=60
```

Example 6-55 Rediscover Pool Disks and Reconfigure the Storage Pool

The following example shows how to forcibly re-discover the pool disks and reconfigure the specified storage pool.

```
@> chstoragepool SP1 --reconfig --force
```

6.2.5.3 lspooldisk

List pool disks.

Purpose

The `lspooldisk` command displays information about Exadata storage server pool disks in the Exascale cluster.

Syntax

```
lspooldisk [ pooldisk-id [ pooldisk-id ] ... ] [ -l ] [ --detail ] [ --
attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lspooldisk` command are:

- *pooldisk-id*: Identifies an Exadata pool disk that you want to list information about. If not specified, the command displays information about all pool disks.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

The allowed operators are =, !=, >=, <=, >, and <.

Multiple comma-separated filter conditions are combined using AND logic.

Dates can be specified using the following formats:

- `yyyy-MM-dd''T''HH:mm:ss`
- `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
- `HH:mm:ss` (Date is assumed to be today)

A date can also be followed by a timezone specification.

Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

- Sorting attributes are specified as: `[-]`*`attribute`*. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-56 List Pool Disk Information

The following example shows how to list detailed information for all pool disks in the Exascale cluster.

```
@> lspooldisk --detail
```

Example 6-57 List Information for Specific Pool Disks

The following example shows how to list information about specific pool disks named `DISK3`.

```
@> lspooldisk --filter name=DISK3
```

6.2.5.4 lsstoragepool

List storage pools.

Purpose

The `lsstoragepool` command displays information about Exascale storage pools.

Syntax

```
lsstoragepool [ storagepool-name [ storagepool-name ] ... ] [ -l ] [ --
detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lsstoragepool` command are:

- *storagepool-name*: Identifies an Exascale storage pool that you want to list information about. If not specified, the command displays information about all storage pools.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-58 List Storage Pool Information

The following example shows how to list information for all storage pools in the Exascale cluster.

```
@> lsstoragepool
```

Example 6-59 List Information for a Specific Storage Pool

The following example shows how to list detailed information about the storage pool named `SP1`.

```
@> lsstoragepool SP1 --detail
```

6.2.5.5 lsstoragepooloperation

List ongoing storage pool operations.

Purpose

The `lsstoragepooloperation` command displays information about storage pool operations that are currently active in the Exascale cluster.

Syntax

```
lsstoragepooloperation [ operation-id [ operation-id ] ... ] [ -l ] [ --
detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lsstoragepooloperation` command are:

- *operation-id*: Identifies an Exascale storage pool operation that you want to list information about. If not specified, the command displays information about all active operations.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-60 List Storage Pool Operations

The following example shows how to list information for all ongoing storage pool operations in the Exascale cluster.

```
@> lsstoragepooloperation
```

Example 6-61 List Information for a Specific Operation

The following example shows how to list detailed information about a specific operation.

```
@> lsstoragepooloperation 3192092 --detail
```

6.2.5.6 mkstoragepool

Create a storage pool.

Purpose

The `mkstoragepool` command creates a pool of Exascale storage.

Syntax

```
mkstoragepool storagepool-name [ --attributes
attribute=value[,attribute=value] ... ] [ --nowait ]
```

Command Options

The options for the `mkstoragepool` command are:

- `--attributes`: Optionally specifies one or more attribute settings for the storage pool.

  Use the `describe mkstoragepool` command to view details about the storage pool attributes you can specify with `mkstoragepool`.

- `--nowait`: Specifies that the command returns immediately while the operation completes in the background. Without this option, the command does not return until the storage pool is created.

Usage Notes

Before you create a storage pool, you need some pool disks that are already associated with the storage pool name.

Examples

Example 6-62 Create a Storage Pool

The following example shows creating a storage pool named `POOL1`.

```
@> mkstoragepool POOL1
```

6.2.5.7 rmstoragepool

Delete a storage pool.

Purpose

The `rmstoragepool` command enables you to delete an Exascale storage pool.

Syntax

```
rmstoragepool storagepool-name
```

Command Options

The options for the `rmstoragepool` command are:

- *storagepool-name*: Identifies the Exascale storage pool that you want to delete.

Usage Notes

You can use the `rmstoragepool` command in cases where you want to:

- Remove storage hardware from the Exascale cluster.
- Migrate data to another storage pool.

Examples

Example 6-63 Delete a Storage Pool

The following example shows how to delete a storage pool named `SP1`.

```
@> rmstoragepool SP1
```

6.2.6 Vault Management

This section contains references for the Exascale command line interface (ESCLI) commands that are associated with vault management:

- chvault
  Change the attributes of a vault.
- mkvault
  Create a vault.
- rmvault
  Delete a vault.

6.2.6.1 chvault

Change the attributes of a vault.

Purpose

The `chvault` command modifies the attributes of a vault.

Syntax

```
chvault vault --attributes attribute=value[,attribute=value] ...
```

Command Options

The options for the `chvault` command are:

- *vault*: Identifies the name of the vault that you want to change.
- `--attributes`: Specifies attributes to change.

  Use the `describe chvault` command to view details about all the vault attributes you can modify with `chvault`.

Usage Notes

Note the following information when using this command:

- The command only changes the specified attributes. Other settings are unaffected.
- Following is the list of vault-specific resource provisioning attributes that may be changed:
  - `spaceProvEF`: Provisions the vault with the specified amount of `EF` storage space.
  - `spaceProvHC`: Provisions the vault with the specified amount of `HC` storage space.
  - `iopsProvEF`: Provisions the vault with the specified number of IOPS from `EF` storage.
  - `iopsProvHC`: Provisions the vault with the specified number of IOPS from `HC` storage.
  - `flashCacheProv`: Provisions the vault with the specified amount of flash cache space.
  - `flashLogProv`: Boolean value (`true` or `false`) indicating whether the vault is provisioned with access to Exadata Smart Flash Log. The default value is `true`.
  - `xrmemCacheProv`: Provisions the vault with the specified amount of Exadata RDMA Memory Cache (XRMEM cache) space.

    Use this attribute only on systems with `HC` or `EF` storage, but not both. For systems with `HC` and `EF` storage, use the following media-specific XRMEM cache provisioning attributes.
  - `xrmemCacheProvEF`: Provisions the vault with the specified amount of XRMEM cache space associated with `EF` media.
  - `xrmemCacheProvHC`: Provisions the vault with the specified amount of XRMEM cache space associated with `HC` media.
- To specify an amount of space, you can:
  - Use an integer value representing the number of bytes. The maximum value is `4503599626321920`.

  - Specify a space value using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
  - Use `unlimited` to specify an unlimited amount of space.
- To specify a number of IOPS, you can:
  - Use an integer value. The maximum value is `4294967295`.
  - Use `unlimited` to specify an unlimited number of IOPS.
- Specifying `0` (zero) as the value for a cache provisioning attribute (`flashCacheProv`, `xrmemCacheProv`, `xrmemCacheProvEF`, or `xrmemCacheProvHC`) effectively disables use of the corresponding cache for the vault.

Examples

Example 6-64 Set the Amount of Space Provisioned for a Vault

The following example shows setting the amount of provisioned `HC` storage space to 10 terabytes for the vault named `MYDATA`.

```
@> chvault MYDATA --attributes spaceProvHC=10T
```

6.2.6.2 mkvault

Create a vault.

Purpose

The `mkvault` command creates an Exascale vault.

Syntax

```
mkvault [@]vault [ --attributes attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `mkvault` command are:

- *vault*: Specifies the name of the vault.
- `--attributes`: Optionally specifies attributes to set.

  Use the `describe mkvault` command to view details about all the vault attributes you can set with `mkvault`.

Usage Notes

Note the following information when using this command:

- The default value for all resource provisioning attributes is effectively unlimited. Consequently, a vault that is created without any resource provisioning attribute settings has access to all of the resources in the Exascale cluster.

  Following is the list of vault-specific resource provisioning attributes that may be set:
  - `spaceProvEF`: Provisions the vault with the specified amount of `EF` storage space.
  - `spaceProvHC`: Provisions the vault with the specified amount of `HC` storage space.

  - `iopsProvEF`: Provisions the vault with the specified number of IOPS from `EF` storage.
  - `iopsProvHC`: Provisions the vault with the specified number of IOPS from `HC` storage.
  - `flashCacheProv`: Provisions the vault with the specified amount of flash cache space.
  - `flashLogProv`: Boolean value (`true` or `false`) indicating whether the vault is provisioned with access to Exadata Smart Flash Log. The default value is `true`.
  - `xrmemCacheProv`: Provisions the vault with the specified amount of Exadata RDMA Memory Cache (XRMEM cache) space.

    Use this attribute only on systems with `HC` or `EF` storage, but not both. For systems with `HC` and `EF` storage, use the following media-specific XRMEM cache provisioning attributes.
  - `xrmemCacheProvEF`: Provisions the vault with the specified amount of XRMEM cache space associated with `EF` media.
  - `xrmemCacheProvHC`: Provisions the vault with the specified amount of XRMEM cache space associated with `HC` media.

- Prior to consumption, all provisioned resources are logically provisioned. For example, the `spaceProvHC` attribute specifies the maximum amount of `HC` storage space that the vault can consume. However, physical space is only consumed when data is written to files in the vault.
- To specify an amount of space, you can:
  - Use an integer value representing the number of bytes. The maximum value is `4503599626321920`.
  - Specify a space value using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
  - Use `unlimited` to specify an unlimited amount of space.
- To specify a number of IOPS, you can:
  - Use an integer value. The maximum value is `4294967295`.
  - Use `unlimited` to specify an unlimited number of IOPS.
- If you specify a value for any provisioning attributes, then the vault is limited to use the provisioned resources from the corresponding storage pools and the other storage pools are effectively disabled.

  If you specify a space provisioning attribute (`spaceProvEF`, or `spaceProvHC`) without a corresponding IOPS provisioning attribute (`iopsProvEF`, or `iopsProvHC`), then the vault can consume the specified space and use unlimited IOPS in the associated storage pool.

  If you specify an IOPS provisioning attribute without a corresponding space provisioning attribute, then the vault can consume unlimited space in the associated storage pool but the I/O bandwidth is limited by the IOPS provisioning attribute.
- Specifying `0` (zero) as the value for a cache provisioning attribute (`flashCacheProv`, `xrmemCacheProv`, `xrmemCacheProvEF`, or `xrmemCacheProvHC`) effectively disables use of the corresponding cache for the vault.

Examples

Example 6-65 Create a Vault with Unlimited Storage

The following example shows creating a vault named `MYDATA` that has unlimited access to all of the resources in the Exascale cluster.

```
@> mkvault @MYDATA
```

Example 6-66 Create a Vault with Provisioned Storage

The following example shows creating a vault named `MYDATA2` that is provisioned with 5 terabytes of `HC` storage space. The resulting vault is implicitly provisioned with unlimited IOPS.

```
@> mkvault @MYDATA2 --attributes spaceProvHC=5T
```

Example 6-67 Create a Vault with Provisioned Storage

The following example shows creating a vault named `MYDATA3` that is provisioned to use `EF` and `HC` storage media. From the `EF` storage pool, the vault can consume 10 terabytes of storage space and unlimited IOPS. From the `HC` storage pool, the vault can consume unlimited storage space and 100000 IOPS.

```
@> mkvault @MYDATA3 --attributes spaceProvEF=5T,iopsProvHC=100000
```

6.2.6.3 rmvault

Delete a vault.

Purpose

The `rmvault` command deletes a vault.

Syntax

```
rmvault [@]vault [ --force ] [ --nowait ]
```

Command Options

The options for the `rmvault` command are:

- *vault*: Identifies the vault that is being deleted.
- `--force`: Also deletes all files in the vault.
- `--nowait`: Instead of waiting for the command to finish, the command returns immediately while the operation completes in the background.

Examples

Example 6-68 Delete a Vault

The following example shows deleting the vault named `MYDATA`.

```
@> rmvault @MYDATA
```

6.2.7 Dataset Management

This section contains references for the Exascale command line interface (ESCLI) commands that are associated with dataset management:

- lsdataset
  List datasets.

6.2.7.1 lsdataset

List datasets.

Purpose

The `lsdataset` command displays information about Exascale datasets. It can also list the files contained in a specific dataset.

Syntax

```
lsdataset [ -l ] [ --detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ] [ --files [ --recursive ]] [ dataset-id ]
```

Command Options

The options for the `lsdataset` command are:

- *dataset-id*: The identifier for a dataset that you want to display information about.

  Each system-defined dataset has a unique composite identifier, which contains unique identifiers for the associated entities. The dataset identifier has one of the following formats:

  - *@Vault-name*: Identifies the vault-level dataset for a specific named vault.
  - *@Vault-name*:*GI-cluster-ID*: Identifies the dataset for a specific Oracle Grid Infrastructure (GI) cluster, which consumes storage space in the specified vault.
  - *@Vault-name*:*GI-cluster-ID*:*CDB-ID*: Identifies the dataset for a specific Oracle multitenant container database (CDB) that belongs to the specified GI cluster.
  - *@Vault-name*:*GI-cluster-ID*:*CDB-ID*:*PDB-ID*: Identifies the dataset for an Oracle pluggable database (PDB) that is associated with the specified CDB, GI cluster, and vault.

  The asterisk (*) can be used for wildcard searches. For example, *@Vault-name*:*GI-cluster-ID*:* displays information about all CDB datasets associated with the specified GI cluster and vault.

  If *dataset-id* is not specified, the command displays information about all datasets in all vaults.

- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.

- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.
- `--files`: Lists files that are in the specified dataset.
- `--recursive`: Also lists files that are in descendant datasets.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with -.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

- You must specify a *dataset-id* in conjunction with the `--files` option.

Examples

Example 6-69 Display All Datasets

You can use the `lsdataset` command with no other options to display a list of all datasets across all vaults.

```
@> lsdataset
id                                                                  name
@MYDATA                                                             @MYDATA
@MYDATA:a5b4997a027d6f91ffd9729702ff6ec5                            @MYDATA/
cluster1
@MYDATA:a5b4997a027d6f91ffd9729702ff6ec5:1427076301                 @MYDATA/
cluster1:mydb
@MYDATA:a5b4997a027d6f91ffd9729702ff6ec5:1427076301:1402749181      @MYDATA/
cluster1:mydb.PDB$SEED
@MYDATA:a5b4997a027d6f91ffd9729702ff6ec5:1427076301:2132037342      @MYDATA/
cluster1:mydb.CDB$ROOT
@MYDATA:a5b4997a027d6f91ffd9729702ff6ec5:1427076301:2164757665      @MYDATA/
cluster1:mydb.MYDB_PDB1
```

```
@VAULT2                                                                    @VAULT2
...
```

Example 6-70 Display Additional Dataset Attributes

```
@> lsdataset --attributes id,name,fileUsageHC,datasetUsageHC
id
name                                fileUsageHC  datasetUsageHC
@MYDATA
@MYDATA                                  0.0000         3.4961G
@MYDATA:a5b4997a027d6f91ffd9729702ff6ec5                                   @MYDATA/
cluster1                           294.5339M         3.4961G
@MYDATA:a5b4997a027d6f91ffd9729702ff6ec5:1427076301                        @MYDATA/
cluster1:mydb                       25.2656M         3.2085G
@MYDATA:a5b4997a027d6f91ffd9729702ff6ec5:1427076301:1402749181             @MYDATA/
cluster1:mydb.PDB$SEED             650.6719M       650.6719M
@MYDATA:a5b4997a027d6f91ffd9729702ff6ec5:1427076301:2132037342             @MYDATA/
cluster1:mydb.CDB$ROOT               1.9056G         1.9056G
@MYDATA:a5b4997a027d6f91ffd9729702ff6ec5:1427076301:2164757665             @MYDATA/
cluster1:mydb.MYDB_PDB1            658.1719M       658.1719M
@VAULT2
@VAULT2                                  0.0000       110.8750M
...
```

Example 6-71 Display Files in a Dataset

```
@> lsdataset --files
@MYDATA:a5b4997a027d6f91ffd9729702ff6ec5:1427076301:2164757665
DATAFILE/mydb_pdb1_db.f
DATAFILE/mydb_pdb1_tmp.f
DATAFILE/mydb_pdb1_xdb.f
DATAFILE/mydb_pdb1_ax.f
```

Example 6-72 Using the Recursive Display Option

The following examples illustrate using the `--recursive` option.

Without the `--recursive` option, the following example shows the files belonging directly to the specified GI cluster dataset.

```
@> lsdataset --files @MYDATA:a5b4997a027d6f91ffd9729702ff6ec5
cluster1.ocr
cluster1/vfile1
```

With the `--recursive` option, the following example shows the files belonging to the specified GI cluster and all of the associated CDB and PDB datasets.

```
@> lsdataset --files --recursive @MYDATA:a5b4997a027d6f91ffd9729702ff6ec5
DATAFILE/mydb_pdb0_ax.f
dbs/xspfile.ora
DATAFILE/mydb_pdb0.xml
DATAFILE/mydb_pdb1_db.f
DATAFILE/t_ax1.f
DATAFILE/t_db1.f
```

```
CLUSTER1-A5B4997A027D6F91FFD9729702FF6EC5/MYDB/TEMPFILE/TEMP.OMF.2012709C
DATAFILE/t_xdb1.f
DATAFILE/mydb_pdb1_tmp.f
dbs/t_cf1.f
DATAFILE/mydb_pdb0_xdb.f
DATAFILE/mydb_pdb0_tmp.f
DATAFILE/t_undo1.f
cluster1.ocr
DATAFILE/mydb_pdb1_xdb.f
CLUSTER1-A5B4997A027D6F91FFD9729702FF6EC5/MYDB/PASSWORD/pwdMYDB.4ABC8604
cluster1/vfile1
DATAFILE/mydb_pdb0_db.f
DATAFILE/mydb_pdb1_ax.f
...
```

6.2.8 File Management

This section contains references for the Exascale command line interface (ESCLI) commands that are associated with file management:

- cd
  Change the current directory.
- chfile
  Change file attributes.
- clonefile
  Clone one or more files.
- extentmap
  Display storage extent information.
- getfile
  Copy a file from Exascale to the local file system.
- ls
  List files and vaults.
- lssnapshots
  List snapshots associated with a file.
- mkfile
  Create a file.
- putfile
  Copy a file from the local file system to Exascale.
- rmfile
  Delete a file.
- snapshotfile
  Create snapshots.

6.2.8.1 cd

Change the current directory.

Purpose

The `cd` command allows you to change the current directory for file operations. File operations are performed relative to the specified directory if absolute paths are not used.

Syntax

```
cd directory
```

Command Options

directory: Specifies the directory to change to. You can specify an absolute directory path or a directory path that is relative to the current directory.

Examples

Example 6-73 Change Directories using an Absolute Directory Path

You can change to a specified absolute directory path, such as `@VAULT/dir`.

```
@> cd @VAULT/dir
@VAULT/dir/>
```

Example 6-74 Change Directories using a Relative Directory Path

You can change directories using a specified relative directory path, such as `../dir2`.

```
@VAULT/dir/> cd ../dir2
@VAULT/dir2/>
```

6.2.8.2 chfile

Change file attributes.

Purpose

The `chfile` command changes attributes of a file.

Syntax

```
chfile filename --attributes attribute=value[,attribute=value] ...
```

Command Options

The options for the `chfile` command are:

- *filename*: Specifies the name of the file you are changing.
- `--attributes`: Specifies attributes to change.

Use the `describe chfile` command to view details about all the file attributes you can modify with `chfile`.

Examples

Example 6-75 Change a File

The following example shows changing the size of the file named `x` to 100 KB.

```
@MYDATA/> chfile x --attributes size=100k
File altered.

@MYDATA/> ls -l
Total 1
100.0k  29 Jun 21:58   x
```

6.2.8.3 clonefile

Clone one or more files.

Purpose

The `clonefile` command allows you to create a writable clone of a file or group of files.

Syntax

```
clonefile source target [ source target ]... [ --exclude exclude-spec ]...
```

Command Options

The options for the `clonefile` command are:

- *source*: Specifies the name of the file or files that are the clone source.
- *target*: Specifies the name of the file or files that are the clone destination.
- `--exclude`: Specifies the name of the file or files that are excluded from the clone operation.

Usage Notes

Note the following information when using this command:

- You can use a wildcard (`*`) in the *source* to specify multiple source files, in which case the corresponding *target* must also contain a matching wildcard.
- All files in a clone operation must be in the same vault, otherwise and error is returned.
- Multiple *source* and *target* pairs are permitted. In this case, the source file specifications are considered in order, and only the first match is used.
- Files specified using the `--exclude` option are excluded from the clone operation. Such exclusions apply across all of the *source* and *target* pairs.
- If you specify a null (empty) string as a *target* value, no clones are created for the corresponding *source*. Specifying a null *target* is an alternative method of excluding files from the clone operation.

  The *source* and null *target* is considered in order along with all other *source* and *target* pairs, and only the first match is used.

- All clones created in the same operation are point-in-time consistent.

Examples

Example 6-76 Clone a File

The following example shows cloning a file. The source file `@MYDATA/file-1` is and the clone is `@MYDATA/clone-file-1`.

```
@> clonefile @MYDATA/file-1 @MYDATA/clone-file-1
```

Example 6-77 Clone a Group of Files

The following example shows cloning a group of files using a wildcard. Assuming the existence of files named `@MYDATA/file1`, `@MYDATA/file2`, and so on, the example creates a clone of `@MYDATA/file1` named `@MYDATA/clone-dir/file1`, `@MYDATA/file2` named `@MYDATA/clone-dir/file2`, and so on.

```
@> clonefile @MYDATA/file* @MYDATA/clone-dir/file*
```

Example 6-78 Clone Multiple File Groups

The following example shows cloning multiple groups of files. The example clones `@MYDATA/a*` and `@MYDATA/b*`, with the resulting clones located under `@MYDATA/clone/`.

```
@> clonefile @MYDATA/a* @MYDATA/clone/a* @MYDATA/b* @MYDATA/clone/b*
```

Example 6-79 Ordering Significance

These examples show the significance of the order in which the *source* and *target* pairs are specified.

In the first command, the clones for files matching `@MYDATA/a*` are created in `@MYDATA/clone/a`, and the clones for the other files are created in `@MYDATA/clone/other`.

```
@> clonefile @MYDATA/a* @MYDATA/clone/a/a* @MYDATA/* @MYDATA/clone/other/*
```

In the following command, all of the clones are created in `@MYDATA/clone/other` because all of the files match `@MYDATA/*`. In this case, the second *source* and *target* pair is never used.

```
@> clonefile @MYDATA/* @MYDATA/clone/other/* @MYDATA/a* @MYDATA/clone/a/a*
```

Example 6-80 Clone a Group of Files with Exclusions

The following examples show cloning files in `@MYDATA`, except for those matching with `@MYDATA/a*` or `@MYDATA/b*`.

```
@> clonefile @MYDATA/* @MYDATA/clone/* --exclude @MYDATA/a* --exclude
@MYDATA/b*
```

```
@> clonefile @MYDATA/a* "" @MYDATA/b* "" @MYDATA/* @MYDATA/clone/*
```

Example 6-81 Invalid Commands

The following examples show invalid commands. The first and second commands are invalid because multiple vaults are referenced. The final command is invalid because the clone destination does not contain a wildcard to match the source specification.

```
@> clonefile @MYDATA/a* @MYDATABACKUP/a*


@> clonefile @MYDATA/a* @MYDATA/clone/a* @VAULT2/a* @VAULT2/clone/a*


@> clonefile @MYDATA/withwc* @MYDATA/clone/withoutwc
```

6.2.8.4 extentmap

Display storage extent information.

Purpose

The `extentmap` command enables Exascale cluster administrators to display storage extent information for Exascale file and vaults.

Syntax

```
extentmap [ --name filename --file-offset offset [ --mirror mirror ]] [ --
extId extent --target target ]
```

Command Options

The options for the `extentmap` command are:

- `--name`: A file or vault name for which you want to display extent information.

  Vault names are preceded with the `@` symbol.

- `--file-offset`: Specifies the offset location for which to display information.

  The value is expressed in bytes and can be a single number or a range. Use this option when you specify a file name for translation.

- `--mirror`: Optionally specifies the mirror number for which to display information.

  For normal (2-way) redundancy, the mirror number can be `0` or `1`. For high (3-way) redundancy, the mirror number can be `0`, `1`, or `2`.

  You can use this option when you specify a file name for translation.

- `--extId`: An extent identifier for which you want to display information.

  You can use this option to translate an extent identifier to a vault or file name.

- `--target`: Specifies the translation target; either `vaults` or `files`.

  You can use this option when you specify an extent identifier for translation.

Usage Notes

This command requires the cluster administrator privilege.

Examples

Example 6-82 Display Extent Information for a Vault

The following example shows how to display extent information for the specified vault.

```
@> extentmap --name @MYDATA
Vault Extent ID: c000_0000_003a:0000_0000_0000:00000000:00000000
```

Example 6-83 Display Extent Information for a Specific File Offset

The following example shows how to display extent information for a specified byte offset in a specified file.

```
@> extentmap --name @MYDATA/file1 --file-offset 123
extId: c000_0000_003a:8000_0000_000a:00000000:00000000
rpmId: 3222802720
extentEndOffset: 8388608
mirrorHealth: OK
translationPath:
o/192.0.2.1;192.0.2.1;192.0.2.1;192.0.2.1:41681/cell1_CD_exdisk6_EDSCELL1
mirrorNumber: 0

extId: c000_0000_003a:8000_0000_000a:00000000:00000000
rpmId: 3222802720
extentEndOffset: 8388608
mirrorHealth: OK
translationPath:
o/192.0.2.3;192.0.2.3;192.0.2.3;192.0.2.3:41681/cell3_CD_exdisk1_EDSCELL3
mirrorNumber: 1

extId: c000_0000_003a:8000_0000_000a:00000000:00000000
rpmId: 3222802720
extentEndOffset: 8388608
mirrorHealth: OK
translationPath:
o/192.0.2.6;192.0.2.6;192.0.2.6;192.0.2.6:41681/cell6_CD_exdisk9_EDSCELL6
mirrorNumber: 2
```

Example 6-84 Display Extent Information for a Specific Mirror

The following example shows how to display extent information for a specific extent mirror, using a specified byte offset in a specified file.

```
@> extentmap --name @MYDATA/file1 --file-offset 123 --mirror 1
extId: c000_0000_003a:8000_0000_000a:00000000:00000000
rpmId: 3222802720
extentEndOffset: 8388608
mirrorHealth: OK
translationPath:
o/192.0.2.3;192.0.2.3;192.0.2.3;192.0.2.3:41681/cell3_CD_exdisk1_EDSCELL3
mirrorNumber: 1
```

Example 6-85 Display Vault Name for a Specific Extent

The following example shows how to display the vault name for a specific extent.

```
@> extentmap --extId c000_0000_003a:8000_0000_000a:00000000:00000000 --target
vaults
vaultName: MYDATA
SysEdsURL: 192.0.2.3;192.0.2.3;192.0.2.3;192.0.2.3:41681
```

Example 6-86 Display File Name for a Specific Extent

The following example shows how to display the file name for a specific extent.

```
@> extentmap --extId c000_0000_003a:8000_0000_000a:00000000:00000000 --target
files
vaultName: MYDATA
fileName: file1
UserEdsURL: 192.0.2.3;192.0.2.3;192.0.2.3;192.0.2.3:41681
```

6.2.8.5 getfile

Copy a file from Exascale to the local file system.

Purpose

The `getfile` command copies a file from an Exascale vault to the local file system.

Syntax

```
getfile exa-file local-file
```

Command Options

The options for the `getfile` command are:

- *exa-file*: Specifies a file in an Exascale vault.
- *local-file*: Specifies a regular file location in the local file system.

Examples

Example 6-87 Get a File

The following example copies the Exascale file at `@MYDATA/file-1` to `/tmp/file-1.copy` on the local file system.

```
@> getfile @MYDATA/file-1 /tmp/file-1.copy
```

6.2.8.6 ls

List files and vaults.

Purpose

The `ls` command displays information about Exascale files and vaults.

Syntax

```
ls [ filename [ filename ] ... ] [ -l ] [ --detail ] [ --attributes
attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ] [ -t ]
```

Command Options

The options for the `ls` command are:

- *filename*: A file or vault name that you want to display information for. Vault names are preceded with the `@` symbol. The asterisk (`*`) can be used for wildcard searches. If not specified, then information is displayed about all files or vaults in the current level of the ESCLI file hierarchy.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.
- `-t`: Sorts files or vaults by ascending creation time.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  Attributes can be file attributes, or vault attributes with the `vault.` prefix.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  File sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

  For example, the following filter only includes files created after noon today and with a size greater than 10 megabytes:

  ```
  --filter creationTime>12:00:00,size>10M
  ```

- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-88 Display All Vaults

If you use the `ls` command at the root level, you get a listing of all vaults.

```
@> ls
MYDATA
VAULT2
```

Example 6-89 Display Files in a Vault

You can specify the name of a vault followed by the slash character (`/`) to list all the files within the vault.

```
@> ls @MYDATA/
x
y
z1
z2
z3
```

Example 6-90 Display File Information in a Long Format

If you use the `-l` option, the output includes additional information about each file or vault.

```
@MYDATA/> ls -l
Total 5
  10.0M   05 Jan 12:59 x
  19.5k   05 Jan 13:08 y
   5.0k   05 Jan 13:09 z1
  10.0M   05 Jan 13:23 z2
  20.0G   05 Jan 13:14 z3
```

Example 6-91 Sorting the Output of the ls Command

The following example shows the files in the vault `MYDATA`, with the most recent appearing first.

```
@MYDATA/> ls --sort -createTime
z2
z3
z1
y
x
```

Example 6-92 Listing Files Using a Wildcard Search

You can use the `*` symbol as a wildcard when specifying the files to list.

```
@MYDATA> ls z*
z1
z2
z3
```

Example 6-93 Display Detailed File Information

If you use the `--detail` option, the output includes detailed information about the files and vaults. This example shows the detailed output for the file `x` in the vault `MYDATA`.

```
@> ls @MYDATA/x --detail
name                    x
size                    0
createTime              2019-01-05 12:59:57 GMT
fileType                8
vault.name              MYDATA
vault.createTime        2018-11-08 22:41:07 GMT
```

Example 6-94 Display Specific Attributes for Files or Vaults

The following example shows the size, name, and vault name attributes for the files in the `MYDATA` vault.

```
@MYDATA/> ls -l --attributes size,name,vault.name
Total 5
      0 x MYDATA
  19.5k y MYDATA
   5.0k z1 MYDATA
  10.0M z2 MYDATA
  20.0G z3 MYDATA
```

6.2.8.7 lssnapshots

List snapshots associated with a file.

Purpose

The `lssnapshots` command displays information about snapshots that are associated with the specified file.

Syntax

```
lssnapshots filename { --all | --tree | [ -l ] [ --detail ] [ --attributes
attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --count value ] }
```

Command Options

The options for the `lssnapshots` command are:

- *filename*: Identifies the file that is the subject of the command. The specified file can be a snapshot, clone, or regular file.
- `--all`: Lists information about all files, clones, and snapshots belonging to the snapshot tree that contains the specified file. The output is presented in a series of lists. The first list contains the files and clones in the snapshot tree. The remaining output lists snapshots that are grouped by their source file.
- `--tree`: Lists information about all files, clones, and snapshots belonging to the snapshot tree that contains the specified file. The output is arranged in a graphical tree.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the output.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- If you do not specify the command options `--all` or `--tree`, then the command displays information about immediate snapshots of the specified source file.
- You cannot use any other command options in conjunction with `--all` or `--tree`.
- Filters are specified as: `<attribute><operator><value>`.

  Attributes can be file attributes, or vault attributes with the `vault.` prefix. Multiple filter conditions are delimited by commas.

  The allowed operators are =, >=, <=, >, and <.

  Dates can be specified using the following formats:

  – `yyyy-MM-dd''T''HH:mm:ss`
  – `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  – `HH:mm:ss` (Date is assumed to be today)

  Any of these formats can be followed by a timezone specification.

  File sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

  For example, to filter the results to show only snapshots created after noon today, with a size greater than 10 megabytes, you could use the following:

  ```
  --filter creationTime>12:00:00,size>10M
  ```

- Output is ordered according to the snapshot creation time, starting with the most recent.

Examples

Example 6-95 List Basic Snapshot Information

The following example shows how to list information about all of the snapshots that are based on the specified file.

```
@> lssnapshots @MYDATA/x
```

Example 6-96 List Detailed Snapshot Information

The following example shows how to list detailed information about all of the snapshots that are based on the specified file.

```
@> lssnapshots @MYDATA/x --detail
```

Example 6-97 List Snapshot Information with Specific Attributes

The following example shows how to list specific attributes about all of the snapshots that are based on the specified file.

```
@> lssnapshots @MYDATA/x --detail --attributes
name,vault.name,createTime,vault.createTime
```

Example 6-98 Filter the List of Snapshots

The following example shows how to specify a filter that constrains the command output. In the example, the output is limited to the snapshots that were created after the specified date.

```
@> lssnapshots @MYDATA/x --filter createTime>2020-01-01
```

Example 6-99 List the Latest Snapshots

The following example shows how to specify a count that constrains the command output. In the example, the output is limited to the 5 most recently created snapshots.

```
@> lssnapshots @MYDATA/x --count 5
```

Example 6-100 Display a Snapshot Tree

The command output in this example assumes a group of related files created using the following command sequence:

```
@TEST/> putfile somelocalfile file1
Putting file somelocalfile to @TEST/file1
Success.

@TEST/> snapshotfile file1 snap1_of_file1
Success.

@TEST/> clonefile file1 clone1
Success.

@TEST/> snapshotfile file1 snap2_of_file1
Success.

@TEST/> snapshotfile clone1 snap1_of_clone1
Success.
```

The following example shows how to display all files in a snapshot tree using the `--tree` command option.

```
@> lssnapshots @TEST/clone1 --tree

+---- [1] @TEST/snap1_of_file1
+---- [2] @TEST/?8000_0000_000a:00000001
|
|`---+---- [4] @TEST/snap1_of_clone1
|    +---- [6] @TEST/clone1
|
+---- [3] @TEST/snap2_of_file1
+---- [5] @TEST/file1
```

Note that the command output includes an internal snapshot (`@TEST/?8000_0000_000a:00000001`), which was implicitly created as part of the cloning operation to create `@TEST/clone1`.

Example 6-101 List all Files in a Snapshot Tree

The following example shows how to display all files in a snapshot tree using the `--all` command option. This example assumes the same scenario as the previous example.

```
@> lssnapshots @TEST/clone1 --all

---FILE/CLONES:---

2022-05-02 06:14:41 UTC @TEST/file1
2022-05-02 06:15:15 UTC @TEST/clone1

---SNAPSHOTS:---

@TEST/file1
  2022-05-02 06:15:06 UTC @TEST/snap1_of_file1
  2022-05-02 06:15:15 UTC @TEST/?8000_0000_000a:00000001
  2022-05-02 06:15:24 UTC @TEST/snap2_of_file1

@TEST/clone1
  2022-05-02 06:15:45 UTC @TEST/snap1_of_clone1
```

6.2.8.8 mkfile

Create a file.

Purpose

The `mkfile` command creates a file within a vault.

Syntax

```
mkfile filename [ --template template ] [ --attributes
attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `mkfile` command are:

- *filename*: Specifies the name of the file you are creating. The file name cannot contain wildcard characters. Only one file name can be specified.
- `--template`: Creates the file using the specified template.
- `--attributes`: Optionally specifies attributes to change.

  Use the `describe mkfile` command to view details about all the file attributes you can set with `mkfile`.

Usage Notes

Note the following information when using this command:

- You can set the file size by specifying the `size` attribute.

  File sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

  Regardless of the file size setting, storage space is physically materialized only when the file contents is written. So, after the `mkfile` command, the file initially consumes no physical storage space. The file only consumes physical storage space when data is written to it.
- If you set any of the file storage attributes (`contentType`, `mediaType`, and `redundancy`), the specified value overrides the setting in the default template.

  For a description of the file storage attributes, see File Storage Attributes.
- You can set the Exascale file type by specifying the `fileType` attribute.

  Every Exascale file type is associated with an Exascale template. So, you can view a complete list of the Exascale file types by using the ESCLI lstemplate command. For example:

```
@> lstemplate --cluster


@> lstemplate --vault VAULT1
```

Examples

Example 6-102 Create a File in a Vault

The following example shows creating the file named `x` within the vault `MYDATA`.

```
@MYDATA/> mkfile x --attributes size=20k
File created.

@MYDATA/> ls -l
  20.0k   08 Jan 15:53 x
```

Related Topics

- File Storage Attributes

- lstemplate
  List file templates.

6.2.8.9 putfile

Copy a file from the local file system to Exascale.

Purpose

The `putfile` command copies a file from the local file system to an Exascale vault.

Syntax

```
putfile local-file exa-file
```

Command Options

The options for the `putfile` command are:

- *local-file*: Specifies a regular file in the local file system.
- *exa-file*: Specifies a file location in an Exascale vault.

Examples

Example 6-103 Put a File

The following example copies the local file at `/tmp/file-1` to `@MYDATA/file-1.copy` on Exascale storage.

```
@> putfile /tmp/file-1 @MYDATA/file-1.copy
```

6.2.8.10 rmfile

Delete a file.

Purpose

The `rmfile` command deletes a file in an Exascale vault.

Syntax

```
rmfile filename [ --force ] [ --nowait ] [ --with-snapshots ]
```

Command Options

The options for the `rmfile` command are:

- *filename*: Identifies the file to be deleted.
- `--force`: Deletes the specified file even if it is in use.
- `--nowait`: Instead of waiting for the command to finish, the command returns immediately while the operation completes in the background.
- `--with-snapshots`: Also deletes snapshots associated with the specified file.

Examples

Example 6-104 Delete a File

The following example deletes the file at `@MYDATA/file-1`.

```
@> rmfile @MYDATA/file-1
```

6.2.8.11 snapshotfile

Create snapshots.

Purpose

The `snapshotfile` command creates a snapshot of a file or group of files.

Syntax

```
snapshotfile source1 target1 [ sourceN targetN ]... [ --exclude exclude-
spec ]...
```

Command Options

The options for the `snapshotfile` command are:

- *source1-N*: Specifies the name of the file or files that are the snapshot source.
- *target1-N*: Specifies the name of the file or files that are the snapshot destination.
- `--exclude`: Specifies the name of the file or files that are excluded from the snapshot operation.

Usage Notes

Note the following information when using this command:

- You can use a wildcard (`*`) in the *source* to specify multiple source files, in which case the corresponding *target* must also contain a matching wildcard.
- All files in a snapshot operation must be in the same vault, otherwise and error is returned.
- Multiple *source* and *target* pairs are permitted. In this case, the source file specifications are considered in order, and only the first match is used.
- Files specified using the `--exclude` option are excluded from the snapshot operation. Such exclusions apply across all of the *source* and *target* pairs.
- If you specify a null (empty) string as a *target* value, no snapshots are created for the corresponding *source*. Specifying a null *target* is an alternative method of excluding files from the snapshot operation.

  The *source* and null *target* is considered in order along with all other *source* and *target* pairs, and only the first match is used.
- All snapshots created in the same operation are point-in-time consistent.

Examples

Example 6-105 Snapshot a File

This example shows creating a snapshot. The source file `@MYDATA/file-1` is and the snapshot is `@MYDATA/snap-file-1`.

```
@> snapshotfile @MYDATA/file-1 @MYDATA/snap-file-1
```

Example 6-106 Snapshot a Group of Files

This example shows creating snapshots for a group of files using a wildcard. Assuming the existence of files named `@MYDATA/file1`, `@MYDATA/file2`, and so on, the example creates a snapshot of `@MYDATA/file1` named `@MYDATA/snap-dir/file1`, `@MYDATA/file2` named `@MYDATA/snap-dir/file2`, and so on.

```
@> snapshotfile @MYDATA/file* @MYDATA/snap-dir/file*
```

Example 6-107 Snapshot Multiple File Groups

This example shows creating snapshots for multiple groups of files using wildcards. The source file groups are `@MYDATA/a*` and `@MYDATA/b*`, with the resulting snapshots located under `@MYDATA/snap/`.

```
@> snapshotfile @MYDATA/a* @MYDATA/snap/a* @MYDATA/b* @MYDATA/snap/b*
```

Example 6-108 Ordering Significance

These examples show the significance of the order in which the *source* and *target* pairs are specified.

In the first command, the snapshots for files matching `@MYDATA/a*` are created in `@MYDATA/snap/a`, and the snapshots for the other files are created in `@MYDATA/snap/other`.

```
@> snapshotfile @MYDATA/a* @MYDATA/snap/a/a* @MYDATA/* @MYDATA/snap/other/*
```

In the following command, all of the snapshots are created in `@MYDATA/snap/other` because all of the files match `@MYDATA/*`. In this case, the second *source* and *target* pair is never used.

```
@> snapshotfile @MYDATA/* @MYDATA/snap/other/* @MYDATA/a* @MYDATA/snap/a/a*
```

Example 6-109 Snapshot a Group of Files with Exclusions

The following examples show how to snapshot files in `@MYDATA`, except for those matching with `@MYDATA/a*` or `@MYDATA/b*`.

```
@> snapshotfile @MYDATA/* @MYDATA/snap/* --exclude @MYDATA/a* --exclude
@MYDATA/b*
```

```
@> snapshotfile @MYDATA/a* "" @MYDATA/b* "" @MYDATA/* @MYDATA/snap/*
```

Example 6-110 Invalid Commands

The following examples show invalid commands. The first and second commands are invalid because multiple vaults are referenced. The final command is invalid because the snapshot destination does not contain a wildcard to match the source specification.

```
@> snapshotfile @MYDATA/a* @MYDATABACKUP/a*


@> snapshotfile @MYDATA/a* @MYDATA/snap/a* @VAULT2/a* @VAULT2/snap/a*


@> snapshotfile @MYDATA/withwc* @MYDATA/snap/withoutwc
```

6.2.9 Template Management

This section contains references for the Exascale command line interface (ESCLI) commands that are associated with template management:

- chtemplate
  Change an attribute for an existing template.
- lstemplate
  List file templates.
- mktemplate
  Create a file template.
- rmtemplate
  Delete a file template.

6.2.9.1 chtemplate

Change an attribute for an existing template.

Purpose

The `chtemplate` command changes an attribute for an existing Exascale file template.

Syntax

```
chtemplate  [ --vault vault ] [ --cluster ] [ --file-type file-type ] [ --
name template-name ] [ --content-type content-type ] [ --media-type media-
type ] [ --redundancy redundancy ]
```

Command Options

Specify one or more of the following `chtemplate` command options to identify the templates that are the subject of the operation:

- `--vault`: Alters templates associated with the specified vault.
- `--cluster`: Alters templates associated with the cluster.
- `--file-type`: Alters templates associated with the specified file type.
- `--name`: Alter a user-defined template having the specified name.

Specify one or more of the following File Storage Attributes, which are associated with the template:

- `--media-type`: Specifies the physical media type that is used to store the file. Exascale uses this attribute to place the file in a storage pool that uses the specified media type. Possible values are:
  - `HC`: Identifies high capacity storage media, which uses hard disk drives (HDDs).
  - `EF`: Identifies extreme flash storage media, which uses flash devices.
- `--redundancy`: Specifies the number of data copies that are maintained. Possible values are:
  - `normal`: Maintains 2 mirrored copies of the file data.
  - `high`: Maintains 3 mirrored copies of the file data.
- `--content-type`: Specifies the type of content in the file. Exascale internally uses this attribute to place file extents on physically separate devices in a way that maximizes availability if failures occurs. Possible values are:
  - `DATA`
  - `RECO`

Examples

Example 6-111 Change a User-Defined Cluster-Wide Template

In this example, the content type is set to `DATA` for the user-defined template named `T1`. The template is associated with the cluster, as neither `--vault` or `--cluster` are specified, and the current working directory in the ESCLI session is the root directory.

```
@> chtemplate --name T1 --content-type DATA
```

Example 6-112 Change a User-Defined Vault-Specific Template

In this example, the media type is set to `HC` and the redundancy is set to `high` for the user-defined template named `T2`. The template is associated with the vault named `VAULT1`, as neither `--vault` or `--cluster` are specified, and the current working directory in the ESCLI session is inside `VAULT1`.

```
@VAULT1> chtemplate --name T2 --media-type HC --redundancy high
```

Example 6-113 Change a User-Defined Vault-Specific Template

In this example, the media type, content type, and redundancy are all set for the user-defined template named `T3` that is associated with the vault named `VAULT1`.

```
@> chtemplate --name T3 --vault VAULT1 --content-type DATA --media-type HC --
redundancy high
```

Example 6-114 Change a User-Defined Cluster-Wide Template

In this example, the media type, content type, and redundancy are all set for the cluster-wide user-defined template named `T4`. The `--cluster` option overrides the fact that the current working directory in the ESCLI session is inside `VAULT1`.

```
@VAULT1> chtemplate --name T4 --cluster --content-type DATA --media-type HC --
redundancy high
```

Example 6-115 Change a Template for a Specific Vault and File Type

This example changes the template associated with the `DATAFILE` file type and the vault named `VAULT1`.

```
@> chtemplate --file-type DATAFILE --vault VAULT1 --content-type DATA --media-
type HC --redundancy high
```

Example 6-116 Change a Cluster-Wide Template for a Specific File Type

This example changes the cluster-wide template associated with the `DATAFILE` file type.

```
@> chtemplate --file-type DATAFILE --cluster --content-type DATA --media-type
HC --redundancy high
```

Related Topics

- File Storage Attributes

6.2.9.2 lstemplate

List file templates.

Purpose

The `lstemplate` command displays information about Exascale file templates that are associated with the vault or the Exascale cluster.

Syntax

```
lstemplate [ template-name ] [ --vault vault [ --vault-level-only ]] [ --
cluster ] [ -l ] [ --detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
```

Command Options

The options for the `lstemplate` command are:

- *template-name*: Optionally limits the output to templates that match the specified template name.
- `--vault`: Lists templates associated with the specified vault.
- `--vault-level-only`: List only the vault-specific templates associated with the specified vault.
- `--cluster`: Lists templates associated with the cluster.

- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.

Usage Notes

Note the following information when using this command:

- If `--vault` is specified without `--vault-level-only`, then the output contains all templates affecting the specified vault, including the vault-specific templates and any cluster-level templates not overridden by vault-specific templates.
- If neither `--vault` nor `--cluster` is specified, and the user entered a vault in the ESCLI session, then the output contains all templates affecting the specified vault, including the vault-specific templates and any cluster-level templates not overridden by vault-specific templates.
- If neither `--vault` nor `--cluster` is specified, and the user hasn't entered a vault in the ESCLI session (ESCLI prompt shows `@>`), then templates associated with the cluster are listed.
- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.
- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-117 List Cluster-Wide Templates

These examples list all templates associated with the cluster.

```
@> lstemplate
```

```
@VAULT1> lstemplate --cluster
```

Example 6-118 List Effective Vault-Specific Templates

These examples list the effective templates associated with the vault `VAULT1`. The output includes templates associated with the vault, and templates associated with the cluster that are not overridden by vault-specific templates.

```
@VAULT1> lstemplate


@> lstemplate --vault VAULT1
```

Example 6-119 List Only Templates Affecting a Vault

This examples lists only the vault-specific templates affecting files in the vault `VAULT1`.

```
@> lstemplate --vault VAULT1 --vault-level-only
```

Example 6-120 List a Specific Cluster-Wide Template

This example displays detailed information about the cluster-wide template named `DATAFILE`.

```
@> lstemplate --detail DATAFILE
```

Example 6-121 List Specific Cluster-Wide Templates

This example displays detailed information about the cluster-wide templates with names ending with `FILE`, whose content type is `DATA`, and sort the list by names (ascending) and media types (descending).

```
@> lstemplate --detail *FILE --sort name,-mediaType --filter contentType=DATA
```

6.2.9.3 mktemplate

Create a file template.

Purpose

The `mktemplate` command creates an Exascale file template.

Syntax

```
mktemplate  [ --vault vault | --cluster ] { --file-type file-type | --name
template-name } --content-type content-type --media-type media-type --
redundancy redundancy
```

Command Options

Specify the following `mktemplate` command options to identify the template and its scope:

- `--vault`: Create a template associated with the specified vault.
- `--cluster`: Create a template associated with the cluster.
- `--file-type`: Create a template associated with the specified file type. If you specify this option, the template name is the same as the file type.

- `--name`: Create a user-defined template having the specified name.

Specify the following File Storage Attributes, which are associated with the template:

- `--media-type`: Specifies the physical media type that is used to store the file. Exascale uses this attribute to place the file in a storage pool that uses the specified media type. Possible values are:
  - `HC`: Identifies high capacity storage media, which uses hard disk drives (HDDs).
  - `EF`: Identifies extreme flash storage media, which uses flash devices.
- `--redundancy`: Specifies the number of data copies that are maintained. Possible values are:
  - `normal`: Maintains 2 mirrored copies of the file data.
  - `high`: Maintains 3 mirrored copies of the file data.
- `--content-type`: Specifies the type of content in the file. Exascale internally uses this attribute to place file extents on physically separate devices in a way that maximizes availability if failures occurs. Possible values are:
  - `DATA`
  - `RECO`

Usage Notes

Note the following information when using this command:

- You can define the scope of the template by specifying the `--vault` option or the `--cluster` option. You cannot specify both options simultaneously.

  If neither option is specified, then the scope of the template is inferred from the current working directory in the ESCLI session. If the current working directory in the ESCLI session is the root directory, then the template is associated with the cluster. Otherwise, the template is associated with the vault that is referenced in the current working directory.
- You must define the template as a user template by specifying the `--name` option, or you must associate the template with a file type by using the `--file-type` option. You cannot specify both options simultaneously.
- You can view a list of Exascale file types by using the ESCLI lstemplate command. For example:

```
@> lstemplate --cluster


@> lstemplate --vault VAULT1 --all
```

Examples

Example 6-122 Create a User-Defined Cluster-Wide Template

This example shows creating a user-defined template named `T1`. The template is associated with the cluster, as neither `--vault` or `--cluster` are specified, and the current working directory in the ESCLI session is the root directory.

```
@> mktemplate --name T1 --content-type DATA --media-type HC --redundancy high
```

Example 6-123 Create a User-Defined Vault-Specific Template

This example shows creating a user-defined template named `T2`. The template is associated with the vault named `VAULT1`, as neither `--vault` or `--cluster` are specified, and the current working directory in the ESCLI session is inside `VAULT1`.

```
@VAULT1> mktemplate --name T2 --content-type DATA --media-type HC --
redundancy high
```

Example 6-124 Create a User-Defined Vault-Specific Template

This example shows creating a user-defined template named `T3` that is associated with the vault named `VAULT1`.

```
@> mktemplate --name T3 --vault VAULT1 --content-type DATA --media-type HC --
redundancy high
```

Example 6-125 Create a User-Defined Cluster-Wide Template

This example shows creating a user-defined cluster-wide template named `T4`. The `--cluster` option overrides the fact that the current working directory in the ESCLI session is inside `VAULT1`.

```
@VAULT1> mktemplate --name T4 --cluster --content-type DATA --media-type HC --
redundancy high
```

Example 6-126 Create a Template for a Specific Vault and File Type

This example shows creating a template associated with the `DATAFILE` file type and the vault named `VAULT1`.

```
@> mktemplate --file-type DATAFILE --vault VAULT1 --content-type DATA --media-
type HC --redundancy high
```

Example 6-127 Create a Cluster-Wide Template for a Specific File Type

This example shows creating a cluster-wide template associated with the `DATAFILE` file type.

```
@> mktemplate --file-type DATAFILE --cluster --content-type DATA --media-type
HC --redundancy high
```

Related Topics

- File Storage Attributes
- lstemplate
  List file templates.

6.2.9.4 rmtemplate

Delete a file template.

Purpose

The `rmtemplate` command deletes an Exascale file template that is associated with the vault or the cluster.

Syntax

```
rmtemplate [ --vault vault ] [ --cluster ] [ --file-type file-type ] [ --name
template-name ]
```

Command Options

Specify one or more of the following `rmtemplate` command options to identify the template to delete:

- `--vault`: Specifies a template associated with the specified vault.
- `--cluster`: Specifies a template associated with the cluster.
- `--file-type`: Specifies a template associated with the specified file type.
- `--name`: Specifies a user-defined template having the specified name.

Usage Notes

Note the following information when using this command:

- File type templates associated with the cluster cannot be deleted.

Examples

Example 6-128 Delete a User-Defined Cluster-Wide Template

This example shows deleting a user-defined template named `T1`. The template is associated with the cluster, as neither `--vault` or `--cluster` are specified, and the current working directory in the ESCLI session is the root directory.

```
@> rmtemplate --name T1
```

Example 6-129 Delete a User-Defined Vault-Specific Template

This example shows deleting a user-defined template named `T2`. The template is associated with the vault named `VAULT1`, as neither `--vault` or `--cluster` are specified, and the current working directory in the ESCLI session is inside `VAULT1`.

```
@VAULT1> rmtemplate --name T2
```

Example 6-130 Delete a User-Defined Vault-Specific Template

This example shows deleting a user-defined template named `T3` that is associated with the vault named `VAULT1`.

```
@> rmtemplate --name T3 --vault VAULT1
```

Example 6-131 Delete a User-Defined Cluster-Wide Template

This example shows deleting a user-defined cluster-wide template named `T4`. The `--cluster` option overrides the fact that the current working directory in the ESCLI session is inside `VAULT1`.

```
@VAULT1> mktemplate --name T4 --cluster
```

Example 6-132 Delete a Template for a Specific Vault and File Type

This example shows deleting a template associated with the `DATAFILE` file type and the vault named `VAULT1`.

```
@> rmtemplate --file-type DATAFILE --vault VAULT1
```

6.2.10 Resource Profile Management

This section contains references for the Exascale command line interface (ESCLI) commands that are associated with resource profile management:

- chresourceprofile
  Change a resource profile associated with a vault.
- lsresourceprofile
  List resource profiles.
- mkresourceprofile
  Create a resource profile.
- rmresourceprofile
  Delete a resource profile.

6.2.10.1 chresourceprofile

Change a resource profile associated with a vault.

Purpose

The `chresourceprofile` command changes the attribute settings for an existing resource profile that is associated with an Exascale vault.

Syntax

```
chresourceprofile [vault-name/]resource-profile-name --attributes
attribute=value[,attribute=value] ...
```

Command Options

The options for the `chresourceprofile` command are:

- *resource-profile-name*: Identifies the resource profile being changed.
- *vault-name*: Identifies the vault that the resource profile is associated with.

  If not specified, then the current working directory is used to identify the vault.
- `--attributes`: Specifies attributes to change. The following list outlines all of the available attributes:
  - `iopsShareEF`: Specifies the relative share of I/O bandwidth (IOPS) from extreme flash (EF) storage media that is available to each client associated with the resource profile. Each client's share is relative to all other client shares. A higher share value implies higher priority. The range of valid values is `1-100`, and the default is `1`.
  - `iopsLimitEF`: Specifies the upper limit of the I/O bandwidth (IOPS) from extreme flash (EF) storage media that is available to each client associated with the resource profile. The value represents a fraction out of 10000. The range of valid values is `1-10000`. The default value is `10000` (effectively unlimited).
  - `iopsShareHC`: Specifies the relative share of I/O bandwidth (IOPS) from high capacity (HC) storage media that is available to each client associated with the resource profile. Each client's share is relative to all other client shares. A higher share value implies higher priority. The range of valid values is `1-100`, and the default is `1`.
  - `iopsLimitHC`: Specifies the upper limit of the I/O bandwidth (IOPS) from high capacity (HC) storage media that is available to each client associated with the resource profile. The value represents a fraction out of 10000. The range of valid values is `1-10000`. The default value is `10000` (effectively unlimited).
  - `enableFlashCache`: Enables or disables use of flash cache for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).
  - `flashCacheMin`: Specifies the guaranteed minimum fraction of flash cache available to each client associated with the resource profile. The range of valid values is `0-10000`. The default value is `0` (no set minimum). If the sum of all values across all resource profiles exceeds 10000, then all values are proportionally scaled down. This option is valid only when `enableFlashCache=true`.
  - `flashCacheMax`: Specifies the maximum fraction of flash cache available to each client associated with the resource profile. This option is valid only when `enableFlashCache=true`. The range of valid values is `0-10000`. The default value is `10000`. This option is valid only when `enableFlashCache=true`.
  - `enableXrmemCache`: Enables or disables use of Exadata RDMA Memory Cache (XRMEM cache) for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).
  - `xrmemCacheMin`: Specifies the guaranteed minimum fraction of XRMEM cache available to each client associated with the resource profile. The range of valid values is `0-10000`. The default value is `0` (no set minimum). If the sum of all values across all resource profiles exceeds 10000, then all values are proportionally scaled down. This option is valid only when `enableXrmemCache=true`.
  - `xrmemCacheMax`: Specifies the maximum fraction of XRMEM cache available to each client associated with the resource profile. This option is valid only when `enableXrmemCache=true`. The range of valid values is `0-10000`. The default value is `10000`. This option is valid only when `enableXrmemCache=true`.
  - `enableFlashLog`: Enables or disables use of flash log for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).

Usage Notes

In addition to regular user-defined resource profiles, you can also modify the system-reserved resource profile named `$UNASSIGNED`. All Exascale clients not explicitly associated with a resource profile are automatically governed by the `$UNASSIGNED` profile. The `$UNASSIGNED` resource profile contains only two modifiable attributes:

- `flashCacheMax`: Specifies the maximum fraction of flash cache shared by clients associated with the `$UNASSIGNED` profile. The range of valid values is `0-10000`. The default value is `10000`.
- `xrmemCacheMax`: Specifies the maximum fraction of XRMEM cache shared by clients associated with the `$UNASSIGNED` profile. The range of valid values is `0-10000`. The default value is `10000`.

All Exascale clients governed by the `$UNASSIGNED` profile share the corresponding cache resources. The behavior differs from regular resource profiles, where each application of the resource profile defines the resource allocation for one associated client.

You must create the `$UNASSIGNED` resource profile before you can modify it. If the `$UNASSIGNED` resource profile does not exist, all unassigned Exascale clients share any unassigned flash cache space and XRMEM cache space. If there is no unassigned space to share, the system automatically reserves 5% of the cache space for unassigned Exascale clients.

Examples

Example 6-133 Change a Resource Profile

In this example, the `iopsShareHC` attribute is set to `11` for the `SILVER` resource profile in conjunction with the `VAULT1` vault.

```
@> chresourceprofile VAULT1/SILVER --attributes iopsShareHC=11
```

Example 6-134 Change a Resource Profile

In this example, the `enableFlashLog` attribute is set to `false` for the `SILVER` resource profile in conjunction with the `VAULT1` vault. In this example, the vault association is not specified in the command but is derived from the current working directory in the ESCLI session.

```
@VAULT1> chresourceprofile SILVER --attributes enableFlashLog=false
```

6.2.10.2 lsresourceprofile

List resource profiles.

Purpose

The `lsresourceprofile` command displays information about Exascale resource profiles that are associated with a vault.

Syntax

```
lsresourceprofile [ [vault-name/]resource-profile-name [, [vault-
name/]resource-profile-name ] ... ]
     [ -l ] [ --detail ] [ --attributes attribute[,attribute] ... ]
```

```
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
```

Command Options

The options for the `lsresourceprofile` command are:

- *resource-profile-name*: Identifies a specific resource profile name.
- *vault-name*: Identifies a specific vault.

  If not specified, then the current working directory is used to specify the vault.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.
- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with -.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-135 List Resource Profiles

This example lists the resource profiles associated with the vault named `VAULT1`.

```
@> lsresourceprofile --filter vault=VAULT1
```

Example 6-136 List Resource Profiles

This example lists the resource profiles associated with the vault named `VAULT1`. In this example, the vault association is not specified in the command but is derived from the current working directory in the ESCLI session.

```
@VAULT1> lsresourceprofile
```

Example 6-137 List a Specific Resource Profile

This example shows two ways to display detailed information about the resource profile named `SILVER` that is associated with the vault named `VAULT1`.

```
@> lsresourceprofile VAULT1/SILVER --detail
```

```
@> lsresourceprofile SILVER --filter vault=VAULT1 --detail
```

6.2.10.3 mkresourceprofile

Create a resource profile.

Purpose

The `mkresourceprofile` command creates a resource profile.

Syntax

```
mkresourceprofile [vault-name/]resource-profile-name [ --attributes
attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `mkresourceprofile` command are:

- *resource-profile-name*: Specifies the name for the resource profile being created.
- *vault-name*: Specifies the vault that the resource profile is associated with.

  If not specified, then the current working directory is used to specify the vault.
- `--attributes`: Optionally specifies values for attributes of the resource profile. The following list outlines all of the available attributes:
  - `iopsShareEF`: Specifies the relative share of I/O bandwidth (IOPS) from extreme flash (EF) storage media that is available to each client associated with the resource profile. Each client's share is relative to all other client shares. A higher share value implies higher priority. The range of valid values is `1-100`, and the default is `1`.
  - `iopsLimitEF`: Specifies the upper limit of the I/O bandwidth (IOPS) from extreme flash (EF) storage media that is available to each client associated with the resource profile. The value represents a fraction out of 10000. The range of valid values is `1-10000`. The default value is `10000` (effectively unlimited).
  - `iopsShareHC`: Specifies the relative share of I/O bandwidth (IOPS) from high capacity (HC) storage media that is available to each client associated with the resource profile. Each client's share is relative to all other client shares. A higher share value implies higher priority. The range of valid values is `1-100`, and the default is `1`.
  - `iopsLimitHC`: Specifies the upper limit of the I/O bandwidth (IOPS) from high capacity (HC) storage media that is available to each client associated with the resource profile. The value represents a fraction out of 10000. The range of valid values is `1-10000`. The default value is `10000` (effectively unlimited).
  - `enableFlashCache`: Enables or disables use of flash cache for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).

- `flashCacheMin`: Specifies the guaranteed minimum fraction of flash cache available to each client associated with the resource profile. The range of valid values is `0-10000`. The default value is `0` (no set minimum). If the sum of all values across all resource profiles exceeds 10000, then all values are proportionally scaled down. This option is valid only when `enableFlashCache=true`.
- `flashCacheMax`: Specifies the maximum fraction of flash cache available to each client associated with the resource profile. This option is valid only when `enableFlashCache=true`. The range of valid values is `0-10000`. The default value is `10000`. This option is valid only when `enableFlashCache=true`.
- `enableXrmemCache`: Enables or disables use of Exadata RDMA Memory Cache (XRMEM cache) for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).
- `xrmemCacheMin`: Specifies the guaranteed minimum fraction of XRMEM cache available to each client associated with the resource profile. The range of valid values is `0-10000`. The default value is `0` (no set minimum). If the sum of all values across all resource profiles exceeds 10000, then all values are proportionally scaled down. This option is valid only when `enableXrmemCache=true`.
- `xrmemCacheMax`: Specifies the maximum fraction of XRMEM cache available to each client associated with the resource profile. This option is valid only when `enableXrmemCache=true`. The range of valid values is `0-10000`. The default value is `10000`. This option is valid only when `enableXrmemCache=true`.
- `enableFlashLog`: Enables or disables use of flash log for clients associated with the resource profile. The value is Boolean, and the default is `true` (enabled).

Usage Notes

In addition to regular user-defined resource profiles, you can also create a system-reserved resource profile named `$UNASSIGNED`. All Exascale clients not explicitly associated with a resource profile are automatically governed by the `$UNASSIGNED` profile. The `$UNASSIGNED` resource profile contains only two modifiable attributes:

- `flashCacheMax`: Specifies the maximum fraction of flash cache shared by clients associated with the `$UNASSIGNED` profile. The range of valid values is `0-10000`. The default value is `10000`.
- `xrmemCacheMax`: Specifies the maximum fraction of XRMEM cache shared by clients associated with the `$UNASSIGNED` profile. The range of valid values is `0-10000`. The default value is `10000`.

All Exascale clients governed by the `$UNASSIGNED` profile share the corresponding cache resources. The behavior differs from regular resource profiles, where each application of the resource profile defines the resource allocation for one associated client.

If the `$UNASSIGNED` resource profile does not exist, all unassigned Exascale clients share any unassigned flash cache space and XRMEM cache space. If there is no unassigned space to share, the system automatically reserves 5% of the cache space for unassigned Exascale clients.

Examples

Example 6-138 Create a Resource Profile with Default Attribute Values

This example shows two ways to create a resource profile with default attribute values, which is named `SILVER` and is associated with the vault named `VAULT1`.

```
@> mkresourceprofile VAULT1/SILVER


@> mkresourceprofile SILVER --attributes vault=VAULT1
```

Example 6-139 Create a Resource Profile with Default Attribute Values

This example shows how to create a resource profile with default attribute values, which is named `BRONZE` and is associated with the vault named `VAULT1`. In this example, the vault association is not specified in the command but is derived from the current working directory in the ESCLI session.

```
@VAULT1> mkresourceprofile BRONZE
```

Example 6-140 Create a Resource Profile with Specific Attribute Values

This example shows how to create a resource profile that contains specific attribute settings.

```
@> mkresourceprofile VAULT1/GOLD --attributes
iopsShareHC=11,enableFlashLog=false
```

6.2.10.4 rmresourceprofile

Delete a resource profile.

Purpose

The `rmresourceprofile` command deletes a resource profile that is associated with an Exascale vault.

Syntax

```
rmresourceprofile [vault-name/]resource-profile-name
```

Command Options

The options for the `rmresourceprofile` command are:

- *resource-profile-name*: Identifies the name for the resource profile to delete.
- *vault-name*: Identifies the vault that the resource profile is associated with.

  If not specified, then the current working directory is used to identify the vault.

Examples

Example 6-141 Delete a Resource Profile

This example shows how to delete a resource profile named `SILVER` that is associated with the vault named `VAULT1`.

```
@> rmresourceprofile VAULT1/SILVER
```

Example 6-142 Delete a Resource Profile

This example shows how to delete a resource profile named `BRONZE` that is associated with the vault named `VAULT1`. In this example, the vault association is not specified in the command but is derived from the current working directory in the ESCLI session.

```
@VAULT1> rmresourceprofile BRONZE
```

6.2.11 Extended Attribute Management

This section contains references for the Exascale command line interface (ESCLI) commands that are associated with extended attribute management:

- chxattr
  Set an extended attribute for a file or vault.
- lsxattr
  List an extended attribute for a file or vault.
- rmxattr
  Delete an extended attribute for a file or vault.

6.2.11.1 chxattr

Set an extended attribute for a file or vault.

Purpose

The `chxattr` command allows you to set an extended attribute for an Exascale file or vault.

Syntax

```
chxattr { file-name | vault-name } --name attribute-name { --value attribute-
value | --bin-value-file file-name }
```

Command Options

The options for the `chxattr` command are:

- `{ file-name | vault-name }`: Specifies the name of the file or vault that is the subject of the operation.
- `--name`: Specifies the name of the extended attribute.
- `--value`: Specifies the value of the extended attribute.
- `--bin-value-file`: Specifies the regular file from which to read the value of the extended attribute. The value is assumed to be binary.

Examples

Example 6-143 Set an Extended Attribute for a Vault

You can add an extended attribute to a vault, such as `@MYDATA`. The name of the attribute is `custom_attr`, and its value is `myvalue`.

```
@> chxattr @MYDATA --name custom_attr --value myvalue
```

Example 6-144 Set an Extended Attribute for a File

You can add an extended attribute to a file, such as `@MYDATA/file1`. The name of the attribute is `custom_attr`, and its value is `myvalue`.

```
@> chxattr @MYDATA/file1 --name custom_attr --value myvalue
```

Example 6-145 Set a Binary Extended Attribute for a File

You can add a binary extended attribute to a file, such as `@MYDATA/file1`. The name of the attribute is `binary_attr`, and its value is the contents of the file at `/home/user/bin-val-in.dat`.

```
@> chxattr @MYDATA/file1 --name binary_attr --bin-value-file /home/user/bin-val-in.dat
```

6.2.11.2 lsxattr

List an extended attribute for a file or vault.

Purpose

The `lsxattr` command displays information about non-standard attributes created for Exascale files and vaults.

Syntax

```
lsxattr { file-name | vault-name } --name attribute_name [ --bin-value-file file-name ]
```

Command Options

The options for the `lsxattr` command are:

- `{ file-name | vault-name }`: Specifies the name of the file or vault for which you want to list the specified extended attribute value.
- `--name`: Specifies the name of the extended attribute.
- `--bin-value-file`: Specifies the file where you want to write the value of the extended attribute.

Usage Notes

If you do not specify the `--bin-value-file` option, binary values are output in Base64 format.

Examples

Example 6-146 List an Extended Attribute

You can use the following commands to list the `custom_attr` extended attribute.

```
@> lsxattr @MYDATA --name custom_attr


@> lsxattr @MYDATA/file1 --name custom_attr
```

Example 6-147 List a Binary Extended Attribute

You can use the following commands to list the `binary_attr` extended attribute and write the binary value to a regular file at `/home/user/bin-val-out.dat`.

```
@> lsxattr @MYDATA/file1 --name binary_attr --bin-value-file /home/user/bin-
val-out.dat
```

6.2.11.3 rmxattr

Delete an extended attribute for a file or vault.

Purpose

The `rmxattr` command deletes non-standard attributes created for Exascale files and vaults.

Syntax

```
rmxattr { file-name | vault-name } --name attribute_name
```

Command Options

The options for the `rmxattr` command are:

- `{ file-name | vault-name }`: Specifies the name of the file or vault for which you want to delete the specified extended attribute value.
- `--name`: Specifies the name of the extended attribute.

Examples

Example 6-148 Delete an Extended Attribute

You can use the following commands to delete the `custom_attr` extended attribute.

```
@> rmxattr @MYDATA --name custom_attr


@> rmxattr @MYDATA/file1 --name custom_attr
```

6.2.12 Block Store Management

This section contains references for the Exascale command line interface (ESCLI) commands that are associated with block store management:

- acfsctl
  Control registration for Exascale-managed ACFS file systems.
- chvolume
  Change attributes of a volume.
- chvolumeHAVIP
  Change attributes of a block store virtual IP.
- chvolumebackup
  Change attributes of a volume backup.
- chvolumesnapshot
  Change attributes of a volume snapshot.
- lsacfsfilesystem
  List Exascale-managed ACFS file systems.
- lsinitiator
  List EDV initiator information.
- lsvolume
  List volumes.
- lsvolumeHAVIP
  List block store virtual IPs.
- lsvolumeattachment
  List volume attachments.
- lsvolumebackup
  List volume backups.
- lsvolumesnapshot
  List volume snapshots.
- mkacfsfilesystem
  Create an Exascale-managed ACFS file system.
- mkvolume
  Create a volume or clone a volume snapshot.
- mkvolumeHAVIP
  Create a block store virtual IP.
- mkvolumeattachment
  Create an attachment for an Exascale volume.
- mkvolumebackup
  Create a backup of a volume snapshot.
- mkvolumesnapshot
  Create a snapshot of a volume.
- rmacfsfilesystem
  Delete an Exascale-managed ACFS file system.
- rmvolume
  Delete volumes.

- rmvolumeHAVIP
  Delete a block store virtual IP.
- rmvolumeattachment
  Delete a volume attachment.
- rmvolumebackup
  Delete a volume backup.
- rmvolumesnapshot
  Delete volume snapshots.

6.2.12.1 acfsctl

Control registration for Exascale-managed ACFS file systems.

Purpose

The `acfsctl` command controls registration for Exascale-managed implementations of Oracle Advanced Cluster File System (ACFS).

Syntax

```
acfsctl register volume-id mount-path [ --attributes
attribute=value[,attribute=value] ... ]


acfsctl deregister acfs-id [ --force ]
```

Command Options

The options for the `acfsctl register` command are:

- *volume-id*: Identifies the volume that hosts the file system. You can use the lsacfsfilesystem command to find the volume identifier associated with each Exascale-managed ACFS file system.
- *mount-path*: Specifies the file system mount path.
- `--attributes`: Optionally specifies attributes for the file system registration.

The options for the `acfsctl deregister` command are:

- *acfs-id*: Identifies the ACFS file system being deregistered. You can use the lsacfsfilesystem command to find the identifier for each Exascale-managed ACFS file system.
- `--force`: Optionally ignores any errors and forces deregistration of the file system, even if it is use.

Usage Notes

Note the following information when using the `acfsctl register` command:

- The specified volume (*volume-id*) must have an associated EDV attachment and existing ACFS file system, otherwise the command fails.
- If the EDV attachment is a cluster-wide attachment, the ACFS file system is mounted on every node in the Oracle Grid Infrastructure (GI) cluster. If the EDV attachment is a node-specific attachment, the file system is mounted only on that node. In all cases, the ACFS details are registered with the GI cluster, and the file system is automatically mounted (or remounted) by Oracle Clusterware as required.

Note the following when using the `acfsctl deregister` command:

- If the underlying EDV attachment is a cluster-wide attachment, the ACFS file system is dismounted and the mount point directory is removed on every node in the Oracle Grid Infrastructure (GI) cluster. If the EDV attachment is a node-specific attachment, the file system is dismounted and the mount point directory is removed on that node. The command also removes the ACFS file system registration in the GI cluster.

6.2.12.2 chvolume

Change attributes of a volume.

Purpose

The `chvolume` command allows you to modify the attributes of an Exascale volume.

Syntax

```
chvolume volume-id [ --attributes attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `chvolume` command are:

- *volume-id*: Identifies the volume being changed. You can use the lsvolume command to find the identifier for each volume.
- `--attributes`: Optionally specifies attributes to change.

  Use the `describe chvolume` command to view details about the volume attributes you can modify with `chvolume`.

Usage Notes

The `chvolume` command only proceeds if the specified volume is not attached.

Examples

Example 6-149 Change the Volume Size

This example shows changing the size of the volume with the volume ID `2:50e52177583f4be4bad68ac20b65001e`.

```
@> chvolume 2:50e52177583f4be4bad68ac20b65001e --attributes size=200m
```

Example 6-150 Change the Volume Owners

A volume can have up to two owners. The following examples show different ways to change the volume owners.

- The following command sets *scott* as the sole owner of the specified volume:

  ```
  @> chvolume 2:50e52177583f4be4bad68ac20b65001e --attributes owners=scott
  ```

- The following command adds *peter* as an owner of the specified volume:

  ```
  @> chvolume 2:50e52177583f4be4bad68ac20b65001e --attributes owners=+peter
  ```

- The following command removes *dave* as an owner of the specified volume:

```
@> chvolume 2:50e52177583f4be4bad68ac20b65001e --attributes owners=-dave
```

6.2.12.3 chvolumeHAVIP

Change attributes of a block store virtual IP.

Purpose

The `chvolumeHAVIP` command allows you to modify the attributes of an Exascale block store high-availability (HA) virtual IP (VIP).

Syntax

```
chvolumeHAVIP block-store-vip-id [ --attributes
attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `chvolumeHAVIP` command are:

- *block-store-vip-id*: Identifies the block store VIP being changed. You can use the lsvolumeHAVIP command to find the identifier for each block store VIP.
- `--attributes`: Optionally specifies attributes to change.

  The following attributes can be changed:

  - `name`

Examples

Example 6-151 Change the Name of a Block Store VIP

This example shows changing the name of the specified block store VIP to `HAVIP_1`.

```
@> chvolumeHAVIP 2:66f187eb86824f56a5c92003f922e97f --attributes name=HAVIP_1
```

6.2.12.4 chvolumebackup

Change attributes of a volume backup.

Purpose

The `chvolumebackup` command allows you to modify the attributes of an Exascale volume backup.

Syntax

```
chvolumebackup volume-backup-id [ --attributes
attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `chvolumebackup` command are:

- *volume-backup-id*: Identifies the volume backup being changed. You can use the lsvolumebackup command to find the identifier for each volume backup.
- `--attributes`: Optionally specifies attributes to change.

  The following attributes can be changed:

  – `name`

Examples

Example 6-152 Change the Name of a Volume Backup

This example shows changing the name of the specified volume backup to `MYDATA_BACKUP`.

```
@> chvolumebackup 2b1:50e52177583f4be4bad68ac20b65001e --attributes
name=MYDATA_BACKUP
```

6.2.12.5 chvolumesnapshot

Change attributes of a volume snapshot.

Purpose

The `chvolumesnapshot` command allows you to modify the attributes of an Exascale volume snapshot.

Syntax

```
chvolumesnapshot volume-snapshot-id [ --attributes
attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `chvolumesnapshot` command are:

- *volume-snapshot-id*: Identifies the volume snapshot being changed. You can use the lsvolumesnapshot command to find the identifier for each volume snapshot.
- `--attributes`: Optionally specifies attributes to change.

  The following attributes can be changed:

  – `name`

Examples

Example 6-153 Change the Name of a Volume Snapshot

This example shows changing the name of the specified volume snapshot to `VOL_SNAP_1`.

```
@> chvolumesnapshot 1.1:50e52177583f4be4bad68ac20b65001e --attributes
name=VOL_SNAP_1
```

6.2.12.6 lsacfsfilesystem

List Exascale-managed ACFS file systems.

Purpose

The `lsacfsfilesystem` command displays information about Exascale-managed implementations of Oracle Advanced Cluster File System (ACFS).

Syntax

```
lsacfsfilesystem [ acfs-id [ acfs-id ] ... ] [ -l ] [ --detail ] [ --
attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lsacfsfilesystem` command are:

- *acfs-id*: Identifies the ACFS file system that you want to list information about. If not specified, the command displays information about all file systems.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with -.

For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

```
--sort -name,createTime
```

6.2.12.7 lsinitiator

List EDV initiator information.

Purpose

The `lsinitiator` command displays information about Exascale Direct Volume (EDV) initiators.

Syntax

```
lsinitiator [ edv-initiator-id ] [ --attributes attribute[,attribute] ... ]
[ --detail ] [ -l ]
```

Command Options

The options for the `lsinitiator` command are:

- *edv-initiator-id*: Identifies the EDV initiator that you want to list information about. If not specified, the command displays information about all EDV initiators.
- `--attributes`: Lists the specific attributes to display.
- `--detail`: Lists all attributes in a detailed form.
- `-l`: Returns output in a long, tabular form.

Examples

Example 6-154 List EDV Initiator Information

The following example shows how to list all EDV initiators in the Exascale cluster.

```
@> lsinitiator
```

Example 6-155 List Information for a Specific EDV Initiator

The following example shows how to list detailed information about the EDV initiator with the ID: `3d8c1b1c-2fa8-fc6e-3d8c-1b1c2fa8fc6e`.

```
@> lsinitiator 3d8c1b1c-2fa8-fc6e-3d8c-1b1c2fa8fc6e
```

6.2.12.8 lsvolume

List volumes.

Purpose

The `lsvolume` command displays information about Exascale volumes.

Syntax

```
lsvolume [ volume-id [ volume-id ] ... ] [ -l ] [ --detail ] [ --attributes
attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lsvolume` command are:

- *volume-id*: Identifies an Exascale volume that you want to list information about. If not specified, the command displays information about all volumes.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-156 List Information for a Specific Volume

The following example shows how to list detailed information for volume ID: `2:50e52177583f4be4bad68ac20b65001e`.

```
@> lsvolume 2:50e52177583f4be4bad68ac20b65001e --detail
```

Example 6-157 List Specific Volume Attributes

The following example shows how to list specific volume attributes for volume ID: `2:50e52177583f4be4bad68ac20b65001e`.

```
@> lsvolume 2:50e52177583f4be4bad68ac20b65001e --attributes
id,size,iopsProvisioned
```

Example 6-158 List Volumes Matching a Filter

The following example shows how to list detailed volume information for volumes that are sized greater than 100 MB. The output is also sorted by volume size.

```
@> lsvolume --detail --filter size>100M --sort size
```

6.2.12.9 lsvolumeHAVIP

List block store virtual IPs.

Purpose

The `lsvolumeHAVIP` command displays information about Exascale block store high-availability (HA) virtual IPs (VIPs).

Syntax

```
lsvolumeHAVIP [ block-store-vip-id [ block-store-vip-id ] ... ] [ -l ] [ --
detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lsvolumeHAVIP` command are:

- *block-store-vip-id*: Identifies the block store VIP that you want to list information about. If not specified, the command displays information about all block store VIPs.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.

- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-159 List Detailed Block Store VIP Information

The following example shows how to list detailed block store VIP information.

```
@> lsvolumeHAVIP --detail
```

Example 6-160 List a Specific Attributes of a Block Store VIP

The following example shows how to list specific attributes for all block store VIPs. In the example, the attributes being listed are `networkIPaddress` and `creationTime`.

```
@> lsvolumeHAVIP --attributes networkIPaddress,creationTime
```

6.2.12.10 lsvolumeattachment

List volume attachments.

Purpose

The `lsvolumeattachment` command displays information about Exascale volume attachments.

Syntax

```
lsvolumeattachment [ --protocol edv ] [ attachment-id [ attachment-id ] ... ]
[ -l ] [ --detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ] [ --count value ]
```

```
lsvolumeattachment --protocol iscsi [ attachment-id [ attachment-id ] ... ]
[ -l ] [ --detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ] [ --count value ]
```

Command Options

The options for the `lsvolumeattachment` command are:

- `--protocol`: Specifies the attachment protocol, either `edv` (Exascale Direct Volume) or `iscsi`. If not specified, `edv` is assumed.
- *attachment-id*: Identifies the volume attachment that you want to list information about. If not specified, the command displays information about all volume attachments.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with -.

For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

```
--sort -name,createTime
```

Examples

Example 6-161 List an EDV Attachment Using an Attachment ID

The following example shows how to list attachment information for an Exascale Direct Volume by using an attachment ID. In the example, the attachment ID is `3:50e52177583f4be4bad68ac20b65001e`.

```
@> lsvolumeattachment 3:50e52177583f4be4bad68ac20b65001e
```

Example 6-162 List Specific EDV Attachment Attributes

The following example shows how to list specific attributes of all EDV attachments. In the example, the attributes being listed are `id` and `devicePath`.

```
@> lsvolumeattachment --attributes id,devicePath
```

Example 6-163 List an iSCSI Volume Attachment Using an Attachment ID

The following example shows how to list information for an iSCSI volume attachment by using an attachment ID. In the example, the attachment ID is `2-2:50e52177583f4be4bad68ac20b65001e`.

```
@> lsvolumeattachment --protocol iscsi 2-2:50e52177583f4be4bad68ac20b65001e
```

Example 6-164 List Specific Attributes of an iSCSI Volume Attachment

The following example shows how to list specific attributes of an iSCSI volume attachment. In the example, the attachment ID is `2-2:50e52177583f4be4bad68ac20b65001e`, and the attributes being listed are `initiatorIQN` and `targetPortal`.

```
@> lsvolumeattachment --protocol iscsi 2-2:50e52177583f4be4bad68ac20b65001e --
attributes initiatorIQN,targetPortal
```

6.2.12.11 lsvolumebackup

List volume backups.

Purpose

The `lsvolumebackup` command displays information about Exascale volume backups.

Syntax

```
lsvolumebackup [ volume-backup-id [ volume-backup-id ] ... ] [ -l ] [ --
detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
```

```
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lsvolumebackup` command are:

- *volume-backup-id*: Identifies an Exascale volume backup that you want to list information about. If not specified, the command displays information about all volume backups.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

  Dates can be specified using the following formats:

  - `yyyy-MM-dd''T''HH:mm:ss`
  - `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
  - `HH:mm:ss` (Date is assumed to be today)

  A date can also be followed by a timezone specification.

  Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
- Sorting attributes are specified as: `[-]attribute`. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-165 List Information for Volume Backups

The following example shows how to list detailed information about all volume backups.

```
@> lsvolumebackup --detail
```

Example 6-166 List Information for Specific Volume Backups

The following example shows how to list detailed information about two volume backups with specific IDs.

```
@> lsvolumebackup 2b1:fedd311081ee490481b7e19bdb691999
2b2:fedd311081ee490481b8f2acab692334 --detail
```

Example 6-167 List Information for Specific Volume Backups

The following example shows how to list information about volume backups with names starting with `MYDATA_BACKUP`.

```
@> lsvolumebackup --filter name=MYDATA_BACKUP*
```

6.2.12.12 lsvolumesnapshot

List volume snapshots.

Purpose

The `lsvolumesnapshot` command displays information about Exascale volume snapshots.

Syntax

```
lsvolumesnapshot [ volume-snap-id [ volume-snap-id ] ... ] [ -l ] [ --
detail ] [ --attributes attribute[,attribute] ... ]
     [ --filter filter[,filter] ... ] [ --sort [-]attribute[,
[-]attribute] ... ]
     [ --count value ]
```

Command Options

The options for the `lsvolumesnapshot` command are:

- *volume-snap-id*: Identifies an Exascale volume snapshot that you want to list information about. If not specified, the command displays information about all volume snapshots.
- `-l`: Returns output in a long, tabular form.
- `--detail`: Lists all attributes in a detailed form.
- `--attributes`: Lists the specific attributes to display.
- `--filter`: Used to specify conditions for filtering the list output.
- `--sort`: Used to sort the output using the specified attributes.
- `--count`: Specifies the maximum number of results to report.

Usage Notes

Note the following information when using this command:

- Filter conditions are specified as: `<attribute><operator><value>`.

  The allowed operators are =, !=, >=, <=, >, and <.

  Multiple comma-separated filter conditions are combined using AND logic.

Dates can be specified using the following formats:

- `yyyy-MM-dd''T''HH:mm:ss`
- `yyyy-MM-dd` (Time is assumed to be 00:00 AM)
- `HH:mm:ss` (Date is assumed to be today)

A date can also be followed by a timezone specification.

Sizes can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

- Sorting attributes are specified as: `[-]`*`attribute`*. Multiple sort attributes are comma-delimited. The default sort order is ascending. For descending sort order, prefix the attribute name with `-`.

  For example, use the following to primarily sort by name in descending order, and use creation time in ascending order to further sort entries with the same name:

  ```
  --sort -name,createTime
  ```

Examples

Example 6-168 List Information for Volume Snapshots

The following example shows how to list detailed information about all of the volume snapshots.

```
@> lsvolumesnapshot --detail
```

Example 6-169 List Information for a Specific Volume Snapshot

The following example shows how to list detailed information about volume snapshot ID: `2.1`.

```
@> lsvolumesnapshot 2.1:50e52177583f4be4bad68ac20b65001e
```

Example 6-170 List Specific Attributes for Volume Snapshots

The following example shows how to list specific attributes of all the volume snapshots.

```
@> lsvolumesnapshot --attributes id,volume,size
```

6.2.12.13 mkacfsfilesystem

Create an Exascale-managed ACFS file system.

Purpose

The `mkacfsfilesystem` command creates an Exascale-managed implementation of Oracle Advanced Cluster File System (ACFS) on an Exascale volume using Exascale Direct Volumes (EDV).

Syntax

```
mkacfsfilesystem volume-id mount-path [ --force ] [ --attributes
attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `mkacfsfilesystem` command are:

- *volume-id*: Identifies the volume that hosts the file system. You can use the lsvolume command to find the identifier for each volume.
- *mount-path*: Specifies the file system mount path.
- `--force`: Optionally instructs the command to ignore and overwrite any existing file system on the volume.
- `--attributes`: Optionally specifies attributes for the file system.

Usage Notes

Note the following information when using this command:

- The specified volume (*volume-id*) must have an associated EDV attachment, otherwise the command fails.
- If the EDV attachment is a cluster-wide attachment, the ACFS file system is mounted on every node in the Oracle Grid Infrastructure (GI) cluster. If the EDV attachment is a node-specific attachment, the file system is mounted only on that node. In all cases, the ACFS details are registered with the GI cluster, and the file system is automatically mounted (or remounted) by Oracle Clusterware as required.
- Before using the `mkacfsfilesystem` command, ensure that ACFS is configured appropriately on the target system.

  For instance, you can create a file system using ACFS encryption by specifying the `encryptionEnabled`, `encryptionAlgorithm`, and `encryptionKeyLength` attributes in the `mkacfsfilesystem` command. For example:

```
@> mkacfsfilesystem 1:bbd6fb4c75e2411b9bf366fe702eabaf /mnt/acfs1
    --attributes
encryptionEnabled=true,encryptionAlgorithm=AES,encryptionKeyLength=192
```

  However, the command fails if ACFS encryption is not initialized on the target system. To initialize ACFS encryption, the system administrator must run the following command before the file system is created:

```
# acfsutil encr init
```

6.2.12.14 mkvolume

Create a volume or clone a volume snapshot.

Purpose

The `mkvolume` command allows you to create a new Exascale volume, clone a volume snapshot, or directly clone an existing volume.

Syntax

To create a new volume:

```
mkvolume size --vault vault [ --attributes
attribute=value[,attribute=value] ... ]
```

To clone a volume snapshot:

```
mkvolume --attributes volumeSnapshot=parent_snapshot_id[,name=clone_name]
[,iopsProvisioned=integer_value][,iopsInherited={true|false}]
```

To directly clone an existing volume:

```
mkvolume --attributes volumeSource=parent_volume_id[,name=clone_name]
[,iopsProvisioned=integer_value][,iopsInherited={true|false}]
```

> **Note:**
>
> To directly clone an existing volume, you must use Oracle Exadata System Software release 25.1.0 or later.

Command Options

The options for the `mkvolume` command are:

- *size*: Specifies the size of the volume. The *size* can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.

  > **Note:**
  >
  > If you intend to use the volume to support Oracle ACFS, note that ACFS requires a minimum volume size of 512 MB.

- `--vault`: Specifies the vault that the volume is created in.
- `--attributes`: Optionally specifies attributes for the volume.

  Use the `describe mkvolume` command to view details about all the volume attributes you can set with `mkvolume`.

Usage Notes

- You can optionally specify up to two volume owners by setting the `owners` attribute to the name(s) of the Exascale user(s) that you want to own the volume.

  To set a sole owner, use:

  ```
  @> mkvolume ... --attributes owners=owner1
  ```

To set two owners, use:

```
@> mkvolume ... --attributes owners=owner1,owner2
```

- The following usage notes apply only when the `mkvolume` command is used to clone a volume.
  - When cloning a volume snapshot, `volumeSnapshot=parent_snapshot_id` identifies the volume snapshot that you want to clone.
  - When directly cloning an existing volume, `volumeSource=parent_volume_id` identifies the volume that you want to clone.
  - When cloning a volume, either directly or using a volume snapshot, you can optionally specify the following attribute settings. No additional command options or attribute settings are required or permitted.
    - `name=clone_name`: Optionally specifies the name of the volume clone, which makes it easier for you to identify later. If not specified, a system-generated name is assigned.
    - `iopsProvisioned=integer_value`: Optionally specifies the I/O bandwidth provisioned for the volume clone. The I/O bandwidth is expressed as the number of I/Os per second (IOPS).
    - `iopsInherited={true|false}`: Optionally specifies whether the volume clone inherits I/O bandwidth from the nearest ancestor in the volume hierarchy with provisioned (not inherited) I/O bandwidth.
  - When cloning a volume, either directly or using a volume snapshot, the following describes the relationship between the `iopsProvisioned` and `iopsInherited` attribute settings:
    - If `iopsProvisioned` is unspecified and `iopsInherited` is not set to `false` (`iopsInherited` is unspecified or `iopsInherited=true`), then:
      - The volume clone inherits I/O bandwidth from its nearest ancestor.
      - The volume clone has the setting: `iopsInherited=true`.
    - If `iopsProvisioned` is unspecified and `iopsInherited=false`, then:
      - The volume clone is provisioned with unlimited `iopsProvisioned`.
      - The volume clone has the setting: `iopsInherited=false`.
    - If you specify a value for `iopsProvisioned` and `iopsInherited` is not set to `true` (`iopsInherited` is unspecified or `iopsInherited=false`), then:
      - The volume clone is governed by the specified `iopsProvisioned` value.
      - The volume clone has the setting: `iopsInherited=false`.
    - An error occurs if you specify a value for `iopsProvisioned` and `iopsInherited=true`. This is an invalid combination.

Examples

Example 6-171 Create a Volume

This example shows creating a volume that is 100 MB in size and located in the vault named `MYVOLS`.

```
@> mkvolume 100m --vault MYVOLS
```

Example 6-172 Create a Volume with Attribute Settings

This example shows creating a volume that is 200 MB in size and located in the vault named `MYVOLS`. The example also includes the following specific attribute settings:

- The volume name is set to `vol2`.
- The volume is provisioned with 1000 IOPS (IOs per second).
- The volume is created with two owners (Exascale users named `scott` and `dave`).

```
@> mkvolume 200m --vault MYVOLS --attributes
name=vol2,iopsProvisioned=1000,owners=scott,dave
```

6.2.12.15 mkvolumeHAVIP

Create a block store virtual IP.

Purpose

The `mkvolumeHAVIP` command creates a block store high-availability (HA) virtual IP (VIP).

Syntax

```
mkvolumeHAVIP --attributes networkIPaddress=ip-address[,attribute=value] ...
```

Command Options

The options for the `mkvolumeHAVIP` command are:

- `--attributes`: Specifies attribute settings for the block store VIP:
  - `networkIPaddress`: Specifies the IP address for the block store VIP. This attribute must be specified.
  - `networkSubnet`: Optionally specifies the network subnet mask.
  - `networkGateway`: Optionally specifies the network gateway.
  - `networkBroadcast`: Optionally specifies the network broadcast address.
  - `name`: Optionally specifies a name that you can use to identify the block store VIP.
  - `networkType`: Optionally specifies the network type. Valid values are `client` and `storage`.

Examples

Example 6-173 Create a Block Store VIP

The following example shows creating a block store VIP with just an IP address.

```
@> mkvolumeHAVIP --attributes networkIPaddress=192.0.2.21
```

Example 6-174 Create a Block Store VIP

The following example shows creating a block store VIP with an IP address and optional attributes.

```
@> mkvolumeHAVIP --attributes
networkIPaddress=192.0.2.21,networkSubnet=24,networkBroadcast=192.0.2.255,netw
orkGateway=192.0.2.0
```

6.2.12.16 mkvolumeattachment

Create an attachment for an Exascale volume.

Purpose

The `mkvolumeattachment` command allows you to create an attachment for an Exascale volume.

Syntax

```
mkvolumeattachment volume-id device-name --attributes { giClusterId=cluster-
id | initiator=edv-initiator-id } [ --protocol edv ]


mkvolumeattachment volume-id iscsi-initiator-id --protocol iscsi [ --
attributes attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `mkvolumeattachment` command are:

- *volume-id*: Specifies the identifier for the Exascale volume for which you want to create an attachment. You can use the lsvolume command to find the identifier for each volume.
- *device-name*: For an Exascale Direct Volume (EDV) attachment only, specifies the device name to use in conjunction with the attachment. This is a user-supplied name, which is applied to the device file that is associated with the attachment. After attachment, the corresponding device file is located under `/dev/exc/`.
- `--protocol`: Specifies the attachment protocol, either `edv` or `iscsi`. If not specified, `edv` is assumed.
- *iscsi-initiator-id*: For an iSCSI attachment only, specifies the identifier used by iSCSI initiators to identify the attachment.
- `--attributes`: Specifies different attributes depending on the attachment type.
  - For a cluster-wide EDV attachment only, the `giClusterId` attribute specifies the Oracle Grid Infrastructure (GI) cluster ID associated with the volume attachment. When you

specify this attribute, the corresponding device file is created on all nodes in the Oracle GI cluster.

- For a node-specific EDV attachment only, the `initiator` attribute specifies the EDV initiator ID associated with the volume attachment. When you specify this attribute, the corresponding device file is only created on the cluster node associated with the specified EDV initiator ID.
- For an iSCSI attachment only, specifies various attributes for the attachment. As a minimum, the following attributes are required for CHAP (Challenge Handshake Authentication Protocol):
  - `chapUserId`
  - `chapPassword`
  - `chapMutualUserId`
  - `chapMutualPassword`

Usage Notes

The following notes apply only to EDV attachments:

- During initial system deployment with Oracle Exadata Deployment Assistant (OEDA), the Exascale Direct Volume (EDV) service is configured on each Exadata compute node (bare-metal or VM) and is related to the Exascale user that manages the Oracle Grid Infrastructure (GI) cluster. To create an EDV attachment, you must use the Exascale user linked with the EDV service.

  If the GI cluster uses a non-role-separated user configuration with one Oracle OS user account, then the associated Exascale user is related to the EDV service. If the GI cluster uses a role-separated configuration with a Grid OS user account and an Oracle OS user account, then the EDV service is linked to the Exascale user associated with the Grid OS account.

  To find the Exascale user linked with the EDV service, use the ESCLI lsinitiator command with the `--detail` option and examine the `user` attribute.

- Each EDV attachment also has a kernel device file at `/dev/exc-devN`, where *N* is the minor number of the device. The kernel device name is contained as an attribute of the EDV attachment and is visible using the ESCLI lsvolumeattachment command. The relationship between the kernel device file and the user-named device file (under `/dev/exc/`) is also recorded in the udev database and is visible using the following Linux command:

  ```
  # udevadm info device-file
  ```

  For the *device-file* value, you can specify either the kernel device file (`/dev/exc-devN`) or the user-named device file (under `/dev/exc/`).

- By default, read and write access to EDV device files is only available to the `root` operating system user and members of the `disk` group. Depending on your use case, you may need to modify the permissions on the EDV device files before using them.

  For example, to make the EDV device file at `/dev/exc/myvol` readable and writable by the `oracle` user and `dba` group, you could configure it using a udev rule similar to the following:

  ```
  # cat /etc/udev/rules.d/57-edv-user.rules
  KERNEL=="exc-*", ENV{EXC_ALIAS}=="myvol", OWNER="oracle", GROUP="dba",
  MODE="0660"
  ```

- To facilitate the management of udev rules related to EDV devices, each EDV client node is configured with a template udev rules file at `/etc/udev/rules.d/57-edv-user.rules`, which you can modify to fulfill your requirements. To maintain existing udev rules, `/etc/udev/rules.d/57-edv-user.rules` is preserved whenever the EDV client software is updated.
- Each EDV client node can support a maximum of 1024 attachments at the same time. This limit includes the total of all cluster attachments involving the server, as well as local attachments specific to the server.

Examples

Example 6-175 Create a Cluster-Wide EDV Volume Attachment

You can create a cluster-wide EDV volume attachment by specifying a *volume-id*, *device-name*, and *cluster-id*.

In the following example, the *volume-id* is `7:50e52177583f4be4bad68ac20b65001e`, the *device-name* is `myvol`, and the *cluster-id* is `72071863FA3E7FCAFF9F42A96957F4C5`.

```
@> mkvolumeattachment 7:50e52177583f4be4bad68ac20b65001e myvol --attributes
giClusterId=72071863-fa3e-7fca-ff9f-42a96957f4c5
```

Example 6-176 Create a Node-Specific EDV Volume Attachment

You can create a node-specific EDV volume attachment by specifying a *volume-id*, *device-name*, and *edv-initiator-id*.

In the following example, the *volume-id* is `7:50e52177583f4be4bad68ac20b65001e`, the *device-name* is `myvol`, and the *edv-initiator-id* is `b0b057aa-1f2c-0f48-b0b0-57aa1f2c0f48`.

```
@> mkvolumeattachment 7:50e52177583f4be4bad68ac20b65001e myvol --attributes
initiator=b0b057aa-1f2c-0f48-b0b0-57aa1f2c0f48
```

Example 6-177 Create an iSCSI Volume Attachment

You can create an iSCSI volume attachment by specifying the `--protocol iscsi` command option along with the *volume-id*, *initiator-id*, and required CHAP attributes.

In the following example, the *volume-id* is `2:50e52177583f4be4bad68ac20b65001e`, and the *initiator-id* is `iqn.1988-10.com.oracle.vm01`. The required CHAP attributes are specified following the `--attributes` command option.

```
@> mkvolumeattachment 2:50e52177583f4be4bad68ac20b65001e
iqn.1988-10.com.oracle.vm01 --protocol iscsi
     --attributes chapUserId=X1234567-user-1,chapPassword=X1234567-
pass-1,chapMutualUserId=X1234567-user-2,chapMutualPassword=X1234567-pass-2
```

6.2.12.17 mkvolumebackup

Create a backup of a volume snapshot.

Purpose

The `mkvolumebackup` command creates a backup of an Exascale volume snapshot.

Syntax

```
mkvolumebackup --attributes volumeSnapshot=volume-snapshot-
id[,attribute=value] ... ]
```

Command Options

The command options for the `mkvolumebackup` command are:

- `--attributes`: Specifies attributes settings for the volume backup:
  - `volumeSnapshot`: Identifies the volume snapshot to back up. This attribute must be specified. You can use the lsvolumesnapshot command to find the identifier for each volume snapshot.
  - `name`: Optionally specifies a name that you can use to identify the volume backup.

Examples

Example 6-178 Create a Volume Snapshot Backup

This example shows creating a volume snapshot backup for snapshot ID `2.1:50e52177583f4be4bad68ac20b65001e`.

```
@> mkvolumebackup --attributes
volumeSnapshot=2.1:50e52177583f4be4bad68ac20b65001e
```

Example 6-179 Create a Volume Snapshot Backup with a User-Specified Name

This example shows creating a volume snapshot backup for snapshot ID `2.1:50e52177583f4be4bad68ac20b65001e`. The resulting backup is named `MYDATA_BACKUP`.

```
@> mkvolumebackup --attributes
volumeSnapshot=2.1:50e52177583f4be4bad68ac20b65001e,name=MYDATA_BACKUP
```

6.2.12.18 mkvolumesnapshot

Create a snapshot of a volume.

Purpose

The `mkvolumesnapshot` creates a snapshot of the specified Exascale volume.

Syntax

```
mkvolumesnapshot volume-id [ --attributes
attribute=value[,attribute=value] ... ]
```

Command Options

The options for the `mkvolumesnapshot` command are:

- *volume-id*: Identifies the volume that you want to snapshot. You can use the lsvolume command to find the identifier for each volume.
- `--attributes`: Optionally specifies attributes for the volume snapshot.

Usage Notes

The `mkvolumesnapshot` command fails if the specified volume supports Exascale Direct Volume (EDV) attachments and Oracle Advanced Cluster File System (ACFS) is currently mounted. To create a volume snapshot in this situation, you must either:

- Unmount the affected file system on all servers before you use the `mkvolumesnapshot` command. Then, mount the file system again afterward.
- Use the Oracle ACFS command-line utility (`acfsutil`) to synchronize the file system and create the volume snapshot. On an EDV client node, run the `acfsutil volsnap create` command as the `root` user and specify the ACFS mountpoint or EDV device.

  For example:

  ```
  # acfsutil volsnap create /mnt/myacfs
  ```

Examples

Example 6-180 Create a Volume Snapshot

This example shows creating a snapshot of volume `1:50e52177583f4be4bad68ac20b65001e`.

```
@> mkvolumesnapshot 1:50e52177583f4be4bad68ac20b65001e
```

Example 6-181 Create a Volume Snapshot with Attribute Settings

This example shows creating a snapshot of volume `1:50e52177583f4be4bad68ac20b65001e` with specific attribute settings.

```
@> mkvolumesnapshot 1:50e52177583f4be4bad68ac20b65001e --attributes
name=snap1,compartmentId=myCompartment123
```

6.2.12.19 rmacfsfilesystem

Delete an Exascale-managed ACFS file system.

Purpose

The `rmacfsfilesystem` command deletes an Exascale-managed implementation of Oracle Advanced Cluster File System (ACFS).

Syntax

```
rmacfsfilesystem acfs-id
```

Command Options

The options for the `rmacfsfilesystem` command are:

- *acfs-id*: Identifies the ACFS file system being deleted. You can use the lsacfsfilesystem command to find the identifier for each Exascale-managed ACFS file system.

Usage Notes

Note the following information when using this command:

- If the underlying EDV attachment is a cluster-wide attachment, the ACFS file system is dismounted and the mount point directory is removed on every node in the Oracle Grid Infrastructure (GI) cluster. If the EDV attachment is a node-specific attachment, the file system is dismounted and the mount point directory is removed on that node. The command also removes the ACFS file system registration in the GI cluster. Furthermore, the file system is removed from the underlying volume, effectively deleting the files in the file system. The command does not alter the underlying volume or volume attachment in any other way.

6.2.12.20 rmvolume

Delete volumes.

Purpose

The `rmvolume` command deletes Exascale volumes.

Syntax

```
rmvolume volume-id [ volume-id ] ...
```

Command Options

The options for the `rmvolume` command are:

- *volume-id*: Identifies the volume being deleted. You can use the lsvolume command to find the identifier for each volume.

Usage Notes

Note the following information when using this command:

- You must detach the volume before it can be deleted.

Examples

Example 6-182 Delete a Volume

This example shows deleting the volume with the volume ID `1:50e52177583f4be4bad68ac20b65001e`.

```
@> rmvolume 1:50e52177583f4be4bad68ac20b65001e
```

6.2.12.21 rmvolumeHAVIP

Delete a block store virtual IP.

Purpose

The `rmvolumeHAVIP` command deletes a block store high-availability (HA) virtual IP (VIP).

Syntax

```
rmvolumeHAVIP block-store-vip-id
```

Command Options

The options for the `rmvolumeHAVIP` command are:

- *block-store-vip-id*: Identifies the block store VIP to delete. You can use the lsvolumeHAVIP command to find the identifier for each block store VIP.

Examples

Example 6-183 Delete a Block Store VIP

The following example shows deleting the block store VIP with the identifier `3`.

```
@> rmvolumeHAVIP 3
```

6.2.12.22 rmvolumeattachment

Delete a volume attachment.

Purpose

The `rmvolumeattachment` command allows you to delete an Exascale volume attachment.

Syntax

```
rmvolumeattachment attachment-id [ --protocol edv ] [ --force ]


rmvolumeattachment attachment-id iscsi-initiator-id --protocol iscsi [ --
force ]
```

Command Options

The options for the `rmvolumeattachment` command are:

- *attachment-id*: Identifies the volume attachment that is the subject of the operation. You can use the lsvolumeattachment command to find the identifier for each volume attachment.
- `--protocol`: Specifies the attachment protocol, either `edv` (Exascale Direct Volume) or `iscsi`. If not specified, `edv` is assumed.
- *iscsi-initiator-id*: For an iSCSI attachment only, specifies the iSCSI initiator identifier, which must be the same one that was used to create the attachment.
- `--force`: Optionally removes a volume attachment even if it is being used.

Examples

Example 6-184 Delete an EDV Attachment

You can delete an Exascale Direct Volume (EDV) attachment by specifying just the *attachment-id*. In the following example, the *attachment-id* is `3:50e52177583f4be4bad68ac20b65001e`.

```
@> rmvolumeattachment 3:50e52177583f4be4bad68ac20b65001e
```

Example 6-185 Delete an iSCSI Attachment

You can delete an iSCSI volume attachment by specifying the `--protocol iscsi` command option and the *attachment-id* and *iscsi-initiator-id*. In the following example, the *attachment-id* is `2-2:50e52177583f4be4bad68ac20b65001e`, and the *iscsi-initiator-id* is `iqn.1988-10.com.oracle.vm01`.

```
@> rmvolumeattachment 2-2:50e52177583f4be4bad68ac20b65001e
iqn.1988-10.com.oracle.vm01 --protocol iscsi
```

6.2.12.23 rmvolumebackup

Delete a volume backup.

Purpose

The `rmvolumebackup` command deletes a backup of an Exascale volume.

Syntax

```
rmvolumebackup volume-backup-id
```

Command Options

The options for the `rmvolumebackup` command are:

- *volume-backup-id*: Identifies the volume backup that is being deleted. You can use the lsvolumebackup command to find the identifier for each volume backup.

Examples

Example 6-186 Delete a Volume Backup

This example shows deleting the volume backup having the ID `2b1:50e52177583f4be4bad68ac20b65001e`.

```
@> rmvolumebackup 2b1:50e52177583f4be4bad68ac20b65001e
```

6.2.12.24 rmvolumesnapshot

Delete volume snapshots.

Purpose

The `rmvolumesnapshot` command allows you to delete one or more Exascale volume snapshots.

Syntax

```
rmvolumesnapshot volume-snap-id [ volume-snap-id ] ...
```

Command Options

The options for the `rmvolumesnapshot` command are:

- *volume-snap-id*: Identifies the volume snapshot being deleted. You can use the lsvolumesnapshot command to find the identifier for each volume snapshot.

Usage Notes

You must detach the volume snapshot before it can be deleted.

Examples

Example 6-187 Delete a Volume Snapshot

This example shows deleting the volume snapshot having the ID `2.1:50e52177583f4be4bad68ac20b65001e`.

```
@> rmvolumesnapshot 2.1:50e52177583f4be4bad68ac20b65001e
```

7
Using XSH

This chapter describes how to use the Exascale shell command line interface (XSH):

- Start and Use XSH
- XSH Command Reference

7.1 Start and Use XSH

This topic describes how to start and use the Exascale shell command line interface (XSH).

XSH is a command-line tool that is located on Exadata compute nodes and storage servers. You can use XSH to perform Linux-like commands on Exascale files and storage.

To use XSH you must have access to an Exascale user account and the digital key store (wallet) for the Exascale user account.

If you do not specify a wallet in the XSH command, XSH uses the first Exascale user wallet available in the following search path:

1. `$OSSCONF/eswallet`
2. `$ORACLE_BASE/admin/eswallet`
3. `/etc/oracle/cell/network-config/eswallet`

To run an XSH command, use the following command line syntax:

```
$ xsh [ XSH-command ]
```

In the command line, *XSH-command* specifies an XSH command to run. For example:

```
$ xsh ls
@MYDATA
$ xsh ls @MYDATA
@MYDATA/x  @MYDATA/y  @MYDATA/z1  @MYDATA/z2  @MYDATA/z3
$
```

If no command is specified, XSH displays a command list and basic help information. For example:

```
$ xsh
cat
clone
cp
dd
ls
man
hexdump
rm
scrub
```

```
snap
strings
template
touch
xattr
version
chacl
mv
lsacl

Enter 'xsh man <command>' or 'xsh man -e <command>' for details
E.g., Enter 'xsh man dd' or 'xsh man -e dd'
Enter 'xsh man man' to see other options accepted by 'xsh man'
```

To facilitate troubleshooting, you can enable tracing by setting the `$XSH_TRACE_LEVEL` environment variable. You can set the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing). For example:

```
# sh/bash/ksh
$ export $XSH_TRACE_LEVEL=3
```

A trace file is written for each XSH command issued while tracing is enabled using the `$XSH_TRACE_LEVEL` environment variable.

If the `$ADR_BASE` environment variable is set, the trace file is written to:

```
$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc
```

Otherwise, the trace file is written to:

```
/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc
```

To stop and deactivate tracing, you can set the `$XSH_TRACE_LEVEL` environment variable to `0` (zero). For example:

```
# sh/bash/ksh
$ export $XSH_TRACE_LEVEL=0
```

Apart from using the `$XSH_TRACE_LEVEL` environment variable, you can enable tracing for a single command by including the `--trace` or `-T` option in the XSH command. For example:

```
$ xsh ls --trace 3 @MYDATA
```

7.2 XSH Command Reference

This section contains references for the Exascale shell command line interface (XSH) commands:

- cat
  Dump the contents of a file to standard output.

- chacl
  Modify the access control list (ACL) for a file.
- clone
  Clone files.
- cp
  Copy files.
- dd
  Convert and copy a file.
- hexdump
  Dump the contents of a file to standard output in hex format.
- ls
  List file/directory details.
- lsacl
  Display the access control list (ACL) for files or vaults.
- man
  Display online manual pages.
- mv
  Rename files.
- rm
  Remove files.
- scrub
  On-demand file scrubbing for logical consistency.
- snap
  Snapshot files.
- strings
  Display strings of printable characters in files.
- template
  Display vault-level template attributes.
- touch
  Create a file or modify attributes of an existing file.
- version
  Print XSH version information to standard output.
- xattr
  List and modify extended attributes for files and vaults.

7.2.1 cat

Dump the contents of a file to standard output.

Syntax

```
cat [{ -w | --wallet } wallet-location ]
    [{ -T | --trace } trace-level ]
    [ --aio=naio ] filename
```

Command Options

The options for the `cat` command are:

- *filename*: Specifies the file being output.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`
- `--aio`: Optionally specifies the number of async I/Os to use. The default value is 4.

Examples

Example 7-1 Dump the contents of an Exascale file

The following example dumps the contents of the file at `@MYDATA/myfile` to standard output.

```
$ xsh cat @MYDATA/myfile
```

7.2.2 chacl

Modify the access control list (ACL) for a file.

Syntax

```
chacl [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]
        file-name acl-string
```

Command Options

The options for the `chacl` command are:

- *file-name*: Specifies the name of the file that is the subject of the operation.
- *acl-string*: Specifies an ACL string having the following format.

  `[+]userID1:acl-priv[;userID2:acl-priv] ...`

  In the ACL string:

  – The optional plus sign (+) at the beginning of the ACL string indicates that the specified ACL string is merged into the existing ACL. In this case, users previously listed in the ACL are updated, and new users are added. Without the optional plus sign, the previous ACL is overwritten.

  – *userIDn*: Specifies an Exascale user ID.

Depending on the user creation method, the user ID may be a system-generated value (for example, `96a68014-5762-4579-86ee-29eb743decbd`) or a user-specified value (for example, `scott`).

  - *acl-priv*: Specifies an ACL privilege, which can be one of the following:
    * `I | inspect`: Specifies that the user can view attributes of the file but not its contents.
    * `R | read`: Specifies that the user can read contents of the file. Also confers the `inspect` permission.
    * `U | use`: Specifies that the user can write to the file. Also confers all preceding permissions.
    * `M | manage`: Specifies that the user can manage the file. Also confers all preceding permissions.
    * `0 | none`: Removes all existing permissions from the specified user.

- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Examples

Example 7-2 Replace a File ACL

In this example, the ACL string for the file is replaced with the new ACL string. Under the new ACL, `scott` is permitted to read and inspect the file. No other user can access this file unless permitted by the vault ACL.

```
$ xsh chacl @VAULT/file scott:R
```

Example 7-3 Change a File ACL

In this example, the plus sign (`+`) at the beginning of the ACL string indicates that the specified ACL string is merged into the existing file ACL. In this case, any pre-existing permissions for `jason` are overwritten, and `jason` is permitted to inspect, read, write, and manage the file. No other user permissions are changed.

```
$ xsh chacl @VAULT/file +jason:M
```

Example 7-4 Remove User Privileges from a File ACL

In this example, the plus sign (+) at the beginning of the ACL string indicates that the specified ACL string is merged into the existing file ACL. However, in this case, any pre-existing permissions for `scott` are removed. No other user permissions are changed.

```
$ xsh chacl @VAULT/file +scott:none
```

Example 7-5 Replace a File ACL using an ACL String that Specifies Multiple Users

In this example, the ACL string for the file is replaced with the new ACL string that specifies permissions for multiple users. Under the new ACL, `scott` can inspect the file, and `jason` can read and inspect the file. No other user can access this file unless permitted by the vault ACL.

```
$ xsh chacl @VAULT/file scott:inspect;jason:read
```

7.2.3 clone

Clone files.

Syntax

```
clone [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]
      source1 target1 [ sourceN targetN ]...
```

Command Options

The options for the `clone` command are:

- *source1-N*: Specifies the source file(s) being cloned.
- *target1-N*: Specifies the location of the clone(s).
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Usage Notes

Note the following information when using this command:

- You can use a wildcard (`%`) in the *source* to specify multiple source files, in which case the corresponding *target* must also contain a matching wildcard.
- All files in a clone operation must be in the same vault.
- Multiple *source* and *target* pairs are permitted. In this case, the source file specifications are considered in order, and only the first match is used.

- All clones created in the same operation are point-in-time consistent.

7.2.4 cp

Copy files.

Syntax

```
cp [ -f | --force ] [ --ftype=file-type ] [ --bs=block-size ]
   [ --template=template-name ] [ --content-type=content-type ]
   [ --media-type=media-type ] [ --redundancy=redundancy ]
   [[ -s | --snap ] | [ -c | --clone ]] [ --aio=naio ]
   [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]
   source1 target1 [ sourceN targetN ]...
```

Command Options

The options for the `cp` command are:

- *source1-N*: Specifies the source file being copied.
- *target1-N*: Specifies the location of the file copy. The value cannot be a directory.
- `-f`, `--force`: Forces the target to be overwritten if it exists. This is the default and only behavior.
- `--ftype`: Optionally specifies the Oracle Database file type. The valid values are:
  - `ctrl` - control file
  - `data` - data file
  - `olog` - on-line log file
  - `alog` - archived log file
  - `temp` - temporary sort file
  - `init` - initialization parameter file
  - `pswd` - password file
  - `flog` - flashback log file
  - `ctrk` - change tracking file

  This option only applies when copying into Exascale storage.
- `--bs`: Optionally specifies the block size (in bytes) to use for copying the file(s).
- `--template`: Optionally specifies the name of the Exascale template to use when copying new files into Exascale storage.

  This option only applies when copying new files into Exascale storage.
- `--content-type`: Optionally specifies the content type setting for file creation on Exascale storage.

  If not specified, the value is determined by the file type and the associated template. If specified, this setting overrides the template setting. This option only applies when copying new files into Exascale storage.

  Permitted values are:
  - `DATA`

  - `RECO`

- `--media-type`: Optionally specifies the media type setting for file creation on Exascale storage.

  If not specified, the value is determined by the file type and the associated template. If specified, this setting overrides the template setting. This option only applies when copying new files into Exascale storage.

  Permitted values are:

  - `HC`: Identifies high capacity storage media, which uses hard disk drives (HDDs).
  - `EF`: Identifies extreme flash storage media, which uses flash devices.

- `--redundancy`: Optionally specifies the redundancy setting for file creation on Exascale storage.

  If not specified, the value is determined by the file type and the associated template. If specified, this setting overrides the template setting. This option only applies when copying new files into Exascale storage.

  Permitted values are:

  - `normal`: Indicates 2 mirrored copies of the file data.
  - `high`: Indicates 3 mirrored copies of the file data.

- `-s`, `--snap`: Creates a read-only snapshot of the specified source file.

  This option only applies when copying into Exascale storage.

- `-c`, `--clone`: Creates a writable thin clone of the specified source file.

  This option only applies when copying into Exascale storage.

- `--aio`: Optionally specifies the number of async I/Os to use. The default value is 4.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Usage Notes

Note the following information when using this command:

- If both the source and target locations reside on Exascale storage, you can use a wildcard (`%`) in the source location to specify multiple source files, in which case the corresponding target location must also contain a matching wildcard. For example, see Example 7-8.

Examples

Example 7-6 Copy a File to Exascale

The following example copies the local file at `/tmp/file-1` to `@MYDATA/file-1.copy` on Exascale storage.

```
$ xsh cp /tmp/file-1 @MYDATA/file-1.copy
```

Example 7-7 Copy a File from Exascale

The following example copies the Exascale file at `@MYDATA/file-2` to `/tmp/file-2.copy` on the local file system.

```
$ xsh cp @MYDATA/file-2 /tmp/file-2.copy
```

Example 7-8 Copy Multiple Files on Exascale Storage

The following example uses the wildcard character (`%`) to copy multiple files from `@MYDATA/location1/prefix1%` to `@MYDATA/location2/anotherlocation/prefix2%`. Using the example command, a file named `@MYDATA/location1/prefix1mydata` would be copied to `@MYDATA/location2/anotherlocation/prefix2mydata`.

```
$ xsh cp @MYDATA/location1/prefix1% @MYDATA/location2/anotherlocation/prefix2%
```

7.2.5 dd

Convert and copy a file.

Syntax

```
dd [ --if=name ] [ --of=name ] [ --bs=bytes ] [ --count=num ]
   [ --ibs=bytes ] [ --obs=bytes ] [ --seek=num ] [ --skip=num ]
   [ --imirror=num ] [ --omirror=num ] [ --mirror=num ]
   [ --truncate ] [ --aio=naio ] [ --block ]
   [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]
```

Command Options

The options for the `dd` command are:

- `--if`: Specifies the name of the input file to use instead of standard input.
- `--of`: Specifies the name of the output file to use instead of standard output.
- `--bs`: Specifies the number of bytes to read and write in each block. The default value is `512`.
- `--count`: Specifies the number of input blocks.
- `--ibs`: Specifies the number of bytes to read at a time. The default value is `512`.
- `--obs`: Specifies the number of bytes to write at a time. The default value is `512`.
- `--seek`: Specifies the number of `obs`-sized blocks to skip at the start of the output.

- `--skip`: Specifies the number of `ibs`-sized blocks to skip at the start of the input.
- `--imirror`: Specifies the input mirror number. The first mirror is `0`, the second mirror is `1`, and so on. The default value is `255`, which means 'read any mirror.'
- `--omirror`: Specifies the output mirror number. The first mirror is `0`, the second mirror is `1`, and so on. The default value is `255`, which means 'write all mirrors.'
- `--mirror`: Specifies the input and output mirror number. The first mirror is `0`, the second mirror is `1`, and so on. The default value is `255`, which means 'all mirrors.'
- `--truncate` Resizes the output file.
- `--aio`: Specifies the number of async I/Os to use. The default value is 4.
- `--block`: Specifies that the I/O offsets should be divisible by the relevant I/O block size.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

7.2.6 hexdump

Dump the contents of a file to standard output in hex format.

Syntax

```
hexdump [ -j bytes | --skip-bytes=bytes ] [ -N bytes | --read-bytes=bytes ]
   [ --aio=naio ] [ -x ] [{ -w | --wallet } wallet-location ]
   [{ -T | --trace } trace-level ] filename
```

Command Options

The options for the `hexdump` command are:

- *filename*: Specifies the file being output.
- `-j`, `--skip-bytes`: Optionally specifies the number of input bytes to skip.
- `-N`, `--read-bytes`: Optionally specifies the number of bytes to read and output.
- `--aio`: Optionally specifies the number of async I/Os to use. The default value is 4.
- `-x`: Displays the file contents in hex notation. This is the default behavior anyway.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

If the `$ADR_BASE` environment variable is set, the trace file is written to:

`$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Otherwise, the trace file is written to:

`/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Examples

Example 7-9 Dump the contents of an Exascale file

The following example dumps the contents of the file at `@MYDATA/myfile` to standard output in hex format.

```
$ xsh hexdump @MYDATA/myfile
```

7.2.7 ls

List file/directory details.

Syntax

```
ls [ -h ] [ -l ] [ -r ] [ -t ] [ -a ] [ -1 ] [ -s ] [ -d | --detail ]
   [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]
   [ name ]
```

Command Options

The options for the `ls` command are:

- *name*: Specifies the file or vault that is the subject of the command.

  If no *name* is specified, the command lists all accessible vaults. If the *name* is a specific file, the command lists details for the specified file.

  If the specified *name* is a vault name and the `-s` option is not specified, the command lists all of the files in the vault.

  A wildcard (`%`) is permitted when specifying either a vault name or file name.
- `-h`: Displays output values in a human readable format (for example, `10M`, `22K`, `300G`, and so on). This options is used in conjunction with the `-l` option.
- `-l`: Displays a longer listing including file size and creation date.
- `-r`: Reverses the sort order.
- `-t`: Sort by creation time instead of alphabetically.
- `-a`: Displays hidden/special entries.
- `-1`: List one entry on each line of output.
- `-s`: Displays vault statistics instead of the vault contents.

  This option applies only when the *name* value specifies a vault name. The option is ignored when the *name* value specifies a file name.

- `-d`, `--detail`: Displays a detailed listing of attributes.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Examples

Example 7-10 List all Exascale vaults

The following example lists all vaults visible to the Exascale user associated with the XSH command invocation.

```
$ xsh ls
```

Example 7-11 List the contents of an Exascale vault

The following example lists the files contained in the `@MYDATA` vault.

```
$ xsh ls @MYDATA
```

7.2.8 lsacl

Display the access control list (ACL) for files or vaults.

Syntax

```
lsacl [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]
        [ file-name | vault-name ]...
```

Command Options

The options for the `lsacl` command are:

- `[ file-name | vault-name ]...`: Specifies the name of the file(s) or vault(s) that is the subject of the operation.

  A wildcard (`%`) is permitted when specifying a file name.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Otherwise, the trace file is written to:

`/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Examples

Example 7-12 Display a File ACL

The following command shows the ACL string for the file named `@VAULT/myfile1`.

```
$ xsh lsacl @VAULT/myfile1
```

Example 7-13 Display File ACLs Using a Wildcard

The following command shows the ACL string for all files in `@VAULT` having names starting with `myfile`.

```
$ xsh lsacl @VAULT/myfile%
```

7.2.9 man

Display online manual pages.

Syntax

`man [ -a ] [ -c ] [ -e ] [ -f ] [ -o ] [ -x ] [ entry ]`

Command Options

The options for the `man` command are:

- *entry*: Specifies the manual entry that you want to display. If none is specified, the command lists all manual pages.
- `-a`: Displays all details, including examples.
- `-c`: Displays the command name.
- `-e`, `-x`: Displays example information.
- `-f`: Displays details except for examples. This is the default behavior if *entry* is specified without any other options.
- `-o`: Displays brief (one line) information.

Examples

Example 7-14 Display command details

The following example displays details about the `dd` command.

```
$ xsh man dd
```

7.2.10 mv

Rename files.

Syntax

```
mv [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]
   source1 target1 [ sourceN targetN ]...
```

Command Options

The options for the `mv` command are:

- *source1-N*: Specifies the source file being moved.
- *target1-N*: Specifies the target location of the file. The value cannot be a directory.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Usage Notes

Note the following information when using this command:

- The source and target locations must reside in the same Exascale vault.
- You can use a wildcard (`%`) in the source location to specify multiple source files, in which case the corresponding target location must also contain a matching wildcard. For example, see Example 7-15.

Examples

Example 7-15 Move Multiple Files on Exascale Storage

The following example uses the wildcard character (`%`) to move multiple files from `@MYDATA/location1/prefix1%` to `@MYDATA/location2/anotherlocation/prefix2%`. Using the example command, a file named `@MYDATA/location1/prefix1mydata` would be moved to `@MYDATA/location2/anotherlocation/prefix2mydata`.

```
$ xsh mv @MYDATA/location1/prefix1% @MYDATA/location2/anotherlocation/prefix2%
```

7.2.11 rm

Remove files.

Syntax

```
rm [ -f ] [ -i ] [ -v ] [{ -w | --wallet } wallet-location ]
   [{ -T | --trace } trace-level ] filename
```

Command Options

The options for the `rm` command are:

- *filename*: Specifies the files being removed.

  You can use a wildcard (`%`) in the *filename* to specify multiple files.
- `-f`: Forces removal of open files.
- `-i`: Interactively prompt for confirmation before deleting a file.
- `-v`: Print the names of the files being deleted.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Examples

Example 7-16 Delete an Exascale file

The following example deletes `@MYDATA/myfile`.

```
$ xsh rm @MYDATA/myfile
```

7.2.12 scrub

On-demand file scrubbing for logical consistency.

Syntax

```
scrub [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]
   [ -a ] [ -i interval] [ -n requests] [ -r count]
   [ -f offset] [ -t offset] [ -s ] [ -v[v[v]] ] filename
```

Command Options

The options for the `scrub` command are:

- *filename*: Specifies the files being scrubbed.

  You can use a wildcard (`%`) in the *filename* to specify multiple files.

- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

- `-a`: Specifies that the scrub operation performs HARD checks on all regions, regardless of the file type or mirror status.
- `-i`: Specifies the interval (in seconds) to wait before re-scrubbing a region.
- `-n`: Specifies the queue depth for the scrubbing operation. This option controls the number of concurrent read requests performed by the scrubbing operation. Larger values enable faster scrubbing but consume more resources, which may affect other workloads.
- `-r`: Specifies the number of re-scrubbing attempts permitted while awaiting concurrent writes on the region. After the specified number of re-scrubbing attempts is exhausted, the corresponding region is reported in the command output and the scrubbing operation continues.
- `-f`: Enables scrubbing on part of a file by specifying the file offset that scrubbing starts from. The offset value is specified in megabytes from the start of the file.
- `-t`: Enables scrubbing on part of a file by specifying the file offset that scrubbing operates to. The offset value is specified in megabytes from the start of the file.
- `-s`: Specifies that when considering multiple sets of data as the source for a repair, the scrub operation should use the highest system change number (SCN) as the repair source.
- `-v[v[v]]`: Increases the verbosity of the command output:
  - `-v`: Generates detailed output.
  - `-vv`: Generates more detailed output.
  - `-vvv`: Generates the most detailed output.

Examples

Example 7-17 Scrub an Exascale file

The following example performs logical scrubbing on `@MYDATA/myfile`.

```
$ xsh scrub @MYDATA/myfile
```

7.2.13 snap

Snapshot files.

Syntax

```
snap [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]
     source1 target1 [ sourceN targetN ]...
```

Command Options

The options for the `snap` command are:

- *source1-N*: Specifies the snapshot source file(s).
- *target1-N*: Specifies the location of the snapshot(s).
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Usage Notes

Note the following information when using this command:

- You can use a wildcard (`%`) in the *source* to specify multiple source files, in which case the corresponding *target* must also contain a matching wildcard.
- All files in a snapshot operation must be in the same vault.
- Multiple *source* and *target* pairs are permitted. In this case, the source file specifications are considered in order, and only the first match is used.
- All snapshots created in the same operation are point-in-time consistent.

7.2.14 strings

Display strings of printable characters in files.

Syntax

```
strings [ -n num ] [ --aio=naio ] [{ -w | --wallet } wallet-location ]
        [{ -T | --trace } trace-level ] filename
```

Command Options

The options for the `strings` command are:

- *filename*: Specifies the file being processed.
- `-n`: Optionally specifies the minimum output string length. The default value is 4.
- `--aio`: Optionally specifies the number of async I/Os to use. The default value is 4.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Examples

Example 7-18 Display printable strings in an Exascale file

The following example displays printable strings contained in `@MYDATA/myfile` to standard output.

```
$ xsh strings @MYDATA/myfile
```

7.2.15 template

Display vault-level template attributes.

Syntax

```
template [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]
        @vault templatename
```

Command Options

The options for the `template` command are:

- *@vault*: Specifies the vault that contains the template being displayed. The vault name must include the leading at sign (`@`) character.
- *template*: Specifies the template being displayed.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Otherwise, the trace file is written to:

`/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Examples

Example 7-19 Display information about an Exascale template

The following example displays information about `mytemplate`, which is contained in `@MYDATA`.

```
$ xsh template @MYDATA mytemplate
```

7.2.16 touch

Create a file or modify attributes of an existing file.

Syntax

```
touch [{ -w | --wallet } wallet-location ]
      [ -s num | --size=num ]
      [ -t ftype | --type=ftype ]
      [ -r { true | false } | --read-only={ true | false }]
      [{ -T | --trace } trace-level ] filename
```

Command Options

The options for the `touch` command are:

- *filename*: Specifies the file being created or modified.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-s`, `--size`: Optionally specifies the size of the file in bytes.

  The file size can be also specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
- `-t`, `--ftype`: Optionally specifies the Oracle Database file type. The valid values are:
  - `ctrl` - control file
  - `data` - data file
  - `olog` - on-line log file
  - `alog` - archived log file
  - `temp` - temporary sort file
  - `init` - initialization parameter file
  - `pswd` - password file
  - `flog` - flashback log file
  - `ctrk` - change tracking file
- `-r`, `--read-only`: Optionally specifies that the file is read-only. When set to `true`, the file contents and metadata cannot be modified.

- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

Examples

Example 7-20 Create an Exascale file

Assuming that `@DATA/myfile` does not already exist, the following example creates a 1 gigabyte file at `@DATA/myfile`.

```
$ xsh touch --size=1G @DATA/myfile
```

If the file (`@DATA/myfile`) already existed, the command would set the size of the preexisting file to 1 gigabyte.

7.2.17 version

Print XSH version information to standard output.

Syntax

```
version
```

7.2.18 xattr

List and modify extended attributes for files and vaults.

Syntax

```
xattr { -a | --add } name xname xval
      [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]


xattr { -d | --delete } name xname
      [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]


xattr { -l | --list } name [ -v | --verbose ] [ -u | --unlimited ]
      [{ -w | --wallet } wallet-location ] [{ -T | --trace } trace-level ]
```

Command Options

The options for the `xattr` command are:

- *name*: Specifies the file or vault that is the subject of the command.

- *xname*: Specifies the name of the extended attribute that is the subject of the command.
- *xval*: Specifies the value of the extended attribute.
- `-a`, `--add`: Adds (or updates) an extended attribute to the specified file or vault.
- `-d`, `--delete`: Deletes an extended attribute from the specified file or vault.
- `-l`, `--list`: Displays extended attributes associated with the specified file or vault.
- `-v`, `--verbose`: Displays extended attributes with size information.
- `-u`, `--unlimited`: Displays complete extended attribute values. If not specified, the output of extended attribute values is limited to 80 characters.
- `-w`, `--wallet`: Optionally specifies the path to the Exascale wallet directory.
- `-T`, `--trace`: Optionally enables tracing and sets the trace level to `1` (minimum tracing), `2` (medium tracing), or `3` (maximum tracing).

  If the `$ADR_BASE` environment variable is set, the trace file is written to:

  `$ADR_BASE/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

  Otherwise, the trace file is written to:

  `/tmp/diag/EXC/xsh_<username>/<hostname>/trace/xsh_<date>.trc`

8
Using Exascale Utility Programs

This chapter describes how to use various utility programs included with Exascale:

- edvmkvol
  Runs ESCLI commands to make an Exascale volume and associated EDV attachment.
- edvrmvol
  Runs ESCLI commands to remove an EDV attachment and associated Exascale volume previously created using the `edvmkvol` command.

8.1 edvmkvol

Runs ESCLI commands to make an Exascale volume and associated EDV attachment.

Purpose

The `edvmkvol` command provides a simple interface to create an Exascale volume and an associated Exascale Direct Volume (EDV) attachment. The command performs the ESCLI `mkvolume` and `mkvolumeattachment` commands.

Syntax

```
$ edvmkvol volume-name vault-name volume-size [ --local | --cluster ] [ --verbose ]
```

Command Options

The options for the `edvmkvol` command are:

- *volume-name*: Specifies the name of the volume. The value is also used as the device name for the volume attachment.
- *vault-name*: Specifies the vault that the volume is created in.
- *volume-size*: Specifies the size of the volume. The *volume-size* can be specified using suffixes `K`, `KB`, `M`, `MB`, `G`, `GB`, `T`, `TB`. The suffix is not case-sensitive.
- `--local`: Specifies the creation of a node-specific EDV attachment residing on the server running the `edvmkvol` command.
- `--cluster`: Specifies the creation of a cluster-wide EDV attachment on the Oracle Grid Infrastructure (GI) cluster that contains the server running the `edvmkvol` command.
- `--verbose`: Instructs the command to display verbose output, including the underlying ESCLI commands and their output.

Usage Notes

- The command must run on an EDV client node where the attachment will reside.
- The command uses the user credentials associated with the first Exascale user wallet available in the following search path:

1. `$OSSCONF/eswallet`
2. `$ORACLE_BASE/admin/eswallet`
3. `/etc/oracle/cell/network-config/eswallet`

- If neither of the `--local` or `--cluster` options are specified, then `--cluster` is the default option when the host running the command contains an active Oracle Grid Infrastructure (GI) cluster. Otherwise, `--local` is the default option.

Example 8-1 Create a Volume and Cluster-Wide EDV Attachment

The example shows a simple command to create a volume and cluster-wide EDV attachment. In the example, the volume is named `myvol1` and is created in the vault named `testVault`. The volume size is 1 TB.

```
$ edvmkvol myvol1 testVault 1T --cluster

Created volume with id 2:1312220a4a9841a1be28a6c60762702a
Created edv attachment with id 1:57172474ab234589a411178f92e4acc8
Created device: /dev/exc/myvol1
```

Example 8-2 Create a Volume and Cluster-Wide EDV Attachment with Verbose Output

The example shows a command to create a volume and cluster-wide EDV attachment with verbose command output. The example is the same is the previous example, except that the command is also directed to display verbose output.

```
$ edvmkvol myvol1 testVault 1T --cluster --verbose

edvutil lsinitiator
id:                                 b68b97b1-ab59-6a6e-b68b-97b1ab596a6e
hostName:                           db01vm01
state:                              ONLINE
giClusterID:                        deadbeef-badc-0fee-dead-beefbadc0fee
giClusterName:                      edvTestCluster
EDV Driver Base Version Info:
  EDV Driver Version                25.1.0.0.0.241108
EDV Driver Online Patch Version Info:
  EDV Online Patch Driver Version: None

ESCLI> mkvolume 1T --vault testVault --attributes name=myvol1
Created volume with id 2:1312220a4a9841a1be28a6c60762702a

ESCLI> mkvolumeattachment 2:1312220a4a9841a1be28a6c60762702a myvol1 --
giClusterId deadbeef-badc-0fee-dead-beefbadc0fee
Created edv attachment with id 1:57172474ab234589a411178f92e4acc8
Created device: /dev/exc/myvol1
```

Related Topics

- mkvolume
  Create a volume or clone a volume snapshot.
- mkvolumeattachment
  Create an attachment for an Exascale volume.

- edvrmvol
  Runs ESCLI commands to remove an EDV attachment and associated Exascale volume previously created using the `edvmkvol` command.

8.2 edvrmvol

Runs ESCLI commands to remove an EDV attachment and associated Exascale volume previously created using the `edvmkvol` command.

Purpose

The `edvrmvol` command provides a simple interface to remove an Exascale Direct Volume (EDV) attachment and associated Exascale volume that were created using the `edvmkvol` command. The command performs the ESCLI `rmvolumeattachment` and `rmvolume` commands.

Syntax

```
$ edvrmvol volume-name [ --verbose ]
```

Command Options

The options for the `edvrmvol` command are:

- *volume-name*: Specifies the name of the volume being removed. This can be either the name specified in the `edvmkvol` command (for example, `myvol1`), or the device path (for example, `/dev/exc/myvol1`).
- `--verbose`: Instructs the command to display verbose output, including the underlying ESCLI commands and their output.

Usage Notes

- The command uses the user credentials associated with the first Exascale user wallet available in the following search path:
  1. `$OSSCONF/eswallet`
  2. `$ORACLE_BASE/admin/eswallet`
  3. `/etc/oracle/cell/network-config/eswallet`

Example 8-3 Remove an EDV Attachment and Associated Volume

The example shows a simple command to remove an EDV attachment and associated volume. In the example, the volume is named `myvol1`.

```
$ edvrmvol myvol1

Removed edv attachment with id 1:57172474ab234589a411178f92e4acc8
Removed volume with id 2:1312220a4a9841a1be28a6c60762702a
Removed device: /dev/exc/myvol1
```

Example 8-4 Remove an EDV Attachment and Associated Volume with Verbose Output

The example shows a command to remove an EDV attachment and associated volume with verbose output. In the example, the volume is identified using the EDV device name `/dev/exc/myvol1`.

```
$ edvrmvol /dev/exc/myvol1 --verbose

ESCLI>lsvolumeattachment --filter deviceName=myvol1
id                                  volume
deviceName attachTime
1:70b18560f4714f178d80940dc643f453 2:87e95b8e9e4a4989a543c713769a867b
myvol1     2024-11-19T14:35:30+00:00

ESCLI>rmvolumeattachment 1:70b18560f4714f178d80940dc643f453
Removed edv attachment with id 1:70b18560f4714f178d80940dc643f453

ESCLI>rmvolume 2:87e95b8e9e4a4989a543c713769a867b
Removed volume with id 2:87e95b8e9e4a4989a543c713769a867b
```

Related Topics

- edvmkvol
  Runs ESCLI commands to make an Exascale volume and associated EDV attachment.
- rmvolumeattachment
  Delete a volume attachment.
- rmvolume
  Delete volumes.

A

Exascale-Specific Information in the Oracle Data Dictionary

A.1 Oracle Database Dictionary Views

The Oracle Database dictionary includes views that contain information about specific objects in Oracle Exadata Exascale.

A.1.1 V$EXA_FILE

The `V$EXA_FILE` view contains information about Oracle Database files stored in Exascale.

Table A-1 V$EXA_FILE Columns and Descriptions

Column	Datatype	Description
FULL_PATH	VARCHAR2(1024)	Full file path, including the vault name and file path inside the vault.
VAULT_NAME	VARCHAR2(256)	Name of the Exascale vault containing the file.
FILE_NAME	VARCHAR2(1024)	File path inside the vault.
REDUNDANCY	VARCHAR2(20)	File redundancy setting. Possible values are: • `normal`: Indicates 2 mirrored copies of the file data. • `high`: Indicates 3 mirrored copies of the file data.
CONTENT_TYPE	VARCHAR2(20)	File content type. Possible values are: • `DATA` • `RECO`
MEDIA_TYPE	VARCHAR2(20)	Storage media type used by the file. The supported media types are: • `HC`: Identifies high capacity storage media, which uses hard disk drives (HDDs). • `EF`: Identifies extreme flash storage media, which uses flash devices.

Table A-1 (Cont.) V$EXA_FILE Columns and Descriptions

Column	Datatype	Description
CREATE_TIME	DATE	File creation time.
FILE_TYPE	VARCHAR2(30)	Oracle Database file type.
SIZE_IN_BYTES	NUMBER	File size in bytes.
SPACE_USED	NUMBER	Storage space used by the file.
USED_BY	VARCHAR2(64)	Identifies the Oracle Grid Infrastructure cluster or Oracle Database associated with the file.
CON_ID	NUMBER	Oracle Database container ID.

A.1.2 V$EXA_TEMPLATE

The `V$EXA_TEMPLATE` view contains information about Exascale templates.

Table A-2 V$EXA_TEMPLATE Columns and Descriptions

Column	Datatype	Description
TEMPLATE_NAME	VARCHAR2(32)	Exascale template name.
VAULT_NAME	VARCHAR2(256)	Name of the Exascale vault containing the template.
REDUNDANCY	VARCHAR2(20)	File redundancy setting. Possible values are: • `normal`: Indicates 2 mirrored copies of the file data. • `high`: Indicates 3 mirrored copies of the file data.
CONTENT_TYPE	VARCHAR2(20)	File content type. Possible values are: • `DATA` • `RECO`
MEDIA_TYPE	VARCHAR2(20)	Storage media type used by the file. The supported media types are: • `HC`: Identifies high capacity storage media, which uses hard disk drives (HDDs). • `EF`: Identifies extreme flash storage media, which uses flash devices.
CON_ID	NUMBER	Oracle Database container ID.

A.1.3 V$EXA_VAULT

The `V$EXA_VAULT` view contains information about Exascale vaults.

Table A-3 V$EXA_VAULT Columns and Descriptions

Column	Datatype	Description
VAULT_NAME	VARCHAR2(256)	Exascale vault name.
CREATE_TIME	DATE	Vault creation time.
EF_SPACE_USED	NUMBER	Vault space used on extreme flash (`EF`) storage media.
EF_SPACE_PROV	NUMBER	Vault space provisioned on `EF` storage media.
EF_IOPS_PROV	NUMBER	Vault I/Os per second (IOPS) provisioned on `EF` storage media.
HC_SPACE_USED	NUMBER	Vault space used on high capacity (`HC`) storage media.
HC_SPACE_PROV	NUMBER	Vault space provisioned on `HC` storage media.
HC_IOPS_PROV	NUMBER	Vault I/Os per second (IOPS) provisioned on `HC` storage media.
FLASH_CACHE_PROV	NUMBER	Flash Cache space provisioned for the vault.
XRMEM_CACHE_PROV	NUMBER	Exadata RDMA Memory (XRMEM) Cache space provisioned for the vault.
CON_ID	NUMBER	Oracle Database container ID.