Purity//FA CLI Reference Guide

Version 6.8.5

PURESTORAGE®

Copyright Statement

Pure Storage, Inc. 2555 Augustine Drive, Santa Clara, CA 95054 *http://www.purestorage.com*

Direct comments to mailto: *DocumentFeedback@purestorage.com*.

Version 1

Table of Contents

Chapter 1
About this Guide

The Pure Storage® FlashArray User Guide is written for array administrators who view and manage the Pure Storage FlashArray storage system.

FlashArray arrays are administered through the Purity for FlashArray (Purity//FA) graphical user interface (GUI) or command line interface (CLI). Users should be familiar with system, storage, and networking concepts, and have a working knowledge of Windows or UNIX.

Organization of the Guide

The guide is organized into the following major sections:

CLI Overview

Describes the basics of the Purity CLI: command syntax and conventions, syntax help and man pages, and working with CLI output.

Purity//FA CLI Commands

Describes the purpose and usage of each CLI command, with descriptions, options, arguments, and examples.

Deprecated CLI Commands

Describes commands that were once in use, but have been removed from Purity.

CLI Getting Started

Describes typical administrative tasks such as creating and connecting volumes and hosts; resizing, renaming, destroying, and eradicating volumes; working with file systems, directory services, and snapshots; and monitoring array performance.

A Note on Format and Content

FlashArray technology is evolving rapidly. As with all advanced information technologies, once basic architecture is in place, implementation develops at different rates in its different facets, each preceded by feature-by-feature detailed design.

This edition of the guide describes the properties and behavior of arrays that run the Purity//FA 6.8.5 release. It may include information about planned capabilities whose external form has been specified at the time of the release. Such material is included to provide users of this release with information for design planning purposes, and is subject to change as new functionality is implemented. Material relating to not-yet-implemented functionality is identified as such in the text.

Related Documentation

Refer to the following related guides to learn more about the FlashArray:

- **Purity//FA Administration Guide.** The Purity//FA interface (GUI) is a graphical, browser-based interface used to query and administer the FlashArray. The Purity//FA Administration Guide describes using the GUI to administer, configure, and monitor the FlashArray and also discusses FlashArray storage concepts, features, and conventions.
- **Pure Storage REST API Guide.** The Pure Storage REpresentational State Transfer (REST) API uses HTTP requests to interact with the FlashArray resources. The Pure Storage REST API Guide provides an overview of the REST API and a list of all available resources.
- **Pure Storage SMI-S Provider Guide.** Purity//FA includes the Pure Storage Storage Management Initiative Specification (SMI-S) provider, which allows FlashArray administrators to manage the array using an SMI-S client over HTTPS. The Pure Storage SMI-S Provider Guide describes functionality the provider supports and information on connecting to the provider.
- **Third-party plugin guides.** Pure Storage packages and plug-ins extend the functionality of the FlashArray. Available packages and plug-ins include, but are not limited to, VSS Hardware Provider, FlashArray OpenStack Cinder Volume Driver, FlashArray Storage Replication Adapter, Management Plugin for vSphere, and vRealize Operations Management Pack.

All related guides are available on the Knowledge site at `https://support.purestorage.com`.

Contact Us

Pure Storage is always eager to hear from you.

Documentation Feedback

We welcome your feedback about Pure Storage documentation and encourage you to send your questions and comments to **`<documentfeedback@purestorage.com>`**.

Product Support

If you are a registered Pure Storage user, log in to the Pure Storage Technical Services website at **`https://support.purestorage.com`** to browse our knowledge base, view the status of your open support cases, and view the details of past support cases.

You can also contact Pure Storage Technical Services at **`<support@purestorage.com>`**.

General Feedback

For all other questions and comments about Pure Storage, including products, sales, service, and just about anything that interests you about data storage, email `<info@purestor-age.com>`.

Chapter 2
CLI Overview

The Purity//FA command line interface (CLI) is a non-graphical, command-driven interface used to query and administer the FlashArray. The Purity//FA CLI is comprised of built-in commands specific to the Purity//FA operating environment.

This chapter covers general Purity//FA CLI concepts and conventions.

CLI Command Syntax and Conventions

Purity//FA CLI commands have the general form:

```
command subcommand --options OBJECT-LIST
```

The parts of a command are:

COMMAND

Type of FlashArray object to be acted upon, prefixed by "pure". For example, the **`purevol`** command acts on Purity//FA virtual storage volumes.

Run the **`pureman`** command to see a list of Purity//FA CLI commands.

SUBCOMMAND

Action to be performed on the specified object. Most CLI subcommands are common to some or all object types.

For example, **`purehost list`** lists all hosts on the array, while **`purevol list`** lists all volumes on the array.

The following subcommands are common to most or all object types:

`create`

Creates and names one or more FlashArray objects.

`delete`

Deletes one or more specified objects.

`list`

Lists information about one or more objects. To list information for all objects, do not specify the object in the command.

`listobj`

Creates whitespace-separated lists of objects or attributes related to one or more objects.

For example, `purevol listobj --type host` creates a list of the hosts to which volumes have connections. The `listobj` subcommand is primarily used to create lists of object and attribute names for scripting purposes.

`rename`

Changes the name of the specified object. Purity//FA identifies the object name in administrative operations and displays. The new name is effective immediately and the old name is no longer recognized in Purity//FA GUI and CLI interactions. In the Purity//FA GUI, the new name appears upon page refresh. Hardware object names cannot be changed.

`setattr`

Changes the attribute values of the specified objects.

`OPTIONS`

Options that specify attribute values or modify the action performed by the subcommand.

For example, in the following command, the **`--addvollist`** option adds volumes `VOL1`, `VOL2`, and `VOL3` to protection group PGROUP1:

```
purepgroup setattr --addvollist VOL1,VOL2,VOL3 PGROUP1
```

Some options, such as **`--hostlist`**, inherently apply only to a single object. Other options, such as **`--size`**, can be set for multiple objects in a single command.

Some options can be multi-valued. For example, in the following command, the **`--hostlist`** option associates multiple hosts (`HOST1`, `HOST2`, and `HOST3`) with the host group `HGROUP1`.

```
purehgroup setattr --hostlist HOST1,HOST2,HOST3 HGROUP1
```

`Object-List`

Object or list of objects upon which the command is to be operated.

If a subcommand changes the object state, then at least one object must be specified. Examples of subcommands that change the object state include `create`, `delete`, and `setattr`. For example, `purehost create HOST1` creates host `HOST1`. In the command synopses, OBJECT specifications that are not enclosed in square brackets (for example, "***HOST***") represent ones that are required.

Passive subcommands, such as `list`, which do not change object state, do not require object specification. Leaving out the object is equivalent to specifying all objects of the type. For example, `purevol list` with no volumes specified displays information about all volumes in an array. In the command synopses, OBJECT specifications enclosed in square brackets (for example, "***[HOST]***") represent ones that are optional.

Most subcommands act on a single object. For example, in the following command, the `setattr` subcommand can only be run on a single host group to change the attributes of that host group.

```
purehgroup setattr --addhostlist HOST1 HGROUP1
```

Certain subcommands can operate on multiple objects. For example, the following command connects volume `VOL1` to host groups `HGROUP1` and `HGROUP2`.

```
purehgroup connect --vol VOL1 HGROUP1 HGROUP2
```

In the command synopses, OBJECT specifications that are followed by ellipses (for example, ***HGROUP*...**) indicate that multiple objects can be entered.

The following list describes the conventions used in CLI help documentation:

- Text in fixed-width (Courier) font must be entered exactly as shown. For example, `purehost list`.
- Text not enclosed in brackets represents mandatory text.
- Text inside square brackets ("[ ]") represents optional items. Do not type the brackets.
- Text inside curly braces ("{ }") represents text, where one (and only one) item must be specified. Do not type the braces.
- The vertical bar (|) separates mutually exclusive items.
- Uppercase italic text represents entered text whose value is based on the nature of the subcommand or option. For example, `--size` ***SIZE***, where SIZE represents the value to be entered, such as `100g`.

CLI Output

Various Purity//FA CLI subcommands, including `list` and `monitor`, generate list outputs. With the ability to create and manage hundreds of volumes and snapshots, list outputs can become very long.

The Purity//FA CLI includes options to control the way a list output is displayed and formatted. The CLI also includes sort, filter, and limit options to control the results you see and the order in which you see them.

Interactive Paging

Pagination divides a large output into discrete pages. Pagination is disabled by default and is only in effect if the **--page** option is specified and the number of lines in the list output exceeds the size of the window.

To interactively move through a paginated list output:

- Press the [**Right Arrow**] key or space bar to move to the next page.
- Press the [**Left Arrow**] key to move to the previous page.

To quit interactive paging and exit the list view, press **q** .

With pagination, each page of the CLI list output begins with the column titles. To suppress the column titles, run the command with the **--notitle** option.

Using Wildcards

When performing a list or monitor operation, include the asterisk in the name argument to expand the list results. For example, the **purehost list *cat*** command displays a list of all hosts that contain "cat", such as `cat`, `catnap`, `happycats`, and `lolcat`. If the asterisks were not included, only the host named `cat` would be returned. As another example, the **purevol list *vol*** command displays a list of all volumes that contain "vol", including ones that begin or end with "vol".

```
$ purevol list *vol*
Name      Size   Source   Created
Myvol     100G   -        2016-04-11 10:18:07 PDT
MyVol-01  100G   -        2016-04-11 10:18:07 PDT
vol       1G     -        2016-04-11 11:19:19 PDT
vol01     100G   -        2016-04-11 10:17:23 PDT
Volume    100G   -        2016-04-11 10:17:23 PDT
```

If Purity//FA cannot find matches for the wildcard argument specified, an error message appears.

Asterisks are also allowed in Purity//FA CLI options that take a list of object names. For example, the **purevol list --snap --pgrouplist *pg*** command displays a list of all volume snapshots that are created as part of a protection group snapshot with "pg" in the protection group name. As another example, the following command displays a list of volume snapshots for all volumes that end with "01" and are created as part of a protection group snapshot with "pg" in the protection group name.

```
$ purevol list --snap --pgrouplist *pg* *01
Name                    Size  Source  Created
pgroup01.001.vol01      100G  vol01   2016-04-13 11:44:46 PDT
pgroup01.123.vol01      100G  vol01   2016-04-13 11:44:46 PDT
pgroup01.abc.vol01      100G  vol01   2016-04-13 11:44:46 PDT
pgroup01.backup.vol01   100G  vol01   2016-04-13 11:44:45 PDT
pgroup01.snap001.vol01  100G  vol01   2016-04-13 11:44:45 PDT
pgroup01.snap002.vol01  100G  vol01   2016-04-13 11:44:42 PDT
pgroup01.suffix.vol01   100G  vol01   2016-04-13 11:44:46 PDT
```

Formatting

Format options control the way list outputs are displayed. The following format options are common to most `list` and `monitor` subcommands:

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Here is a comparison between the `purehost list` output with formatted column titles and data, and the same output with unformatted column titles and data:

```
$ purehost list --space
Name  Size  Thin Provisioning   Data Reduction ...
H001  60T   57%                 1.5 to 1 ...


$ purehost list --space --raw
name  size            thin_provisioning   data_reduction ...
H001  65970697666560  0.5712341           1.5123 ...
```

Sorting

When running a list or monitor operation, include the **`--sort`** option to sort a column of the output in ascending or descending order.

The sort option adheres to the following syntax:

`--sort SORT`

`SORT` represents a comma-separated list of columns to sort. Use the unformatted title name (**`--raw`**) of the column to represent each column listed. To sort a column in descending order, append the minus (**`-`**) sign to the column name.

For example, run the following commands to get the unformatted column titles in the **`purehost list`** output, and then sort the same output by host group in descending order:

```
$ purehost list --raw
name                        ...  hgroup
ESXi-GRP-Cluster02-H0001    ...  ESXi-GRP-Cluster02-HG003
ESXi-GRP-Cluster02-H0002    ...  ESXi-GRP-Cluster02-HG003
ESXi-IT-Cluster01-H0001     ...  ESXi-IT-Cluster01-HG001
ESXi-IT-Cluster02-H0001     ...  ESXi-IT-Cluster02-HG002
ESXi-STG-Cluster03-H0001    ...  ESXi-STG-Cluster03-HG005
```

```
$ purehost list --sort hgroup-
Name                        ...  Host Group
ESXi-STG-Cluster03-H0001    ...  ESXi-STG-Cluster03-HG005
ESXi-IT-Cluster02-H0001     ...  ESXi-IT-Cluster02-HG002
ESXi-IT-Cluster01-H0001     ...  ESXi-IT-Cluster01-HG001
ESXi-GRP-Cluster02-H0001    ...  ESXi-GRP-Cluster02-HG003
ESXi-GRP-Cluster02-H0002    ...  ESXi-GRP-Cluster02-HG003
```

If multiple columns are specified, the output is sorted by the order of the columns listed.

If a sorted column contains non-unique values and a secondary sort argument is not specified, Purity//FA automatically performs a secondary sort using the default sort criteria (typically by **`Name`**).

Examples

Example 1: Display a list of volumes sorted in descending order by physical space occupied.

```
$ purevol list --space --sort "total-"
```

Example 2: Display a list of protection groups sorted in ascending order by source array name.

```
$ purepgroup list --sort "source"
```

Example 3: Display a list of volumes that are sorted in descending order by volume size. If any of volumes are identical in size, sort those volumes in descending order by physical space occupied.

```
$ purevol list --space --sort size-,total-
```

Filtering

When running a list or monitor operation, include the **`--filter`** option to narrow the results of the output to only the rows that meet the filter criteria. The following commands support the **`--filter`** option: puredir, purefs, purehgroup, purehost, purepgroup, purepolicy, pureport, and purevol. Filtering can be performed on any column of the list output.

Filtering with Operators

When filtering with operators, use the following syntax:

`--filter "RAW_TITLE OPERATOR VALUE"`

`RAW_TITLE` represents the unformatted title name of the column by which to filter. Run the list or monitor operation with the **`--raw`** option to get the unformatted title name.

`OPERATOR` represents the type of filter match (**`=`**, **`!=`**, **`<`**, **`>`**, **`<=`**, or **`>=`**) used to compare RAW_ TITLE to VALUE.

`VALUE` represents the value (number, date, or string) that determines the results to be included in the list output. Literal strings must be wrapped in quotes.

Filtering supports the following operators:

=	Equals
!=	Does not equal
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Filtering supports the asterisk wildcard character. For example, the value "**`*esx`***" in the **`purepgroup list --filter "hosts=*esx*"`** command displays a list of all protection groups with hosts members that contain "esx" in the host name.

The AND and OR operators can be used to further refine your filter. The AND operator displays only the results that meet all of the filter criteria in the command. The OR operator displays the results that meet at least one of the filter criteria in the command.

Examples

Example 1: Display a list of protection groups configured on source array **`pure-001`**.

```
$ purepgroup list --filter "source = 'pure-001'"
```

Example 2: Display a list of volumes that were created on or before **`2016-05-23 13:09:39`**.

```
$ purevol list --filter "created <= '2016-05-23 13:09:39 PDT'"
```

Example 3: Display a list of volume snapshots that are greater than 2 gigabytes in size.

```
$ purevol list --space --snap --filter "snapshots > '2G'"
```

Example 4: Display a list of volumes that are currently inactive.

```
$ purevol monitor --filter "output_per_sec=0 and input_per_sec=0"
```

Example 5: Display a list of volumes that begin with "vol10" or are greater than 100 gigabytes in size.

```
$ purevol list --filter "name = 'vol10*' or size > '100G'"
```

Filtering with Functions

The filter option supports the CONTAINS and NOT functions.

`--filter FUNCTION(PARAMETERS)`

Function	Description
`contains(raw_title,string)`	Contains the enclosed string. Takes exactly two parameters, where **`raw_title`** represents the unformatted title name of the column by which to filter and **`string`** represents the string to search within the column. Run the list or monitor operation with the **`--raw`** option to get the unformatted title name.
`not(expression)`	Inverse of the enclosed expression.

Examples

Example 1: Get a list of all volumes that include "cluster03" in the volume name.

```
$ purevol list --filter "contains(name, 'cluster03')"
```

Example 2: Get a list of all hosts that are associated with host groups with "PROD" in the host group name.

```
$ purehost list --filter "not(contains(hgroup, 'PROD'))"
```

Example 3: Get a list of all hosts that are not associated with host groups with "PROD" in the host group name.

```
$ purehost list --filter "not(contains(hgroup, 'GRP'))"
```

Existence Checks

The Purity//FA CLI supports existence checks. For example, the `purevol list --filter "name"` command checks to see if the "name" column exists in the volume list output.

Examples

Example 1: Get a list of all hosts associated with a host group.

```
$ purehost list --filter "hgroup"
```

Example 2: Get a list of all volumes that were not created from another source.

```
$ purevol list --filter "not(source)"
```

Example 3: Get a list of all hosts that are not associated with Fibre Channel WWNs.

```
$ purehost list --filter "not(wwn)"
```

Setting Limits

When running a list or monitor operation, include the `--limit` option to restrict the result size to the limit specified. The following commands support the `--limit` option: puredir, purefs, pure-hgroup, purehost, purepgroup, purepolicy, pureport, and purevol.

The limit option adheres to the following syntax:

`--limit ROWS`

`ROWS` represents the maximum number of rows to return.

For example, run the following command to list only the first 5 rows of the `purevol list` output:

```
$ purevol list --limit 5
Name                   Size  Source  Created     Serial
ESXi-Cluster01-vol001  100G  -       2016-05-23 13:09:40 PDT B143778B8ACE487B00011021
ESXi-Cluster01-vol002  100G  -       2016-05-23 13:09:40 PDT B143778B8ACE487B0001101D
ESXi-Cluster01-vol003  100G  -       2016-05-23 13:09:40 PDT B143778B8ACE487B00011024
ESXi-Cluster01-vol004  100G  -       2016-05-23 13:09:40 PDT B143778B8ACE487B00011018
ESXi-Cluster01-vol005  100G  -       2016-05-23 13:09:40 PDT B143778B8ACE487B00011025
```

Combining Sorting, Filtering, and Limit Options

The `--sort`, `--filter`, and `--limit` options can all be combined together.

Examples

Example 1: Display the top 10 volumes that occupy the largest amount of physical space and include "esx" in the volume name.

```
$ purevol list --space --sort "total-" --limit 10 --filter "name = '*esx*'"
```

Example 2: Display the top 10 volumes that have the largest physical space occupied by data unique to one or more snapshots.

```
purevol list --space --sort "snapshots-" --limit 10
```

Example 3: Display the top 10 volumes with the highest data reduction ratios.

```
$ purevol list --space --sort "data_reduction-" --limit 10
```

Example 4: Display the top 10 volumes that use the most read bandwidth.

```
$ purevol monitor --sort "output_per_sec-" --limit 10
```

Example 5: Display the top 10 volumes with the largest provisioned sizes and have shared connections to host groups with "PROD" in the host group name.

```
$ purevol list --connect --filter "hgroup = '*PROD*'" --sort size- --limit 10
```

CLI Login

Log in to the Purity//FA CLI to query and administer the FlashArray. Logging in to the Purity//FA CLI requires a virtual IP address or fully-qualified domain name (FQDN) and a login username and password; this information is determined during the FlashArray installation. The CLI has been tested with the following remote access packages:

- SSH (all common UNIX and Linux distributions)
- PuTTY

Logging in to the Purity//FA CLI

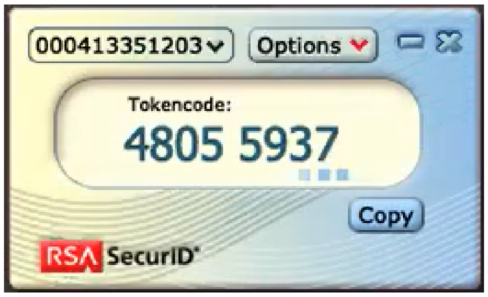
To log in to the Purity//FA CLI, select one of the following two options:

- **UNIX.** Start a secure shell (SSH) session, and then connect to the array using the login username and password obtained during the FlashArray installation.
- **Windows.** Start a remote terminal emulator (such as PuTTY), and then connect to the array using the login username and password obtained during the FlashArray installation.

Logging in to the Purity//FA CLI with Multi-factor Authentication

When multi-factor authentication is enabled on the array, instead of a password, you supply a passcode based on an RSA SecurID® tokencode. See Figure 2-1.

Figure 2-1. **Example RSA SecurID Tokencode**



Contact your RSA SecurID administrators for your organization's RSA instructions.

Follow these steps to log in when multi-factor authentication is enabled on the array:

1. The following notification appears (assuming user `user2` is logging in):

```
Password required for user2.
```

2. Hit **Enter** to proceed.

   The `Password:` prompt appears.

```
Password required for user2.
```

```
Password:
```

Despite the wording (`Password:`), enter your passcode at this prompt and hit **Enter**.

Logging out of the Purity//FA CLI

Type `exit` or `logout` to log out of Purity//FA and exit the shell or terminal emulator.

CLI Help

The Purity//FA CLI includes documentation for how to invoke and use each Purity//FA CLI command. Two types of interactive help are available through the Purity//FA CLI:

- The built-in Purity//FA CLI command help facility provides brief descriptions and usage information for each Purity//FA CLI command, subcommand, and option.
- The Purity//FA CLI man pages (or manual pages) is a more formal version of documentation that provides detailed information for each Purity//FA CLI command.

Run the `pureman` command to see a list of Purity//FA CLI commands.

Purity//FA CLI Command Help

CLI command help is available at both the command and subcommand levels. To use it, type the Purity//FA CLI command or subcommand followed by `-h` or `--help`. Type the Purity//FA CLI command with the `-h` or `--help` switch to display usage information, supported syntax, and a list of subcommands for the specified command.

```
COMMAND -h
```

For example (truncated for brevity),

```
$ purevol -h
usage: purevol [-h]
       {connect,copy,create,destroy,disconnect,eradicate,
list,listobj,monitor,recover,rename,setattr,snap,truncate}
       ...
positional arguments:
{connect,copy,create,destroy,disconnect,eradicate,
list,listobj,monitor,recover,rename,setattr,snap,truncate}
connect              connect one or more volumes to a host
copy                 copy a volume or snapshot to one or more volumes
create               create one or more volumes
destroy              destroy one or more volumes or snapshots
disconnect           disconnect one or more volumes from a host
eradicate            eradicate one or more volumes or snapshots
...
optional arguments:
-h, --help           show this help message and exit
```

Type the Purity//FA CLI subcommand with `-h` or `--help` switch to display usage information, supported syntax, and a list of options for the specified subcommand.

```
COMMAND SUBCOMMAND -h
```

For example,

```
$ purevol create -h
usage: purevol create [-h] --size SIZE VOL ...
positional arguments:
VOL            volume name

optional arguments:
-h, --help     show this help message and exit
--size SIZE    virtual size as perceived by hosts (e.g., 100M, 10G, 1T)
```

Purity//FA CLI Man Pages

CLI man pages display extensive help for each CLI command. To display the man page for a Purity//FA CLI command, run the `pureman` command with the Purity//FA CLI command or sub-command name.

```
pureman COMMAND
```

or

```
pureman COMMAND-SUBCOMMAND
```

For example (truncated for brevity),

```
$ pureman purevol
PUREVOL(1)                    Purity//FA CLI Man Pages                    PUREVOL(1)
NAME
purevol, purevol-copy, purevol-create, purevol-destroy,
purevol-eradicate
...
- manage the creation, naming, and destruction of Purity//FA
virtual storage volumes and snapshots of their contents,
as well as the reclamation of physical storage occupied
by the data in them
purevol-monitor - monitor volume I/O performance
SYNOPSIS
purevol copy [--overwrite] SOURCE TARGETVOL
purevol create --size SIZE VOL...
purevol destroy VOL...
purevol eradicate VOL...
...
ARGUMENTS
SOURCE
Volume or snapshot from where data is copied.
The data is copied to the TARGETVOL volume.
TARGETVOL
```

```
Volume to where data is copied.
The data is copied from the SOURCE volume or snapshot.
VOL
Volume to be created, destroyed, eradicated, recovered,
or snapped.
...
OPTIONS
--interval SECONDS (purevol monitor only)
Sets the number of seconds between displays of data.
At each interval, the system displays a point-in-time snapshot
of the performance data.
If omitted, the interval defaults to every 5 seconds.
--overwrite
Allows purevol copy to overwrite an existing volume.
Without this option, if the target volume already exists,
the command fails.
...
DESCRIPTIONS
This page describes management of volumes. The purevol create command...
EXAMPLES
Example 1
purevol create --size 100G vol1 vol2 vol3
Creates three volumes called vol1, vol2, and vol3, each
with a host-visible capacity of 100 gigabytes (10^2 * 2^30 bytes).
...
SEE ALSO
purevol-list(1), purevol-setattr(1), purehgroup(1),
purehgroup-connect(1), purehost(1), purehost-connect(1)
AUTHOR
Pure Storage Inc.
```

Chapter 3
Purity//FA CLI Commands

This chapter describes each of the Purity//FA CLI commands. The commands listed in this chapter administer and/or manage the FlashArray, with the exception of the following information-only command:

pureman

Describes the pureman command and displays a list of available Purity//FA commands.

puread

puread, puread-account, puread-account-create, puread-account-delete – manages the creation and deletion of Active Directory accounts for directory services

puread-account-list – displays information about Active Directory account attributes

Synopsis

puread account create --domain ***DOMAIN*** [--join-existing-account | --join-ou ***JOIN_OU***] [--tls {optional,required}] [--computer-name ***COMPUTER_NAME***] ***NAME***

puread account delete [--local-only] ***NAME...***

puread account list [--cli | --csv | --nvp] [--notitle | --page | --raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [***NAME...***]

puread account setattr [--tls {optional,required}] ***NAME***

Options

-h | --help

Can be used with any command or subcommand to display a brief syntax description.

--computer-name ***COMPUTER_NAME***

The AD account name that represents FlashArray file.

--domain ***DOMAIN***

The AD domain name.

--join-existing-account

Enable this option if the account already exists. The domain is searched for a pre-existing computer account to join and a new account will not be created within the domain. The “Join OU” option cannot be used when joining a pre-existing computer account.

`--join-ou` ***`JOIN_OU`***

Optional, when adding a new account. Specifies the organizational unit for the new account, in distinguished name format. For example, **`OU=Dev,OU=Sweden,DC=purestorage,DC=com`**. The **`DC=...`** components of the distinguished name can be optionally omitted. If the option is omitted, the organizational unit defaults to **`CN=Computers`**.

`--local-only`

Deletes only the Active Directory account on the array, without deleting the computer account in the Active Directory domain.

`--tls`

TLS mode for communication with domain controllers. Valid values are **`required`** and **`optional`**. If the value **`required`** is specified, this forces TLS communication with the domain controller. If the value **`optional`** is specified, this allows the use of non-TLS communication; however, TLS is still preferred. If not specified, this option defaults to **`required`**.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Arguments

NAME

The Active Directory member configuration name.

Description

Join the FlashArray to Active Directory (AD) to allow clients access to FlashArray file services using AD for authentication. Note that a Domain Name System (DNS) server must be accessible and the AD account (computer name) must be different from the array name. You can join an existing AD account or join and create a new account in one operation. If joining a pre-existing account, the account must have “Read all properties" and "Reset password" permissions. Use only one account to represent FlashArray file. If you have more than one file VIFs or more than one DNS aliases for file, then add these as separate ServicePrincipalNames (SPN) to the Computer Name in the AD.

The **`puread account create`** command creates an Active Directory (AD) member for directory services, and joins the array to the domain DNS specified with the **`--domain`** option and argument. Valid options include **`--join-ou`** to specify the organizational unit for the new account, and **`--computer-name`** to specify the machine account name.

You can join Active Directory using an existing computer account. The requirements for joining an existing AD account include a domain name, security credentials, and a computer account name. When specifying the **`--join-existing-account`** option, the **`puread account create`** command searches within the domain for an existing computer account. When the array joins the domain using an existing computer account, the organizational unit, and encryption type are inherited from the pre-existing computer account. If the account is not found, the prospective account created will fail.

The **`puread account setattr`** command allows you to change the TLS mode for a connected Active Directory account. Valid values for **`--tls`** are **`required`** and **`optional`**.

To display information about AD attributes, use the **`puread account list`** command.

To delete an existing AD member, use the **`puread account delete`** command, followed by one or more names of AD members to be deleted. This deletes the AD account both on the local array and the computer account in the AD domain. To delete the AD account on the local array only, use the **`--local-only`** option.

Examples

Example 1

```
puread account create --domain ad1.local file-fa-1
```

Creates and joins a new AD member named **`file-fa-1`** for file, on the domain named **`ad1.local`**.

Example 2

```
puread account list
```

Lists information about Active Directory attributes.

Example 3

```
puread account delete file-fa-1
```

Deletes the Active Directory member named **`file-fa-1`**.

Example 4

```
puread account create --domain ad1.local --join-existing-account array1
```

Joins Active Directory using an existing computer account.

See Also

pureds

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

pureadmin

pureadmin, pureadmin-add, pureadmin-create, pureadmin-delete, pureadmin-global, pureadmin-list, pureadmin-refresh, pureadmin-remove, pureadmin reset, pureadmin-setattr — management of administrative accounts

Synopsis

pureadmin add --access-policy ***ACCESS_POLICY...*** ***USER***

pureadmin create {--access-policy ***ACCESS_POLICY...*** | --api-token} [--timeout ***PERIOD***] ***USER...***

pureadmin delete [--api-token] ***USER...***

pureadmin global list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [***USER...***]

pureadmin global disable {--single-sign-on} {--active-management}

pureadmin global enable {--single-sign-on} {--active-management}

pureadmin global setattr [--min-password-length ***LENGTH***] [--ssh-required-authentication ***SSH_AUTH_METHODS***] --max-login-attempts ***ATTEMPT*** --lockout-duration ***DURATION*** --active-management-role ***ROLE***

pureadmin list [--api-token] [--expose] [--publickey] [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--lockout][***USER...***]

pureadmin refresh [--clear] [***USER...***]

pureadmin remove --access-policy ***ACCESS_POLICY...*** ***USER***

pureadmin reset [--lockout] [***USER...***]

pureadmin setattr {--password | --publickey} ***USER...***

Arguments

USER

User login name. Sometimes referred to as sAMAccountName.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--active-management-role` ***ROLE***

Sets the Active Management role to **`readonly`**, **`storage_admin`**, or **`array admin`**.

`--access-policy` ***ACCESS_POLICY***

With the **`pureadmin add`** and **`pureadmin create`** commands, attaches one or more management access policies to the user. An access policy defines a set of permissions and scope (in which array or realm) for those permissions. (See "Management Access Policies" on page 453 in the **`purepolicy`** chapter.) Specify multiple access policies in a comma-separated list.

`--api-token`

Displays users with API tokens, or generates an API token, granting the user access to the REST API. When listing users with API tokens, optionally include the **`--expose`** option to unmask the API token.

`--clear`

Clears the user permission cache.

`--expose`

Unmasks the current user's API token.

`--lockout`

Unlocks the specified user after the maximum allowed login attempts have failed. Run the **`pureadmin reset --lockout`** command to unlock the user. Only the array administrator can utilize this option. When used with **`pureadmin list --lockout`**, displays the current locked users.

`--lockout-duration` ***DURATION***

Sets the lockout duration after the user reaches the maximum allowed login attempts. The duration is specified as an integer, followed by one of the suffix letters **`s`** (seconds), **`m`** (minutes), **`h`** (hours), **`d`** (days), or **`w`** (weeks).

`--max-login-attempts` ***`ATTEMPT`***

Sets the maximum allowed login attempts before the user account is locked.

`--min-password-length` ***`LENGTH`***

Displays or sets the global minimum character limit for local account passwords. New passwords must be at least ***`LENGTH`*** characters long to be accepted. The minimum password length must be greater than 0 characters. Empty passwords are not allowed. The default value is 1 character. Minimum password length changes do not apply to existing passwords.

`--password`

Changes the password for the specified local user. The password is entered through interactive prompt. Local users can change their own passwords through the **`pureadmin setattr --password`** command. Only array administrators can change the password on behalf of other users. By default, the new password must be at least 1 character and cannot exceed 100 characters in length. If a global minimum password length has been specified, the new password must abide by that setting.

`--publickey`

Displays users with public keys, or sets their public keys for the specified user for SSH access. Only array administrators can change public keys on behalf of other users. If changing the public keys, the public keys are entered through an interactive prompt. If no users are specified, the public key change will be for the current user. When setting the public keys separate each one with a new line and at the end add a new line and press CTRL-D. Every time the public keys of a user are set they will override all of the user’s public keys. If you wish to list a user's public keys you must specify the user, only one user can be specified for the keys to be shown.

`--ssh-required-authentication`

Sets the global authentication methods that are required for all users on the array. Valid values will be taken in a list like array. A list containing more then one value will mean that the users require both factors for authentication (2 Factor Authentication). The valid authentication methods allowed are: **`default`**,**`password`**, **`key`**, and **`password,key`**. Any other combination will not be accepted. Only an Array Admin can set the global authentication methods.

An array is by default set to the **`default`** authentication method, this method means that the user can log into their array by a password, or by a publickey if a publickey is set. RSA SecurID is also only allowed to be set if the global authentication method is set to **`default`**.

When the global authentication method is set to **`password`**, this means that users will only be able to log in to their accounts with the use of their passwords, even if a public key is set. RSA SecurID will also not be able to be set if the global authentication method is set to **`password`**.

When the global authentication method is set to **`key`**, this means that users will only be able to log in to their accounts with the use of their private key. If a public key is not set and admin will have to set for the users account an admin will have to set the key themselves. This method of authentication cannot be set unless pureuser and the admin changing the authentication method have public keys set. RSA SecurID will also not be able to be set if the global authentication method is set to **`key`**.

When the global authentication method is set to **`password,key`**, this means that users will only be able to log in to their accounts after using their password and with the use of a valid private key. All users will require the two factors of authentication to access their accounts. If a public key is not set and admin will have to set for the users account an admin will have to set the key themselves. This method of authentication cannot be set unless pureuser and the admin changing the authentication method have public keys set. RSA SecurID will also not be able to be set if the global authentication method is set to **`password,key`**.

`--single-sign-on`

Enables or disables single sign-on (SSO) from Pure1 Manage and other cloud applications to the current array.

Enabling single sign-on gives LDAP users the ability to navigate seamlessly from cloud applications to the current array through a single login. Enabling and disabling single sign-on takes effect immediately. By default, single sign-on is not enabled.

`--timeout PERIOD`

Sets the validity period of the API token. The validity period is specified as an integer, followed by one of the suffix letters **`s`** (seconds), **`m`** (minutes), **`h`** (hours), **`d`** (days), or **`w`** (weeks).

Note: With the **`pureadmin create`** command, the `--role` option is deprecated and replaced by `--access-policy`.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Note on the Default Password

For security purposes, Pure Storage recommends that the password for the **`pureuser`** account be changed immediately upon first login.
Use the **`pureadmin setattr --password`** command, which prompts for the current password and a new password.

Description

The FlashArray array is delivered with a single administrative account named **`pureuser`**. The account is password protected and may alternatively be accessed using a public-private key pair.

The **`pureuser`** account is set to the array administrator role, which has array-wide permissions. The **`pureuser`** account cannot be deleted.

Users can be added to the array either locally by creating and configuring a local user directly on the array, or through Lightweight Directory Access Protocol (LDAP) by integrating the array with a directory service, such as Active Directory or OpenLDAP. Account information, such as group membership or password policy, for users that are added to the array through LDAP integration are managed through the directory. For more information about configuring directory services and LDAP integration, refer to pureds.

Local users can only be created by array administrators. The name of the local user must be unique to all other users, including both local users and LDAP users.

If an LDAP user appears with the same name as a local user, the local user always has priority.

The `pureadmin create` command creates a local user. The name of the new user must be between 1 and 32 characters (alphanumeric and '-') in length and begin and end with a letter or number. The name must include at least one letter or '-'. All letters must be in lowercase. Include the required `--access-policy` option to attach a management access policy to the user, thereby assigning a set of permissions and scope (in which array or realm) for those permissions. (See "Management Access Policies" on page 453 in the **`purepolicy`** chapter.)

To log in to the Purity//FA system, each user requires a login password. For local users, this password is manually assigned by the array administrator through interactive command prompt during user creation. Local users can change their own passwords through the **`pureadmin setattr --password`** command.

The `pureadmin create --api-token` command generates a REST API token, granting the specified user access to the REST API. The regenerated API token replaces any token that has already been created for the user. Users have permission to manage their own API tokens. By default, API tokens are created without expiry dates. To set an expiry date for an API token, include the `--timeout` option when creating the token. The validity period is specified as an integer, followed by one of the suffix letters **`s`** (seconds), **`m`** (minutes), **`h`** (hours), **`d`** (days), or **`w`** (weeks). When an API token expires and is therefore no longer valid, the user cannot access the REST API until the token is recreated.

The `pureadmin delete` command deletes a local user. Once deleted, the user is no longer able to access the array. Deleted users cannot be restored. To delete the API token of a user, include the `--api-token` command. Once the API token has been deleted, the user can no longer access the REST API. The API token can be recreated at any time.

The `pureadmin global` command displays and changes global administrative account configuration. The `pureadmin global setattr --min-password-length` command sets the minimum password length of all local account passwords. Existing passwords are not affected, but all future password assignments and changes must meet the new minimum password length requirement. Before the user account becomes locked, set the maximum login attempts by using the `pureadmin global setattr --max-login-attempts` command. The **`lockout-duration`** option sets the lockout duration after the maximum allowed login attempts have failed. The duration is specified as an integer, followed by one of the suffix letters **`s`** (seconds), **`m`** (minutes), **`h`** (hours), **`d`** (days), or **`w`** (weeks).

Once the user has failed the maximum allowed login attempts, run the **`pureadmin reset --lockout`** command to unlock the user. Only the array administrator can utilize this option.

The **pureadmin global enable --single-sign-on** command enables single sign-on on the current array. Enabling single sign-on gives LDAP users the ability to navigate seamlessly from Pure1 Manage to the current array through a single login. The **pureadmin global disable --single-sign-on** command disables single sign-on. Enabling and disabling single sign-on takes effect immediately. By default, single sign-on is not enabled. Enabling single sign-on is a two-step process: first, configure single sign-on and LDAP integration through Pure1 Manage, and second, enable single sign-on on the array through Purity//FA. For more information about SSO and LDAP integration with Pure1 Manage, refer to the Pure1 Manage - SSO Integration article on the Knowledge site at https://support.purestorage.com. The **pureadmin global list** command displays the global configuration.

Use the **pureadmin list** command to display a list of Purity user accounts and their attributes. The list displays the following types of users:

- **pureuser** administrative account.
- Local users that have been created on the array.
- LDAP users with a public key and/or API token. LDAP users that do not have a public key or API token do not appear in the list.

The Type column displays the way in which the user was added to the array, either as Local or LDAP. The Access Policy column displays the access policy or policies attached to each user. array. Use the **purepolicy management-access list** command to display the permission and scope for an access policy. (See "Management Access Policies" on page 453 in the **purepolicy** chapter.)

Include the **--api-token** option to display the API token details for each user. The Created column displays the date the API token was created while the Expires column displays the date the API token expires. When the API token expires and is therefore is no longer valid, the user cannot access the REST API until the API token has been recreated. A dash (-) in the Expires column indicates an API token with no expiry date, meaning the API token is valid until it is recreated or deleted. Combine the **--api-token** option with the **--expose** option to unmask the current user's API token. Include the **--publickey** option to determine which users have public keys configured.

Directory service enabled accounts are also subject to policy-based access control, but the permission level of those users correlate to the configured directory group(s) to which they are members. To prevent binding and querying the directory server too frequently, permissions are cached on the array. Run the **pureadmin refresh** command to refresh the cache entries for the specified user. Cache entries are also automatically updated for a user when starting a new session. Include the **--clear** option to empty the entire permissions cache for all users. After running the **pureadmin refresh --clear** command, the first action by each user causes a query to the directory service, both to confirm that the user has permission for that action and to refresh that user's permission cache entry. These queries to the directory service eventually refresh the permission cache entries for all active users.

Use the **`pureadmin add --access-policy`** command to attach one or more management access policies to an existing user. The following example attaches the `realm1_admin_policy` access policy to an existing user named `admin1`. First, the **`purepolicy management-access list realm1_admin_policy`** command shows the permissions and scope for the access policy. In this example, the `realm1_admin_policy` access policy grants `admin` permissions for the realm namded `realm1`.

```
$ purepolicy management-access list realm1_admin_policy
Name                 Type               Enabled  Role   Aggregation St.  Resource Name
realm1_admin_policy  management-access  True     admin  all-permissions  realm1
```

```
$ pureadmin add --access-policy realm1_admin_policy admin1
Name    Type         Access Policy
admin1  local-admin  realm1_admin_policy
```

With the `realm1_admin_policy` access policy attached, user `admin1` now has `admin` permissions for `realm1` in addition to any permissions `admin1` had previously. (See "Management Access Policies" on page 453 in the **`purepolicy`** chapter.)

Use the **`pureadmin remove --access-policy`** command to detach one or more access policies from a user.

The **`pureadmin setattr`** command changes the attributes of existing users.

Include the **`--password`** option to change the password for the specified local user. The password is entered through interactive prompt. Local users can change their own passwords. Array administrators can change the password on behalf of other users. By default, the new password must be between 1 and 100 characters in length. If a minimum password length has been specified, the new password must abide by that setting. Password management for directory service enabled accounts is performed in the directory.

Include the **`--publickey`** option to set the public key for the specified user for SSH access. The public key is entered through interactive prompt. If no users are specified, the public key change will be for the current user. A new public key is typically entered by copying a value from a key generation application running in a local window on the administrative workstation and pasting it into the administrative session window. Each public key must correspond to a private key in the account from which a session is being conducted. Public key access can be configured for any user on the array, local or LDAP.

Logging into the Purity//FA CLI with Yubikeys using PIV or FIDO2

Yubikey hardware can be used for authentication purposes in the FA. To do so you must generate a FIDO2 key pair using your security key, or a PIV certificate. If you went the way of a PIV certificate you must extract the public key from it.

From there the public key from the certificate or key pair must be copied and pasted into the users public keys, this can be done with the following commend (assuming user is the one who needs their public key set):

1. `pureadmin setattr --publickey user`
2. Enter in the users public keys and then press enter and CTRL-D

Now the user can log in using their YubiKey for passwordless authentication.

Exceptions

None.

Examples

Example 1

```
pureadmin list --publickey
```

Lists the current administrative accounts for which SSH key access has been configured.

Example 2

```
pureadmin create --access-policy storage_policy localuser1
Enter password:
Retype password:
```

Creates a local user named **`localuser1`** with the manually-created login password that was entered twice through interactive prompt. The password is not shown while typing. If the passwords do not match, the user is not created. The access policy named **`storage_policy`** is attached to the user **`localuser1`**.

Example 3

```
pureadmin create --api-token pureuser
```

Creates an API token for the pureuser administrative account to grant REST API access.

Example 4

```
pureadmin create --api-token --timeout 1h pureuser
```

Creates an API token for the pureuser administrative account to grant REST API access, and sets the API token to expire one hour after being created.

Example 5

```
pureadmin delete localuser1
```

Deletes the **`localuser1`** user from the system. The **`localuser1`** user is no longer able to access the array.

Example 6

```
pureadmin setattr --password pureuser
Enter old password:
Enter new password:
Retype new password:
```

Sets the password for local user **`pureuser`** to the new password. The old and new passwords are entered through interactive prompts. The password is not shown while typing.

Example 7

```
pureadmin setattr --publickey
Please enter key: followed by Enter and then Ctrl-D:
```

Indicates a request to change the public keys for the workstation account from which the session is being conducted. The public keys are entered through an interactive prompt. The public keys are typically copied from a key generation tool running in a local window on the administrative workstation. Each key should be separated by a new line.

Example 8

```
pureadmin refresh --clear
```

Clears the contents of the entire user permission cache.

Example 9

```
pureadmin global enable --single-sign-on
```

Enables single sign-on (SSO) from Pure1 Manage and other cloud applications to the current array. SSO and LDAP integration must already be configured through Pure1 Manage before single sign-on can be enabled on the array.

Example 10

```
pureadmin reset user1 --lockout
```

Unlocks the user after maximum allowed login attempts have failed.

Example 11

```
pureadmin global setattr --ssh-required-authentication password,key
```

Sets the global authentication method to `password,key`. Users will only be able to log in to their accounts after using their password and with the use of a valid private key.

See Also

pureds, purepolicy

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purealert

purealert-code, purealert-flag, purealert-list, purealert-rule, purealert-unflag, purealert-watcher – manages alert history and the list of designated email addresses to which Purity//FA sends alert messages when significant events occur in an array

Synopsis

```
purealert code list [--snooze] [--context REMOTES]

purealert code setattr [--reminder-frequency REMINDER_FREQUENCY] [--
snooze-period SNOOZE_PERIOD] [--context REMOTE]

purealert flag [--context REMOTE] ID...

purealert list [ --cli | --csv | --nvp] [--notitle] [--raw] [--filter
FILTER] [--page] [--page-with-token] [--token TOKEN] [--sort SORT] [--
events] [--flagged] [--context REMOTES] [ID...]

purealert rule create --parameter PARAMETER --value VALUE [--context
REMOTE] CODE

purealert rule delete --parameter PARAMETER [--context REMOTE] CODE

purealert rule list [--catalog] [--code CODE] [--context REMOTES]

purealert rule setattr --parameter PARAMETER --value VALUE [--context
REMOTE] CODE

purealert unflag [--context REMOTE] ID...

purealert watcher create [--context REMOTE] ADDRESS...

purealert watcher delete [--context REMOTE] ADDRESS...

purealert watcher disable [--context REMOTE] ADDRESS...

purealert watcher enable [--context REMOTE] ADDRESS...

purealert watcher list [ --cli | --csv | --nvp ] [--notitle] [--page]
[--raw] [--page-with-token] [--token TOKEN] [--context REMOTES]
ADDRESS...
```

```
purealert watcher listobj [--csv] [--type address] ADDRESS...
purealert watcher test [--context REMOTE] ADDRESS...
```

Arguments

ADDRESS
: Any valid email address.

CODE
: The specific alert code.

ID
: A unique, numeric ID assigned by the array to each alert.

FILTER
: The row numbers to be filtered.

PARAMETER
: The parameter used to adjust the rule.

SORT
: Ascending or descending.

VALUE
: The value the rule will be set to.

Options

`-h | --help`
: Can be used with any command or subcommand to display a brief syntax description.

`--catalog` (**`purealert rule list`**)
: Lists alerts and parameters available for customization.

`--context` ***REMOTE***

Used to specify the remote array on which a command is processed. ***REMOTE*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

If used with **`purealert list`**, `--context` can specify multiple arrays and ***REMOTE*** is a comma-separated list of array names.

`--events` (**`purealert list`**)

Displays a history of alert events.

`--filter` (**`purealert list`**)

Displays only the rows that meet the filter criteria specified.

`--flagged` (**`purealert list`**)

Displays only flagged alerts.

`--parameter` (**`purealert rule create, purealert rule delete, purealert rule setattr`**)

Sets the parameter for the custom rule.

`--reminder-frequency` (**`purealert code setattr`**)

Sets the frequency with which to send reminder email notifications about an alert code. Set to the default reminder frequency or use "" to clear. The period must be expressed as the number followed by a letter designating the unit of time (e.g. 15m, 1h, 1d). Accepted units of measurement are seconds, minutes, hours, and days (s,m,h, and d respectively).

`--snooze` (**`purealert code list`**)

Lists all the alert codes that currently have a snooze option set.

`--snooze-period` (**`purealert code setattr`**)

Sets the period for which to mute reminder email notifications about an alert code. The period must be expressed as the number followed by a letter designating the unit of time (e.g. 15m, 1h, 1d) to a maximum 30 days. Accepted units of measurement are seconds, minutes, hours, and days (s,m,h, and d respectively).

`--sort` (**`purealert list`**)

Sorts the list output in ascending or descending order by the column specified.

`--tag` (**`purealert list`**)

Displays tags.

`--type {address}` (**`purealert watcher listobj`**)

Outputs a whitespace-separated list containing the specified email addresses. If no addresses are specified, contains all addresses designated to receive alert messages, whether enabled or not.

`--value`(**`purealert rule create, purealert rule setattr`**)

The value to set the rule to.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--page-with-token`

Used to paginate output with a continuation token (**`--token TOKEN`**). Results are printed in `stdout` and the token is printed in `stderr`.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Description

Purity//FA generates log records called alerts when significant events occur within an array. Administrators can designate email addresses to which Purity//FA will send email messages when alert-generating events occur.

By default, the address flasharray-alerts@purestorage.com is designated to receive alerts on FlashArray systems. This address can be disabled (see below), temporarily suspending the transmission of alert messages to Pure Storage Technical Services, but cannot be deleted.

The **`purealert flag`** command identifies an alert as flagged. Alert messages are flagged by default and can be unflagged as necessary and when alerts are closed.

The **`purealert list`** command lists alerts on the array, including the alert ID, when it was created, severity, state, code, summary, and a description of the issue.

The **`purealert code list`** command provides a list of all the types of alerts that can fire on the array. **`purealert code list --snooze`** provides a list of all alert codes that currently have a snooze option set. You can set the snooze option with purealert code setattr --snooze-period. You can also change the frequency with which reminder email notifications are sent about a given alert with **`purealert code setattr --reminder-frequency`**.

Custom alert rules can be created using the **`purealert rule`** commands.

- **`purealert rule create`**: Creates a customized alert rule.
- **`purealert rule delete`**: Removes the specified alert rule.
- **`purealert rule list`**: Lists all existing configured alert rules.
- **`purealert rule setattr`**: Adds the attributes specified to the customized rule.

The **`purealert unflag`** command identifies an alert as unflagged. Unflagging an alert is equivalent to dismissing it from the Open Alerts panel in the GUI dashboard. The output of the **`purealert list --flagged`** does not show unflagged alerts.

The **`purealert watcher`** commands modify or list the email addresses to which alerts are sent and test an array's ability to send email to any email address (designated or not). The following subcommands are available:

- **`purealert watcher create`**: Designates any valid email address to receive Purity//FA alert messages. Up to 20 addresses can be designated in an array (19 in addition to the built-in flasharray-alerts@purestorage.com).
- **`purealert watcher delete`**: Removes email addresses from the list of designated addresses.
- **`purealert watcher disable`**: Disables the sending of alert messages to one or more designated email addresses. If no email address arguments are specified, use of this argument disables the sending of alert messages to all designated addresses, including the built-in flasharray-alerts@purestorage.com.
- **`purealert watcher enable`**: Enables the sending of alert messages to one or more designated email addresses. If no email address arguments are specified, use of this argument enables the sending of alert messages to all designated addresses, including the built-in flasharray-alerts@purestorage.com.

- `purealert watcher list`: Lists any or all designated email addresses and their states (enabled or disabled).
- `purealert watcher listobj`: Creates whitespace-separated (no formatting option specified) or comma-separated (`--csv` specified) lists of designated email addresses for scripting. If no addresses are specified, produces a list of all designated addresses. The same output list is produced whether or not the `--type` option is specified.
- `purealert watcher test`: Sends a test email to one or more email addresses.

The sending account name for Purity//FA email alert messages is the array name. This name and the sender domain for alert messages can be viewed and altered by the `purearray list` and `purearray setattr` commands respectively.

Examples

Example 1

```
purealert watcher test admin1@mydomain.com
```

Sends a test message to admin1@mydomain.com. After receipt of the message at the destination has been verified by external means, designates admin1@mydomain.com to receive Purity//FA alert messages. The `purealert watcher test` command sends test messages to all designated addresses, whether they are enabled or disabled. The only console response to the purealert watcher test subcommand is the next Purity//FA prompt.

Example 2

```
purealert watcher disable admin2@mydomain.com
purevol connect --host HOST1 VOL1 VOL2
purealert watcher enable admin2@mydomain.com
```

The `purealert watcher disable` command prevents Purity//FA from sending alert messages to admin2@mydomain.com, so that the account does not receive the messages generated when the private connections between HOST1 and VOL1 and HOST1 and VOL2 are established. The `purealert watcher enable` command re-enables sending to the admin2@mydomain.com account after the connections are established, so that the account receives subsequent alert messages.

Example 3

```
purealert rule list --catalog
purealert rule create --parameter warning --value 85 25
```

The **`purealert rule list --catalog`** command lists all the alerts and parameters available for customization for a specific alert. The **`purealert rule create`** command creates a custom rule for alert **`25`** with a **`warning`** parameter and a value of **`85`**.

Example 4

```
purealert code setattr --reminder-frequency 7d 60
```

Sets the reminder frequency for alert 60 to 7 days. Now if that alert comes up, an additional reminder email will be sent about it after 7 days.

Example 5

```
purealert code setattr --snooze-period 2d 4
```

Sets the snooze period of alert 4 to 2 days.

See Also

purearray

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

pureapiclient

pureapiclient, pureapiclient-create, pureapiclient-enable, pureapiclient-delete, pureapiclient-disable, pureapiclient-list – manages API clients

Synopsis

pureapiclient `create --access-policy` ***ACCESS_POLICY...*** `[--issuer` **ISSUER**`] --public-key [--accesstoken-ttl` **TIME**`]` ***NAME***

pureapiclient `delete` ***NAME***

pureapiclient `disable` ***NAME***

pureapiclient `enable` ***NAME***

pureapiclient `list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--filter` **FILTER**`] [--limit` **LIMIT**`] [--sort` **SORT**`] [--public-key] [`***NAME***`]`

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--access-policy` ***ACCESS_POLICY***

With the **pureapiclient create** command, attaches one or more management access policies to the API client. An access policy defines a set of permissions and scope (in which array or realm) for those permissions. (See "Management Access Policies" on page 453 in the **purepolicy** chapter.) Specify multiple access policies in a comma-separated list.

`--access-token-ttl` ***TIME***

Allows optional API Client request for TTL (Time to Live) in exchange for access tokens via OAuth 2.0. The duration is specified as an integer, followed by one of the suffix letters s (seconds), m (minutes), h (hours), or d (days).

`--issuer` ***`ISSUER`***

Displays the name of the identity provider. When **`--issuer`** is not specified, the name of the identity provider will default to the API client name..

`--public-key`

Displays the API client public key, or sets the public key for the specified user for SSH access. Only array administrators can change public keys on behalf of other users. If changing the public key, the public key is entered through interactive prompt. If no users are specified, the public key change is for the current user.

When used with the **`pureapiclient create`** command, creates the RSA public key in PEM format. When used with the **`pureapiclient list`** command, displays the public key.

Note: The `--max-role` option is deprecated and replaced by `--access-policy`.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Arguments

NAME

The API client name.

Description

The **`pureapiclient`** command manages API clients on the FlashArray.

The **`pureapiclient create`** command creates an API client to communicate with the FlashArray while using its own API tokens. Newly created API clients are disabled by default. The API tokens issued by the API client allows trusted access to the REST API. The **`--access-policy`** option attaches a management access policy to the API client, thereby assigning a set of permissions and scope (in which array or realm) for those permissions. (See "Management Access Policies" on page 453 in the **`purepolicy`** chapter.) Use the **`--public-key`** option to set the public key for the API client or issuer. When an API client is enabled and the public key is set, the FlashArray trusts the API client to issue ID tokens on behalf of the users. The issuing entity that generates ID tokens is specified by using the **`--issuer`** option. Pure Storage uses the OAuth 2.0 standard to authenticate to the Pure Storage REST API.

Use the **`pureapiclient create --access-token-ttl`** command to set the TTL (time-to-live) for the access tokens exchanged via OAuth 2.0. The duration is specified as an integer, followed by one of the suffix letters s (seconds), m (minutes), h (hours), or d (days). Note that an API client must be created or enabled before using the ID token.

The API client is disabled by default. To enable an API client use the `pureapiclient enable` command. Specify the **`NAME`** of the API client.

To display the current API clients run the `pureapiclient list` command.

For more information about the Pure Storage REST API, refer to the FlashArray REST API Reference Guide on the Knowledge site at https://support.purestorage.com.

Examples

Example 1

```
pureapiclient create --access-policy r1_admin_policy --public-key myClient
Please enter public key followed by Enter and then Ctrl-D:
```

Creates an API client and sets the public key for the specified user `myClient`.

Paste the public key, press `Enter`, and then press `Control-D`.

`--access-policy` names an existing management access policy **`r1_admin_policy`**, which **`pureapiclient`** attaches to `myClient`.

Example 2

```
pureapiclient enable myClient
```

Enables the newly created API client. The API client is disabled by default.

Example 3

```
pureapiclient delete myClient
```

Deletes the specified API client.

Example 4

```
pureapiclient list myClient
```

Displays the specified API client.

See Also

pureadmin, purepolicy

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

pureapp

pureapp, pureapp-disable, pureapp-enable, pureapp-list, pureapp-node — displays and manages Purity apps

Synopsis

```
pureapp disable APP
pureapp enable APP
pureapp list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--
filter FILTER] [--limit LIMIT] [--sort SORT] [ APP...]
```

Arguments

APP

App name.

Options

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Description

The Purity Run platform extends array functionality by integrating add-on services into the Purity//FA operating system. Each service that runs on the platform is provided by an app.

In the following example, an app named `linux` has been installed on the array. The `linux` app represents an Ubuntu Linux distribution with Docker pre-installed, giving `pureuser` the ability to run Docker-based containers on the array.

```
$ pureapp list
Name   Version  Status   Description                                   Details
linux  1.0.0    healthy  Ubuntu Linux 16.04 pre-installed with Docker  -
```

Note that if an app migrates between controllers, it briefly stops and restarts.

Apps require CPU, memory, network, and storage resources. For this reason, apps by default are not installed. Apps are installed through the `puresw app` command. To determine if apps are installed on your array, run the `pureapp list` command. If apps are installed on your array, Purity//FA displays a list of the installed apps.

Once an app has been installed, perform the following tasks:

1. View the app attributes (**`pureapp list`**) to verify that the app is running.
2. View the volume attributes (**`purevol list`**) to verify that the boot and data volumes have been created.
3. View the host attributes (**`purehost list`**) to verify that the app host has been created, and view the host connection details (**`purehost list --connect`**) to verify that the boot and data volumes are connected to the app host.
4. View the app interfaces (**`purenetwork list`**) to verify that they have been created.
5. Configure the app interface(s) by assigning an IP address to the interface (**`purenetwork eth setattr --address`**) and then enabling the interface (**`purenetwork eth enable`**).
    - To give **`pureuser`** the ability to log into the app, configure the app management interface.
    - To use a separate interface to transfer data in high volumes and at high speed, configure the app data interface.
6. Optionally connect FlashArray volumes to the app host (**`purevol connect --host`** or **`purehost connect --vol`**).

Viewing App Attributes

The `pureapp list` command displays a list of apps that are installed on the array, along with the following attributes for each app:

- **Name**: App name. The app name is pre-assigned and cannot be changed.
- **Enabled**: App enable/disable status. An app must be enabled so the array can reach the app service. Apps are disabled by default.
- **Version**: App version that is currently installed on the array.
- **Status**: App status. A status of `healthy` means the app is running. A status of `unhealthy` means the app is not running.

- There are various factors that contribute to an unhealthy app. In most cases, the unhealthy status is temporary, such as when the app is being restarted; upon successful restart, the app returns to healthy status. The app might also be unhealthy if, upon enabling the app, Purity//FA determines that there are insufficient resources to run it. An accompanying message appears in the Details column stating that there are insufficient resources to operate the app. Disable any apps that are currently not in use to free up some resources and try to enable the app again.

  If the app is in an unhealthy status for a longer than expected period of time, contact Pure Storage Technical Services.
- **Description**: Description of the app.
- **Details**: Additional details specific to the app.

Viewing App Volumes

For each app that is installed, a boot volume is created. For some apps, a data volume is also created. Boot and data volumes are known as app volumes. Run the **`purevol list`** command to see a list of volumes, including app volumes.

Boot and data app volume names begin with a distinctive `@` symbol. The naming convention for app volumes is `@APP_boot` for boot volumes and `@APP_data` for data volumes, where `APP` denotes the app name.

The following example displays the boot and data volumes for the `linux` app.

```
$ purevol list @linux*
Name Size Source Created Serial
...
@linux_boot 15G - 2016-11-09 15:13:37 MST AC97A330F2544A3C00011010
@linux_data 16T - 2016-11-09 15:14:17 MST AC97A330F2544A3C00011012
```

The boot volume represents a copy of the boot drive of the app. Do not modify or save data to the boot volume. When an app is upgraded, the boot volume is overwritten, completely destroying its contents including any other data that is saved to it. The data volume is used by the app to store data.

The following example shows that the drives were correctly mounted inside the `linux` app.

```
pureuser@linux:~$ df
Filesystem 1K-blocks Used Available Use% Mounted on
```

```
udev 8198768 0 8198768 0% /dev
tmpfs 1643272 8756 1634516 1% /run
/dev/sda1 15348720 1721392 12824616 12% /
/dev/sdb 17177782208 33608 17177748600 1% /data
```

Disk device `/dev/sdb`, which corresponds to the app data volume, is mounted on `/data`, meaning the data will be saved to the data volume (and not the boot volume), and disk device `/dev/sda1`, which corresponds to the app boot volume, is mounted on `/`.

Viewing App Hosts

Each app has a dedicated host, known as an app host. The app host is connected to the associated boot and data volumes. The app host is also used to connect FlashArray volumes to the app.

App host names begin with a distinctive `@` symbol. The naming convention for app hosts is `@APP`, where `APP` denotes the app name.

The following example displays the app host for the `linux` app.

```
$ purehost list @linux*
Name WWN IQN Host Group
@linux - - -
```

The following example displays one app host named `@linux` and its associated boot and data volumes.

```
$ purehost list --connect @linux*
Name LUN Vol Host Group
@linux 1 @linux_boot -
@linux 2 @linux_data -
```

Unlike regular FlashArray hosts, app hosts cannot be deleted, renamed, or modified in any way. Furthermore, app hosts cannot be added to host groups or protection groups.

Viewing App Interfaces

For each app that is installed, one app management interface is created per array management interface. An app data interface may also be created for high-speed data transfers.

The naming convention for app interfaces is `APP.datay` for the app data interface, and `APP.mgmty` for the app management interface, where *APP* denotes the app name, and *y* denotes the interface.

In the following example, for the `linux` app, two app management interfaces, one named `linux.mgmt0` and another named `linux.mgmt1`, have been created to correspond to each of the array management interfaces. An app data interface (named `linux.data0`) with a data transfer rate of 10 gigabytes per second has also been created.

```
Name Enabled Address Mask ... Speed Services
ct0.eth0 True 10.7.102.60 255.255.255.0 ... 1.00 Gb/s management
ct0.eth1 False - - ... 1.00 Gb/s management
ct0.eth2 True 10.7.102.62 255.255.255.0 ... 10.00 Gb/s iscsi
ct1.eth0 True 10.7.102.70 255.255.255.0 ... 1.00 Gb/s management
ct1.eth1 False - - ... 1.00 Gb/s management
ct1.eth2 True 10.7.102.72 255.255.255.0 ... 10.00 Gb/s iscsi
linux.data0 False - - ... 10.00 Gb/s app
linux.mgmt0 False - - ... 1.00 Gb/s app
linux.mgmt1 False - - ... 1.00 Gb/s app
vir0 True 10.7.102.80 255.255.255.0 ... 1.00 Gb/s management
vir1 False - - ... 1.00 Gb/s management
```

Configuring App Interfaces

Configure an app interface to give `pureuser` the ability to log into the app or transfer data through a separate interface. Configuring an app interface involves assigning an IP address to the interface and then enabling the interface.

Optionally set the gateway. Note that only one of the app interfaces of a particular app can have a gateway set.

Before you configure an app interface, make sure the corresponding external interface is physically connected.

Configure one or more of the following app interfaces:

- **App Management Interface**

  Configure the app management interface to give `pureuser` the ability to log into the app with the same Purity//FA login credentials. If a public key has been created for the user, it can be used to log into the app. Purity//FA password changes are automatically applied to the app.

  To configure the app management interface, run the **`purenetwork eth setattr --address`** command to assign an IP address to one of the app management interfaces, and then run the **`purenetwork eth enable`** command to enable the interface.

- **App Data Interface**

  Configure the app data interface to use a separate interface for high-speed data transfers.

  To configure the app data interface, run the **`purenetwork eth setattr --address`** command to assign an IP address to the app data interface, and then run the **`purenetwork eth enable`** command to enable the interface.

In the following example, app management interface `linux.mgmt0` has been enabled and IP address `10.8.102.96` has been assigned to the interface, giving `pureuser` the ability to log directly into the `linux` app with the same Purity//FA login credentials.

```
$ purenetwork list
Name Enabled Address ... Speed Services
...
linux.data0 False - ... 10.00 Gb/s app
linux.mgmt0 True 10.8.102.96 ... 1.00 Gb/s app
linux.mgmt1 False - ... 1.00 Gb/s app
...
```

Establishing Connections between FlashArray Volumes and Apps

FlashArray volumes are connected to apps via the app host.

To connect a FlashArray volume to an app, connect the volume to the app host that is associated with the app. In the following example, five FlashArray volumes, along with the `linux` boot and data volumes, are connected to the `@linux` app via the `@linux` app host.

```
$ purehost list --connect @linux*
Name LUN Vol Host Group
@linux 1 @linux_boot -
@linux 2 @linux_data -
@linux 3 app_vol001 -
@linux 4 app_vol002 -
@linux 5 app_vol003 -
@linux 6 app_vol004 -
@linux 7 app_vol005 -
```

FlashArray volumes are connected to app hosts in the same way that they are connected to regular FlashArray hosts. Run the **`purevol connect --host`** command to connect one or more volumes to an app host. The connection can also be established by running the **`purehost connect --vol`** command.

A FlashArray volume can only be connected to one app host at a time. Furthermore, the FlashArray volume cannot be connected to other hosts or host groups while it is connected to an app host.

After a FlashArray volume has been connected to an app host, rescan the SCSI bus to ensure the newly-connected volumes are visible from inside the app. The following example displays five FlashArray volumes (and their target LUNs) as SCSI devices from inside the `linux` app, ready to be mounted.

```
pureuser@linux:~$ cat /proc/scsi/scsi
Attached devices:
Host: scsi2 Channel: 00 Id: 01 Lun: 03
Vendor: PURE Model: FlashArray Rev: 9999
Type: Direct-Access ANSI SCSI revision: 06
Host: scsi2 Channel: 00 Id: 01 Lun: 04
Vendor: PURE Model: FlashArray Rev: 9999
Type: Direct-Access ANSI SCSI revision: 06
Host: scsi2 Channel: 00 Id: 01 Lun: 05
Vendor: PURE Model: FlashArray Rev: 9999
```

```
Type: Direct-Access ANSI SCSI revision: 06
Host: scsi2 Channel: 00 Id: 01 Lun: 06
Vendor: PURE Model: FlashArray Rev: 9999
Type: Direct-Access ANSI SCSI revision: 06
Host: scsi2 Channel: 00 Id: 01 Lun: 07
Vendor: PURE Model: FlashArray Rev: 9999
Type: Direct-Access ANSI SCSI revision: 06
```

To disconnect a FlashArray volume from an app host, run the `purevol disconnect --host` or `purehost disconnect --vol` command.

Enabling and Disabling Apps

The `pureapp enable` command enables an app. An app must be enabled so the array can reach the app service. Apps are disabled by default.

The `pureapp disable` command disables an app, effectively stopping the service.

Installing Apps

The `puresw app install` command installs a new app or upgrades a currently installed app. For more information about the app installation process, refer to puresw.

Examples

Example 1

```
pureapp list
```

Displays a list of apps that have been installed on the array and the attributes of each app, including name, enabled status, version number, description, and details.

Example 2

```
purevol list @linux*
purehost list --connect @linux
```

```
purenetwork eth setattr --address 10.8.108.165 linux.mgmt0
purenetwork eth enable linux.mgmt0
```

Displays the boot and data volumes for the `linux` app, displays the names of all volumes that are connected to app host `linux`, assigns IP address `10.8.108.165` to app management interface `linux.mgmt0`, and enables interface `linux.mgmt0`, giving *pureuser* the ability to log into the `linux` app.

Example 3

```
purevol connect --host @linux app_vol001
```

Connects FlashArray volume `app_vol001` to the `linux` app via app host `@linux`. Note that the connection can also be established by running the **`purehost connect --vol`** command.

Example 4

```
purehost disconnect --vol app_vol002 @linux
```

Breaks the connection between FlashArray volume `app_vol002` and the `linux` app. Note that the connection can also be established by running the **`purevol disconnect --host`** command.

Example 5

```
pureapp disable linux
```

Disables the `linux` app.

See Also

purehost, purenetwork, puresw, purevol, "purevol-list

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purearray

purearray, purearray-list, purearray-rename, purearray-replace, purearray-setattr, purearray-eradication-config – manages attributes that pertain to the array as a whole

purearray-cloud-capacity – lists or sets the capacity of a Cloud Block Store array

purearray-cloud-config model – manages hardware non-disruptive upgrades for Cloud Block Store arrays.

purearray-default-protection – manages enabling automatic protection group assignment for new volumes

purearray-disable, purearray-enable, purearray-phonehome, purearray-remoteassist – manages support-related functions

purearray-diagnose, purearray-test – performs array diagnostic functions

purearray-connect, purearray-disconnect, purearray-connection – manages connections between arrays

purearray-encryption-list – provides the encryption module version and data-at-rest encryption status

purearray-eradication-config – manages and displays the current eradication configuration

purearray-eula – manages the Pure Storage End User Agreement (EUA)

purearray-erase – performs a factory reset on a Cloud Block Storearray

purearray-factory-reset-token – manages the authorization token for a factory reset on a Cloud Block Store array

purearray-monitor – monitors array I/O performance

purearray-suspend – suspends a Cloud Block Store for AWS array

purearray-tag – adds a tag to public cloud

purearray-untag – removes a tag from public cloud

Synopsis

purearray authorize factory-reset

purearray cloud-capacity list [--supported] [--cli | --csv | --nvp] [--page] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--notitle] [--sort SORT]

purearray cloud-capacity setattr --requested-capacity ***REQUESTED_ CAPACITY***

purearray cloud-config model list [--supported]

purearray cloud-config model setattr [--requested-model ***REQUESTED_ MODEL***] [--override ***CHECK_NAME, ARGS...***]

purearray connect --management-address ***MANAGEMENT-ADDRESS*** [--replication-address ***REPLICATION-ADDRESS***] --type {async-replication | sync-replication} [--replication-transport {fc | ip}] [--encrypted] [--connection-key] [--context ***REMOTE***]

purearray connection disable --encrypted [--context ***REMOTE***] ***ARRAY***

purearray connection enable --encrypted [--context ***REMOTE***] ***ARRAY***

purearray connection list [--cli | --csv | --nvp] [--notitle] [--raw] [--page] [--page-with-token] [--token ***TOKEN***] [--path] [--controller] [--throttle] [--context ***REMOTES***]

purearray connection refresh [--renew-encryption-key] [--context ***REMOTE***] ***ARRAY***

purearray connection setattr [--management-address ***MANAGEMENT_ADDRESS***] [--replication-address ***REPLICATION_ADDRESS***] [--type {async-replication | sync-replication}] [--context ***REMOTE***] ***ARRAY***

purearray connection throttle [--default-limit ***DEFAULT_LIMIT***] [--window ***WINDOW***] [--window-limit ***WINDOW_LIMIT***] [--context ***REMOTE***] ***ARRAY***

purearray default-protection list [--pending | --pending-only] [--cli | --csv | --nvp] [--filter ***FILTER***] [--limit ***LIMIT***] [--notitle] [--page] [--raw] [--sort SORT] [***CONTAINER...***]

purearray default-protection set [--pgroup ***PGROUP***] ***CONTAINER...***

purearray diagnose {--email | --log | --phonehome}

purearray disable {console-lock | phonehome}

purearray disconnect [--context ***REMOTE***] ***ARRAY***

purearray enable {console-lock | phonehome | security-token}

purearray encryption list {--data-at-rest | --module-version} [--csv | --nvp] [--notitle] [--page]

purearray eradication-config list [--page-with-token] [--token ***TOKEN***] [--context ***REMOTES***]

purearray eradication-config setattr [--disabled-delay DISABLED_DELAY] [--enabled-delay ENABLED_DELAY] [--context ***REMOTE***]

purearray erase --factory-reset-token ***FACTORY-RESET-TOKEN*** --eradicate-all-data [--context ***REMOTE***]

purearray eula {accept | list [--acceptance]}

purearray factory-reset-token { create | delete | list }

purearray list [--banner] [--cloud-provider] [--connect] [--connection-key] [--console-lock] [--controller] [--encrypted] [--expected-resume-time] [--historical {1h,1y,24h,30d,3h,7d,90d}] [--idle-timeout] [--ntp] [--ntpserver] [--path] [--phonehome] [--proxy] [--relayhost] [--scsi-timeout] [--security-token] [--senderdomain] [--service] [--space] [--syslogserver] [--tags] [--throttle] [--time] [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--page-with-token] [--token ***TOKEN***] [--context ***REMOTES***] [--installed-capacity] [***ARRAY ...***]

purearray monitor [--protocol {nfs | smb} | --protocol-group {block | file}] [--historical {1h | 3h | 24h | 7d | 30d | 90d | 1y}] [--interval ***SECONDS***] [--latency] [--mirrored] [--link] [--repeat ***REPEAT-COUNT***] [--size] [--csv] [--notitle | --raw] [--page]

purearray phonehome {--cancel | --send-all | --send-today | --send-yesterday | --status}

purearray remoteassist {--connect | --disconnect | --status}

purearray rename [--context ***REMOTE***] ***NEW-NAME***

purearray replace {security-token}

purearray report {security-token}

purearray setattr [--banner | --idle-timeout ***IDLE-TIMEOUT*** | --ntpserver ***NTPSERVER*** | --ntp-symmetric-key | --proxy ***PROXY*** | --scsi-timeout SCSI-TIMEOUT | --syslogserver ***SYSLOG-SERVER***} [--context ***REMOTE***] `purearray suspend [--expected-resume-time` ***`EXPECTED-RESUME-TIME]`*** `[--allow-unsafe-suspend-with-cmk]` ***`ARRAY`***

`purearray tag --cloud-provider --key` ***`KEY`*** `--value` ***`VALUE`***

`purearray test {ntp | phonehome | pinghome | security-token | syslogserver}`

`purearray throttle --connect {--default-limit` ***`DEFAULT-LIMIT`*** `--window` ***`WINDOW`*** `--window-limit` ***`WINDOW-LIMIT`***`}` ***`ARRAY`***

`purearray untag --cloud-provider --key` ***`KEY`***

Options

`--acceptance`

Displays the acceptance details for the Pure Storage End User Agreement. Acceptance details include the name and title of the individual at the company who accepted the terms of the agreement, company name, and agreement acceptance date.

`--allow-unsafe-suspend-with-cmk`

When used with the **`purearray suspend`** command, allows unsafe suspensions despite use of CMK for default EBS encryption. This operation is not recommended since CMK usage can block resume.

`--banner`

Displays or sets text that Purity users see in the GUI at login, and in the CLI just before the password prompt.

`--cancel`

Cancels any in-progress transmission of event logs to Pure Storage Technical Services.

`--cloud-provider`

Specifies a public cloud to tag. When used with the **`purearray list`** command, displays currently configured tags. When used with the purearray tag command along with the **`--key`** and **`--value`** options, configures a tag for the specified cloud provider. When used with the **`purearray untag`** command along with the **`--key`** option, removes a tag for the specified cloud provider.

`--connect`

When used with **`purearray remoteassist`**, enables Pure Storage Technical Services to initiate a remote assistance session.

The `purearray throttle` command must be used with any of the **`purearray throttle`** options to set network bandwidth throttle limits.

Deprecated for use with `purearray list`, instead use **`purearray connection list`**.

`--connection-key`

When used with `purearray list`, displays a connection key that can be used to connect to this array from another array. Combined with **`--encrypted`**, the displayed “connection key” includes a pre-shared key (PSK) for encrypted connection, for one time use only.

Deprecated for use with `purearray connect`, since the command prompts for the encrypted connection key when used with the **`--encrypted`** option, or connection key when encryption is not used.

`--context` ***`REMOTE`***

Used to specify the remote array on which a command is processed. ***`REMOTE`*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

If used with `purearray list` or `purearray connection list`, `--context` can specify multiple arrays and ***`REMOTES`*** is a comma-separated list of array names.

`--controller`

When used with the `purearray list` command, displays the name, type, mode, FlashArray model, Purity//FA version, and status information for the array controllers.

When used with the `purearray connection list` command, displays the remote array name, transport type, local controller, remote controller, and the number of connected paths for the array controllers.

`--data-at-rest`

When used with the `purearray encryption list` command, displays the encryption algorithm in use, operating mode, and key length.

`--default-limit` ***`DEFAULT-LIMIT`***

Sets the default maximum network bandwidth threshold for outbound traffic when transferring data to the specified remote array. Once the threshold is exceeded, bandwidth throttling occurs.

`--disabled-delay` ***`DISABLED_DELAY`***

Used with the `purearray eradication-config setattr` command, configures the eradication delay (the length of the eradication pending period) for destroyed objects that *are* protected by SafeMode (objects for which eradication is disabled). Takes effect only when SafeMode is enabled.

DISABLED_DELAY is an integer number of days, from 1 day to 30 days, such as 2d, 14d, or 3w.

DISABLED_DELAY is specified in the pattern `N[s/m/h/d/w]`, representing seconds, minutes, hours, days, or weeks, preceded by a positive integer. Together the number and time unit provided must result in an integer number of days from 1 day to 30 days. For example: 1440m, 24h, 48h, 10d, or 2w.

`--disconnect`

Terminates an in-progress remoteassist session with Pure Storage Technical Services.

`--email`

Sends subcommand output to Pure Storage Technical Services.

`--enabled-delay` ***ENABLED_DELAY***

Used with the **`purearray eradication-config setattr`** command, configures the eradication delay (the length of the eradication pending period) for destroyed objects that *are not* protected by SafeMode (objects for which eradication is enabled).

ENABLED_DELAY is an integer number of days, from 1 day to 30 days, such as 2d, 14d, or 3w.

ENABLED_DELAY is specified in the pattern `N[s/m/h/d/w]`, representing seconds, minutes, hours, days, or weeks, preceded by a positive integer. Together the number and time unit provided must result in an integer number of days from 1 day to 30 days. For example: 1440m, 24h, 48h, 10d, or 2w.

`--encrypted`

When used with **`purearray connect`**, establishes an encrypted connection to the remote array. When omitted, the connection is unencrypted. Encryption cannot be selected when the connection type is **`sync-replication`** or replication transport is Fibre Channel (**`fc`**).

When used with **`purearray connection enable`**, or **`purearray connection disable`**, enables or disables encryption for an existing connection.

When used with **`purearray list --encryption-key`**, returns a new encrypted connection key. The encrypted connection key includes the pre-shared key (PSK) and is for one time use only. A new key is generated when you obtain the key, which is needed each time if you reconnect.

`--eradicate-all-data`

Confirms a factory reset by specifying this option to eradicate all data on the Cloud Block Store array.

`--expected-resume-time`

When used with the **`purearray list`** command, displays the expected resumption time for the Cloud Block Store array if it is to be suspended now.

When used with the **`purearray suspend`** command, specifies the expected resumption time for the Cloud Block Store array to be suspended.

This option must be specified from a Cloud Block Store appliance.

`--factory-reset-token` ***`FACTORY-RESET-TOKEN`***

Specifies the token generated from a factory reset authorization to execute a factory reset on the Cloud Block Store array.

`--historical`

When used with **`purearray list --space`**, displays the amount of usable physical storage or effective used capacity on the array and the amount occupied by data and metadata over the specified range of time. Valid time ranges include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year.

When used with **`purearray monitor`**, displays historical performance data over the specified range of time. Valid time ranges include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year.

`--idle-timeout` ***`IDLE-TIMEOUT`***

Displays or sets an idle time limit in minutes. CLI and GUI sessions that are idle for more than IDLE-TIMEOUT minutes are automatically logged out. Values between 5 and 180 minutes are valid. Specifying a value of zero disables the automatic logout feature. The default timeout value is 30 minutes. Idle timeout changes apply to existing sessions.

`--installed-capacity`

Displays the usable capacity of the array without capacity limits in place.

`--interval` ***`SECONDS`***

Sets the number of seconds between displays of real-time performance data. At each interval, the system displays a point-in-time snapshot of the performance data. If omitted, the interval defaults to every 5 seconds.

`--key` **`KEY`**

Specifies the key of a tag to be added to public clouds. Must be used with the **`--value`** option.

`--latency`

Displays real-time and historical I/O latency information across the entire array.

`--link`

Used with the **`--mirrored`** option to display synchronous replication statistics for writes per synchronized replication link.

`--log`

Sends subcommand output to the array's log.

`--management-address` ***`MANAGEMENT-ADDRESS`***

Specifies the management address. Enter the virtual (**`vir0`**) IP address or FQDN of the remote array.

`--mirrored`

In an ActiveCluster replication configuration, includes performance data for mirrored writes (that is, writes to volumes in stretched pods). If one array of a stretched pod is offline, the writes processed on the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again. Include the **`--link`** option to display writes per synchronized replication link.

`--module-version`

When used with the **`purearray encryption list`** command, displays the version of the Purity Encryption Module installed on the array.

`--ntp`

Displays a list of NTP servers and the NTP symmetric key configured when used with the **`purearray list`** command.

`--ntpserver [`***`NTP-SERVER`***`]`

Displays or sets the hostnames or IP addresses of the NTP servers currently being used by the array to maintain reference time.

When used with **`purearray setattr`**, specify up to four NTP servers. If specifying an IP address, for IPv4, specify the IP address in the form **`ddd.ddd.ddd.ddd`**, where **`ddd`** is a number ranging from **`0`** to **`255`** representing a group of 8 bits. For IPv6, specify the IP address in the form **`xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx`**, where **`xxxx`** is a hexadecimal number representing a group of 16 bits. When specifying an IPv6 address, consecutive fields of zeros can be shortened by replacing the zeros with a double colon (`::`).

`--ntp-symmetric-key`

Sets the value for the NTP symmetric key to be used for NTP authentication.

When used with the **`purearray setattr`** command, the NTP symmetric key must follow a specific format. If it is an ASCII string, it cannot contain the character "**#**" and cannot be longer than 20 characters. If it is a hex-encoded string, it cannot be longer than 64 characters.

`--override` ***`CHECK_NAME, ARGS...`***

Allows you to temporarily bypass specific upgrade checks that might otherwise prevent the upgrade from proceeding. This can be useful if you are aware of certain conditions that would cause a check to fail but are confident that they will not impact the upgrade process. Used with the **`purearray cloud-config model setattr`** command.

`--path`

Used with the **`purearray list`** or **`purearray connection list`** commands, displays the connection paths between this array and each of the connected arrays.

`--pgroup` ***`PGROUP`***

Comma-separated list of default protection groups.

`--phonehome`

For **`purearray diagnose`**, sends subcommand output to Pure Storage Technical Services via the phonehome channel.

For **`purearray list`**, displays the current state of the Purity//FA phonehome automatic hourly log transmission facility (enabled or disabled).

`--protocol-group {block | file}`

Displays array-wide I/O performance information for the specified protocol group only.

`--protocol {nfs | smb}`

Displays array-wide I/O performance information for the specified protocol only.

`--proxy [`***`PROXY`***`]`

Displays or sets the proxy host for the phonehome facility when https is the phonehome protocol (the phonehome facility itself determines which protocol to use). The format for the option value in the **`purearray setattr`** subcommand is:

`HTTP(S)://HOSTNAME:PORT`, where **`HOSTNAME`** is the name of the proxy host, and **`PORT`** is the TCP/IP port number used by the proxy host.

`HTTP(S)://USERNAME@HOSTNAME:PORT`, where **`USERNAME`** is the required username, can also be used. The CLI prompts for a password interactively.

`--renew-encryption-key`

Used with the **`purearray connection refresh`** command to renew the key for an encrypted connection.

`--repeat` ***`REPEAT-COUNT`***

Sets the number of times to log I/O statistics or display real-time performance data. If omitted, the repeat count is 1.

`--replication-address` ***`REPLICATION-ADDRESS`***

Specifies the replication address. Enter the IP address or FQDN of the replication bond (**`replbond`**) interface on the target array. Optionally omitted for auto-discovery when NAT is not used.

`--replication-transport { fc | ip }`

Specifies the transport mechanism between two FlashArrays. Supported transport types include fc and ip, where fc denotes Fibre Channel transport and ip denotes Ethernet transport. Only one transport type is allowed between two FlashArrays.

`--requested-capacity` ***`REQUESTED-CAPACITY`***

Specifies a new capacity setting for a Cloud Block Store array. Used with the **`purearray cloud-capacity setattr`** command.

`--requested-model` ***`REQUESTED_MODEL`***

Specifies the model to which an upgrade is being requested for a CBS array. Used with the **`purearray cloud-config model setattr`** command.

`--scsi-timeout`

Displays or sets the amount of time (in seconds) that can lapse during an I/O interruption before the target ports log out of the fabric. The default timeout value is 60 seconds.

The **`--scsi-timeout`** option is an advanced option. Changing the default timeout value may cause an initiator to mistakenly interpret the status of the FlashArray as failed or generate a host timeout.

Contact the Pure Storage Technical Services team before you change the **`--scsi-timeout`** value.

`--security-token`

Displays the status of the Rapid Data Locking feature (enabled or disabled) and the signature of the attached security token. For more information about the Rapid Data Locking (RDL) feature, refer to the FlashArray Enhanced Data Security Guide on the Knowledge site at **`https://support.purestorage.com`**.

`--send-all`

Sends all log information stored in the array to Pure Storage Technical Services via the phonehome channel.

`--send-today`

Sends log information from the current day (in the array's time zone) to Pure Storage Technical Services via the phonehome channel.

`--send-yesterday`

Sends log information from the previous day (in the array's time zone) to Pure Storage Technical Services via the phonehome channel.

`--size`

Displays the average I/O sizes per read and write operation.

`--service`

Displays the array service type, `FlashArray` for purchased arrays, `Evergreen//One` for subscription storage.

`--space` (for purchased arrays)

Displays the amount of usable physical storage on the array and the amount of storage occupied by data and metadata.

Name

Name by which Purity//FA identifies the FlashArray.

Capacity

Total physical usable space on the array. Replacing a drive may result in a dip in usable capacity. This is intended behavior. RAID striping splits data across an array for redundancy purposes, spreading a write across multiple drives. A newly added drive cannot use its full capacity immediately but must stay in line with the available space on the other drives as writes are spread across them. As a result, usable capacity on the new drive may initially be reported as less than the amount expected because the array will not be able to write to the unallocatable space. Over time, usable capacity fluctuations will occur, but as data is written to the drive and spreads across the array, usable capacity will eventually return to expected levels.

Parity

Percentage of data that is fully protected. The percentage value will drop below 100% if the data isn't fully protected, such as when a module is pulled and the array is rebuilding the data to bring it back to full parity.

Provisioned Size

Total provisioned size of all volumes connected to the host group. Represents storage capacity reported to hosts.

`--space` (for Evergreen//One™ subscription arrays)

Displays capacity metrics reflecting storage consumption based on effective used capacity (EUC).

Name

Name by which Purity//FA identifies the FlashArray.

Capacity

Estimated total effective used capacity available from a host’s perspective, including both consumed and unused storage.

Provisioned Size

The sum of the sizes of all volumes on the array.

Displays a ‘-‘ sign for arrays when a file system on the array has unlimited provisioned size.

Virtual

The amount of data that the host has written to the volume as perceived by the array, before any data deduplication or compression.

Thin Provisioning

The empty un-addressed capacity that remains in the volume from the host's perspective as a percentage of the Provisioned Size.

Unique

Effective used capacity data of both volumes and file systems after removing clones, but excluding metadata and snapshots.

Snapshots

Effective used capacity consumed by data unique to one or more snapshots.

Shared

Effective used capacity consumed by cloned data, meaning that the space is shared with cloned volumes and snapshots as a result of data deduplication.

Replication

Effective used capacity consumed for either of the ActiveDR or ActiveCluster pod-based replication features.

Total

Total effective used capacity containing user data, including Shared, Snapshots, and Unique storage.

`--status`

For `purearray phonehome --status`, displays the status of in-progress log transmission jobs. A status of "running" indicates a log transmission of the displayed action type (send-all, send-today, or send-yesterday) is in progress.

For `purearray remoteassist --status`, displays the status of a remote assist session as enabled or disabled.

`--stop`

Stops logging I/O statistics.

`--supported`

When used with the **`purearray cloud-capacity list`** command, lists the capacities supported on a Cloud Block Store array. When used with the **`purearray cloud-config model list`** command, lists models supported as an upgrade target.

`--syslogserver [`***`SYSLOG-SERVER`***`]`

Displays or sets the URI of the remote syslog server. For example, **`tcp://MyHost.com`**.

Syslog servers deleted using this command will not receive an audit record on the remote syslog server. Instead, use the **`purelog`** command.

Note: The **`--syslogserver`** option is going to be replaced by the **`purelog`** command in a future release. Therefore, **`--syslogserver`** is considered deprecated and should not be used. To set remote syslog servers, run the **`purelog`** command.

`--tags`

When used with the **`purearray list`** command and the **`--cloud-provider`** option, displays all currently configured tags, not including those configured with Pure1.

`--throttle`

Used with the **`purearray connection list`** command, displays the network bandwidth throttle settings for all connected arrays.

`--time`

Displays the date, time, and timezone of an array.

`--type`

Specifies the type of connection. Valid connection types include **`async-replication`** for asynchronous replication, which supports encryption, and **`sync-replication`** for both synchronous replication and asynchronous replication.

`--value` ***`VALUE`***

Specifies the value of a key that applies to a tag. Must be used with the **`--key`** option.

`--window` ***`WINDOW`***

Range of time the **`--window-limit`** threshold is in effect: **`HH[am/pm]-HH[am/pm]`**. For example, **`4-5`** or **`3am-12pm`**.

`--window-limit` ***`WINDOW-LIMIT`***

Sets the maximum network bandwidth threshold for outbound traffic during the specified **`--window`** time range. For example: **`1M`**, **`2K`**, or **`4G`**. Once exceeded, bandwidth throttling occurs.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--page-with-token`

Used to paginate output with a continuation token (**`--token TOKEN`**). Results are printed in `stdout` and the token is printed in `stderr`.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Arguments

`ARRAY`

Name of the remote array.

`CONTAINER`

A comma-separated list of the container object or objects to which the default protection group list applies. Values include one or more pod names or **`""`** (empty double quotes) for the root of the array.

`KEY`

Key of the tag.

`NEW-NAME`

Name by which the array is to be known after the command executes.

`VALUE`

Value of the key.

Object Names

Valid characters are letters (A-Z and a-z), digits (0-9), and the hyphen (-) character. The first and last characters of the name must be alphanumeric, and the name must contain at least one letter or '-'.

Most objects in Purity//FA that can be named, including host groups, hosts, volumes, protection groups, volume and protection group suffixes, SNMP managers, and subnets, can be 1-63 characters in length.

Array names can be 1-56 characters in length. The array name length is limited to 56 characters so that the names of the individual controllers, which are assigned by Purity//FA based on the array name, do not exceed the maximum allowed by DNS.

Names are case-insensitive on input. For example, **`vol1`**, **`Vol1`**, and **`VOL1`** all represent the same volume. Purity//FA displays names in the case in which they were specified when created or renamed.

All objects in a pod have a fully qualified name that include the pod name and object name. The fully qualified name of a volume in a pod is **`POD::VOLUME`**, with double colons (`::`) separating the pod name and volume name. The fully qualified name of a protection group in a pod is **`POD::PGROUP`**, with double colons (`::`) separating the pod name and protection group name. For example, the fully qualified name of a volume named **`vol01`** in a pod named **`pod01`** is **`pod01::vol01`**, and the fully qualified name of a protection group named **`pgroup01`** in a pod named **`pod01`** is **`pod01::pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target array, the fully qualified name of the protection group on the target array is **`POD:PGROUP`**, with single colons (`:`) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to target array **`array02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod01:pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target pod, the fully qualified name of the protection group on the target pod is **`POD:PGROUP`**, with single colons (`:`) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to a target pod **`pod02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod02::pod01:pgroup01`**.

All objects in a volume group have a fully qualified name that includes the volume group name and the object name, separated by a forward slash (`/`). For example, the fully qualified name of a volume named **`vol01`** in a volume group named **`vgroup01`** is **`vgroup01/vol01`**.

Realms, pods, and volume groups provide a namespace with unique naming conventions. All objects in a realm have a fully qualified name that include the realm name and object name or realm, pod, and object names. The fully qualified name of a pod in a realm is **`REALM::POD`**, with double colons (`::`) separating the realm name and pod name. The fully qualified name of a volume in a pod in a realm is **`REALM::POD::VOLUME`**, with double colons (`::`) separating the realm, pod, and volume names. The fully qualified name of a protection group in a pod in a realm is **`REALM::POD::PGROUP`**, with double colons (`::`) separating the realm, pod, and protection group names. The fully qualified name of an object in a realm but not in a pod is **`REALM::HOST`**, using host as an example.

Description

For **purearray cloud-capacity** commands, which apply only to Cloud Block Store arrays, see "Managing Capacity on Cloud Block Store Arrays" on page 92.

The **purearray diagnose** subcommand executes a series of CLI subcommands that capture a snapshot of an array's I/O performance, configuration, and hardware status to be collected by Pure Storage Technical Services. The output format must be declared with one of the following options:

`--email`

Mail the output to Pure Storage Technical Services.

`--log`

Write the output to the array's log.

`--phonehome`

Transmit the output to Pure Storage Technical Services via the secure channel.

The **purearray enable phonehome** and **purearray disable phonehome** commands enable and disable, respectively, automatic hourly transmission of array logs to Pure Storage Technical Services.

The **purearray enable console-lock** and **purearray disable console-lock** commands enable and disable, respectively, root user login at the physical (serial or VGA) console.

The **purearray enable security-token** enables Rapid Data Locking if the required hardware is present. The command must be run on the primary controller.

The **purearray encryption list** provides information about an array's encryption status. Used with the **purearray encryption list --module-version** command, it displays the version of the Purity Encryption Module installed on the array. Used with the **purearray encryption list --data-at-rest** command, it displays if the data-at-rest encryption is enabled on the array. If it is, the response indicates the encryption algorithm in use, its operating mode, and key length.

The following example displays the output on a standard (encryption enabled) build of Purity:

```
$:~# purearray encryption list --data-at-rest
Enabled Algorithm
True AES-256-CTR
root@vm-pureuser-ct0:~# purearray encryption list --module-version
Module Version
```

```
FA-1.3
```

The `purearray eradication-config` command manages and displays the eradication delay and manual eradication settings. See "Eradication Delays" on page 88.

The `purearray list` command with no option specified displays the array name, the array ID, and the Purity//FA version and revision numbers. Options can be specified to display controller and configuration information, such as connected arrays, the NTP (Network Time Protocol) server name, the state of and proxy host for the phonehome facility, the state of the console lock, space metrics, and array service type.

The `purearray list --connect` command displays a list of arrays that are connected to this array. In the list output, the management and replication addresses only appear for the arrays from where an array connection was made. If the array connection was made from its peer array, the Management Address and Replication Address columns display dashes (-). Include the **`--path`** option to display the connection paths for each of the connected arrays. The **`--path`** option is only valid when run on the array from where the array connection was made. A status of **`connected`** means that the path between the source and destination is healthy. A status of **`connecting`** means that the connection is unhealthy, possibly due to network issues. When a path goes into **`connecting`** status, the array generates an alert citing a degraded connection, which may result in replication issues. Check the network connectivity between the replication interfaces. A status of **`quarantined`** means that the path is unstable and has been temporarily embargoed for synchronous replication connections. When a path goes into quarantined status, the array generates an alert citing an unhealthy path detected, which is closed when the path is healthy again.

The `purearray list --controller` command displays the name, mode, FlashArray model, Purity//FA version, and status information for the array's controllers.

```
purearray list --controller
Name Type             Mode     Model  Version Status  Current Mode Since       Internal Details
CT0  array_controller primary  VMware 99.9.9   ready  2024-01-04 10:11:30 PST
```

The following are the controller modes:

`primary`

Running as the primary controller.

`secondary`

Running as the secondary controller.

`not present`

Not installed.

`offline`

Installed but not running Purity//FA at the moment. For example, represents a controller that is powered off or is undergoing a Purity//FA or firmware upgrade.

The Status column contains one of the following:

`ready`

Functioning normally.

`not ready`

In the process of starting Purity//FA.

`updating`

Undergoing a firmware upgrade.

`unknown`

Status could not be determined.

For arrays to which a second controller has never been connected, only a single controller is displayed. Once an additional controller has been connected, its information is always shown.

The **`purearray list --service`** command displays the array service type, which indicates whether the array is purchased or is part of subscription storage. On a purchased array, the **`purearray list --service`** command displays `FlashArray`.

```
$ purearray list --service
Type
FlashArray
```

On subscription storage, the **`purearray list --service`** command displays `Evergreen//One`.

```
$ purearray list --service
Type
Evergreen//One
```

The **`purearray list --space`** command, on purchased arrays, displays current physical storage capacity, both total configured and occupied by data. On subscription storage, the **`purearray list --space`** command reflects storage consumption based on effective used capacity (EUC). Include the **`--historical`** option to display historical space data at the specified resolution.

The **`purearray list --time`** command outputs the date, time, and timezone of an array.

Run the **`purearray list --installed-capacity`** command to display the full installed capacity of an array with no capacity limits in place, showing both the entitled and installed capacities.

The `purearray monitor` command displays real-time and historical I/O performance information for the whole array. The output includes the following data about bandwidth, IOPS, and latency:

- **Time**: Current time.
- **B/s**: Bandwidth. Number of bytes read/written.
- **op/s**: IOPS. Number of read, write, or mirrored write requests processed per second.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve a read, write, or mirrored write I/O request. Latency does not include SAN time or QoS rate limit time.
- **B/op**: IOPS. Average I/O size per read, write, mirrored write, and both read and write (all) operations. Must include the `--size` option to see the B/op columns.

To display I/O performance information for block or for file exclusively, include the `--protocol-group` option and specify `block` or `file`. Alternatively, for a single file protocol, use the `--protocol` option and specify `nfs` or `smb`. Refer to the `puredir monitor` command for I/O performance information on directory level.

When displaying information for file only, metadata operations are shown as “other”.

By default, the `purearray monitor` command displays real-time performance data. Include the `--repeat` option to specify the number of times to repeat the real-time update. If not specified, the repeat value defaults to 1. Include the `--interval` option to specify the number of seconds between each real-time update. At each interval, the system displays a point-in-time snapshot of the performance data. If not specified, the interval value defaults to every 5 seconds. The `--interval` and `--repeat` options can be combined.

In an ActiveCluster replication configuration, include the `--mirrored` option to display performance data for mirrored writes (that is, writes to volumes in stretched pods). If one array of a stretched pod is offline, the writes processed of the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again. Include the `--link` option to display writes per synchronized replication link.

The `purearray monitor --historical` command displays historical performance data over any of the following ranges of time: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, or 1 year.

The `purearray monitor --latency` command displays real-time and historical I/O latency information across the entire array. The output includes the following data:

- **Time**: Current time.

- **SAN us/op**: SAN time. Average time, measured in microseconds, required to transfer data between the initiator and the array. Slow data transfers will result in higher SAN times.
- **Queue us/op**: Queue time. Average time, measured in microseconds, that an I/O request spends in the array waiting to be served. The time is averaged across all I/Os of the selected types.
- **QoS Rate Limit us/op**: QoS rate limit time. Average time, measured in microseconds, that all reads, writes, or mirrored writes spend in queue as a result of bandwidth limits reached on one or more volumes.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve read, write, or mirrored write I/O requests across all volumes. Latency does not include SAN time or QoS rate limit time.

When displaying information for file only, metadata operations are shown as “other”.

Include the `--mirrored` option to display latency data for mirrored writes (i.e., writes to volumes in stretched pods).

The `purearray phonehome` subcommand initiates immediate transmission of event logs stored in an array to Pure Storage Technical Services. Options determine whether the current day, previous day, or all log information in the array are transmitted. Only one transmission may be in progress at any instant. A transmission in progress can be canceled by specifying the `--cancel` option in the subcommand. The phonehome facility automatically selects either the ssh or the https protocol for communicating with Pure Storage Technical Services.

If a proxy host is required by https, use the `purearray setattr --proxy` command to specify its hostname and port. Specify the full URL string as described in the OPTIONS section above. Use the `purearray setattr --proxy https://localhost:8080` command to configure a proxy without a username and password. When a username and password are required, run the `purearray setattr --proxy https://username@localhost:8080` command. Only the username should be specified. The CLI prompts for a password interactively.

Purity//FA can transmit an array's log contents to Pure Storage Technical Services (a) automatically once per hour, or (b) on demand. Pure Storage Technical Services stores log contents for immediate or later analysis.

On-demand log transmission may be complete (`--send-all` option) or selective (`--send-today` or `--send-yesterday` options). Run the `--status` option to check the status of an in-progress log transmission. Run the `--cancel` option to cancel an in-progress log transmission.

Purity//FA can collect samples of hardware status, monitored performance, and array configuration on demand, and either (a) mail them to enabled alert recipients, (b) write them into the array's log, or (c) transmit them immediately to Pure Storage Technical Services for analysis.

The `purearray remoteassist` subcommand enables or terminates a detached remote assistance session with Pure Storage Technical Services. When the `purearray remoteassist --connect` command enables a session, the connection status is displayed. The administrative session from which the `purearray remoteassist --connect` command is issued is unaffected, and can terminate the session at any time via the `purearray remoteassist --disconnect` subcommand.

One remote assistance session can be active at a time. Executing the `purearray remoteassist --connect` command while a session is active displays the current status. The `purearray remoteassist --status` command displays remote assistance status (disconnected, connecting, connected, or session-active).

The `purearray authorize factory-reset` command authorizes Pure Storage Technical Services to reset a FlashArray. Resetting the array involves the following steps:

- Ensure you are connected to the correct array.
- Ensure all volumes on the array are deleted.
- Run `purearray authorize factory-reset --eradicate-all-data` to generate an authorization code. Send the code to Pure Storage Technical Services to authorize resetting the array.

Use the `purearray rename` command to change an array's name. See the Object Names section below for information on Purity//FA object naming rules.

Use the `purearray setattr` command to change other attributes. To set a null value, use an empty string (`""`).

If Rapid Data Locking was enabled during `puresetup`, the `purearray replace` command substitutes new security tokens for those currently in use. Additional security tokens should be connected to both controllers prior to issuing the command. The command updates flash module keys using the passphrase on the new security tokens.

The `purearray tag` command provides the ability to tag public clouds. The `purearray untag` command provides the ability to remove configured tags.

Purity//FA can test certain array functionality on demand. The `purearray test ntp` command verifies that the configured NTP servers can be used to synchronize time on both controllers with Coordinated Universal Time (UTC). The `purearray test pinghome` command tests an array's ability to connect with Pure Storage Technical Services by establishing a secure shell or secure http connection. The `purearray test phonehome` command extends the

purearray test pinghome command by additionally verifying that test messages can be exchanged. The **purearray test syslogserver** command sends a test message to configured syslog servers.

The **purearray report security-token** command tests connectivity and reports the current status of the attached security token.

The **purearray test security-token** command tests the functionality of attached security tokens by regenerating and displaying an enhanced secret used to encrypt flash modules. The command verifies that attached security tokens are accessible and contain the passphrase used to generate the flash module keys currently in use. The command returns an error if:

- Rapid Data Locking is not enabled (configured during **puresetup**).
- The security token is not present or not functional.
- The attached security token does not contain the correct passphrase.

For more information about security tokens and the Rapid Data Locking (RDL) feature, refer to the FlashArray Enhanced Data Security Guide on the Knowledge site at **https://support.purestorage.com**.

Connecting Arrays

To transfer data from one array to another, the two arrays must be connected. For example, arrays must be connected to replicate data from one array to another.

Arrays are connected using a connection key, which is supplied from one array and entered into the other array.

When two arrays are connected, the array where data is being transferred from is called the local array, and the array to which the data is being transferred is called the remote array. A local array can connect to multiple remote arrays.

Connecting two arrays for data replication involves the following steps:

1. From the target array, obtain the connection key.
   - For unencrypted Async or Sync connections: **purearray list --connection-key**
   - For encrypted Async connections: **purearray list --connection-key --encrypted**
2. From the local array, connect the local array to the target array (**purearray connect --management-address --type**).

The `purearray connect --management-address --type` command connects the local array to a remote array. The `--management-address` option represents the virtual (`vir0`) IP address or FQDN of the remote array. The command prompts for the connection key.

To obtain the connection key, run `purearray list --connection-key` from the remote array. Include the `--encrypted` option to instead generate an encrypted connection key. Each time `purearray list --connection-key --encrypted` is executed, a new encrypted connection key is generated.

The `--type` option represents the type of connection. Array connections set to `async-replication` support asynchronous replication only, while array connections set to `sync-replication` support both synchronous replication and asynchronous replication. Async and ActiveDR supports both connection types. If ActiveCluster is used, you should specify the `sync-replication` type for the array connection. Otherwise, the `async-replication` type, which also supports encryption, is sufficient.

Optionally, use the `--replication-transport` option to specify the transport mechanism between the two arrays. Supported transport types include Ethernet (`ip`) and Fibre Channel (`fc`). Only one transport type is allowed between two FlashArrays. If you are using Network Address Translation (NAT) on Purity//FA replication interfaces, use the `--replication-address` option and specify the NAT IP addresses that are assigned to the remote array.

Once two arrays are connected, optionally use the `purearray connection throttle` command to configure network bandwidth throttling to set maximum threshold values for outbound traffic.

In case of network configuration changes, the `purearray connection` command allows you to manage and update existing array connections without repeating the `purearray connect` process. Should this return an error “Connection key not found” then the existing connection cannot be updated and instead, it needs to be created again with the new configuration.

Use the `purearray setattr` command to change connection attributes such as `--management-address`, `--replication-address`, or `--type`. To enable or disable encryption (asynchronous IP connections only), use the `purearray enable` or `purearray disable` commands with the `--encrypted` option. A new connection key or encrypted connection key is required if you switch connection type or enable or disable encryption, see the `purearray list` command.

Refreshing the connection is helpful in cases such as network configuration changes on either side of the connection, to synchronize the changes. The request must be issued from the array that initiated the connection. To refresh the connection, use the `purearray refresh` command. Include the `--renew-encryption-key` option to renew the connection key for an encrypted connection.

Disconnected Arrays

The `purearray disconnect` command disconnects the local array from the specified remote array and can only be run on the array that established the connection.

Disconnecting two arrays suspends any in-progress data transfer processes. The process resumes when the arrays are reconnected.

If two arrays are connected for replication purposes, disconnecting the arrays does not delete data from the replication target. However, after the two arrays have been disconnected, removing a target array from a protection group immediately deletes all of the data on the target array that was sent from its source array. As a result, replicating data again from the source array to the target array would cause another baseline transfer.

Protection Groups on Disconnected Target Arrays

If protection groups remain on a disconnected target array, it is recommended to reconnect the disconnected target array to the source array and then destroy and eradicate those protection groups from the source array. If reconnecting is not feasible, use the **`purepgroup destroy`** command (and also **`purepgroup eradicate`** or **`purepgroup recover`**, if needed) on the target array. See "Destroying Protection Groups" on page 334 for the **`purepgroup`** commands.

Automatic SafeMode™ Default Protection for Volumes

Purity//FA provides automatic protection group membership for newly created volumes and copied volumes. Default protection is implemented with configurable lists of one or more default protection groups. Each pod has its own separate list of default protection groups, and an additional list at the root of the array, for volumes that are not members of a pod. Newly created volumes, including copied volumes, are automatically placed in each of the protection groups in the appropriate list. Each list as well as its protection groups and the protection configuration of the groups can be customized as needed.

These automatic protection groups start with the following protection configuration:

- Snapshots are scheduled every 6 hours and retained for the first 3 days, with one snapshot retained for the next 5 days.
- Replication is not enabled.
- Retention lock is ratcheted.
- Manual eradication is disabled.

Initially, for new volumes not in a pod, the initial configuration of its root array default protection group list contains one protection group, named **`pgroup-auto`**. If the root array list has not been customized, the initial configuration of its default protection group list contains one protection group, named **`<pod-name>::pgroup-auto`**.

Each protection group list can be configured separately. When customizing a default protection group list, any new protection group must be created before it can be included in the list.

When a new pod is created, the initial default protection group list for the pod is a copy of the current root default protection group list, modified with the pod name. Purity//FA creates (in the pod) each protection group named in the pod default protection group list. For example, if the root array default protection group list contains **`pgroup-auto,pgroup-B`**, the pod default protection group list is created with **`<pod-name>::pgroup-auto,<pod-name>::pgroup-B`**. When a volume is created in a pod, the new volume is given membership in all protection groups contained in its default protection group list.

The **`purearray default-protection list`** command displays the current default protection lists for the array and its pods. This example shows each pod has its own default protection group list, in addition to the separate list for volumes at the root of the array (denoted by "**`-`**").

```
$ purearray default-protection list
Container Name Container Type Default Protection Name      Default Protection Type
-              -              pgroup-auto                  protection_group
pod1           pod            pod1::pgroup-auto            protection_group
pod2           pod            pod2::pgroup-auto,pgroup-B   protection_group
```

The **`purearray default-protection set`** command replaces the contents of a default protection list with a new list. The protection groups in the new list must already exist. This example replaces the default protection group list for **`pod2`**with a new list that contains the protection groups **`pod2::pg1`** and **`pod2::pg2`**.

```
$ purearray default-protection set --pgroup pod2::pg1,pod2::pg2  pod2
Container Name Container Type Default Protection Name Default Protection Type
pod2           pod            pod2::pg1                                    pro-
tection_group
                              pod2::pg2               protection_group
```

Use an empty list to opt out of default protection, as in **`purearray default-protection set --pgroup "" pod2`** for a pod or **`purearray default-protection set --pgroup "" ""`** for the root of the array. Opting out is only possible when protection groups in the default protection group list are empty and unlocked. You opt out of default protection separately for each pod and for the root of the array. Contact Pure Storage Technical Services to opt out when a ratcheted protection group is involved.

Eradication Delays

Purity supports two types of eradication delays, one for SafeMode-protected objects and one for non-protected objects:

- **Disabled delay**: The eradication delay for SafeMode-protected objects. Sets the length of the eradication pending period for SafeMode-protected objects and only takes effect when SafeMode is enabled. Known as the "disabled" eradication delay because manual eradication is disabled on those objects. Default 8 days. 14 days are recommended.
- **Enabled delay**: The eradication delay for objects for which eradication is enabled, that is, objects not protected by SafeMode. Sets the length of the eradication pending period for objects the are not protected by SafeMode.This is the eradication value used in previous Purity versions.

Use the `purearray eradication-config list` command to view the current eradication delay values as well as the manual eradication status.

```
# purearray eradication-config list
Enabled Delay  Disabled Delay   Manual Eradication
1d             8d               all-enabled
```

The Enabled Delay column displays the length of the eradication pending period for objects that *are not* protected by SafeMode and is configured with the `purearray eradication-config setattr --enabled-delay` ***`ENABLED_DELAY`*** command.

The Disabled Delay column displays the length of the eradication pending period for objects that *are* protected by SafeMode and is configured with the `purearray eradication-config setattr --disabled-delay` ***`DISABLED_DELAY`*** command.

The following example sets the eradication pending period for objects not protected by SafeMode to 2 days and sets the eradication pending period for SafeMode-protected objects to 14 days:

```
# purearray eradication-config setattr --enabled-delay 2d --disabled-delay 14d
Enabled Delay  Disabled Delay   Manual Eradication
```

```
2d              14d                 all-enabled
```

Notes about eradication delays:

- When an object is destroyed, the length of its eradication pending period is set to the number of days of the appropriate eradication delay field.
- As a protection against accidental or other deletion of important data, the disabled eradication delay cannot be decreased when there are objects under SafeMode protection.
- With array-wide SafeMode, where all array objects are SafeMode protected, both eradication delay parameters (enabled delay and disabled delay) are present, but only the SafeMode eradication delay (disabled delay) value takes effect. The enabled delay value is ignored in this case.
- With per-group SafeMode, including Auto-on SafeMode, the SafeMode-specific eradication delay applies to the SafeMode protected snapshots of the protection group.

Extending or Decreasing an Eradication Pending Period

Except for file systems, if the eradication pending period is increased, items already pending eradication immediately inherit the new pending period, and if the eradication pending period is decreased, items pending eradication keep their higher pending period.

File systems are an exception, as the eradication pending period for a destroyed file system is not affected by a later increase or decrease of an enabled delay or disabled delay setting.

The disabled eradication delay cannot be decreased when SafeMode is enabled, as a protection against accidental or other deletion of important data.

Eradication Delay Settings after an Upgrade

When upgrading from an earlier version of Purity that has only one eradication delay setting, not both an enabled delay and an disabled delay, the eradication delay value after the upgrade depends on whether SafeMode is enabled or not.

If SafeMode is enabled at the time of the upgrade, then after the upgrade, both the enabled delay and the disabled delay are set to the pre-upgrade eradication delay setting.

If SafeMode is not enabled before the upgrade, then after the upgrade, the enabled delay is set to the pre-upgrade eradication delay setting and the disabled delay is set to the higher of the pre-upgrade eradication delay setting and the post-upgrade default of 8 days.

Manual Eradication Status

Use the **`purearray eradication-config list`** command to view the manual eradication status as well as the current eradication delays. Manual eradication status can have the following values:

- `all-enabled` (default): Destroyed objects can be eradicated by `pureuser` or other array admin.
- `all-disabled`: Destroyed objects cannot be eradicated by `pureuser` or other array admin, and eradication of an object can only occur through the expiration of the eradication pending period for that object.

The manual eradication status cannot be altered by the user.

Network Bandwidth Throttling

Once two arrays are connected, optionally configure **`purearray throttle --connect`** to regulate when and how much data should be transferred between the arrays. The **`purearray throttle --connect`** command must be used with at least one of the following options: **`--default-limit`**, **`--window-limit`**, or **`--window`**. Inbound I/O is not affected by the throttling setting because arrays with inbound I/O (replicated to) can also act as outbound I/O (replicated from).

Network bandwidth throttling on the local array sets a network bandwidth cap for the amount of outbound data, measured by number of bytes per second, that can be transferred from the local array to a remote array before network bandwidth throttling occurs.

Two different network bandwidth limits can be set:

- The **`--default-limit`** option sets a default maximum network bandwidth threshold for outbound traffic.

  and/or

- The **`--window-limit`** option sets a maximum network bandwidth threshold for outbound traffic during the specified **`--window`** time range.

If both thresholds are set, the “window” limit overrides the “default” limit.

The maximum threshold must be between one megabyte per second and four gigabytes per second. The maximum threshold value represents an average data rate, so actual data transfer rates can fluctuate slightly above the configured limit.

The `--window-limit` setting overrides the `--default-limit` setting.

The `--default-limit` and `--window-limit` values are specified as integers, followed by one of the suffix letters `M` or `G`, denoting 512-byte sectors MiB and GiB, respectively.

The `--window` start and end times must be set on the hour, and the time can be specified in 12-hour or 24-hour format.

For example, run the following command to set a default network bandwidth cap of 4 gigabytes per second of data to be transferred from the local array to array `ARRAY1`, except between the hours of 11:00am and 3:00pm, when the network bandwidth limit drops to 2 gigabytes per second.

```
$ purearray throttle --connect --default-limit 4gb
--window 11am-3pm --window-limit 2gb ARRAY1
```

To clear the `--default-limit`, `--window-limit`, or `--window` settings, use an empty string ("").

To completely stop the data transfer process, set the `default-limit` or `window-limit` option to "0". During this time, all in-progress and scheduled data transfer processes are aborted.

For example, run the following command to stop all data transfers from the local array to `ARRAY1` between 9:00am and 5:00pm.

```
$ purearray throttle --connect --window 9am-5pm --window-limit 0 ARRAY1
```

Bandwidth limit changes take effect immediately.

The configured bandwidth limit is applied to each network interface individually. So for example, in the most common configuration where replication uses two network interfaces per controller, setting a limit of 10MB/s allows up to 20MB/s of bandwidth (only the primary interfaces are active.)

Less commonly, replication can be configured to use a single "replbond" interface which uses two physical interfaces in an active/passive configuration. In this case, the throttle would be applied to the single replbond interface. So in this case a 10MB/s configured limit would result in 10MB/s maximum throughput.

End User Agreement (EULA)

The Pure Storage End User Agreement (EULA) represents a contract between Pure Storage and users of Pure Storage products. The most recent version of the agreement governs use of the FlashArray hardware and software and can be found at **`http://www.purestorage.com/legal/productenduserinfo.html`**.

Run the `purearray eula list` command to view the terms of the Pure Storage End User Agreement. Include the `--acceptance` option to view the acceptance status for the Pure Storage End User Agreement. If the agreement has not been accepted the Accepted column will display a dash (-).

Run the `purearray accept` command to accept the terms of the agreement. Only array administrators (i.e., users with the Array Admin role) have the necessary permissions to run the command.

Note on Array Time

The installation technician sets the proper time zone for an array when it is installed. During operation, arrays maintain time synchronization by interacting with a Network Time Protocol (NTP) server (by default, time.purestorage.com). The `purearray setattr --ntpserver NTP-SERVER` command can be executed to specify an alternate NTP server by IP address or hostname.

Managing Capacity on Cloud Block Store Arrays

The `purearray cloud-capacity` commands display and manage array capacity onCloud Block Store arrays. Capacity values are shown in bytes and represents the total nominal capacity of the array, which

might not match the installed capacity. The `purearray cloud-capacity list` command displays the array capacity, requested capacity (if any), and array status. This example shows information for an array that is expanding to a new requested capacity.

```
$ purearray cloud-capacity list
Requested Capacity     Current Capacity    Status      Details
21990232555520         10995116277760      expanding
```

When appropriate, the Details column displays progress information and error messages, if any.

The `purearray cloud-capacity list --supported` command displays the capacity value supported on the array:

```
purearray cloud-capacity list --supported
```

```
Supported Capacity
10995116277760
21990232555520
43980465111040
87960930222080
```

The **purearray cloud-capacity setattr** command requests a new capacity setting for the array:

```
$ purearray cloud-capacity setattr --requested-capacity 21990232555520
Requested Capacity    Current Capacity    Status      Details
21990232555520        10995116277760      expanding
```

A Status of `idle` indicates the array is not undergoing a capacity change currently. Reducing array capacity is not supported in this release.

Performing Hardware Upgrades on Cloud Block Store Arrays

Beginning with Purity version 6.8.3, Cloud Block Store provides the ability to perform self-service hardware non-disruptive upgrades. This allows for controllers of the same family to be upgraded without downtime. This functionality does not extend to controllers across different VM families. During the upgrade process, the system runs multiple checks to make sure everything is in shape to finish the upgrade successfully.

Use the **purearray cloud-config model list --supported** command to list the supported model hop(s) for your Cloud Block Store array.

```
purearray cloud-config model list --supported
Supported Models
CBS-V20MP2R2
```

To initiate the hardware upgrade, run the **purearray cloud-config model setattr --requested-model** command.

```
purearray cloud-config model setattr --requested-model CBS-V20MP2R2
Current Model Requested Model Status Step Details
CBS-V10MP2R2 CBS-V20MP2R2 upgrading Preparing for running pre-upgrade-check
```

At any point during the upgrade process, you can run **`purearray cloud-config model list`** to see details about the status of the upgrade.

Pre-upgrade checks are the first series of checks through the hardware upgrade procedure. The overall array health is evaluated to ensure successful completion. If all the checks are successful, the system will proceed to the next stage. Should one or more of the checks fail, Hardware upgrade is paused with warning. To help with resolution, the system will identify the failed checks and offer possible next steps.

While paused in this stage, the hardware upgrade can be aborted. To abort an upgrade, run **`purearray cloud-config model setattr --requested-model ''`**.

A failed check is generally designed to be an indication that the upgrade should not proceed. However, in some cases it may be necessary to override a check and proceed with the upgrade. To override a check, use the `--override` ***CHECK_NAME*** option. The --override option can be used on pre-upgrade checks, mid-upgrade checks, and post-upgrade checks.

An example below shows two failed checks, one of which can be overridden (in this case it is the `HostPathsCheck`). This is indicated by the flag `Overridable: True`

```
purearray cloud-config model list
Current Model Requested Model Status Step Details
CBS-V10MP2R2 CBS-V20MP2R2 paused pre-upgrade-check Upgrade has been paused: Finished
pre-upgrade, 16 checks in total, 2 failed (CapacityStepCheck, HostPathsCheck).

CapacityStepCheck
Message: Current disk capacity is not sufficient for upgrading to the CBS-V20MP2R2
model.
Recommended Action: Please abort the hardware upgrade, perform a capacity upgrade with
a required step of at least 7696581394432, then start the hardware upgrade again.
Overridable: False

HostPathsCheck
Host Protocol VLAN Paths to CT0 Paths to CT1 CT0 IOps CT1 IOps Status
h1 iSCSI - 1 0 74.078 33.711 Error! No paths to CT1.

Message: Host(s) only have paths to one controller.
Recommended Action: Manually verify that all hosts are symmetrically connected to both
controllers.
```

```
Override Warning: Since both controllers are rebooted during a Purity upgrade, ignoring
this check will likely lead to an outage for the host(s) in question. Please ensure all
hosts have paths to both controllers before proceeding.
Overridable: True
To proceed with the upgrade, either:
1) Retry upgrade checks - Run "purearray cloud-config model setattr --requested-model
CBS-V20MP2R2 [--override <CheckName>]"
2) Abort the upgrade - Run "purearray cloud-config model setattr --requested-model ''"
3) Contact Pure Technical Services for assistance
```

To override the HostPathsCheck in this example, run the command:

```
purearray cloud-config model setattr --requested-model CBS-V20MP2R2 --override
HostPathsCheck
```

From all the checks performed, one failed and one was overridden. The system no longer informs the user of the next possible steps to take for the overridden check.

Once all pre-upgrade checks pass, the upgrade proceeds for the cloud resources and controllers. Between the first and second controller upgrade, it runs mid-upgrade checks. After the upgrade is complete, it runs post-upgrade checks.

Attention: Once cloud resources are affected, it is no longer possible to abort the upgrade process. If an upgrade is paused after failing mid-upgrade checks, it remains possible to retry, and optionally to retry with overrides, using the CLI.

It is possible the upgrade will complete with warning. That indicates the array is running on the new model with upgraded hardware, all mandatory hardware changes have successfully completed and the array is running on more performant resources; however, the final verification step has failed.

If the failed check cannot be resolved independently, it is recommended to contact Pure Support for further assistance.

Performing a Factory Reset on Cloud Block Store Arrays

A factory reset removes all applications and data that were added to a Cloud Block Store array. Before you can perform a factory reset on the array, prepare the array to satisfy the following pre-requisites:

- Destroy and eradicate all file systems, file system snapshots, volumes, and volume snapshots on the Cloud Block Store array.
- Disconnect all connected peer FlashArrays and targets.
- Ensure that the Cloud Block Store array can transmit log and diagnostic information via the phone home facility.

Note: A factory reset is initiated from a Cloud Block Store appliance.

To perform a factory reset:

1. Authorize a factory reset by using the `purearray factory-reset-token create` command to generate a token.

   This is a one-time token that can be used to reset the Cloud Block Store array; a new token overrides the existing token. For example:

   ```
   $ purearray factory-reset-token create
   Name Token
   vm-pureFA 4671919
   ```

   After the factory reset authorization, you cannot configure the Cloud Block Store array, such as creating volumes or connecting to peer FlashArrays.

2. Execute the factory reset by using the `purearray erase` command with the following required options.

   - Specify the token generated in the previous step with the --factory-reset-token option.
   - Specify the –eradicate-all-data option to confirm the factory reset.

   For example:

   ```
   $ purearray erase --factory-reset-token 4671919 --eradicate-all-data
   Name
   vm-pureFA
   ```

After the factory reset authorization, you can execute the factory reset at any time, but not when the factory reset is canceled. To cancel a factory reset, run the `purearray factory-reset-token delete` command.

To `create`, `display`, or `delete` a factory reset token, specify the create, list, or delete subcommand, respectively, with the `purearray factory-reset-token` command, as shown in the following example.

```
$ purearray factory-reset-token create
Name Token
vm-pureFA 4441611
$ purearray factory-reset-token list
Name Token
vm-pureFA 4441611
$ purearray factory-reset-token delete
Name Token
vm-pureFA -
```

Suspending Cloud Block Store Arrays on AWS

This feature allows you to suspend Cloud Block Store arrays on AWS when the data is not accessed.

To suspend a Cloud Block Store array on AWS:

1 Display the expected resumption time of the array by running the `purearray list --expected-resume-time` command.

For example,

```
$ purearray list --expected-resume-time
Output
6h
```

2 Suspend the array by running the `purearray suspend --expected-resume-time` command with the expected resumption time obtained from the previous step.

The following example suspends the ARRAY1 array with the expected resumption time of six hours.

```
$ purearray suspend --expected-resume-time 6h ARRAY1
Status
```

```
Array has started suspension and will shut down soon.
```

The suspension process will succeed only when the array is fully resumed and does not have other suspension or resumption processes in progress.

To suspend a Cloud Block Store array on AWS, allowing an unsafe suspension even when CMK is used for EBS encryption, run the `purearray suspend --expected-resume-time` ***`EXPECTED RESUME TIME`*** `--allow-unsafe-suspend-with-cmk` ***`ARRAY`*** command, specifying the desired resumption time and the array being suspended.

Exceptions

The `purearray enable security-token`, `purearray replace security-token`, and `purearray setattr --ntpserver` commands are not supported on Cloud Block Store.

Examples

Note: Some commands are shown on two lines because of space limitations. Commands are to be entered on a single line.

Example 1

```
purearray enable phonehome
```

Enables automatic hourly transmission of log contents to the Pure Storage Technical Services web site.

Example 2

```
purearray diagnose --phonehome
```

Takes a snapshot of performance, configuration, and hardware status and transmits it to Pure Storage Technical Services via the phonehome channel.

Example 3

```
purearray list --phonehome
```

Displays the current state of the phonehome automatic hourly log transmission facility (enabled or disabled).

Example 4

```
purearray list --space --historical 3h --csv
```

On purchased arrays, displays a CSV output of the amount of usable physical storage on the array and the amount of storage occupied by data and metadata over the past 3 hours.

On subscription storage, the CSV output includes metrics on effective used capacity over the past 3 hours.

Example 5

```
purearray list --throttle --connect
```

Displays network bandwidth limits, if set, for all connected arrays.

Example 6

```
purearray monitor --repeat 20 --interval 10
```

Displays real-time performance data for the whole array. Twenty (20) point-in-time updates are displayed, with each update taken every ten (10) seconds.

Example 7

```
purearray monitor --historical 24h --csv
```

Displays a CSV output of historical performance data for the local array over the past 24 hours.

Example 8

```
purearray monitor --protocol-group file
```

Displays real-time performance data for the whole array, for file services only.

Example 9

```
purearray phonehome --send-today
```

Initiates immediate transmission of the current day's event logs to Pure Storage Technical Services via the phonehome network channel.

Example 10

```
purearray remoteassist --connect
```

Enables a Pure Storage Technical Services representative to initiate a remoteassist session with the array.

Example 11

```
purearray rename NEWARRAYNAME
```

Changes the name of an array to **`NEWARRAYNAME`**.

Example 12

```
purearray setattr --ntpserver MyNTPServer1.com,MyNTPServer2.com
```

Assigns the NTP servers **`MyNTPServer1.com`** and **`MyNTPServer2.com`** as the array's sources for reference time. Supersedes any previous NTP server assignments.

Example 13

```
purearray list --connect --path
```

Displays the connection path details on the array, including the array name, local port, local address, remote port, remote address, connection status, and transport type.

Example 14

```
purearray list --connect --controller
```

Displays the number of connected paths for each controller on the connected arrays, in addition to the array name, transport type, local controller, and remote controller.

Example 15

```
purearray list --connection-key
```

Displays the connection key of the array.

Example 16

```
purearray connect --management-address 10.78.0.188 --type async-replication
                  --connection-key
```

Prompts the user to enter the connection key of the target array. Connects the local array to remote array `10.78.0.188` for asynchronous replication using the connection key.

Example 17

```
purearray disconnect 10.78.0.188
```

Disconnects array `10.78.0.188` from the local (current) array. To be run on the source array.

Example 18

```
purearray disconnect --decommissioned 10.78.0.188
```

Disconnects array `10.78.0.188` from the local (current) array. To be run on the target array when the source array is not reachable. This command fails if the source array is reachable.

Example 19

```
purearray throttle --connect --default-limit 2mb
                   --window 11pm-5am --window-limit 4mb ARRAY1
```

Sets the default maximum network bandwidth limit for outbound traffic through the network interface to 2 MB/s for array **`ARRAY1`**, and sets the maximum network bandwidth limit for outbound traffic through the network interface to 4 MB/s between 11:00pm and 5:00am.

Example 20

```
purearray eula list
```

Displays the terms of the Pure Storage End User Agreement.

Example 21

```
purearray eula list --acceptance
```

Displays the acceptance date details for the Pure Storage End User Agreement.

Example 22

```
purearray eula accept
```

Accepts the Pure Storage End User Agreement. The individual who accepts the terms of the agreement must have the authority to do so.

Example 23

```
purearray eradication-config list
```

Displays the currently configured eradication timer and manual eradication settings.

Example 24

```
purearray eradication-config setattr --disabled-delay 14d
```

Sets the length of the eradication pending period for objects protected by SafeMode to 14 days.

Example 25

```
purearray tag --cloud-provider --key key01 --value val01
```

Applies the specified tag to a public cloud.

Example 26

```
purearray untag --cloud-provider --kefy key01
```

Removes the specified tag from a public cloud.

Example 27

```
purearray default-protection list
```

Displays all default protection group lists on the array, including for the root of the array and for each pod, if any.

Example 28

```
purearray default-protection list pod1,pod2
```

Displays the default protection group lists for **`pod1`** and **`pod2`**.

Example 29

```
purearray default-protection set --pgroup pg3,pg4 ""
```

Sets the default protection group list for the root of the array to protection groups **`pg3`** and **`pg4`** (which must already exist). The contents of the previous protection group list, if any, are replaced. New pods will be created with a copy of this default protection group list, with protection groups modified with the pod name.

Example 30

```
purearray default-protection set --pgroup  "" pod2
```

Opts out of default protection for **`pod2`** by setting an empty default protection group list for that pod.

The protection groups in the previous default protection group list for the pod must be empty and unlocked. Contact Pure Storage Technical Services to opt out when a ratcheted protection group is involved.

Example 31

```
purearray list --ntp
```

Displays a list of NTP servers and the NTP symmetric key used for authentication.

Example 32

```
purearray cloud-capacity list
```

For a Cloud Block Store array, displays the current array capacity (and requested capacity, if any).

Example 33

```
purearray cloud-capacity list --supported
```

For a Cloud Block Store array, displays which capacities are supported on that array.

Example 34

```
purearray cloud-capacity setattr --requested-capacity 21990232555520
```

For a Cloud Block Store array, requests that the array capacity to changed to the specified value.

Example 35

```
purearray cloud-config model list
```

Provides information about the Cloud Block Store array upgrade, including current model, requested model, status, step in the upgrade process, and relevant details.

Example 36

```
purearray cloud-config model list --supported
```

Provides the supported upgrade path for the Cloud Block Store array.

Example 37

```
purearray cloud-config model setattr --requested-model CBS-V20MP2R2
```

For a Cloud Block Store array, requests an upgrade to model **`CBS-V20MP2R2`**.

Example 38

```
purearray cloud-config model setattr --requested-model CBS-V20MP2R2 --override
HostIOCheck
```

For a Cloud Block Store array, requests an upgrade to model **`CBS-V20MP2R2`** while also overriding potential HostIOCheck warnings that could prevent the upgrade.

See Also

purealert, puredns
puredir, purehgroup, purehost, purepgroup, purepod, purevol

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

pureaudit

pureaudit, pureaudit-list – displays and manages audit records

Synopsis

```
pureaudit list [ --csv | --nvp ] [--notitle] [--page] [--page-with-
token] [--token TOKEN] [--raw]
[--filter FILTER] [--limit LIMIT] [--sort SORT] [--context REMOTE]
```

Options

`--context REMOTE`

Used to specify the remote array on which a command is processed. ***REMOTE*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

Options that control display format:

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--page-with-token`

Used to paginate output with a continuation token (**`--token TOKEN`**). Results are printed in `stdout` and the token is printed in `stderr`.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Description

Purity//FA generates *audit trail* records, which represent administrative actions performed by a Purity//FA user to modify the configuration of the array. For example, creating, destroying, or eradicating a volume creates an audit record.

The **`pureaudit list`** command displays a list of audit records. For example,

```
pureaudit list
ID  Time                    User     Command  Subcommand Name   Arguments
250 2019-03-25 03:00:40 PDT root     purevol  disconnect vol3   --host host3
251 2019-03-25 03:00:41 PDT root     purevol  connect    vol3   --host host3
252 2019-02-27 12:57:30 PST pureuser purehost setattr    host1  --addwwnlist
253 2019-03-15 23:45:42 PDT os76     purevol  disconnect vol5   --host host2
254 2019-02-24 18:56:21 PST pureuser purevol  create     vol4   --size 10T
255 2018-12-09 17:14:14 PST root     purevol  destroy    vol1
256 2018-12-09 17:14:23 PST root     purevol  eradicate  vol1
```

The **`pureaudit list`** command displays the following audit record details:

`ID`
: Unique number assigned by the array to the audit record. ID numbers are assigned to audit records in chronological ascending order.

`Time`
: Date and time the action occurred to trigger the audit event.

`User`
: User name of the Purity//FA user who issued the command.

`Command`
: Purity//FA CLI command that was issued.

`Subcommand`
: Purity//FA CLI subcommand that was issued.

`Name`
: Name of the object on which the command was issued.

`Arguments`
: Options that were included with the Purity//FA CLI command that was issued.

Examples

Example 1

```
pureaudit list --filter "name = 'hgroup*'"
```

Displays a list of audit records of host groups with the prefix of "hgroup" in the host group name.

Exceptions

None.

See Also

purealert, purearray

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purecert

purecert, purecert-construct, purecert-create, purecert-imported-create, purecert-delete, purecert-list, purecert-imported-setattr, purecert-self-signed-create, purecert-self-signed-setattr – creates and manages FlashArray SSL certificates.

purecert-group-add, purecert-group-create, purecert-group-delete, purecert-group-remove, purecert-group-list - creates and manages certificate groups.

purecert-csr-create - creates a certificate signing request.

Synopsis

purecert csr create [--common-name ***COMMON_NAME***] [--country ***COUNTRY***] {--certificate-signing-request} [--email ***EMAIL***] [--locality ***LOCALITY***] [--organization ***ORG***] [--organizational-unit ***ORG_UNIT***] [--san ***SAN***[,***SAN***...]] [--state ***STATE***] ***NAME***

purecert delete ***NAME***

purecert group add --certificate ***NAME[,NAME...]*** ***CERTIFICATE-GROUPS...***

purecert group create ***CERTIFICATE-GROUPS...***

purecert group delete ***CERTIFICATE-GROUPS...***

purecert group list [--uses] [--cli | --csv | --notitle | --nvp | --raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--offset ***OFFSET***] [***CERTIFICATE-GROUPS...***]

purecert group remove --certificate ***NAME[,NAME...]*** ***CERTIFICATE-GROUP***

purecert list [--certificate | --intermediate-certificate | --group | --uses] [--cli | --csv | --nvp] [--notitle] [--page] [--raw] ***NAME***

purecert imported create [--intermediate-certificate] [--key] [--passphrase] ***NAME***

purecert imported setattr [--intermediate-certificate] [--key] [--passphrase] ***NAME***

purecert self-signed create [--common-name ***COMMON_NAME***] [--country ***COUNTRY***] [--days ***DAYS***] [--email ***EMAIL***] [--key-algorithm ***KEY_ALGO***] [--keysize ***KEY_SIZE***] [--locality ***LOCALITY***] [--organization ***ORG***] [--organizational-unit ***ORG_UNIT***] [--san ***SAN***[,***SAN***...]] [--state ***STATE***] ***NAME***

purecert self-signed setattr [--common-name ***COMMON_NAME***] [--country ***COUNTRY***] [--days ***DAYS***] [--email ***EMAIL***] [--key-algorithm ***KEY_ALGO***] [--key-size ***KEY_SIZE***] [--locality ***LOCALITY***] [--new-key] [--organization ***ORG***] [--organizational-unit ***ORG_UNIT***] [--san ***SAN***[,***SAN***...]] [--state ***STATE***] ***NAME***

Arguments

CERTIFICATE GROUPS
The name of one or more certificate groups to be created, modified, or deleted.

NAME
Name of the certificate to be created, modified, or displayed.

Options

-h | --help
Can be used with any command or subcommand to display a brief syntax description.

--certificate (**purecert list** only)
Lists the certificates.

--common-name
Fully qualified domain name (FQDN) of the current array. For example, the common name for `https://purearray.example.com` is `purearray.example.com`, or `*.example.com` for a wildcard certificate. The common name can also be the management IP address of the array or the short name of the current array. Common names cannot have more than 64 characters.

--country
Country name. Two-letter ISO code for the country where the organization is located.

--days

Number of valid days for the self-signed certificate being generated. If not specified, the self-signed certificate expires after 3650 days. Only used with self-signed certificates.

`--email`

Email address used to contact the organization.

`--group` (**`purecert list`** only)

Displays the names of groups and certificates associated with each group.

`--intermediate-certificate`

Displays, exports, or imports the intermediate certificate.

`--key`

Imports the private key. Paste the key, press **`Enter`**, and then press **`Control-d`**.

`--key-algorithm` ***`KEY_ALGO`***

Specifies the signature algorithm used to generate a key. Valid values are `ec`, `ed25519`, `ed448`, and (default) `rsa`. Only used with self-signed certificates.

`--key-size` ***`KEY_SIZE`***

Specifies key size in bits. Valid values are `256`, `456`, `1024`, `2048`, and `4096`, depending on the specified key algorithm. A key size smaller than 2048 is considered insecure with RSA. Only used with self-signed certificates.

`--locality`

The city where the organization is located.

`--new-key`(**`purecert setattr --self-signed`** only)

Generates a new private key when creating the self-signed certificate. If a new private key is not generated, the certificate uses the existing private key.

`--organization`

Full and exact name in which the organization is legally registered. Organization name should not be abbreviated and should include suffixes such as Inc, Corp, or LLC.

`--organizational-unit`

Name of the department within the organization that is managing the certificate.

`--passphrase`

Passphrase used to decrypt the private key.

`--san`

Subject alternative names. Supported types include IP addresses, URIs and DNS names.

`--state`

Full name of the state or province where the organization is located.

`--uses` (**`purecert list`**)

Displays the names of certificates and any services that are currently using each certificate.

`--uses` (**`purecert group list`**)

Displays the names of certificate groups and any services that are currently using each certificate group.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Description

The **`purecert`** command allows you to perform tasks to create and manage your Pure Storage SSL certificates. Purity//FA creates a self-signed certificate and private key when you start the system for the first time. You can use the default certificate, change the certificate attributes, create a new self-signed certificate, or import an SSL certificate signed by a certificate authority.

To list the attributes of your certificates, run the `purecert list` command. To display the names of certificate groups and the certificates associated with each group, run the `purecert group list` command.

Self-signed Certificate

You can create a new self-signed certificate on the current array by using the `purecert self-signed create` command. The command generates a certificate object that stores the private key and a certificate containing the public key. You can then submit this certificate to a certificate authority (CA) to obtain a CA certificate. To display the self-signed certificate, use the `purecert list --certificate` command.

When you create a self-signed certificate to replace the current certificate, also include the **`--key-size`** option to generate a private key with the specified key size.

Optionally, also include the **`--key-algorithm`** option to specify the signature algorithm used to generate the key. The following signature algorithms are supported:

- `ec`: An elliptic curve signing algorithm using SECP256R1. Key size defaults to 256 bits, which is the only supported key size value for this algorithm.
- `ed25519`: An elliptic curve signing algorithm using EdDSA and Curve25519. Key size defaults to 256 bits, which is the only supported key size value for this algorithm.
- `ed448`: An EdDSA signature algorithm using SHAKE256 and Curve448. Key size defaults to 456 bits, which is the only supported key size value for this algorithm.
- `rsa` (default): The RSA algorithm. Supported key sizes are 1024, 2048 (default), and 4096. The key size 1024 is not considered secure.

By default, self-signed certificates are valid for 3650 days. Include the **`--days`** option to change the valid number of days.

Certificate Signing Request

Signing in with a certificate from a certificate authority (CA) involves importing an SSL certificate issued by the CA.

To obtain a CA certificate, you must first construct a certificate signing request (CSR) on the array. The CSR represents a block of encrypted data specific to your organization. Run the `purecert csr create` command to construct a CSR.

You can change the certificate attributes when you construct the CSR; otherwise, Purity//FA will reuse the attributes of the current certificate (self-signed or imported) to generate the new one. Note that the certificate attribute changes will only be visible after you import the signed certificate from the CA.

Send the CSR to a certificate authority for signing. The certificate authority returns the SSL certificate for you to import. Verify that the signed certificate is PEM formatted (Base64 encoded), includes the **`"-----BEGIN CERTIFICATE-----"`** and **`"-----END CERTIFICATE-----"`** lines, and does not exceed 3000 characters in length. Press `Enter` and then `Control-d` to exit the interactive session.

IIf the intermediate certificate is bundled with the SSL certificate, run the **`purecert imported setattr`** command to import the certificates. If the certificates are sent separately, run the **`purecert imported setattr --intermediate-certificate`** command to import the SSL certificate and its accompanying intermediate certificate. The certificates are entered interactively.

If the certificate is signed with the CSR that was constructed on the current array and you did not change the private key, you do not need to import the key. However, if the CSR was not constructed by the current array, include the **`--key`** option to import the private key. If the private key is encrypted, include the **`--passphrase`** option with **`--key`**.

Changing the Certificate

To change the certificate attributes, run the **`purecert imported setattr`** or **`purecert self-signed setattr`** command. When you change the attributes of a self-signed certificate, Purity//FA replaces the existing certificate with a new certificate, along with its specified attributes. Certificate attributes include organization-specific information, such as country, state, locality, organization, organizational unit, common name, and email address.

You can also generate a new private key for the current certificate by using the **`--new-key`** option. To change the length (in bits) of the private key, include the **`--key-size`** option with the **`--new-key`** option. To change the key signature algorithm of the private key, include the **`--key-algorithm`** option with the **`--new-key`** option.

Certificate Administration

To import the certificate signed by a certificate authority (CA), run the `purecert imported setattr` command.

To export a certificate or an intermediate certificate, such as for backup purposes, run the `purecert list --certificate` or `purecert list --intermediate-certificate` command, respectively.

Certificate Groups

Certificate groups allow you to organize your certificates for different purposes, such as separating certificates used for internal services from those used for external-facing applications. To manage certificate groups, use the `purecert group` command suite. To create new certificate groups, run the `purecert group create` command. This initializes empty groups that you can populate with certificates. To delete certificate groups, use the `purecert group delete` command. Note that you cannot delete groups that are in use.

To add certificates to groups, run the `purecert group add` command. This associates the specified certificates with the given groups. To remove certificates from groups, run the `purecert group remove` command. While you can add or remove certificates from groups, you cannot remove the last certificate from a group if the group is in use. The group must always contain at least one certificate if it is actively being utilized (for example by DSX or Fusion). To view which certificates belong to which groups, run either the `purecert group list` or the `purecert list --group` command. This displays the names of groups and the certificates associated with each group.

Examples

Example 1

```
purecert self-signed setattr cert_1 --common-name db.example.com --country US --state
CA --locality 'Mountain View' --organization 'Example, Inc.'
```

Modifies the existing self-signed certificate `cert_1` with the common name `db.example.com` (the FQDN of the current array) and various attributes. Because the private key is not specified, the new self-signed certificate will use the existing private key. Because key algorithm and key size are not specified, the defaults `rsa` and `2048` are used.

Example 2

```
purecert csr create cert_2 --common-name app.example.com --san 1.2.3.4,app.example.com
```

Constructs a certificate signing request (CSR) with the common name `app.example.com` (the FQDN of the current array) and subject alternative names (SANs) `1.2.3.4` and `app.example.com`. The CSR includes all existing attributes. Because key algorithm and key size are not specified, the defaults `rsa` and `2048` are used.

Example 3

```
purecert imported setattr cert_3 --intermediate-certificate
```

Imports the certificate signed by a certificate authority (CA) along with its intermediate certificate. Because key algorithm and key size are not specified, the defaults `rsa` and `2048` are used.

Example 4

```
purecert self-signed create DSM_cert --common-name c1v7w
purecert list DSM_cert --certificate
```

Creates a new `DSM_cert` self-signed certificate with the common name `c1v7w`, and then displays the certificate. Because key algorithm and key size are not specified, the defaults `rsa` and `2048` are used.

Example 5

```
purecert self-signed create my_cert
purecert csr create my_cert --common-name c1v7w
purecert imported setattr my_cert
```

Creates a new certificate, `my_cert` containing a private and public key pair. Then, it generates the certificate signing request (CSR) with the common name `c1v7w`. Since the key algorithm and key size are not specified, the defaults (`rsa` and `2048`) are used. Finally, the certificate `my_cert` is updated by importing the newly signed certificate.

Example 6

```
purecert self-signed create --key-algorithm ec EC-cert
```

Creates a new self-signed certificate `EC-cert` using the EC key signature algorithm. Key size defaults to `256`.

Example 7

```
purecert list --group test_cert
```

Displays a list of memberships for the certification `test_cert`, showing each group it belongs to.

Example 8

```
purecert group create test_group
```

Creates a new empty certificate group `test_group`. Certificates can now be added to it.

Example 9

```
purecert group add --certificate test_cert test_group
```

Adds the certificate `test_cert` to the certificate group `test_group`.

Example 10

```
purecert group remove --certificate test_cert test_group
```

Removes the certificate `test_cert` from the certificate group `test_group`.

Example 11

```
purecert group delete test_group
```

Deletes the certificate group `test_group`.

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

pureconfig

pureconfig – displays the commands required to reproduce the current FlashArray configuration of hosts, host groups, pods, protection groups, volumes, volume groups, connections, file systems and directories, alert, network, policies, and support.

Synopsis

```
pureconfig list [--all | --object | --system]
```

Options

`-h | --help`
: Can be used with any command or subcommand to display a brief syntax description.

`--all`
: Lists all configurations of the FlashArray.

`--object`
: Lists the object configuration of the FlashArray including hosts, host groups, pods, protection groups, volumes, volume groups, and connections, as well as file systems and directories if file services are enabled.

`--system`
: Lists the system configuration of the FlashArray including alert, network, policies, and support.

Description

Displays an array's current configuration of volumes, hosts, shared and private connections, file systems and directories, administrative network parameters, alert email addresses, array parameters, and policies in the form of CLI commands that would be required to reproduce the configuration on a newly-installed or otherwise unconfigured array. (The output does not contain delete or destroy subcommands).

The output of this command can be captured on the administrative workstation and used as a script to configure a previously unconfigured array identically to the array on which the command is executed.

Executing the **pureconfig list** command is roughly equivalent to executing the following commands in sequence:

```
purevol list --cli
purehost list --cli
purehgroup list --cli
purehost list --connect --cli
purehgroup list --connect --cli
purenetwork list --cli
purealert list --cli
purearray list --cli
```

To list all the configurations of the FlashArray, specify the **--all** option with the **pureconfig list** command. Specify the **--object** option to show the object configuration or the **--system** option to show the system configuration.

Examples

Example 1

```
pureconfig list --object
```

Displays the object configuration of the FlashArray including hosts, host groups, pods, protection groups, volumes, volume groups, and connections, as well as file systems and directories if file services are enabled.

Example 2

```
pureconfig list --system
```

Displays the system configuration of the FlashArray including alert, network, policies, and support.

See Also

purealert, purearray, purehgroup-list, purehost-list, purenetwork, purevol-list

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puredir

puredir, puredir-create, puredir-delete, puredir-rename, puredir-list, puredir-monitor – manages the creation, naming, deletion, listing, and monitoring of managed directories

puredir-export-create, puredir-export-delete, puredir-export-list – manages the creation and displaying of file exports

puredir-lock – manages releasing of file locks

puredir-quota-add, puredir-quota-remove, puredir-quota-list – manages the creation and displaying of directory quotas

puredir-snapshot-create, puredir-snapshot-setattr, puredir-snapshot-rename, puredir-snapshot-destroy, puredir-snapshot-recover, puredir-snapshot-eradicate, puredir-snapshot-list – manages the creation, modification, destruction, and listing of directory snapshots

puredir-snapshot-add, puredir-snapshot-remove – manages the adding and removal of a snapshot policy

Synopsis

puredir create --path ***PATH NAME***

puredir delete ***NAME***...

puredir list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--file-system ***FILE_SYSTEM_NAME***] [--pending | --pending-only] [--space] [--total] [--historical { 1h | 3h | 24h | 7d | 30d | 90d | 1y }] [--usage] [***NAME***...]

puredir lock nlm-reclamation create

puredir monitor [--csv] [--notitle] [--page] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--protocol { nfs | smb }] [--interval ***INTERVAL***] [--repeat ***REPEAT***] [--size] [--total] [--historical { 1h | 3h | 24h | 7d | 30d | 90d | 1y }] [***NAME***...]

puredir rename ***OLD-NAME NEW-NAME***

puredir export create --dir ***DIR*** --policy ***POLICY*** ***EXPORT***

puredir export delete [--policy ***POLICY***] [--dir ***DIR***] ***EXPORT...***

puredir export disable [--policy ***POLICY***] [--dir ***DIR***] ***EXPORT...***

puredir export enable [--policy ***POLICY***] [--dir ***DIR***] ***EXPORT...***

puredir export list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--dir ***DIR***] [--pending | --pendingonly] [***EXPORT...***]

puredir export rename [--policy ***POLICY***] [--dir ***DIR***] ***OLD-NAME*** ***NEW-NAME***

puredir quota add --policy ***POLICY*** [--ignore-usage] ***DIR...***

puredir quota list [--csv | --nvp] [--notitle] [--page] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [***NAME...***]

puredir quota remove [--policy ***POLICY***] ***DIR...***

puredir snapshot add --policy ***POLICY*** ***DIR...***

puredir snapshot create --client-name ***CLIENT_NAME*** [--keep-for ***KEEP_FOR***] [--suffix ***SUFFIX***] ***DIR***

puredir snapshot destroy ***NAME...***

puredir snapshot eradicate ***NAME...***

puredir snapshot list [--csv | --nvp] [--notitle] [--raw] [--filter ***FILTER***] [--page] [--limit ***LIMIT***] [--sort ***SORT***] [--pending | --pending-only] [--space] [***NAME...***]

puredir snapshot recover ***NAME...***

puredir snapshot remove --policy ***POLICY*** ***DIR...***

puredir snapshot rename ***OLD-NAME*** ***NEW-NAME***

puredir snapshot setattr [--policy ***POLICY*** | --keep-for ***KEEP_FOR***] [--client-name ***CLIENT_NAME***] [--suffix ***SUFFIX***] ***NAME...***

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--client-name` ***CLIENT_NAME***

Sets the client visible portion of the snapshot name.

`--dir` ***`DIR`***

Specifies a comma-separated list of one or more managed directory names.

`--file-system` ***`FILE_SYSTEM_NAME`***

Specifies a comma-separated list of one or more file systems.

`--historical` ***`TIME`***

Displays historical data at the given resolution. Valid time ranges include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year.

`--ignore-usage`

Attaches the quota policy to the directory even when the directory usage exceeds the quota limit.

`--interval` ***`INTERVAL`***

Sets the number of seconds between displays of real-time performance data. At each interval, the system displays a point-in-time snapshot of the performance data. If omitted, the interval defaults to every five seconds.

`--keep-for` ***`KEEP_FOR`***

When creating or modifying a snapshot, specifies the time period in which the snapshot is retained before it is eradicated, specified in the format **`N[m|h|d|w|y]`** for example **`15m`**, **`1h`**, **`2d`**, **`3w`**, **`4y`**. The minimum value is five minutes (**`5m`**). If omitted, the snapshot will be kept until manually destroyed.

`--path` ***`PATH`***

Specifies the full directory path.

`--pending`

Includes destroyed managed directories or snapshots that are in the pending eradication state. If not specified, items that are pending eradication are not included.

`--pending-only`

Includes only destroyed managed directories or snapshots that are in the pending eradication state.

`--policy` ***`POLICY`***

The export or quota policy name.

`--protocol` ***`PROTOCOL`***

Specification of protocol. Valid values are **`nfs`** or **`smb`**. If omitted, all available protocols will be displayed.

`--repeat` ***`REPEAT`***

Use with **`puredir monitor`** to specify the number of times to repeat displaying real-time performance data. If omitted, the repeat count is 1.

`--size`

Displays the average I/O sizes per read and write operation.

`--space`

Includes space reporting.

`--suffix` ***`SUFFIX`***

Optionally, when creating or modifying a manual snapshot, specify a suffix string to replace the unique number that Purity//FA creates for the directory snapshot.

`--total`

Follows output lines with a single line containing aggregated column data after displaying realtime updates.

`--usage`

Displays usage, the amount of data currently stored by users.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`
: Displays only the rows that meet the filter criteria specified.

`--limit`
: Limits the size of the list output to the specified maximum number of rows.

`--sort`
: Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Arguments

`DIR`
: The name of the managed directory.

`EXPORT`
: The name of the managed directory export.

`NAME`
: The name of the managed directory or the snapshot to be created, modified, or displayed.

`NEW-NAME`
: Name by which the managed directory or the snapshot is to be known after the command executes.

`OLD-NAME`
: Current name of the managed directory or the snapshot to be renamed.

`POLICY`
: The name of the directory quota policy to be added to a managed directory.

Description

The **`puredir`** command manages the creation, naming, and deletion of managed directories, directory quotas, directory snapshots, and file exports. The path for a managed directory can be up to eight levels deep, counting the root directory as the first level.

Managed Directories

The `puredir create` command creates a file system directory which is a managed directory. Specify `--path` and the full path for the new directory, followed by the name to be used for administration. The name is entered in the form `filesystem:name` where `filesystem` is the name of the file system and `name` is the managed directory name.

Managed directories can be deleted with the `puredir delete` command. Specify the full name of one or more directories to be deleted. To avoid accidental deletion of content, the managed directory can only be deleted when no content exists and all exports are either removed or disabled. Managed directories cannot be deleted by clients.

The `puredir rename` command changes the current name (`OLD-NAME`) of a managed directory to the new name (`NEW-NAME`). The name change is effective immediately and the old name is no longer recognized in CLI, GUI, or REST interactions. In the Purity//FA GUI, the new name appears upon page refresh.

Managed directories can be displayed by using the `puredir list` command. Options include `--pending` to include managed directories pending eradication, and `--pending-only` to only include managed directories pending eradication. If not specified, directories pending eradication are not included. Use the `--space` option to include space reporting. The `--usage` option displays the amount of data currently stored by users. Combine with the `--total` option to display totals, or the `--historical` option to display historical data at the given resolution. Valid values include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year. One or more managed directory names can be specified.

The `puredir monitor` command displays real-time and historical I/O performance information for file. The output includes the following data about bandwidth and IOPS:

- **Directory**: The full name of the managed directory.
- **Time**: Current time.
- **B/s**: Bandwidth. Number of bytes read/written per second.
- **op/s**: IOPS. Number of read, write, or metadata (other) requests processed per second.
- **us/op**: Latency. Average time in microseconds the array takes to serve a read, write or metadata (other) I/O request.
- **B/op**: IOPS. Average I/O size per read, write, and both read and write (all) operations. Must include the `--size` option to see the B/op columns.

By default, the `puredir monitor` command displays real-time performance data for file directories. To display information for a single file protocol exclusively, use the `--protocol` option and specify `nfs` or `smb`. For information about a selection of one or more directories, append their names.

Include the `--total` option to add a line of aggregated column data for all the listed directories. Include the `--repeat` option to specify the number of times to repeat the real-time update. If not specified, the repeat value defaults to `1`. Include the `--interval` option to specify the number of seconds between each real-time update. At each interval, the system displays a point-in-time snapshot of the performance data. If not specified, the interval value defaults to every 5 seconds. The `--repeat` and `--interval` options can be combined.

Include the `--historical` option to display historical performance data over any of the following ranges of time: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, or 1 year.

Exports

Managed directory exports (i.e., shares) are created by adding export policies to managed directories. Export policies are created with the `purepolicy smb` and `purepolicy nfs` commands. Each policy can be re-used for multiple exports, each export having its own unique name. For each managed directory, there can be one or many exports. An SMB and an NFS export may share the same name, since they reside in different namespaces.

To create an export, use the `puredir export create` command, append `--dir` followed by the name of a managed directory, `--policy` followed by the name of an existing SMB or NFS policy, and the name of the new export. Exports are enabled by default, as long as the export policy they are based upon is enabled. To disable or re-enable an export, use the `puredir export disable` or `puredir export enable` commands.

SMB shares can be made hidden by appending a "`$`" to the end of the SMB export name. Hidden shares are typically not listed when Windows obtains a list of shares. The share may still be listed with administrative tools such as Microsoft Management Console. The filtering happens on the client side and as such this is not a feature that should be used for security. To access a hidden share, one needs to know the name of the share, including the "`$`" at the end of the name.

To rename an export, use the `puredir export rename` command followed by the old name and a new name. If the `--policy` or `--dir` option is used, only the exports matching both policy name and directory name will be renamed. Renaming an export is not a disruptive action for clients connected to an export. Renaming is only disruptive if clients try to reconnect to the old export name. If the export is ActiveDR replicated, the copy on the target array is not automatically renamed.

To delete an export, use the `puredir export delete` command. Optionally, specify the policy name with the `--policy` option. Finally, specify one or more exports to be deleted. If the `--policy` option is used, only the exports matching both policy name and export name will be deleted.

To display an export, use the `puredir export list` command. Valid options include `--pending` or `--pending-only` which is used to include managed directories pending eradication, or only include managed directories pending eradication, respectively. If not specified, directories pending eradication are not included.

NFSv3 File Locking

File services enable users or applications to lock a file so that other users cannot perform operations on the same file. For NFS version 3, file locking is handled through the Network Lock Manager (NLM) protocol. NLM locks are considered advisory locks in that, if a client has access to the file, the NFS service itself does not automatically prevent the client from accessing the file. The service does not check for the existence of or try to obtain an NLM lock on a file ahead of time.

In cases where files are unintentionally left in a locked state, run the `puredir lock nlm-reclamation create` command to release all NLM locks for the entire array. By doing this, client applications are notified, allowing them to reclaim the lock.

Directory Quotas

Directory quotas allow restriction of storage space, and notification, using enforced and unenforced quota limits. Use the `purepolicy quota` command to create and manage quota policies.

To add a quota policy to a managed directory, use the `puredir quota add` command. For the policy to be applied, the current directory usage must not exceed the new enforced limit. To override the control of directory usage while applying the policy, use the `--ignore-usage` option. Only one quota policy can be added. The `puredir quota list` command displays a list of all quotas. To remove a quota policy from a managed directory, use the `puredir quota remove` command. During ActiveDR for file systems, snapshot policies will remain stalled.

File System Recalculation

File system recalculation is a recovery process for directory quotas that recalculates all quotas for specified file system. File system recalculation can be initiated manually to change incorrect values reported by quotas and in upgrades. During filesystem recalculation, the **`Usage`** field in the **`puredir list --usage`** and **`puredir quota list`** command output will show a **-** instead of the actual value. This indicates that recalculation is in progress.

```
$ puredir list --usage
Name                         Path   Usage  % Used  Dynamic Limit Limited BY
test-quota::quota-test-fs:dir1 /dir1  -      -       -       -
```

Snapshots

To create a manual snapshot, use the `puredir snapshot create` command, followed by **`--client-name`** and the client visible portion of the snapshot name. Optionally, add **`--keep-for`** to specify a range of time to keep the snapshot. There is no maximum value, the minimum value is five minutes (5m). Valid time ranges include: 15 minutes, 1 hour, 2 days, 3 weeks, and 4 years. If **`--keep-for`** is omitted, the snapshot will be kept until manually destroyed. Append the name of the managed directory to have the snapshot taken.

For each snapshot, a unique number assigned by Purity//FA is added to the name as a suffix. When creating a manual snapshot, the suffix can be manually defined as a string with the **`--suffix`** option. For example, the following command creates a manual snapshot with a suffix named **`backup`**, provided that the full name then becomes unique:

```
puredir snapshot create --client-name manual --suffix backup fs1:Data
Name                    Source   Policy Created                Time Remaining
fs1:Data.manual.backup fs1:Data -      2020-11-17 03:25:50 PST -
```

Snapshot attributes can be modified by using the `puredir snapshot setattr` command. The two options are mutually exclusive: Use the **`--policy`** option to attach the snapshot to an existing policy, or the **`--keep-for`** option to set or change the period that snapshots are retained before they are eradicated. Only manual snapshots can have their keep-for period changed. To clear the policy or clear the keep-for period, set the option to `''` (i.e., an empty string). Optionally, for manual snapshots, use the **`--client-name`** and the **`--suffix`** options to modify the client visible portion of the snapshot name and the snapshot suffix, respectively. Append one or more managed directory snapshot names to be modified.

The **`puredir snapshot rename`** command changes the current name (***`OLD-NAME`***) of a manual snapshot to the new name (***`NEW-NAME`***). The name change is effective immediately and the old name is no longer recognized in CLI, GUI, or REST interactions. In the Purity//FA GUI, the new name appears upon page refresh.

Snapshots are destroyed, eradicated, or recovered using the **`puredir snapshot`** command with the **`destroy`**, **`eradicate`**, or **`recover`** sub-commands, respectively, followed by the full name of one or more snapshots to be destroyed, eradicated, or recovered.

During ActiveDR for file systems, snapshot policies will remain stalled.

The **`puredir snapshot list`** command displays a list of snapshots. Specify the **`--pending`** or **`--pending-only`** option to include snapshots pending eradication or to display only snapshots pending eradication, respectively. If neither of the two options is specified, the command does not display snapshots pending eradication. To display the space usage of individual snapshots, specify the **`--space`** option. To show the aggregated space usage of all snapshots, include the **`--total`** option. The **`--total`** option must be used with the **`--space`** option.

Scheduled Snapshots

Scheduled snapshots are created by having snapshot policies that are added to managed directories. Snapshot policies are created with the **`purepolicy snapshot`** command. Each snapshot policy can be re-used for multiple directories.

To add a snapshot policy to a managed directory, use the **`puredir snapshot add`** command followed by **`--policy`**, the name of the snapshot policy, and one or more managed directories to where the policy is to be added. Similarly, to remove a snapshot policy from a managed directory, use the **`puredir snapshot remove`** command. This can also be done through the **`purepolicy`** command.

Examples

Example 1

```
puredir create --path /data FS1:Data
```

Creates a managed directory **`data`** named **`Data`** on a file system **`FS1`**.

Example 2

```
puredir export create --dir FS1:Data --policy SMBanon Share
```

Creates an export named **`Share`** by adding the export policy **`SMBanon`** to the managed directory **`Data`** on filesystem **`FS1`**.

Example 3

```
puredir export create --dir FS1:Admin --policy SMBadmin Adm$
```

Creates an export for a hidden share named **`Adm$`** by adding the export policy **`SMBadmin`** to the managed directory Admin on filesystem **`FS1`**. To access a hidden share, one needs to know the name of the share, including the "**`$`**" at the end.

Example 4

```
puredir snapshot create --client-name manual --keep-for 2w FS1:Data
```

Creates a manual snapshot, client visible portion of the snapshot being named **`manual`**, of directory **`FS1:Data`**, to keep for two weeks. The full management name of the snapshot is: **`FS1:Data.manual.n`** where **`n`** is a counter starting from one.

Example 5

```
puredir snapshot add --policy Schedule1 FS1:Data FS1:User
```

Creates a snapshot schedule by adding the snapshot policy **`Schedule1`** to two managed directories: **`Data`** and **`User`**.

Example 6

```
puredir snapshot destroy FS1:Data.manual.1
```

Destroys the snapshot **`FS1:Data.manual.1`** which is a snapshot located in the **`Data`** directory on the file system **`FS1`**.

Example 7

```
puredir delete FS1:Data
```

Deletes the directory named **`Data`** on the file system named **`FS1`**.

Example 8

```
puredir snapshot list --space FS1:Data
```

Displays a list of snapshots created in the **`Data`** directory. The list includes space reporting.

Example 9

```
puredir snapshot list --space --total
```

Displays a list of snapshots and the individual space usage followed by the total space usage of all the snapshots.

Example 10

```
puredir monitor --repeat 10 --interval 5 --total --protocol smb
```

Lists SMB statistics for all managed directories. Updates 10 times with an interval of 5 seconds, followed by a single line (total) containing aggregated statistics.

See Also

purefs, purefile, purearray, purepolicy

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puredns

puredns – manages the DNS attributes of an array

Synopsis

puredns create [--domain ***DOMAIN***] [--nameservers ***NAMESERVERS***] [--services ***SERVICES***] [--source ***SOURCE***] [--ca-certificate-group ***CA_CERTIFICATE_GROUP*** | --ca-certificate ***CA_CERTIFICATE***] ***NAME***

puredns delete ***NAME***

puredns list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [***NAME...***]

puredns rename ***OLD-NAME NEW-NAME***

puredns setattr [--domain ***DOMAIN***] [--nameservers ***NAMESERVERS***] [--services ***SERVICES***] [--source ***SOURCE***] [--ca-certificate-group ***CA_CERTIFICATE_GROUP*** | --ca-certificate ***CA_CERTIFICATE***] [***NAME***]

Options

`-h | --help`
: Can be used with any command or subcommand to display a brief syntax description.

`--ca-certificate` ***CA_CERTIFICATE***
: For encrypted DNS connections, Imports and uses the CA certificate generated by a certificate authority (CA) as the server communication certificate. A certificate can be imported using the purecert command. This option is mutually exclusive with the **--ca-certificate-group** option.

`--ca-certificate-group` ***CA_CERTIFICATE_GROUP***

For encrypted DNS connections, the ID of the CA certificate group to use for DNS over HTTPS. For example, `ca_cert_group_1`. A group can be created using the **`purecert`** command. This option is mutually exclusive with the **`--ca-certificate`** option.

`--domain`

Domain suffix to be appended by the array when performing DNS lookups. To remove the domain suffix from Purity//FA DNS queries, set to an empty string (**`""`**).

`--nameservers`

Comma-separated list of up to three DNS server IP addresses.

For IPv4, specify the IP address in the form **`ddd.ddd.ddd.ddd`**, where **`ddd`** is a number ranging from 0 to 255 representing a group of 8 bits. For IPv6, specify the IP address in the form **`xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx`**, where **`xxxx`** is a hexadecimal number representing a group of 16 bits. When specifying an IPv6 address, consecutive fields of zeros can be shortened by replacing the zeros with a double colon (**`::`**).

To unassign the DNS server IP addresses, set to an empty string (**`""`**). Once the DNS server IP addresses have been unassigned, the array no longer makes DNS queries.

`--services` ***`SERVICES`***

Specifies the services that will leverage the DNS configuration. Valid values include **`management`**, **`file`**, or **`management,file`** for both.

`--source` ***`SOURCE`***

Specifies the name of the virtual file network interface used to communicate with the DNS servers. If the source network interface is not specified then the default network interface resolution will be used.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Arguments

`NAME`

The name of the DNS configuration. When modifying a configuration and the name is not specified, the management configuration is modified.

`NEW-NAME`

Name by which the DNS configuration is to be known after the rename command executes.

`OLD-NAME`

Current name of the DNS configuration to be renamed.

Description

The **`puredns`** command manages the DNS attributes for an array's administrative and, optionally, file services network.

The **`puredns setattr`** command sets or modifies DNS parameters. Include the **`--domain`** option to set the domain suffix to be appended to DNS queries. Specify the name servers with the **`--nameservers`** option. The **`--services`** option can be used to specify the types of services that will leverage the DNS configuration. To modify an existing configuration other than the management DNS configuration, specify the name of the configuration to be modified. When modifying a non-management configuration, the **`--source`** option specifies the interface used to communicate with the DNS servers.

The **`puredns create`** command allows you to create additional DNS configurations. Additional configurations must be named. This allows file services to communicate with another Active Directory server. File services and management services must be assigned to only one DNS configuration at a time. The configuration cannot be named “management”. Use the **--source** option only for non-management services.

The **puredns rename** command changes the current name (OLD-NAME) of a DNS configuration to the new name (NEW-NAME). The management configuration cannot be renamed.

The **puredns delete** command deletes a DNS configuration specified by name. The management configuration cannot be deleted.

The **puredns list** command displays the current DNS parameters, including domain suffix and name servers, of the array. Include the **--cli** option to display the CLI command lines that would reproduce the array's current DNS configuration. These can, for example, be copied and pasted to create an identical DNS configuration in another array, or saved as a backup.

Encrypted DNS Connection

To secure the DNS connections with encryption by using DNS over HTTPS (DoH), the following must be in place:

1. Define the DNS servers with a “https://” scheme. The domain part of the address must be an IP address. IPv4 and IPv6 addresses are allowed. Custom ports are allowed, both for IPv4 and IPv6. HTTPS and non-HTTPS servers should not be mixed. That is, all configured servers in the list must either be secure, or not.

   The following are examples of valid IP addresses for encrypted DNS connection:

```
https://10.14.278.132
https://10.14.278.132/dns-query
https://10.14.278.132:4430/dns-query
https://[2001:db8::1]:4430
```

2. Provide a certificate to validate the connection.

When configuring DNS over HTTPS, it is mandatory to use either the **--ca-certificate** or **--ca-certificate-group** options to import and use a CA certificate or certificate group. These options accept the certificate or group IDs, respectively, not the certificate itself. The certificate or the certificate group can be created using the **purecert** command. These, and only these, will be used to verify the secure connection.

Setting either CA certificate or CA certificate group via the **setattr** subcommand automatically replaces previously set value in the other with empty so that both cannot be set at one time.

Whenever a plain non-HTTPS connection is used, the ca-certificate and ca-certificate-group parameters must be blank. Existing certificate associations can be removed by setting the corresponding attribute to an empty string (`""`). Removing the certificate association can only be done when switching from a DoH configuration to a non-DoH configuration.

Exceptions

The `puredns setattr` command is not supported on Pure Cloud Block Store.

Examples

Example 1

```
puredns setattr --domain mydomain.com --nameservers 192.168.0.125,192.168.2.125
```

Sets the DNS parameters for the management service. Specifies the IPv4 addresses of two DNS servers for Purity//FA to use to resolve hostnames to IP addresses, and the domain suffix mydomain.com for DNS searches.

Example 2

```
puredns setattr --nameservers 192.168.0.125,2001:0db8:85a3::ae26:8a2e:0370:7334
```

Specifies IP addresses of two DNS servers for Purity//FA to use to resolve hostnames to IP addresses for the management service.

Example 3

```
puredns create dns1
```

Creates a DNS configuration named **`dns1`**.

Example 4

```
puredns setattr --domain fft-ad1.local --nameservers 10.14.230.132 --services file --
source filevif10 dns1
```

Modifies the DNS configuration named **`dns1`** and uses this configuration for file services.

Example 5

```
puredns setattr --domain ""
```

Removes the domain suffix from Purity//FA DNS management queries.

Example 6

```
puredns setattr --nameservers ""
```

Unassigns DNS server IP addresses for the management service (Purity//FA ceases to make DNS queries).

See Also

purearray, purenetwork

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puredrive

puredrive-admit, puredrive-list – display and manage an array's flash, NVRAM, and cache modules

Synopsis

```
puredrive admit
puredrive list [--csv | --nvp] [--notitle] [--page] [--raw] [--spec |
--installed-capacity] [--total] [DRIVE...]
```

Arguments

DRIVE

Name of a flash, NVRAM, or cache module about which information is to be displayed. Includes the shelf identifier (for example SH1.BAY0, which designates the module in bay #0 of storage shelf #1).

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--installed-capacity`

Displays the full installed capacity of the SSD.

`--mode`

Reserved for Pure Storage Technical Services.

`--spec`

Displays hardware specifications for the module.

`--total`

Follows output lines with a single line containing column totals in columns where they are meaningful.

Options that control display format:

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Description

The FlashArray array consists of flash SSD (solid state drive, also called flash module), NVRAM (non-volatile random access memory), and cache modules. Flash modules are used for the persistent storage of user data, while NVRAM modules are used as non-volatile write caches. The **`puredrive`** command manages the flash and NVRAM modules of an array.

The **`puredrive list`** command lists an array's flash and NVRAM modules. The list output includes the following information:

`Name`

Name by which Purity//FA identifies the module in administrative operations. Flash and NVRAM module names identify physical locations in terms of shelf and bay numbers.

A bay with the name `(`**`unknown`**`)` means that something unexpected has happened to the bay and the array is attempting to bring it back. If the module remains in **`unknown`** status, contact Pure Storage Technical Services.

`Type`

Drive type. Possible drive types include **`SSD`** (Solid State Drive), **`NVRAM`** (Non-Volatile Random-Access Memory), and **`Cache`** modules.

A dash (**`-`**) in the Type column represents a module that this array does not recognize. The module is initiated as a FlashArray module and is functional, but it does not belong to this array. For example, the module may have been pulled from another array and mistakenly placed into this array.

`Status`

Condition of the module. Possible module statuses include:

`healthy`

The module is functioning and responds to I/O commands.

`unadmitted`

The module has been added to the bay, and it is waiting to be admitted into the array. Run **`puredrive admit`** to admit all of the unadmitted modules, including this one. Once a module has been successfully admitted, it changes to **`healthy`** status. If the module returns to **`unadmitted`** status after the admission process, run the puredrive admit command again. If the subsequent "admit" attempt is not successful, contact Pure Storage Technical Services.

`unused`

The bay does not have any modules. New modules can be added to this empty bay.

`unrecognized`

The module is initiated as a FlashArray module and is functional, but it does not belong to this array. For example, the module may have been pulled from another array and mistakenly placed into this array.

`unhealthy`

The module is not working as intended, but it may still be functional. Contact Pure Storage Technical Services.

`missing`

A module was previously present in the bay. The FlashArray considers this module missing based on the current configuration. Contact Pure Storage Technical Services.

`failed`

The module has experienced a failure and is not responding to I/O. As a result, Purity//FA is evacuating or has evacuated data off the module. Contact Pure Storage Technical Services.

`empty`

The FlashArray expects a module to be present in the bay that is configured for a specific module type. Contact Pure Storage Technical Services.

Modules might also go into any of the following temporary statuses:

`identifying`

The module is in the process of coming online as Purity//FA tries to identify the drive. The module is not yet ready to respond to I/O commands. The **`identifying`** status is transitory and will eventually change to **`unadmitted`** or **`healthy`** status. If the module remains in **`identifying`** status for more than 10 minutes, contact Pure Storage Technical Services.

`recovering`

The module is undergoing a drive profiling procedure or rehabilitation. The **`recovering`** status is transitory and will eventually change to **`healthy`** status. If the module remains in **`recovering`** status or transitions to **`failed`** status, contact Pure Storage Technical Servicess.

`updating`

A firmware update is taking place. The updating status is transitory and will eventually return to **`healthy`** status. If the module does not return to **`healthy`** status, contact Pure Storage Technical Services.

`Capacity`

Physical storage capacity of the module.

`Evac Remaining`

For a missing module, the amount of data that remains to be evacuated (reconstructed and stored on other flash modules).

`Last Failure`

Time at which a module became non-responsive.

`Last Evac Completed`

Time at which the evacuation of data from a non-responsive module completed.

`Protocol`

Storage protocol, such as NVMe and SAS. Must include the **`--spec`** option to display the Protocol column.

Capacity Upgrade and Drive Admission

Increase storage capacity by adding data packs or individual drives to a shelf.

Data packs can be added to any shelf that has open space.

Individual drives can be added to any unused slot within a shelf. The drives must be DirectFlash modules, and the shelf must already contain similar-sized DirectFlash modules. The array will not admit individual drives of other types or sizes. Add each drive to the unused slots from left to right, starting with the lowest open slot.

Before you begin the upgrade, ensure your shelf has enough open spaces to hold the new packs or individual drives.

Performing a capacity upgrade is a two-step process: first, add the modules to the array; and second, admit the newly added modules.

When a module has been added to the array, its status changes from **`unused`** to **`identifying`** as Purity works to identify the module. The module transitions to its final **`unadmitted`** status when it has been successfully added to the array.

Run the **`puredrive list`** command to verify that the newly added shelves and drives are in **`unadmitted`** status, indicating that the modules have been successfully connected but not yet admitted to the array.

After a module has been added (connected) to the array, it must be admitted.

Once the drive status changes to **`unadmitted`**, run the **`puredrive admit`** command to admit all modules that have been added (connected) but not yet admitted to the array.

Once a drive has been successfully admitted to the array, its module status changes from **`unadmitted`** to **`healthy`**.

Run the **`puredrive list`** command to verify that all of the shelves and drives are in **`healthy`** status, indicating that the modules have been successfully admitted and are in use by the system. This completes the drive admission process.

If issues arise during the admission process, the module status changes to **`unrecognized`** or **`failed`**, or reverts to **`unadmitted`**. If the module changes to **`unrecognized`** or **`failed`** status, contact Pure Storage Technical Services. If the module returns to the **`unadmitted`** status, run the **`puredrive admit`** command again. If the subsequent “admit” attempt is not successful, contact Pure Storage Technical Services.

Capacity Limits

Capacity limits are implemented to manage the customer's entitled drive capacities. Run the `puredrive list --installed-capacity` command to display the full installed capacity of the drives of any SSDs in an array, showing both entitled and installed capacities. Use with the `--pack`-option to display information about all drives in the same pack as the specified drive.

```
$ puredrive list --installed-capacity
Name      Type Status  Capacity Installed Capacity Details
CH0.Bay0  SSD  healthy 8.80T    17.60T             -
CH0.Bay1  SSD  healthy 8.80T    17.60T             -
CH0.Bay2  SSD  healthy 8.80T    17.60T             -
```

Examples

Example 1

```
puredrive list SH0.BAY5
```

Lists information for the flash module in BAY5 of shelf SH0.

Example 2

```
puredrive list CH0.NVB1
```

Lists information for the NVRAM module in NVRAM bay 1 of shelf CH0.

Example 3

```
puredrive admit
puredrive list
```

Admits all modules that have been added (connected) to the array and are waiting to be admitted. Confirms that all modules that were in **`unadmitted`** status are now in **`healthy`** status, indicating successful admission.

Example 4

```
puredrive list --installed-capacity SH0.BAY5
```

Displays all drives in the same pack as SH0.BAY5 including the types, status, capacities, and additional details, if available.

See Also

n/a

Author

Pure Storage Inc. **`<documentfeedback@purestorage.com>`**

pureds

pureds, pureds-disable, pureds-enable, pureds-list, pureds-role, pureds-setattr, pureds-test – manages FlashArray array integration with a directory service.

pureds-local – manages local users and groups operations.

Synopsis

pureds disable [--check-peer] [***SERVICE-NAME...***]

pureds enable [--check-peer] [***SERVICE-NAME...***]

pureds list [--ca-certificate] [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [***SERVICE-NAME...***]

pureds local group add {--user ***USER*** | --group ***GROUP*** | --external ***MEMBER***} ***NAME***

pureds local group create [--gid ***GID***] [--email ***EMAIL***] ***NAME...***

pureds local group delete ***NAME...***

pureds local group list [--cli | --csv | --nvp] [--notitle] [--raw] [--filter ***FILTER***] [--page] [--limit ***LIMIT***] [--sort ***SORT***] [--member] [--sid] [--email] [--member-sid] [***NAME...***]

pureds local group remove {--user ***USER*** | --group ***GROUP*** | --external ***MEMBER***} ***NAME***

pureds local group rename ***OLD-NAME NEW-NAME***

pureds local group setattr [--gid ***GID***] [--email ***EMAIL***] ***NAME...***

pureds local user add --group ***GROUP*** [--primary] ***NAME***

pureds local user create --primary-group ***PRIMARY_GROUP_NAME*** [--password] [--disable] [--uid ***UID***] [--email ***EMAIL***] ***NAME...***

pureds local user delete ***NAME...***

pureds local user disable [--keep-open-sessions] ***NAME...***

pureds local user enable ***NAME...***

pureds local user list [--cli | --csv | --nvp] [--notitle] [--raw] [--filter ***FILTER***] [--page] [--limit ***LIMIT***] [--sort ***SORT***] [--member] [--sid] [--email] [***NAME...***]

pureds local user remove --group ***GROUP*** ***NAME***

pureds local user rename ***OLD-NAME*** ***NEW-NAME***

pureds local user setattr [--primary-group ***PRIMARY_GROUP_NAME***] [--password] [--uid ***UID***] [--email ***EMAIL***] [--keep-open-sessions] ***NAME...***

pureds role add --access-policy ***ACCESS_POLICY*** ***NAME***

pureds role create [--access-policy ***ACCESS_POLICY***] [--group ***GROUP***] [--group-base ***GROUP-BASE***] ***NAME...***

pureds role delete ***NAME...***

pureds role list [--cli | --csv | --nvp] [--filter ***FILTER***] [--limit ***LIMIT***] [--notitle] [--page] [--raw] [--sort ***SORT***] [***NAME...***]

pureds role remove --access-policy ***ACCESS_POLICY*** ***NAME***

pureds role setattr [--group ***GROUP***] [--group-base ***GROUP-BASE***] ***NAME...***

pureds setattr [--auto-fetch] [--base-dn ***BASE-DN***] [--bind-password] [--binduser ***BIND-USER***] [--ca-certificate] [--ssh-publickey-attribute ***SSH_PUBLICKEY_ATTRIBUTE***] [--trust] [--uri ***URI-LIST***] [--user-login-attribute ***USER_LOGIN_ATTRIBUTE***] [--user-object-class ***USER_OBJECT_CLASS***] [***SERVICE-NAME***]

pureds test

Options

`-h | --help`
: Can be used with any command or subcommand to display a brief syntax description.

`--auto-fetch`
: Attempts to get CA certificate data from the configured URI or the first URI in the list if more than one URI is configured. This option is not for use with the data service.

`--base-dn BASE-DN`

Sets the base of the distinguished name (base DN) of the directory service groups. The base DN must be in a valid DN syntax.

Specify the value for each component. List multiple values in comma-separated format (for example, `OU=people,DC=company,DC=com)`.

To clear the base DN setting, set to an empty string (`""`).

`--bind-password`

Sets the password of the bind user account. The password is entered through interactive prompt.

`--bind-user` ***BIND-USER***

Sets the user name used to bind to and query the directory.

For Active Directory, enter the user name (often referred to as sAMAccountName or User Logon Name) for the account that is used to perform directory lookups. The user name cannot contain the characters `" [ ] : ; | = + * ? < > / \`, and cannot exceed 20 characters in length.

For OpenLDAP, enter the full DN of the user. For example, **`"CN=John,OU=Users,DC=example,DC=com"`**

`--ca-certificate`

Sets the CA certificate data. The data is entered through interactive prompt. The data must be PEM formatted (Base64 encoded) and include the **`"-----BEGIN CERTIFICATE-----"`** and **`"-----END CERTIFICATE-----"`** lines. The certificate cannot exceed 3000 characters in total length. Press **`Enter`** and then **`Control-d`** to exit the interactive session. To clear the certificate, enter blank lines at the prompt. This option is not for use with the data service.

`--check-peer`

Enables or disables server authenticity enforcement with the configured CA certificate. This option can only be enabled if CA certificate data has been provided. If CA certificate data is cleared when this option is enabled, this setting reverts back to disabled. See also "Notes on the Check Peer Option" on page 161.

`--disable`

Disables the local user upon creation. When creating a user, one or both of the **`--password`** or **`--disable`** options must be set.

`--email`

Email address for the local user or group.

`--external` ***MEMBER***

When adding to or removing an external member from a group, MEMBER is a comma-separated list of one or more external members (FQDN) to add to or remove from the group.

`--gid` ***GID***

The ID of the local group, overrides the ID which is automatically set.

`--group` ***GROUP***

For local users and groups: a comma-separated list of one or more group names or IDs.

For directory service configuration:

Sets the common name (CN) of the configured group in the directory tree that maps to a role in the array. The group name should be just the common name of the group without the **`"CN="`** specifier. Common names cannot exceed 64 characters in length.

All users in a configured group map to the same role in the array.

To clear the group setting, set to an empty string (`""`).

`--group-base` ***GROUP-BASE***

Specifies where the configured groups are located in the directory tree. This field consists of organizational units (OU) that, when combined with the base DN and the configured group CNs, complete the full DN of each group.

Specify "OU=" for each OU. List multiple OUs in comma-separated format.

The order of OUs should get smaller in scope from right to left. For example, for "OU=PureGroups,OU=SANManagers", the "SANManagers" OU contains the sub OU "PureGroups".

To clear the group base setting, set to an empty string (`""`).

`--keep-open-sessions`

Keeps any open session open, when setting password or disabling a user account.

`--member`

The user list output includes groups that the user is a member of. Group list output includes members of the group. Any external members are also included.

`--member-sid`

The output includes member SID of the user or group.

`--password`

Sets the password for the user. The password is manually created and typed twice through an interactive prompt. The password is not shown while typing. If the passwords do not match, the user is not created.

When creating a user, one or both of the **`--password`** or **`--disable`** options must be set.

`--primary`

Sets the group as the user's primary group.

`--primary-group` ***PRIMARY_GROUP_NAME***

Sets the primary group of the user.

`--role` ***ROLE***

Sets the Purity//FA RBAC role.
The `--role` option is deprecated but stll available for backwards compatibility. The `--access-policy` option is recommended instead.

`--sid`

Displays the security identifier of the user or group.

`--ssh-publickey-attribute` ***`SSH_PUBLICKEY_ATTRIBUTE`***

Sets the specified configuration name for the ssh public key attribute. This option is not for use with the data service.

`--trust`

Skips the certificate chain trust verification. This option is not for use with the data service.

`--uid` ***`UID`***

The ID of the user.

`--uri` ***`URI-LIST`***

Sets the URIs of the directory servers. Up to 30 URIs can be specified. List multiple URIs in comma-separated format.

Each URI must include the scheme **`ldap://`** or **`ldaps://`** (for LDAP over SSL), a hostname, and a domain name or IP address. For example, **`ldap://ad.company.com`** configures the directory service with the hostname "ad" in the domain "company.com" while specifying the unencrypted LDAP protocol.

If specifying a domain name, it should be resolvable by the configured DNS servers. See puredns for more information.

If specifying an IP address, for IPv4, specify the IP address in the form **`ddd.ddd.ddd.ddd`**, where **`ddd`** is a number ranging from 0 to 255 representing a group of 8 bits.

For IPv6, specify the IP address in the form **`[xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx]`**, where **`xxxx`** is a hexadecimal number representing a group of 16 bits. Enclose the entire address in square brackets (**`[]`**). Consecutive fields of zeros can be shortened by replacing the zeros with a double colon (**`::`**).

If the scheme of the URIs is **`ldaps://`**, SSL is enabled. SSL is either enabled or disabled globally, so the scheme of all supplied URIs must be the same. They must also all have the same domain.

If base DN is not configured and a URI is provided, the base DN will automatically default to the domain components of the URIs.

Optionally specify a port. If a port number is specified, append it to the end of the address. Default ports are 389 for ldap, and 636 for ldaps. Non-standard ports can be specified in the URI if they are in use.

To clear the URI setting, set to an empty string (**`""`**).

`--user` ***`USER`***

Comma-separated list of one or more names or IDs of users to be added to or removed from the group.

`--user-login-attribute` ***`USER_LOGIN_ATTRIBUTE`***

User login attribute in your LDAP structure. Typically, the attribute field that holds the user's unique login name. Defaults to sAMAccountName for Active Directory, or uid for other directory services.

This option is not for use with the data service.

`--user-object-class` ***`USER_OBJECT_CLASS`***

Value of the object class used for the LDAP user. Defaults to User for AD, posixAccount or shadowAccount for OpenLDAP, or person for all others.

This option is not for use with the data service.

Note: These group options are deprecated and are not available: `--array-admin-group`, `--storage-admin-group`, `--ops-admin-group`, and `--readonly-group`. Instead, run the **`pureds role`** commands to map LDAP groups to either built-in or customer-defined access policies.

Arguments

`EMAIL`

email information for the user or for the group.

`NAME`

User or group name, or user or group ID, or name of a group to role mapping.

`NEW-NAME`

Name by which the user or group is to be known after the command executes.

`OLD-NAME`

Current user or group name, user ID or group ID, of the user or group to be renamed.

`ROLE`

Role name. Valid role names are **`readonly`**, **`ops_admin`**, **`array_admin`**, and **`storage_admin`**.

`SERVICE-NAME`

The service name, for example **`management`** or **`data`**.

Description

The **pureds** command manages the integration of FlashArray arrays with an existing LDAP directory service. LDAP integration can be used for array management and for file users accessing NFS exports:

- **Management** - manages the LDAP connection for FlashArray administrative accounts.
- **Data** - manages the LDAP connection for users to access NFS file exports only (AUTH_SYS).

The Purity//FA release comes with a single local administrative account named `pureuser` with array-wide (`array_admin`) permissions. The account is password protected and may alternatively be accessed using a public-private key pair. File storage comes with two local users: Administrator and Guest, neither of which can be used until a password is set.

Additional users can be added to the array by creating and configuring local users directly on the array. For more information about administrative local users, refer to pureadmin. For file services, use the **`pureds local user`** and **`pureds local group`** commands.

Users can also be added to the array through Lightweight Directory Access Protocol (LDAP) by integrating the array with an existing directory service. If a user is not found locally, the directory servers are queried. OpenLDAP and Microsoft's Active Directory (AD) are two implementations of LDAP that Purity//FA supports.

Directory Service for Array Management

With LDAP integration, the array leverages the directory for authentication (validate user's password) and authorization (determine user's role in the array).

Configuring the Pure Storage directory service requires a URI, a base DN, a bind user, a bind password, and at least one group within the LDAP directory.

Including `--ssh-publickey-attribute` with a specified attribute name will allow public keys to be taken from the LDAP server. When the attribute is not specified, the public keys are only taken from the array.

In a case where a user is a local user and is on the LDAP server the public keys are only taken from the local users keys. If the user is specified on the LDAP server but has keys located on the array using pureadmin then the keys from both the array and the LDAP server can be used.

Users

An LDAP user is an individual in the LDAP directory. To be able to access the array, an LDAP user must belong to a configured group in the LDAP directory.

LDAP users log in to the array via `ssh` by entering the following information, where **`<user_name>`** represents the sAMAccountName for Active Directory or uid for OpenLDAP, and **`<array_name>`** represents the name or the IPv4 or IPv6 address of the FlashArray array:

```
<user_name>@<array_name>
```

For IPv4, specify the IP address in the form **`ddd.ddd.ddd.ddd`**, where **`ddd`** is a number ranging from `0` to `255` representing a group of 8 bits.

For IPv6, specify the IP address in the form **`[xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx]`**, where **`xxxx`** is a hexadecimal number representing a group of 16 bits. Enclose the entire address in square brackets (**`[]`**). Consecutive fields of zeros can be shortened by replacing the zeros with a double colon (**`::`**).

For directory service enabled accounts, user passwords to the array are managed through the directory service, while public keys are configured through Purity//FA.

Accounts with user names that conflict with local accounts will not be authenticated against the directory. These account names include, but are not limited to: **`pureuser`**, **`os76`**, **`root`**, **`daemon`**, **`sys`**, **`man`**, **`mail`**, **`news`**, **`proxy`**, **`backup`**, **`nobody`**, **`syslog`**, **`mysql`**, **`ntp`**, **`avahi`**, **`postfix`**, **`sshd`**, **`snmp`**.

If an LDAP user has the same name as a locally created user, the locally created user always has priority.

Users with disabled accounts will not have access to the array.

Groups

A group in the LDAP directory consists of users who share a common purpose.

Each configured group in the directory has a unique distinguished name (DN) representing the entire path of the object's location in the directory tree. The DN is comprised of the following attribute-value pairs:

- DC - Domain component base of the DN. For example, **`DC=mycompany,DC=com`**.
- OU - Organizational unit base of the group. For example, **`OU=PureGroups,OU=SAN,OU=IT`**.
- CN - Common name of the groups themselves. For example, **`CN=purereadonly`**.

For example, **`CN=purereadonly,OU=PureGroups,OU=SAN,OU=IT,DC=mycompany,DC=com`** is the DN for configured group **`purereadonly`** at group base **`OU=PureGroups,OU=SAN,OU=IT`** and with base DN **`DC=mycompany,DC=com`**.

The DN can contain multiple DC and OU attributes.

OUs are nested, getting more specific in purpose with each nested OU.

For OpenLDAP, for group configurations based on the **`groupOfNames`** object class, groups must have the full DN of members in the `member` attribute. For group configurations based on the **`posixAccount`** class, groups must have the **`uid`** of members in the `memberUid` attribute. For group configurations based on the nisNetgroups object class, see the NIS Netgroup section below.

When a user who is a member of a configured group logs in to the array, only the CLI actions that the user has permission to execute will be visible. Similarly, in the GUI, actions the user does not have permission to execute will be grayed out or disabled.

For Active Directory, two types of groups are supported: security groups and distribution groups. Distribution groups are used only with email applications to distribute messages to collections of users. Distribution groups are not security enabled. Security groups assign access to resources on your network. All groups configured on the array must be security groups.

NIS Netgroups

NIS netgroups are an LDAP extension that provides a way to compose hierarchical groups of users or machines on a network, which can then be used for access control and to determine the role of a user. They can be used by FlashArray to configure the group-to-role mapping, such that each named group has an object class of **`nisNetgroup`** on the directory server. All groups must have the same group type. Using other group types together with **`nisNetgroup`** is not allowed.

Typically, `cn` (common name) is the relative distinguished name attribute for the group. Each nisNetgroup is required to define the following two attributes:

- **nisNetgroupTriple** - describes the user, for example: `(,username,).`
- **memberNisNetgroup** - merges the attributes of another netgroup into the current one by listing the name (`cn`) of the merging netgroup.

When using NIS netgroups group-to-role mapping, Purity only recognizes users if they appear in a **`nisNetgroupTriple`** attribute. Only the middle field, the username, is respected, which means that the first and last fields in the triple can be set to any value and will be disregarded. For example the following **`nisNetgroupTriple`** attributes are identical from the perspective of the array: `(machine1,username1,)`, `(-,username1,-)`, `(,username1,)`, and `(machine1,username1,domain1).`

For cascading access control and system groupings, use the **`memberNisNetgroup`** attribute to join NIS Netgroups together. The following is an example that displays how NIS Netgroup entries on the LDAP server are expected to look:

```
cn: parent_netgroup
objectClass: top
objectClass: nisNetgroup
nisNetgroupTriple: (,username1,)
nisNetgroupTriple: (,username2,)
nisNetgroupTriple: (,username3,)
memberNisNetgroup: child_netgroup
description: All QA users in my organization
cn: child_netgroup
objectClass: top
objectClass: nisNetgroup
[...]
```

When NIS netgroups are configured, once the user is authenticated and authorized, the user experience is identical to ordinary LDAP configuration.

Policy-Based Access Control

Policy-based access control restricts the system access and capabilities of each user based on their assigned access policy for either an array or a realm.

Purity//FA supports the following built-in access policies, each of which has array scope:

- **Read Only**. Users with the Read-Only (**`readonly`**) role can perform operations that convey the state of the array. Read Only users cannot alter the state of the array.
- **Ops Admin**. Users with the Ops Admin (**`ops_admin`**) role can perform the same operations as Read Only users plus enable and disable remote assistance sessions. Ops Admin users cannot alter the state of the array.
- **Storage Admin**. Users with the Storage Admin (**`storage_admin`**) role can perform the same operations as Read Only users plus storage related operations, such as administering volumes, hosts, and host groups. Storage Admin users cannot perform operations that deal with global and system configurations.
- **Array Admin**. Users with the Array Admin (**`array_admin`**) role can perform the same

operations as Storage Admin users plus array-wide changes dealing with global and system configurations. In other words, Array Admin users can perform all operations.

All users in the array, whether created locally or added to the array through LDAP integration, are attached one of the built-in access policies or one or more customer-defined access policies.

For LDAP users, role-based access control is achieved by mapping existing groups in the LDAP directory to Purity//FA RBAC roles (`array_admin`, `ops_admin`, `storage_admin`, and `readonly`) on the array. See "Map LDAP Groups to Purity Access Policies" on the next page.

When a user belongs either to an LDAP groups that map to multiple access policies or to multiple LDAP groups that map to different access policies, the resulting permissions depend on whether those access policies use all-permissions or least-common-permissions aggregation strategies. See "Management Access Policies" on page 453 in the "purepolicy" on page 428 chapter.

Directory Service Configuration on the Array

Configuring a directory service for use with Purity//FA involves the following separate steps (each described below):

- Identifying the directory server or servers (through the URI or URIs).
- Describing the directory server internal configuration (Base DN, groups, bind user, password, etc.).
- Defining a mapping of existing directory service groups to Purity roles.
- Validating the directory service configuration and parameters.
- Enabling the use of the directory service.

Configuring the directory service requires a URI, a base DN, a bind user, a bind password, and at least one group within the LDAP directory.

Before you start the configuration process, note the DN of each group within the directory server. Each component of the DN is used to configure directory service use on the array.

When you configure the array to integrate with a directory service, consider the following:

- If the directory service contains multiple groups, each group must have a common name (CN).
- All uniform resource identifiers (URIs) must be in the same, single domain.

Identify the Directory Server or Servers

Run the **pureds setattr** command to configure the URI, base DN, bind user name, and bind password. Include the **--uri** option to set the URIs of the directory servers with LDAP or LDAPS. Up to 30 URIs can be specified. List multiple URIs in comma-separated format. To clear the URI setting, set it to an empty string (""). Include the **--base-dn** option to set the base distinguished name (base DN) of the LDAP group. The base DN must be in a valid DN syntax. Specify the value for each component. List multiple components in comma-separated format (for example, `OU=people,DC=company,DC=com`). To clear the base DN setting, set to an empty string (""). Include the **--bind-user** and **--bind-password** options to specify the user name and password used to bind to and query the directory, respectively. The bind password is entered through interactive prompt. The bind account must be configured to allow the array to read the directory. It is good practice for this account to not be tied to any actual person and to have different password restrictions, such as "password never expires". The bind account should also not be a privileged account, since only read access to the directory is required.

For example, run the following command to set two URIs and set bind user credentials:

```
pureds setattr --uri ldaps://dc01.domain.com:636,ldaps://dc02.domain.com:636
               --base-dn DC=mycompany,DC=com --bind-user ldapreader --bind-password
```

Notes on TLS

Ensure that the directory server URL begins the "`LDAPS`" to use TLS for a secure connection between the array and the directory server.

When "`LDAP`" is used instead of "`LDAPS`" in the directory server URL, TLS is not used for a secure connection between the array and the directory server. However, an exception is when the `--check-peer` option is enabled, as using a CA certificate automatically enforces a TLS connection.

Map LDAP Groups to Purity Access Policies

For an LDAP user to log in to the array, the user must belong to a configured group in the LDAP directory, and that group must be mapped to a Purity//FA access policy, either the `array_admin`, `ops_admin`, `storage_admin`, or `readonly` built-in access policies or a customer-defined access policy. Once an LDAP group is mapped to an access policy, all members of the group are assigned that access policy in Purity//FA.

These mappings are called named group-to-role mappings.

Mappings of multiple LDAP groups to a Purity access policy are supported with the **`pureds role create`** and **`pureds role setattr`** CLI commands. The groups are not required to have the same group base in LDAP.

Note: Beginning with the 6.6.3 release, the meaning and usage of the **`pureds role`** command is changed. The **`pureds role create`** command now creates a named mapping of an existing LDAP group to one of the Purity//FA management access policies, either the `array_admin`, `ops_admin`, `storage_admin`, or `readonly` built-in access policies and customer-defined access policies.

System-defined Mappings

The following are the default named mappings defined in Purity//FA on installation or upgrade:

```
pureds role list
Name           Access Policy   Group   Group Base
array_admin    array_admin     -       -
ops_admin      ops_admin       -       -
readonly       readonly        -       -
storage_admin  storage_admin   -       -
```

Optionally, you may configure and use these system-defined mappings, but that is not required.To use a system defined mapping, configure the mapping group and group base with the **`pureds role setattr`** command (described below). The name or access policy of a system-defined mapping cannot be changed.

Mapping Creation and Configuration

The **`pureds role create`** command creates a named mapping of an existing LDAP group to one or more Purity//FA access policies.

Create a new mapping:

This example creates a mapping, named `adminmap1`, of the LDAP group `group1` to the Purity//FA access policy `realm1_admin_pol`. As a result of this mapping, users who are members of the LDAP group `group1` now are assigned the Purity//FA access policy `realm1_admin_pol`. (The `--group-base` values are truncated in these examples.)

```
pureds role create adminmap1 --access-policy realm1_admin_pol --group group1 --group-
base OU=Pure
```

Different LDAP groups can share the same Purity//FA access policies.

Display group-to-role mappings:

The **pureds role list** command displays the current LDAP group to Purity//FA access policy mappings:

```
pureds role list
Name           Access Policy     Group    Group Base
array_admin    array_admin       -        -
ops_admin      ops_admin         -       -
readonly       readonly          -        -
storage_admin  storage_admin     -        -
adminmap1      realm1_admin_pol  group1   OU=PureStorage,OU=SAN,OU=IT,OU
storagemap2    storage_admin     group2   OU=PureStorage,OU=SAN,OU=IT,OU
storagemap3    storage_admin     group3   OU=PureStorage,OU=SAN,OU=IT,OU
```

The **pureds role list** output includes each mapping's group and group base. A dash (-) in the output represents an unconfigured attribute of the role.

Modify an existing mapping:

Use the **pureds role setattr** command to change the directory service attributes of a named mapping, including a system-defined mapping. With the **pureds role setattr** command, include the `--group` option to map the common name (CN) of the configured group in the directory tree to the specified Purity//FA access policy. The group name should be just the common name of the LDAP group without the `CN=` specifier. To clear the group setting, set it to an empty string (""). Include the `--group-base` option to set the common organizational unit (OU) under which to search for the group. Specify `OU=` for each organizational unit. The order of OUs should get smaller in scope from right to left. List multiple OUs in comma-separated format. To clear the group base setting, set it to an empty string ("").

Assuming the configuration in the previous example, this example replaces group `group3` in the `storagemap3` mapping with group `group3B`. The LDAP users who are members of `group3` no longer are assigned the Purity//FA access policy `storage_admin`. LDAP users who are members of `group3B` are assigned the Purity//FA access policy `storage_admin`.

```
pureds role setattr storagemap3 --group group3B
Name           Access Policy   Group     Group Base
storagemap3    storage_admin   group3B   OU=PureStorage,OU=SAN,OU=IT,OU
```

Modify a system-defined mapping:

These examples show configuring the system-defined mapping for the `readonly` access policy and then adding another LDAP group to the `readonly` access policy. First, modify the system-defined `readonly` mapping to specify the appropriate group and group base settings for the directory server:

```
pureds role setattr readonly --group groupRO --group-base OU=Pure
Name      Access Policy  Group    Group Base
readonly  readonly       groupRO  OU=PureStorage,OU=SAN,OU=IT,OU=US
```

Then, configure another mapping (named `readonly2`) for a second LDAP group (`groupRO2`) of readonly users:

```
pureds role create readonly2 --access-policy readonly --group groupRO2 --group-b
Name       Access Policy   Group     Group Base
readonly2  readonly        groupRO2  OU=PureStorage,OU=SAN,OU=IT,OU=US
```

Now, all LDAP users in the `groupRO` and `groupRO2` groups are assigned the `readonly` access policy in Purity//FA.

```
pureds role list readonly,readonly2
Name       Access Policy  Group     Group Base
readonly   readonly       groupRO   OU=PureStorage,OU=SAN,OU=IT,OU=US
readonly2  readonly       groupRO2  OU=PureStorage,OU=SAN,OU=IT,OU=US
```

Notes about access policies and mappings:

- Each Purity//FA access policies can be mapped to up to 25 LDAP groups.
- Of those groups, different group bases are allowed.
- The permission level of an individual user is cached locally to prevent frequently binding and querying the directory. The cache entries expire after a time limit at which point the directory is queried again and the cache entry is updated. Cache entries can be refreshed on demand using the pureadmin refresh command. Cache entries are also automatically updated when starting a new session.
- In this release, do not use the GUI to either create or modify mappings of multiple groups to a role. Unexpected results can occur in the GUI. The GUI currently supports only mappings of a single group to each Purity//FA role.
- A mapping name can contain alpha-numeric characters, underscore (“_”), and hyphen (“-”). A mapping name cannot begin with an underscore (“_”).

- A mapping name does not have to match the name of an LDAP group or other configuration. Names are for your use and convenience.
- Mapping names are not case sensitive. The mapping names `Map_AA` and `map_aa` refer to the same (single) mapping.

Validate the Directory Service Configuration

After you have configured the directory service and created the group-to-role mappings, test the configuration to:

- Verify the Uniform Resource Identifiers (URIs) can be resolved and that the array can bind and query the tree using the bind user credentials.
- Verify the array can find all the configured groups to ensure the common names (CNs) and group base are correctly configured. For each configured group, the array binds and queries the directory service to find the configured group. If "check peer" is enabled, the initial bind and query test is repeated while enforcing server authenticity using the CA certificate.

The **`pureds test`** command runs a series of tests to verify that the URIs can be resolved and that the array can bind and query the tree using the bind user credentials. The test also verifies that the array can find all the configured groups to ensure the common names and group base are correctly configured. The directory service test can be run at any time. If "check peer" is enabled, the initial bind and query test is repeated while enforcing server authenticity using the CA certificate.

Note: The use of LDAP is not activated until the service is enabled, using the **`pureds enable`** command.

Enable Use of the Directory Service

The **`pureds enable`** command enables the Pure Storage directory service, allowing users in the LDAP directory to log in to the array. Enable the directory service after you have configured the directory service, created the role mappings, and tested the directory service configuration. Include the **`--check-peer`** option to enable server authenticity enforcement using the configured CA certificate.

Notes on the Check Peer Option

The CA certificate must be configured before the **`--check-peer`** option can be enabled.

Note that including the **`--check-peer`** option only enables the "check peer" feature and does not enable use of the directory service with the array. This option uses the CA certificate, which automatically enforces TLS.

If the `--check-peer` option is enabled:

Directory server authenticity is enforced using the CA certificate.

In this case, TLS is used to ensure a secure connection between the array and the directory server, whether or not `LDAPS` appears in the directory server URL.

If the `--check-peer` option is *not* enabled:

The directory server authenticity is *not* enforced.

When used with `LDAPS` in the directory server URL (ensuring a secure TLS connection between the array and the directory server), this combination is useful for self-signed certificates.

Other pureds Commands

The **`pureds disable`** command disables the directory service, stopping all users in the directory server from logging in to the array. Include the **`--check-peer`** option to disable "check peer" functionality. Note that including the **`--check-peer`** option only disables "check peer"; it does not disable the actual directory service.

The **`pureds list`** command displays the current base configuration. Include the **`--ca-certificate`** option to display currently-configured CA certificate data.

LDAP for File Services

The service named **`data`** is reserved for file users. Configuring and then enabling the directory service for data allows users in the LDAP directory to log in to NFS exports using AUTH_SYS authentication.

An LDAP user is an individual account in the LDAP directory. Each user optionally belongs to a group. If an LDAP user has the same name as a locally created user, the LDAP user always has priority.

LDAP users and groups follow the POSIX standard for integration with systems that adhere to POSIX guidelines. The following LDAP attributes are used by the FlashArray for file services:

cn: The common name of a user or a group. For users, it is often used as part of the distinguished name (DN).

uid: The user identifier is a unique identifier for a user within the directory. It is used for logging in and is often part of the user's DN. This attribute is crucial for distinguishing individual users.

uidNumber: The numeric user identifier is typically used for file ownership and permissions, ensuring that each user has a unique numeric ID within the system.

gidNumber: Optional, similar to uidNumber, this is the numeric group identifier, an attribute for both users and groups. It is used to manage group permissions and ownership and is crucial for defining access controls based on group membership.

mail: Optional, the email address of a user or a group, used for communication purposes, allowing the directory service to store and provide email addresses. For example, for quota alerts.

memberUid: The group identifier, specific to groups, is used to list the members of a group. This attribute contains the uid of each user that is a member of the group.

Run the **`pureds setattr`** command to bind the array to LDAP for access to users and groups:

1. Include the **`--uri`** option to set the URIs of the directory servers with LDAP or LDAPS. Up to 30 URIs can be specified. List multiple URIs in comma-separated format. Each URI must include the scheme `ldap://` or `ldaps://` (for LDAP over SSL), a hostname, and a domain name or IP address. If specifying a domain name, it should be resolvable by the configured DNS servers. Optionally specify a port. Append the port number after the end of the entire address. Default ports are 389 for ldap, and 636 for ldaps. Non-standard ports can be specified in the URI if they are in use. To clear the URI setting, set it to an empty string (`""`).
2. Include the **`--base-dn`** option to set the base distinguished name (base DN) of the LDAP group. The base DN must be in a valid DN syntax. Specify the value for each component. List multiple components in comma-separated format (for example, `DC=storage,DC=company,DC=com`). To clear the base DN setting, set it to an empty string (`""`).
3. Include the **`--bind-user`** and **`--bind-password`** options to specify the user name and password used to bind to and query the directory, respectively. The bind password is entered through an interactive prompt. Enter the full DN of the user. For example, "`CN=John,OU=Users,DC=example,DC=com`". The bind account must be configured to allow the array to read the directory. It is good practice for this account to not be tied to any actual person and to have different password restrictions, such as "password never expires". The bind account should also not be a privileged account, since only read access to the directory is required.
4. Specify the service name: **`data`**.

For example, run the following command to set URI and set bind user credentials for use with the **`data`** service:

```
pureds setattr --uri "ldaps://10.15.61.221" --base-dn "DC=example,DC=com" --bind-user
"CN=John,OU=Users,DC=example,DC=com" --bind-password data
```

The data service is created but not enabled. Use the **`enable`** subcommand to enable the **`data`** service:

```
pureds enable data
```

Local Users and Groups

The **`pureds local user`** and **`pureds local group`** commands manage local users and groups for file services, as an alternative to external authentication solutions like Active Directory or LDAP. With local users and groups, clients can connect to the FlashArray File domain and authenticate with their respective credentials.

Permissions are managed from the client side, for example through Windows Explorer or Computer Management, by adding and removing permissions to users or groups.

Local Users for file is a separate concept from FlashArray local users. The main purpose of local users for file is to access file systems via SMB or NFS protocols, while the purpose of FlashArray local users is to manage the array.

A user is a local user account which includes a username and password. Each user is a member of one primary group. Before creating a user, its primary group must be created if it does not already exist. A user can also be a member of other groups, denoted as secondary groups. Users can be enabled or disabled. Enabling a user is only allowed when the password is set. A membership cannot be removed if the containing group is the user's primary group.

A group is a local group account under which one or more users can be gathered for simplified management of permissions. For example, accounting, development, sales, and so on. A group can have many members. Only users can be members of a group, not other groups. Before deleting a group, all members must be removed from the group.

External members, user accounts or groups that reside on external AD or LDAP servers, can be included in local groups as well. The purpose of this is to authenticate external users through the local group, similar to local users, and authenticate the user within the array, rather than the entire domain. Use the **`--external`** option to include an external member.

Run the **`pureds local group list`** or **`pureds local user list`** command to view available groups or users. Valid options include **`--member`**, **`--sid`**, **`--email`**, and, for group, **`--member-sid`** to display the SID of external members.

The following local users and groups are built in. These built-in users and groups cannot be removed or modified:

- **Groups**:
  - Administrators (GID 0)
  - Guests (GID 65534)
  - Backup Operators (GID 65535)
- **Users**:
  - Administrator (UID 0, member of the Administrators group)
  - Guest (UID 65534, disabled by default, member of the Guests group)

To create a group, use the **`pureds local group create`** command with the name of the new group. A unique group GID is automatically set, or use the **`--gid`** option to override. Use the **`--email`** option to include the email address, used for example for quota notifications.

The local group name must be between 1 and 63 characters in length and can contain alphanumeric US ascii, space, or symbols. The name cannot be numbers only, start or end with dot or space, contain any control characters or any of the following characters: `" / \ [ ] : ; | = , + * ? < >`. Names are case-insensitive on input. Purity//FA displays names in the case in which they were specified when created or renamed.

To create a user, use the **`pureds local user create`** command. To set the mandatory primary group, include **`--primary-group`** and the name of the group. A unique user UID is automatically set; use the **`--uid`** option to override. The user must be created with the **`--password`** or **`--disable`** option, or both. When setting the password, the password is manually typed twice through an interactive prompt. The password is not shown while typing. If the passwords do not match, the user is not created. Use the **`--email`** option to include an email address. The local user name must be between 1 and 20 characters, with the same characteristics as for the group name.

The user account can at any time be enabled or disabled with the corresponding subcommand, **`enable`** and **`disable`**. When disabling the account, to avoid closing any open session, use the **`--keep-open-sessions`** option. Enabling a user is only allowed when the password is set.

To add members to an existing group, use the **`pureds local group add`** command. Valid options include **`--user`**, **`--group`**, and **`--external`**, for user, group, or external member, respectively. Members can be one or a comma-separated list of many.

There are two ways to remove memberships between groups and users: Remove a user from one or more groups, or detach a group from one or more users. To remove a user from groups, use the **`pureds local user remove`** command with the **`--group`** option and a comma-separated list of one or more groups. To detach a group from users, use the **`pureds local group remove`** command with one of the options **`--user`**, **`--group`**, or **`--external`**. The list of names is comma-separated. A membership cannot be removed if the containing group is the user's primary group.

Users and groups can be modified with the **`pureds local user setattr`** and **`pureds local group setattr`** commands. For users, valid options include **`--primary-group`**, **`--password`**, **`--uid`**, and **`--email`**. To avoid closing the open sessions for users for which the password is being set, use the **`--keep-open-sessions`** option. For groups, the options include **`--gid`** and **`--email`**.

To rename users or groups, use the **`rename`** subcommand. To delete users or groups, use the **`delete`** subcommand.

Certificates

If the configured directory servers have been issued certificates, the certificate of the issuing certificate authority can be stored on the array to validate the authenticity of the directory servers. When performing directory queries, the certificate presented by the server is validated using CA certificate.

Server authenticity is only enforced if the "check peer" option is enabled. "Check peer" can only be enabled if a CA certificate has been configured. See also "Notes on the Check Peer Option" on page 161.

Only one certificate can be configured at a time, so the same certificate authority should be the issuer of all directory server certificates. The certificate must be PEM formatted (Base64 encoded) and include the **`"-----BEGIN CERTIFICATE-----"`** and **`"-----END CERTIFICATE-----"`** lines. The certificate cannot exceed 3000 characters in total length.

When certificate data with valid syntax is supplied, the certificate trust is checked to determine if the certificate is self-signed or signed by a trusted root certificate authority. If the trust cannot be determined, the certificate data can still be saved, but server authenticity enforcement using the certificate may fail.

As a convenience, the Pure Storage directory service can attempt to automatically fetch CA certificate data from the directory server. If certificate data is successfully retrieved from the server, it undergoes the same trust check as manually entered certificate data, however if trust cannot be determined the operation will not continue. If certificate data is successfully retrieved and the CA certificate is trusted, a final prompt to confirm the data is displayed.

Examples

Example 1

```
pureds setattr --uri ldaps://[2001:0db8:85a3::ae26:8a2e:0370:7334]
               --base-dn DC=mycompany,DC=com --bind-user ldapreader
```

Sets the URI to IPv6 address **`2001:0db8:85a3::ae26:8a2e:0370:7334`** and the scheme to **`ldaps://`** to enable SSL. Also sets the base DN and sets the bind user as the user name used to bind to and query the directory.

Example 2

```
pureds setattr --uri ldaps://ad1.mycompany.com,ldaps://ad2.mycompany.com
               --base-dn DC=mycompany,DC=com
                               --bind-user CN=John,OU=Users,DC=mycompany,DC=com
```

Sets the URI to both **`ad1.mycompany.com`** and **`ad2.mycompany.com`** and the scheme to **`ldaps://`** to enable SSL. Also sets the base DN and sets the bind user as the user name used to bind to and query the directory.

Example 3

```
pureds setattr --bind-password
Enter bind password:
Retype bind password:
```

Shows the interactive prompt for entering a password for the bind user account. The password is not shown while typing, so a confirmation prompt is presented. If the passwords do not match, no change is made.

Example 4

```
pureds setattr data --base-dn dc=mydomain,dc=com --bind-password
--bind-user cn=admin,dc=mydomain,dc=com --uri ldap://192.168.20.2
pureds enable data
```

Prepares the connection of the FlashArray file services to an LDAP server, using the **`data`** service. The data service must be activated by enabling it.

Activates the use of LDAP directory service, allowing users in the LDAP directory to log in to the file services.

Example 5

```
pureds list
```

Displays the current base configuration of the Pure Storage directory service. Attributes include URI, base DN, bind user, "check peer" status, and enabled status.

Example 6

```
pureds role list storage_admin
```

Displays the group and group base configuration for the `storage_admin` access policy in the array.

Example 7

```
pureds test
Output
Feature Status: Enabled
Testing from ct0:
Resolving dc01.domain.com...                  PASSED
Searching ldaps://dc01.domain.com:636...      PASSED
Searching for group CN=purereadonly...        PASSED
Searching for group CN=pureopsadmins...       PASSED
Searching for group CN=pureusers...           PASSED
Searching for group CN=pureadmins...          PASSED
Resolving dc02.domain.com:636...              PASSED
Searching ldaps://dc02.domain.com...          PASSED
Searching for group CN=purereadonly...        PASSED
Searching for group CN=pureopsadmins...       PASSED
Searching for group CN=pureusers...           PASSED
Searching for group CN=pureadmins...          PASSED
```

Tests the current directory service configuration, showing successful test output.

Example 8

```
pureds enable
```

Enables use of the directory service for array authentication. Use only after `pureds test` passes.

Example 9

```
pureds enable --check-peer
```

Enables the check peer feature, which enforces checking the authenticity of the directory server through the CA certificate.

If directory service is not already enabled on the array, the check peer feature will be enabled when directory services are enabled at a later date.

Example 10

```
pureds disable --check-peer
```

Turns off the check peer feature that enforces checking the authenticity of the directory server through the CA certificate. Use of the directory service for array authentication remains enabled.

Example 11

```
pureds disable
```

Disables use of the directory service for array authentication

Example 12

```
pureds local user list
```

Lists the current directory of local users with their associated group and UID information.

Example 13

```
pureds local group list --sid --email
```

Lists the directory of local users, including their automatically generated security identifier and email addresses.

Example 14

```
pureds local group create Group1 --email group1@mydomain.com
```

Creates the group `Group1` with a default GID. An email address is included which is used for notifications, etc.

Example 15

```
pureds local user create User1 --password --email user1@mydomain.com --primary-group
Group1
Enter password:
Retype password:
```

Creates a local user named `User1` with a manually-created login password that is entered twice through the interactive prompt. The password is not shown while typing. If the passwords do not match, the user is not created. The user is assigned `Group1` as its primary group.

Example 16

```
pureds local user add User1 --group Group2,Group3,Group4
```

Adds the user `User1` to several groups.

Example 17

```
pureds local group add Group2 --user User1,User2,User3
```

Adds several users to the group `Group2`.

Example 18

```
pureds local user setattr User1 --password
Enter password:
Retype password:
```

Sets the password for local user `User1` to the new password. The new password is entered twice through the interactive prompt and not shown while typing.

Example 19

```
pureds role create adminmap1 --access-policy array_admin --group GroupA1 --group-base
OU=PureStor
```

Creates a named mapping of the existing LDAP group named `GroupA1` to the Purity//FA access policy `array_admin`. Members of `GroupA1` are assigned the Purity//FA access policy `array_admin`.

(`--group-base` values are truncated in these examples.)

Example 20

```
pureds role create adminmap2 -role array_admin --group GroupA2 --group-base OU=PureStor
pureds role create adminmap3 -role array_admin --group GroupA3 --group-base OU=PureStor
```

Creates additional mappings of other LDAP groups (`GroupA2`, `GroupA3`) to the Purity//FA access policy `array_admin`. Members of those LDAP groups are also assigned the access policy `array_admin`.

Example 21

```
pureds role list
```

List the current LDAP group to Purity//FA access policy mappings on the array.

Example 22

```
pureds role list adminmap2
```

List the details of the specified group to Purity//FA access policy mapping on the array.

Example 23

```
pureds role list --filter "role.name='array_admin'"
```

List the current mappings for the `array_admin` access policy.

Example 24

```
pureds role list --filter "group='groupRO'"
```

List the current mappings that involve the `groupRO` group.

See Also

puread, pureadmin, puredns, purepolicy

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purefile

purefile, purefile-copy – manages fast copying of files from one managed directory to another, within or between file systems

Synopsis

```
purefile copy [--overwrite] --source-dir SOURCE_DIR [--target-dir
TARGET_DIR] --source-path SOURCE_PATH [TARGET_PATH]
```

Options

`-h | --help`
: Can be used with any command or subcommand to display a brief syntax description.

`--overwrite`
: Allows any existing file to be overwritten.

`--source-dir` ***SOURCE_DIR***
: Full name of the source managed directory.

`--source-path` ***SOURCE_PATH***
: Path of the source file relative to source managed directory.

`--target-dir` ***TARGET_DIR***
: Full name of the target managed directory. Only required if the target managed directory is different from the source, or when the target path is not specified. If the target managed directory is not specified, it is equal to the source directory.

Arguments

TARGET_PATH

Path of the file relative to the target managed directory. Only required if the target managed directory is not specified. If the target path is not specified, it will be equal to the source path.

Description

The **`purefile copy`** command creates a copy of a file from one path to another. The source and the target can be located on any managed directory, on any file system on the FlashArray. The advantage of this method over native copy or clone is that the process is faster. This is useful functionality for datastore, restoring a VM, or similar file management tasks, to be able to quickly make a file clone from one location to another. The command can only create a copy of one file at a time.

Use the **`--source-dir`** option to specify the full name of the managed directory where the source file is located. If the copy is to be created on a different managed directory, specify with the **`--target-dir`** option. If the target managed directory is not specified, it is equal to the source directory. In that case, the target path must be specified.

Use the **`--source-path`** option to specify the source path relative to the source managed directory. Specify the target file path, relative to the target managed directory. If the target path is not specified, it will be equal to the source path. The target path must be specified if the target managed directory is not specified.

The target file will be created and the copy is carried out. However, if any of the the source or target managed directory, or source or target path, is invalid, the process is aborted.

Use the **`--overwrite`** option if you want the target file to be overwritten. Otherwise, if the target file exists, the process is aborted and the file copy is not carried out.

If a target file is overwritten with the **`--overwrite`** option, a snapshot is automatically created on the target managed directory, using the client name “copy”. For example: **`FS2:MD2.copy.1`**. The suffix is an incremental number, similar to a regular snapshot. This snapshot is automatically destroyed and is available for restoring the file only until it is automatically eradicated according to your eradication timer settings.

In the following example, specifying the target path **`/dir1/dir2/newfile`** was only necessary since it differs from the source path:

```
purefile copy --source-dir FS1:MD1 --target-dir FS2:MD2 --source-path /dir1/dir2/file
/dir1/dir2/newfile
```

Upon a successful completion, the CLI outputs the full name of the new file:

```
Name
FS2:MD2\dir1/dir2/newfile
```

Examples

Example 1

```
purefile copy --source-dir FS1:Data --source-path /dir1/file /dir1/file2
```

Executes a fast file copy, creating a clone named **`file2`**, to the same managed directory.

Example 2

```
purefile copy --source-dir FS1:Data --target-dir FS2:Data --source-path /dir1/file
```

Executes a fast file copy, creating a clone of **`file`** on a different managed directory and file system..

Example 3

```
purefile copy --overwrite --source-dir FS1:Data --target-dir FS2:Data --source-path
/dir1/file
```

Executes a fast file copy, creating a clone of **`file`** on a different managed directory and file system. If the target file already exists, it will be overwritten.

See Also

purefs, puredir, purearray

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purefleet

purefleet, purefleet-create, purefleet-list, purefleet rename – creates and manages fleets of one or more arrays

purefleet-member – manages array fleet membership

Synopsis

```
purefleet create [--fleet-key] [NAME]
purefleet list [--fleet-key] [--cli | --csv | --nvp] [--notitle] [--raw] [--filter FILTER] [--page] [--limit LIMIT] [--sort SORT]
purefleet member add --fleet FLEET --fleet-key
purefleet member list [--csv | --nvp] [--notitle] [--raw] [--filter FILTER] [--page] [--limit LIMIT] [--sort SORT]
purefleet member remove [--unreachable] ARRAY
purefleet rename NEW_NAME
```

Arguments

ARRAY

The array to be removed.

FLEET

The fleet to be created or listed.

NAME

The fleet name.

A fleet name is 1 to 63 characters in length. Valid characters are letters (A-Z and a-z), digits (0-9), and the hyphen (-) character. The first and last characters of the name must be alphanumeric, and the name must contain at least one letter or '-'. Names are case-insensitive on input. For example, `fleet1`, `Fleet1`, and `FLEET1` all represent the same fleet. Purity//FA displays names in the case in which they were specified when created or renamed.

NEW_NAME

The new name of the fleet.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--fleet`

Name of the fleet to create or join.

`--fleet-key`

An encrypted key generated on any fleet member array.

Required with the **`purefleet member add`** command to confirm that the new array has permission to join the fleet.

With the **`purefleet list`** command, displays the fleet key creation and expiration time stamps.

`--unreachable`

Used with the **`purefleet member remove`** command when an array cannot reach the rest of the fleet. When the **`--unreachable`** option is used, the command has to be run twice, once on the array to removed and once on any array in the rest of the fleet.

.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Overview

Fleets and Remote Management

The `purefleet` command provides preliminary array federation and fleet management for on-premises arrays. Cloud access is not required. For information about creating a fleet and managing fleet member arrays, see "Description" on page 181 and "Steps to Create a Fleet with Multiple Array Members" on page 183.

Once a fleet has been created, remote storage management is supported across arrays that are members of that fleet. With the `--context` ***`REMOTE`*** option, CLI commands can be run on a different array in the fleet, as opposed to on the local array. For information about remote provisioning and management across arrays within a fleet, see "Remote Commands for Fleet Member Arrays" on page 184.

Requirements and Restrictions

The following are requirements for arrays in a fleet:

- All arrays in a fleet must be able to connect to each other via management network ports (ex: vif0).

  All arrays must be able to communicate with each other on the network over port 443.
- Directory services (Active Directory or LDAP) must be enabled with a consistent configuration on each array in the fleet. Remote provisioning operations are restricted to users authenticated by directory services.
- Arrays in a fleet can run on different versions of Purity//FA, with at least one running on a 6.8.1 or later release.

Secure Communication in Arrays

In a fleet of arrays, secure communication is established using mutual Transport Layer Security (mTLS). This requires each array to share a common certificate, ensuring that all communications within the fleet are authenticated and encrypted. To maintain security, certificate rotation typically occurs every 45 days, with a Time-To-Live of 90 days. If an array loses connection to the fleet for an extended period and misses the certificate rotation, it will no longer be able to communicate with the fleet until it is reconnected and updated with the current certificate. This approach ensures that only authorized arrays can participate in the fleet, maintaining the integrity and security of the system.

In the case that the array becomes disconnected from the fleet for longer than the standard 45 days and the certificate expires, the array will not be able to connect to the fleet any longer. In order to renew the certificate, run the **`purefleet member remove --unreachable`** ARRAY command on both the array and any other array in the fleet to grant the array a new certificate.

Fleet Keys

A fleet key is an encrypted key containing the information necessary for another array to join an existing fleet.

Because a fleet key can only be created on an array that already is a member of the fleet, the key serves as confirmation that another array is authorized to join the fleet. A valid fleet key is required with the **`purefleet member add`** command for a new array to join a fleet.

Each fleet key expires after its first use or after an hour if not used.

Description

The **`purefleet create`** command creates a fleet key or creates a fleet. Include the ***`FLEET`*** name argument to create a fleet with that name. To change the name of the fleet use the **`purefleet rename`** command.The new fleet initially includes one array, the array on which the **`purefleet create`** command was run. This example creates a fleet named `fleet-1` consisting of one member array, which is the array on which the command was run.

```
$ purefleet create "fleet-1"
Name
fleet-1
```

Use the **`--fleet-key`** option to create a fleet key, which is required to add other arrays to the fleet. This command must be run on an array that is a member of a fleet, and it creates a fleet key for that fleet.

Make a note of the fleet key in this output, as this is the only time the key is displayed. The fleet key is valid for one use only and expires after an hour if not used.

```
$ purefleet create --fleet-key
Fleet Key           Created               Expires
xyzEXAMPLE_KEYabc   2024-04-25 20:50:33   2024-04-25 21:50:33
```

Running the **`purefleet create --fleet-key`** command a second time deletes the previous key, if any, and creates a new key.

The **`purefleet list`** command displays
information about a fleet key or information about one or more fleets. With no options or

arguments, the **`purefleet list`** command displays the name of the current fleet (the fleet that the current array is a member of). The **`purefleet list`** output is the same when the command is run on any member array of the fleet.

```
purefleet list
Name
fleet-1
```

With the **`--fleet-key`** option, **`purefleet list --fleet-key`** command displays the creation and expiration timestamps for the fleet key, and does not display the key itself.

```
purefleet list --fleet-key
Created               Expires
2024-04-25 20:50:33   2024-04-25 21:50:33
```

The **`purefleet member add`** command adds a new member array to an existing fleet. On the array that is to be added to the fleet, run the **`purefleet member add --fleet`** ***`FLEET`*** **`--fleet-key`** command. Purity//FA prompts you to enter the fleet key, which was displayed in the **`purefleet create --fleet-key`** command output. The command cannot be completed without the correct fleet key, which cannot have been used previously and cannot have expired. In this example, the current array is added to `fleet-1`:

```
$ purefleet member add --fleet fleet-1 --fleet-key
Enter fleet key: <type in the fleet key>
Fleet     Name      Status
fleet-1   Array2    Joined
fleet-1   Array1    Joined
```

The **`purefleet member add`** command may retry multiple times to add a new member array to an existing fleet without success. It will continue retrying until instructed otherwise. If you know that the action will not be successful, use the **`purefleet member remove --unreachable`** command to abort the action to add a new member array.

The **`purefleet member list`** command lists all member arrays in the current fleet with their status. The possible values for Status are the following:

- **Joined**. Array has successfully joined the fleet.
- **Joining**. Array is currently joining the fleet.
- **Removing**. Array is currently being removed from the fleet.

The `purefleet member remove` command removes the specified array from the fleet. Use the `--unreachable` option to remove the current array when the fleet is not reachable. In that case, also run the `purefleet member remove --unreachable` command on a separate array in the fleet in order to update the fleet that the array is removed. When removing a member array that is not the current array, the remove command fails if the current array is unable to reach the fleet, unless `--unreachable` was specified.

If the `purefleet member add` command encounters an issue, it is because the system automatically and continuously retries on error without ever giving up. If you know that the join will not succeed, for example, if the array is not physically connected to the rest of the fleet, you can use the `purefleet member remove --unreachable` command to abort the join.

Steps to Create a Fleet with Multiple Array Members

This section describes the steps involved in setting up a fleet. Those steps are summarized here:

- On an array that is to be a member of the fleet, create a fleet.
- Create the fleet key for that fleet.
- On a second array that is to be a member of the fleet, use the fleet key to join the fleet. A fleet key can only be used once and cannot be used after its expiration.
- Repeat step 3 as needed.

1. On array `array-1`, that is to be a member of the fleet, create the fleet with the `purefleet create NAME` command.

```
$ purefleet create "fleet-1"
Name
fleet-1
```

   Fleet `fleet-1` is created with array `array-1` as the only member array.

2. On array `array-1`, create the fleet key for the new fleet.

```
$ purefleet create --fleet-key
Fleet Key              Created                    Expires
```

```
xyzEXAMPLE_KEYabc   2024-04-25 20:50:33  2024-04-25 21:50:33
```

Creates the fleet key for the current fleet (`fleet-1`). Make a note of this key to use in later steps.

3 To add another array to the fleet, on the array that is to be added to the fleet, run the **`purefleet member add --fleet fleet-1 --fleet-key`** command. Purity//FA prompts you to enter the fleet key, which was displayed in the previous step.

```
$ purefleet member add --fleet fleet-1 --fleet-key
Enter fleet key: <type in the fleet key>
Fleet      Name       Status
fleet-1    Array-2    Joined
fleet-1    Array-1    Joined
```

The current array (`array-2`) is added to `fleet-1`.

4 Repeat step 3 as needed, each time from an array that is to be added to the fleet.

Remote Commands for Fleet Member Arrays

Grouping arrays into a fleet brings increased flexibility in remote resources provisioning and management. On arrays within a fleet, the `--context` ***REMOTE*** option can be used to create, manage, and list objects on a different array in the fleet, and not only on the local array.

For example, the `purevol list` command lists volumes on the local array. The `purevol list --context array2` command lists volumes on `array2`, when `array2` and the local array are members of the same fleet.

The examples below illustrate the scenario in which the local array (where the command is executed) and the remote array, `array2`, are both members of the same fleet. The `--context` ***REMOTE*** option is used with the `purehost`, `purevol`, `purehgroup`, and `pureaudit` commands.

The following command, when run on a different array in the same fleet, creates host `DBHost03` on `array2`:

```
$ purehost create --context array2 DBHost03
Context   Name    WWN IQN   NQN
array2 DBHost03   - - -
```

The following command lists hosts on the remote array `array2`:

```
$ purehost list --context array2
Context   Name       WWN   IQN                             NQN   Host Group
array2    DBHost-03  -     -                               -     -
array2    esx1-a     -     iqn.1998-01.com.vmware:esx1-a   -     esx-hosts
array2    esx1-b     -     iqn.1998-01.com.vmware:esx1-b   -     esx-hosts
array2    esx1-c     -     iqn.1998-01.com.vmware:esx1-c   -     esx-hosts
```

The following commands provision a volume on the remote array `array2` and attach that volume to host `DBHost03`:

```
$ purevol create DBHost03-Vol1 --size 100M --context array2
Context   Name           Size  Source  Created         Serial              Protection
array2    DBHost03-Vol1  1G      -        2024-10-22 PDT  4EB81F1E06200115DB pgroup-
auto

$ purehost connect --vol DBHost03-Vol1 --context array2 DBHost03
Context   Name        Vol              LUN   NSID
array2    DBHost03    DBHost03-Vol1    1     6
```

The following command creates the host group `DBHostGroup` on the remote array `array2` with host `DBHost03` as a member of that hostgroup:

```
$ purehgroup create --hostlist DBHost03 --context array2 DBHostGroup
Context   Name          Hosts
array2    DBHostGroup   DBHost03
```

The following command lists the host groups on the remote array `array2`:

```
$ purehgroup list --context array2
Context   Name          Hosts
array2    DBHostGroup   DBHost03
array2    esx-hosts     esx1-a
```

```
                        esx1-b
                        esx1-c
```

The following command lists the audit entries on the remote array `array2`:

```
$ pureaudit list --context array2
Context   ID    Time            User      Origin   Command    Subcommand  Arguments
array2    268   2024-04-24 UTC  pureuser  array2   purefleet  member add  --fleet
fleet1  --fleet-key
array2    271   2024-04-24 UTC  pureuser  array2   purehost   create      DBHost03
array2    272   2024-04-24 UTC  pureuser  array2   purehgroup create
DBHostGroup
```

Examples

Example 1

```
purefleet create fleet2
```

Creates a fleet named `fleet2` with the current array as its only member.

Example 2

```
purefleet create --fleet-key
```

Creates a fleet key for the current fleet (the fleet for which the array is a member).

Make a note of the fleet key, as it cannot be obtained by other commands. The key is valid for one hour and for one use only.

Example 3

```
purefleet create fleet2 --fleet-key
```

Creates a fleet key for the fleet `fleet2`. The key is valid for one hour and for one use only.

Make a note of the fleet key, as it cannot be obtained by other commands.

Example 4

```
purefleet list
```

Lists the fleet name that the current array is a member of, if any.

Example 5

```
purefleet list --fleet-key
```

Lists the creation and expiration times for the fleet key for the fleet that the current array is a member of, if any.

Example 6

```
purefleet list fleet2 fleet3
```

Lists the fleets named `fleet2` and `fleet3`, if those fleets exist.

Example 7

```
purefleet member add --fleet fleet1 --fleet-key
```

Purity//FA prompts you to enter the fleet key, which was displayed in the **`purefleet create --fleet-key`** command output. Makes the current array a member of `fleet1`.

Example 8

```
purefleet member remove array1
```

Purity//FA prompts you to enter the fleet key, which was displayed in the **`purefleet create --fleet-key`** command output. Removes the specified array from its fleet. The specified array must be reachable from its fleet.

Example 9

```
purefleet member remove array1 --unreachable
```

Purity//FA prompts you to enter the fleet key, which was displayed in the **`purefleet create --fleet-key`** command output. Removes the specified array from its fleet, when the specified array is not reachable from its fleet.

Note: This command must be run twice, on a fleet member array and on a separate array in the fleet in order to update the fleet that the specified array was removed.

Example 10

```
purefleet list --filter "name = 'abc*'"
```

Displays a list of fleets with names that begin with "abc".

See "Filtering" on page 14 for more information on filtering CLI output.

Exceptions

None.

See Also

purearray, pureaudit, purehgroup, purehgroup-connect, purehost, purehost-connect, purevol

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purefs

purefs, purefs-create, purefs-destroy, purefs-eradicate, purefs-move, purefs-recover, purefs-rename, purefs-list – manages the creation, modification, destruction, and listing of FlashArray file systems

Synopsis

purefs `create [--context` ***REMOTE***`]` ***NAME...***

purefs `destroy [--context` ***REMOTE***`]` ***NAME...***

purefs `eradicate [--context` ***REMOTE***`]` ***NAME...***

purefs `move [--context` ***REMOTE***`]` ***NAME... POD***

purefs `recover [--context` ***REMOTE***`]` ***NAME...***

purefs `rename [--context` ***REMOTE***`]` ***OLD-NAME... NEW-NAME...***

purefs `list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token` ***TOKEN***`] [--raw] [--filter` ***FILTER***`] [--limit` ***LIMIT***`] [--sort` ***SORT***`] [--pending | --pending-only] [--context` ***REMOTES***`] [`***NAME...***`]`

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--context` ***REMOTES***

Used to specify the remote array or arrays on which a command is processed. ***REMOTE*** is a comma-separated list of names of arrays within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--pending`

Displays a list of file systems, including those that have been destroyed and are in the pending eradication state. If not specified, file systems that are pending eradication are not shown.

`--pending-only`

Displays only destroyed file systems that are in the pending eradication state.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--page-with-token`

Used to paginate output with a continuation token (**`--token TOKEN`**). Results are printed in `stdout` and the token is printed in `stderr`.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Arguments

NAME

The name of the file system to be created, modified, or displayed.

NEW-NAME

Name by which the file system is to be known after the command executes.

OLD-NAME

Current name of the file system to be renamed.

Description

The **`purefs`** command manages the creation, naming, destruction, and listing of FlashArray file systems.

The **`purefs create`** command, followed by a name, creates a file system. The root directory, which is automatically created, is a managed directory named "root". This directory cannot be deleted, it can only be destroyed along with the file system.

File systems can be created directly in pods using the **`purefs create`** command. Once created, a file system can be moved into another pod (POD) using the **`purefs move`** command. Directory snapshots managed by a snapshot policy will become unmanaged after the file system is moved into a different pod. Policies attached to managed directories in the file system will be copied into the destination pod if no equivalent policy already exists; the policy attachments will be updated to use the equivalent policies in the destination pod. During ActiveDR, file systems can be deleted.

File systems can be destroyed, eradicated, or recovered using the **`purefs destroy`**, **`purefs eradicate`**, and **`purefs recover`** commands, respectively, followed by the name of one or more file systems.

The **purefs rename** command changes the current name (**OLD-NAME**) of a file system to the new name (**NEW-NAME**). The name change is effective immediately and the old name is no longer recognized in CLI, GUI, or REST interactions. In the Purity//FA GUI, the new name appears upon page refresh.

The **purefs list** command displays a list with information about file systems on the array. Add the **--pending** or the **--pending-only** option to include destroyed and pending file systems, or only destroyed and pending file systems, respectively.

Examples

Example 1

```
purefs create filesystem1
```

Creates a file system named **filesystem1**.

Example 2

```
purefs list
```

Displays a list of all file systems in the array.

Example 3

```
purefs destroy filesystem1
```

Destroys the file system.

Example 4

```
purefs eradicate filesystem1
```

Eradicates and completely removes a destroyed file system.

Example 5

```
purefs move filesystem1 pod1
```

Moves file system **filesystem1** into the pod **pod1**.

See Also

puredir

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purehgroup

purehgroup, purehgroup-create, purehgroup-delete, purehgroup-rename, purehgroup-setattr – manage the creation, deletion, naming, and population of Purity//FA host groups.

purehgroup-add, purehgroup-remove – manage adding and removing of Purity//FA host groups to and from protection groups, respectively

purehgroup-monitor – monitor host group I/O performance.

Synopsis

purehgroup create [--hostlist ***HOST-LIST***] [--context ***REMOTE***] ***HGROUP...***

purehgroup delete [--context ***REMOTE***] ***HGROUP...***

purehgroup add [--pgroup ***PGROUP***] ***HGROUP...***

purehgroup remove [--pgroup ***PGROUP***] ***HGROUP...***

purehgroup monitor [--array] [--csv] [--filter ***FILTER***] [--historical {1h | 3h | 24h | 7d | 30d | 90d | 1y}] [--interval ***SECONDS***] [--latency] [--limit ***LIMIT***] [--mirrored] [--notitle] [--page] [--raw] [--repeat ***REPEAT-COUNT***] [--size] [--sort ***SORT***] [--total] [***HGROUP...***]

purehgroup rename [--context ***REMOTE***] ***OLD-NAME NEW-NAME***

purehgroup setattr {--addhostlist ***HOST-LIST*** | --hostlist ***HOST-LIST*** | --remhostlist ***HOST-LIST***} [--context ***REMOTE***] ***HGROUP***

For **purehgroup connect**, see purehgroup-connect.

For **purehgroup list**, see purehgroup-list.

Arguments

HGROUP

Host group name.

NEW-NAME

Name by which the host group is to be known after the command executes.

OLD-NAME

Current name of the host group to be renamed.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--addhostlist`

Comma-separated list of one or more additional hosts to be associated with the host group. Has no effect on hosts already associated with the group.

`--array`

In an ActiveCluster replication configuration, breaks down performance data by the array to which the I/O is directed.

`--context` ***REMOTE***

Used to specify the remote array on which a command is processed. ***REMOTE*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--historical`

Displays historical performance data over the specified range of time. Valid time ranges include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year.

`--hostlist`

Comma-separated list of one or more hosts to be associated with the host group. When specified in the **`purehgroup setattr`** command, replaces the entire membership of a host group. To disassociate all hosts from a host group, set to an empty string (`""`).

`--pgroup` ***PGROUP***

Comma-separated list of protection groups to which the specified host groups are added or from which the specified host groups are removed. Has no effect on host groups already associated with the protection group.

`--interval` ***SECONDS***

Sets the number of seconds between displays of real-time performance data. At each interval, the system displays a point-in-time snapshot of the performance data. If omitted, the interval defaults to every 5 seconds.

`--latency`

Displays real-time and historical I/O latency information.

`--mirrored`

In an ActiveCluster replication configuration, includes performance data for mirrored writes (that is, writes to volumes in stretched pods). If one array of a stretched pod is off-line, the writes processed on the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again.

`--remhostlist`

Comma-separated list of one or more hosts whose associations with the host group are to be broken.

`--repeat` ***REPEAT-COUNT***

Sets the number of times to display real-time performance data. If omitted, the repeat count defaults to 1.

`--size`

Displays the average I/O sizes per read and write operation.

`--total`

Follows output lines with a single line containing column totals in columns where they are meaningful.

Options that control display format:

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **purearray monitor** output, the unformatted version of column title **us/op (read)** is **usec_per_read_op**. The **--raw** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

Object Names

Valid characters are letters (A-Z and a-z), digits (0-9), and the hyphen (-) character. The first and last characters of the name must be alphanumeric, and the name must contain at least one letter or '-'.

Most objects in Purity//FA that can be named, including host groups, hosts, volumes, protection groups, volume and protection group suffixes, SNMP managers, and subnets, can be 1-63 characters in length.

Array names can be 1-56 characters in length. The array name length is limited to 56 characters so that the names of the individual controllers, which are assigned by Purity//FA based on the array name, do not exceed the maximum allowed by DNS.

Names are case-insensitive on input. For example, **vol1**, **Vol1**, and **VOL1** all represent the same volume. Purity//FA displays names in the case in which they were specified when created or renamed.

Pods and volume groups provide a namespace with unique naming conventions.

All objects in a pod have a fully qualified name that include the pod name and object name. The fully qualified name of a volume in a pod is **POD::VOLUME**, with double colons (**::**) separating the pod name and volume name. The fully qualified name of a protection group in a pod is **POD::PGROUP**, with double colons (**::**) separating the pod name and protection group name. For example, the fully qualified name of a volume named **vol01** in a pod named **pod01** is

pod01::vol01, and the fully qualified name of a protection group named **pgroup01** in a pod named **pod01** is **pod01::pgroup01**.

If a protection group in a pod is configured to asynchronously replicate data to a target array, the fully qualified name of the protection group on the target array is **POD:PGROUP**, with single colons (**:**) separating the pod name and protection group name. For example, if protection group **pod01::pgroup01** on source array **array01** asynchronously replicates data to target array **array**02, the fully qualified name of the protection group on target array **array02** is **pod01:pgroup01**.

All objects in a volume group have a fully qualified name that includes the volume group name and the object name, separated by a forward slash (**/**). For example, the fully qualified name of a volume named **vol01** in a volume group named **vgroup01** is **vgroup01/vol01**.

Description

A Purity//FA *host group* is an abstraction that implements *consistent* connections between a set of hosts and one or more volumes. Connections are consistent in the sense that all hosts associated with a host group address a volume connected to the group by the same LUN. Host groups are typically used to provide a common view of storage volumes to the hosts in a clustered application.

Creating Host Groups

The **purehgroup create** command creates a host group and assigns it a name. Optionally, include the **--hostlist** option with the **purehgroup create** command to add hosts during host group creation.

Adding Hosts to Host Groups

Adding a host to a host group automatically connects the host to all volumes associated with the group. A host can only belong to one host group.

Hosts can be added to or removed from a host group at any time.

To add hosts to an existing host group, include the **`--addhostlist`** or **`--hostlist`** option with the **`purehgroup setattr`** command. The **`--addhostlist`** option adds hosts to an existing host group. Hosts that are already associated with the host group are not affected. The **`--hostlist`** option adds hosts to an existing host group, completely replacing any existing hosts that are already associated with the host group.

To remove hosts from a host group, run the **`purehgroup setattr --remhostlist`** command. Alternatively, remove hosts from a host group by running the **`purehgroup setattr --hostlist ""`** command to replace the existing list of hosts with a null set.

Creating Shared Host-Volume Connections

To connect the hosts within a host group to a volume, establish a shared connection by running the **`purehgroup connect`** or **`purevol connect --hgroup`** command. Shared connections can be created or broken at any time. Once a connection is established, the volume is assigned a logical unit number (LUN), which all hosts within the host group use to communicate with the volume.

For more information about shared connections and LUNs, refer to purehgroup-connect.

Deleting Host Groups

The **`purehgroup delete`** command removes host groups that are no longer required. A host group can only be deleted if it is empty, so remove all hosts from a host group before deleting the group.

Renaming Host Groups

Run the **`purehgroup rename`** command to change the current name (**`OLD-NAME`**) of a host group to the new name (**`NEW-NAME`**). The name change is effective immediately and the old name is no longer recognized in CLI, GUI, or REST interactions. In the Purity//FA GUI, the new name appears upon page refresh.

Adding Host Groups to Protection Groups

The `purehgroup add` command adds existing host groups to existing protection groups. Host groups can only be added to protection groups that replicate to target arrays through asynchronous replication. Hosts and host groups are not supported for replication to offload targets.

Multiple host groups can be added to multiple protection groups. Enter multiple protection groups in comma-separated format.

If a protection group already includes other host groups, those host groups are unaffected.

Only members of the same object type can belong to a protection group. For example, a host group cannot be added to a protection group that already contains hosts or volumes.

The `purehgroup add` command only accepts host groups that do not already belong to any of the specified protection groups. If any of the host groups already belong to any of the protection groups, the entire command fails.

Run the `purehgroup list --protect` command to see a list of all protected host groups and their associated protection groups.

The `purehgroup remove` command removes one or more host groups from one or more protection groups. All of the specified host groups must belong to all of the specified protection groups in the command; otherwise, the command fails.

Host groups can also be added to and removed from protection groups through the `purepgroup setattr` command.

Host Group I/O Performance Monitoring

The `purehgroup monitor` command displays real-time I/O performance information for all or specified host groups. The output includes the following data about bandwidth, IOPS, and latency:

- **Name**: Object name.
- **Time**: Current time.
- **B/s**: Bandwidth. Number of bytes read/written.
- **op/s**: IOPS. Number of read, write, or mirrored write requests processed per second.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve a read, write, or mirrored write I/O request. Latency does not include SAN time or QoS rate limit time.

- **B/op**: IOPS. Average I/O size per read, write, mirrored write, and both read and write (all) operations. Must include the `--size` option to see the B/op columns.

Include the `--historical` option to display historical performance data over any of the following ranges of time: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, or 1 year.

Include the `--repeat` option to specify the number of times to repeat the real-time update. If not specified, the repeat value defaults to 1. Include the `--interval` option to specify the number of seconds between each real-time update. At each interval, the system displays a point-in-time snapshot of the performance data. If not specified, the interval value defaults to every 5 seconds. The `--interval` and `--repeat` options can be combined.

In an ActiveCluster replication configuration, include the `--array` option to break down performance data by the array to which the I/O is directed. Include the `--mirrored` option to display performance data for mirrored writes (i.e., writes to volumes in stretched pods). If one array of a stretched pod is offline, the writes processed on the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again.

Host Group Latency Monitoring

The `purehgroup monitor --latency` command displays real-time and historical I/O latency information for volumes connected to host groups. The output includes the following data:

- **SAN us/op**: SAN time. Average time, measured in microseconds, required to transfer data between the initiator and the array. Slow data transfers will result in higher SAN times.
- **Queue us/op**: Queue time. Average time, measured in microseconds, that an I/O request spends in the array waiting to be served. The time is averaged across all I/Os of the selected types.
- **QoS Rate Limit us/op**: QoS rate limit time. Average time, measured in microseconds, that reads, writes, or mirrored writes spend in queue as a result of bandwidth limits reached on one or more volumes.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve a read, write, or mirrored write I/O request. Latency does not include SAN time or QoS rate limit time.

Include the `--mirrored` option to display latency data for mirrored writes (i.e., writes to volumes in stretched pods).

Exceptions

Purity//FA will not create a host group if:

- A host group with the specified name already exists in the array.
- The creation of the host group would exceed the limit of concurrent host groups.

Purity//FA will not delete a host group if:

- Any hosts are associated with the host group or any volumes are connected to it.
- A host cannot be added to a host group if:
- The host is associated with another host group. A host can only be associated with one host group at a time.
- The host has a private connection to a volume associated with the host group.

Examples

Example 1

```
purehost create --wwnlist 0123456789abcde6,0123456789abcde7 HOST6
purehost create --wwnlist 0123456789abcde8,0123456789abcde9 HOST7
purevol create --size 100g VOL1 VOL2 VOL3 VOL4
purehgroup create --hostlist HOST6,HOST7 HGROUP3
purehgroup connect --vol VOL1 HGROUP3
purehgroup connect --vol VOL2 HGROUP3
purehgroup connect --vol VOL3 HGROUP3
purehgroup connect --vol VOL4 --lun 25 HGROUP3
```

Typical usage of the `purehgroup create` command. Creates hosts **`HOST6`** and **`HOST7`**. Creates 100 gigabyte volumes **`VOL1`**, **`VOL2`**, **`VOL3`**, and **`VOL4`**. Creates host group **`HGROUP3`** and associates **`HOST6`** and **`HOST7`** with it. Connects **`VOL1`**, **`VOL2`**, **`VOL3`**, and **`VOL4`** to **`HGROUP3`**. Purity//FA automatically assigns LUNs to volumes **`VOL1`**, **`VOL2`**, and **`VOL3`** by counting down from 254 and assigning the first available numbers to each of the respective volumes. If available, LUN 25 is assigned to **`VOL4`**. Both hosts communicate with the volumes via the same LUNs.

Example 2

```
purehgroup create --hostlist HOST1,HOST2,HOST3 HGROUP1
```

Creates host group **`HGROUP1`** and associates hosts **`HOST1`**, **`HOST2`**, and **`HOST3`** with it. No volumes are connected to the group at the time of creation.

Example 3

```
purehgroup create HGROUP2
```

... *time passes*...

```
purehgroup connect --vol VOL1 --lun 11 HGROUP2
```

... *more time passes*...

```
purehgroup setattr --addhostlist HOST4,HOST5 HGROUP2
```

Creates host group **`HGROUP2`**, but does not associate any hosts with it. At a later time, **`VOL1`** is connected to the group, with LUN 11 assigned to it. Still later, **`HOST4`** and **`HOST5`** are associated with **`HGROUP2`**, causing Purity//FA to establish shared connections between them and **`VOL1`**, using LUN 11 for communication.

Example 4

```
purehgroup monitor --repeat 60 HGROUP1
```

Displays real-time performance data for host group **`HGROUP1`**. Sixty (60) updates are displayed, with each point-in-time update taken at the default interval of every five (5) seconds.

Example 5

```
purehgroup setattr --host "" HGROUP2
purehgroup delete HGROUP2
```

Removes all hosts from **`HGROUP2`** and deletes the host group.

See Also

purehgroup-connect, purehgroup-list

purehost, purepod, purevol

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purehgroup-connect

purehgroup-connect, purehgroup-disconnect, purevol-connect, purevol-disconnect – manage shared connections between host groups and volumes

Synopsis

purehgroup connect --vol ***VOL*** [--lun ***LUN***] [--use-protocol-endpoint ***PROTOCOLENDPOINT***] [--context ***REMOTE***] ***HGROUP...***

purehgroup disconnect --vol ***VOL*** [--context ***REMOTE***] ***HGROUP...***

purevol connect {--host ***HOST*** | --hgroup ***HGROUP***} [--lun ***LUN***] [--use-protocolendpoint ***PROTOCOL-ENDPOINT***] [--context ***REMOTE***] ***VOL...***

purevol disconnect {--host ***HOST*** | --hgroup ***HGROUP***} [--context ***REMOTE***] ***VOL...***

Arguments

HGROUP
: Host group with which a shared connection to the specified volume is to be created or broken.

VOL
: Name of a volume to be connected to or disconnected from a host group.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--context REMOTE`

Used to specify the remote array on which a command is processed. ***`REMOTE`*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--hgroup`

Host group with which shared connections to the specified volumes are to be made or broken by **`purevol connect`** and **`purevol disconnect`** commands. Exactly one host group must be specified by this option. The option is mutually exclusive with the **`--host`** option.

`--host`

Host with which private connections to the specified volumes are to be made or broken by **`purevol connect`** and **`purevol disconnect`** commands. Exactly one host must be specified by this option. The option is mutually exclusive with the **`--hgroup`** option.

`--lun`

Logical unit number (LUN) by which hosts associated with the host group are to address the volume. If not specified, Purity//FA automatically assigns a LUN to the connection. To automatically assign a LUN to a shared connection, Purity//FA starts at `254` and counts down to the minimum `LUN 1`, assigning the first available LUN to the connection. If all LUNs in the `[1...254]` range are taken, Purity//FA starts at `LUN 255` and counts up to the maximum `LUN 4095`, assigning the first available LUN to the connection.

`--use-protocol-endpoint`

Connects the FlashArray virtual volume to VMware ESXi host groups.

For more information about virtual volumes, including configuration steps, refer to the Pure Storage vSphere Web Client Plugin for vSphere User Guide on the Knowledge site at https://support.purestorage.com.

`--vol`

Name of the volume to be connected to or disconnected from the specified hosts host groups. Exactly one volume must be specified.

Description

Makes and breaks shared connections between hosts associated with host groups and volumes.

Private vs. Shared Host-Volume Connections

Purity//FA supports two types of host-volume connections:

`Private`

Connects one volume to one host. Private connections are independent of one another. For example, the sequence:

```
purevol connect --host HOST1 VOL1 VOL2
purehost disconnect --vol VOL1 HOST1
```

connects `HOST1` to `VOL1` and `HOST1` to `VOL2`, and then disconnects `HOST1` and `VOL1`, leaving `HOST1` connected to `VOL2`.

Private connections are typically used for boot volumes and for stand-alone (non-clustered) host applications.

To establish a private connection, run the **`purevol connect --host`** or **`purehost connect --vol`** command. Both commands are functionally identical.

To break a private connection that is no longer required, run the **`purevol disconnect --host`** or **`purehost disconnect --vol`** command. Both commands are functionally identical. When a connection has been broken, its LUN is available for reuse.

For more information about private connections and the **`purehost connect`** command, refer to purehost-connect.

`Shared`

Connects a designated set of hosts (via a host group) to a designated set of volumes, providing the hosts with a consistent "view" of the volumes. All associated hosts use the same LUN to address a given associated volume. All hosts and volumes associated with a host group are automatically connected to each other by virtue of their associations with the group.

To establish a shared connection (assuming the host belongs to a host group), run the **`purevol connect --hgroup`** or **`purehgroup connect --vol`** command. Both commands are functionally identical. For example, the command:

```
purevol connect --hgroup HGROUP1 VOL1 VOL2
```

is equivalent to the sequence:

```
purehgroup connect --vol VOL1 HGROUP1
purehgroup connect --vol VOL2 HGROUP1
```

Both commands establish shared connections between the hosts associated with `HGROUP1` and `VOL1` and `VOL2`.

Shared connections are useful for cluster applications in which several related hosts require consistent (same LUN) connectivity to a set of storage volumes.

To break a shared connection that is no longer required, run the **`purevol disconnect --hgroup`** or **`purehgroup disconnect --vol`** command. Both commands are functionally identical. When a connection has been broken, its LUN is available for reuse.

A host can have only one connection to a given volume at any given time. If you attempt to make a second connection between a host and a volume, private or shared, the attempt will fail.

LUN Management

When a volume is connected to a host group, it is assigned a logical unit number (LUN) ID, which all hosts within the host group use to communicate with the volume. A volume can be connected to multiple host groups as well as to individual hosts simultaneously.

To manually assign a LUN ID to a shared connection, include the **`--lun`** option with the **`purehgroup connect`** or **`purevol connect --hgroup`** command.

Purity//FA automatically assigns a LUN to the shared connection. To do this, Purity//FA starts at LUN `254` and counts down to the minimum LUN `1`, assigning the first available LUN to the connection. If all LUNs in the `[1...254]` range are taken, Purity//FA starts at LUN `255` and counts up to the maximum LUN (`4095`), assigning the first available LUN to the connection.

To change the LUN associated with a shared connection, the connection must first be broken and then recreated by `purehgroup connect`.

Exceptions

Purity//FA will not establish a (shared) connection between a volume and host group if:

- An unavailable LUN was specified.
- The volume is already connected to the host group.
- The volume is already connected to a host associated with the host group.

Examples

Example 1

```
purehgroup connect --vol VOL1 HGROUP1 HGROUP2
```

Establishes shared connections between `VOL1` and the hosts associated with `HGROUP1` and those associated with `HGROUP2`. Purity//FA assigns a LUN to the connections with `HGROUP1`'s hosts and another to those with `HGROUP2`'s hosts; the two LUNs may be the same or different.

Example 2

```
purehgroup connect --vol VOL2 --lun 15 HGROUP3 HGROUP4
```

Establishes shared connections between `VOL2` and the hosts associated with `HGROUP3` and those associated with `HGROUP4`. If LUN 15 is already being used by shared connections to either of the host groups, no connections are made between that group's hosts and `VOL2`.

See Also

purehgroup, purehgroup-list, purehost, purevol

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purehgroup-list

purehgroup-list, purehgroup-listobj – display host groups' attributes and storage consumption

Synopsis

purehgroup list [--cli | --csv | --nvp] [--connect | --protect | --remote | --space] [--filter ***FILTER***] [--limit LIMIT] [--notitle] [--page] [--raw] [--sort ***SORT***] [--context ***REMOTES***] [***HGROUP...***]

purehgroup listobj [--csv] [--remote] [--type {hgroup | host | vol}] [***HGROUP...***]

Arguments

HGROUP

Host group for which the information specified by options is to be displayed.

Options

Options that control information displayed:

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`default (no content option specified in` **`purehgroup list`** `command)`

Displays names and associated hosts for the specified host groups.

`--connect (`**`purehgroup list`**` only)`

Displays volumes associated with the specified host groups, and the LUNs used to address them.

`--context` ***`REMOTES`***

Used to specify the remote array or arrays on which a command is processed. ***`REMOTES`*** is a comma-separated list of names of arrays within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--protect` (**`purehgroup list`** only)

Displays all protected host groups and their associated protection groups.

`--remote`

Includes a list of remote host groups.

`--space` (**`purehgroup list`** only)

For purchased arrays, displays size and space consumption information for the volumes connected to each host group.
Not supported on subscription storage, where effective used capacity information is available in the Purity//FA GUI.

`--type` (**`purehgroup listobj`** only)

Specifies the type of information about the specified host groups that is to be produced in whitespace-separated format suitable for scripting.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Description

The **`purehgroup list`** command displays the information indicated by content options for the specified host groups. If no host groups are specified, the display includes the specified information for all host groups.

If no options are specified, the display lists the hosts associated with each specified host group.

If the **`--connect`** option is specified, the display lists volumes associated with the specified host groups, and the LUNs used to address them.

If the **`--protect`** option is specified, the display lists all protected host groups and their associated protection groups.

If the **`--remote`** option is specified, the display includes a list of remote host groups, meaning a list of host groups that have been created on remote arrays. Host groups on remote arrays have the naming convention **`ARRAY:HGROUP`**, where **`ARRAY`** represents the name of the remote array, and **`HGROUP`** represents the name of the host group on the remote array.

If the **`--space`** option is specified (purchased arrays only), the display lists the following size and space consumption information for the volumes connected to each host group:

`Size`

Total provisioned size of all volumes connected to the host group. Represents storage capacity reported to hosts.

`Thin Provisioning`

Total provisioned size of all volumes connected to the host group. Represents storage capacity reported to hosts.

`Data Reduction`

Ratio of mapped sectors within a volume versus the amount of physical space the data occupies after data compression and deduplication. The data reduction ratio does not include thin provisioning savings.

For example, a data reduction ratio of 5:1 means that for every 5 MB the host writes to the array, 1 MB is stored on the array's flash modules.

`Total Reduction`

Ratio of provisioned sectors within a volume versus the amount of physical space the data occupies after reduction via data compression and deduplication *and* with thin provisioning savings. Total reduction is data reduction with thin provisioning savings.

For example, a total reduction ratio of 10:1 means that for every 10 MB of provisioned space, 1 MB is stored on the array's flash modules.

`Unique`

Physical space that is occupied by data of both volumes and file systems on FlashArray after data reduction and deduplication, but excluding FlashArray metadata and snapshots.

`Snapshots`

Physical space occupied by data unique to one or more snapshots.

`Total`

Total physical space occupied by system, shared space, volume, and snapshot data.

The **`purehgroup listobj`** command creates lists of certain attributes of specified host groups in either whitespace or comma-separated form, suitable for scripting.

If the **`--remote`** option is specified, the display includes a list of remote host groups, meaning a list of host groups that have been created on remote arrays. Host groups on remote arrays have the naming convention **`ARRAY:HGROUP`**, where **`ARRAY`** represents the name of the remote array, and **`HGROUP`** represents the name of the host group on the remote array.

Include the **`--type`** option to display one of the following types of lists:

`--type hgroup (default if` **`--type`** `option not specified)`

Produces a list of the specified host group names. If no host group names are specified, the list contains the names of all host groups.

`--type host`

Produces a list of hosts associated with the specified host groups. If no host groups are specified, the list contains names of all hosts associated with host groups.

`--type vol`

Produces a list of volumes associated with the specified host groups. If no host group argument is specified, the list contains all volumes that are associated with any host group.

Lists are whitespace-separated by default. Specify the **--csv** option to produce a comma-separated list.

Exceptions

None.

Examples

Example 1

```
purehgroup list --connect
```

For all host groups, displays names of associated volumes, and the logical units used by associated hosts to address them.

Example 2

```
purehgroup list
```

Displays the names of all host groups and their member hosts.

Example 3

```
purehgroup list --space --csv HGROUP1 HGROUP2 HGROUP3
```

Displays the virtual and physical space consumption for volumes associated with each of **HGROUP1**, **HGROUP2**, and **HGROUP3**. The output is displayed in comma-separated value format. Not supported on subscription storage.

Example 4

```
purehgroup list --remote
```

Displays a list of all host groups on both the local and remote arrays, and their member hosts.

See Also

purehgroup, purehgroup-connect
purehost, purevol

Author

Pure Storage Inc. **`<documentfeedback@purestorage.com>`**

purehost

purehost, purehost-create, purehost-delete, purehost-rename, purehost-setattr – manage creation, deletion, naming, and attributes of the Purity//FA hosts used to identify computers ("hosts") that use FlashArray storage services

purehost-add, purehost-remove – manage adding and removing of Purity//FA hosts to and from protection groups, respectively

purehost-monitor – monitor host I/O performance

Synopsis

purehost access create ***HOST***

purehost access list [--show-implicit] [--realm ***REALMS***] [--cli | --csv | --nvp] [--notitle] [--raw] [--filter ***FILTER***] [--page] [--limit ***LIMIT***] [--sort ***SORT***] [***HOST ...***]

purehost access delete --realm ***REALMS HOST ...***

purehost add --pgroup ***PGROUP HOST...***

purehost create [--iqnlist ***IQN-LIST***] [--nqnlist ***NQN-LIST***] [--personality ***PERSONALITY***] [--preferred-array ***PREFERRED-ARRAY***] [--wwnlist ***WWN-LIST***] [--vlan ***VLAN***] [--context ***REMOTE***] ***HOST...***

purehost delete [--context ***REMOTE***] ***HOST...***

purehost remove --pgroup ***PGROUP HOST...***

purehost monitor [--array] [--balance] [--csv] [--filter ***FILTER***] [--page][--interval ***INTERVAL***] [--latency] [--limit ***LIMIT***] [--mirrored] [--notitle] [--raw] [--repeat ***REPEAT***] [--size] [--sort ***SORT***] [--total] [***HOST...***]

purehost move --from ***FROM_MEMBER_NAMES*** --to ***TO_MEMBER_NAME*** [--access-create | --access-delete] ***HOST ...***

purehost remove --pgroup ***PGROUP HOST ...***

```
purehost rename [--context REMOTE] OLD-NAME NEW-NAME
```

```
purehost setattr {--addiqnlist IQN-LIST] [--addnqnlist NQN-LIST] [--
addwwnlist WWN-LIST] [--host-password] [--host-user HOST-USER][--
iqnlist IQN-LIST][--nqnlist NQN-LIST][--preferred-array PREFERRED-
ARRAY] [--remiqnlist IQN-LIST] [--remnqnlist NQN-LIST] [--remwwnlist
WWN-LIST] [--target-password] [--target-user TARGET-USER] [--wwnlist
WWN-LIST] [--personality PERSONALITY][--vlan] [--context REMOTE]
HOST...
```

For **`purehost connect`**, see purehost-connect.

For **`purehost disconnect`**, see purehost-connect.

For **`purehost list`**, see purehost-list.

For **`purehost listobj`**, see purehost-list.

Arguments

FROM-MEMBER-NAMES
: Comma-separated list of one or more realms and arrays

HOST
: Host name.

NEW-NAME
: Name by which the host is to be known after the command executes.

OLD-NAME
: Current name of the host to be renamed.

REALMS
: Comma separated list of realm names.

TO-MEMBER-NAME
: Realm or array name.

Object Names

Valid characters are letters (A-Z and a-z), digits (0-9), and the hyphen (-) character. The first and last characters of the name must be alphanumeric, and the name must contain at least one letter or '-'.

Most objects in Purity//FA that can be named, including host groups, hosts, volumes, protection groups, volume and protection group suffixes, SNMP managers, and subnets, can be 1-63 characters in length.

Array names can be 1-56 characters in length. The array name length is limited to 56 characters so that the names of the individual controllers, which are assigned by Purity//FA based on the array name, do not exceed the maximum allowed by DNS.

Names are case-insensitive on input. For example, **`vol1`**, **`Vol1`**, and **`VOL1`** all represent the same volume. Purity//FA displays names in the case in which they were specified when created or renamed.

All objects in a pod have a fully qualified name that include the pod name and object name. The fully qualified name of a volume in a pod is **`POD::VOLUME`**, with double colons (**`::`**) separating the pod name and volume name. The fully qualified name of a protection group in a pod is **`POD::PGROUP`**, with double colons (**`::`**) separating the pod name and protection group name. For example, the fully qualified name of a volume named **`vol01`** in a pod named **`pod01`** is **`pod01::vol01`**, and the fully qualified name of a protection group named **`pgroup01`** in a pod named **`pod01`** is **`pod01::pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target array, the fully qualified name of the protection group on the target array is **`POD:PGROUP`**, with single colons (**`:`**) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to target array **`array02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod01:pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target pod, the fully qualified name of the protection group on the target pod is **`POD:PGROUP`**, with single colons (**`:`**) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to a target pod **`pod02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod02::pod01:pgroup01`**.

All objects in a volume group have a fully qualified name that includes the volume group name and the object name, separated by a forward slash (`/`). For example, the fully qualified name of a volume named **`vol01`** in a volume group named **`vgroup01`** is **`vgroup01/vol01`**.

Realms, pods, and volume groups provide a namespace with unique naming conventions. All objects in a realm have a fully qualified name that include the realm name and object name or realm, pod, and object names. The fully qualified name of a pod in a realm is **`REALM::POD`**, with double colons (`::`) separating the realm name and pod name. The fully qualified name of a volume in a pod in a realm is **`REALM::POD::VOLUME`**, with double colons (`::`) separating the realm, pod, and volume names. The fully qualified name of a protection group in a pod in a realm is **`REALM::POD::PGROUP`**, with double colons (`::`) separating the realm, pod, and protection group names. The fully qualified name of an object in a realm but not in a pod is **`REALM::HOST`**, using host as an example.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--access-create`

Allows creation of resource access between the destination realm and hosts to preserve existing connections.

`--access-delete`

Allows deletion of existing resource access when moving hosts with resource access from the array to a realm.

`--addiqnlist`

Adds the iSCSI Qualified Names (IQNs) in the comma-separated list to those already associated with the specified host.

`--addnqnlist`

Adds the NVMe Qualified Names (NQNs) in the comma-separated list to those already associated with the specified host.

`--addwwnlist`

Adds the Fibre Channel World Wide Names (WWNs) in the comma-separated list to those already associated with the specified host.

`--array`

In an ActiveCluster replication configuration, breaks down performance data by the array to which the I/O is directed.

`--balance`

Displays I/O balance details.

`--context` ***REMOTE***

Used to specify the remote array on which a command is processed. ***REMOTE*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--host-password`

Sets the host password for CHAP authentication. The CLI prompts for the password interactively.

`--host-user`

Sets the host username for CHAP authentication.

`--interval` ***INTERVAL***

Sets the number of seconds between displays of real-time performance data. At each interval, the system displays a point-in-time snapshot of the performance data.

If used with the **`--balance`** option, sets the interval at which I/O balance details are logged. Each I/O balance update consists of the I/O statistics collected during the entire interval.

If omitted, the interval defaults to every 5 seconds.

`--iqnlist`

Comma-separated list of one or more iSCSI Qualified Names (IQNs) to be associated with the specified host. In the **`purehost setattr`** command, this option replaces all IQNs previously associated with the specified host with those in the list.

`--latency`

Displays real-time and historical I/O latency information.

`--nqnlist`

Comma-separated list of one or more NVMe Qualified Names (NQNs) to be associated with the specified host. In the **`purehost setattr`** command, this option replaces all NQNs previously associated with the specified host with those in the list.

`--mirrored`

In an ActiveCluster replication configuration, includes performance data for mirrored writes (that is, writes to volumes in stretched pods). If one array of a stretched pod is offline, the writes processed on the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again.

`--personality`

Determines how the Purity//FA system tunes the protocol used between the array and the initiator. To ensure the array works optimally with the host, set the personality to the name of the host operating or virtual memory system. Valid values are **`aix`**, **`esxi`**, **`hitachi-vsp`**, **`hpux`**, **`oracle-vm-server`**, **`solaris`**, and **`vms`**. If your system is not listed as one of the valid host personalities, do not set the option. By default, the host personality is not set. To clear the host personality setting, set to an empty string (`""`).

`--pgroup` ***`PGROUP`***

Comma-separated list of protection groups to which the specified hosts are added or from which the specified hosts are removed. Has no effect on hosts already associated with the protection group.

`--preferred-array`

Specifies one or more preferred arrays that provide an optimized path, such as a shorter distance or lower latency, to the specified host. If you list more than one array, specify a comma-separated list.

`--realm` ***`REALMS`***

Comma separated list of realm names to which access from hosts are modified using **`purehost access`**.

`--remiqnlist`

Disassociates the iSCSI Qualified Names (IQNs) in the comma-separated list from the specified host.

`--remnqnlist`

Disassociates the NVMe Qualified Names (NQNs) in the comma-separated list from the specified host.

`--remwwnlist`

Disassociates the Fibre Channel World Wide Names (WWNs) in the comma-separated list from the specified host.

`--repeat` ***`REPEAT`***

Sets the number of times to display real-time performance data. If omitted, the repeat count defaults to 1.

`--show implicit`

Includes implicitly shared hosts when using **`purehost access list`**.

`--size`

Displays the average I/O sizes per read and write operation.

`--target-password`

Sets the target password for CHAP authentication. The CLI prompts for the password interactively.

`--target-user`

Sets the target username for CHAP authentication.

`--total`

Follows output lines with a single line containing column totals in columns where they are meaningful.

`--vlan` ***`VLAN`***

Specifies the VLAN ID that the host is associated with. In the **`purehost list`** command, this option lists only the hosts that are configured with the specified VLAN ID.

`--wwnlist`

Comma-separated list of one or more Fibre Channel World Wide Names (WWNs) to be associated with the specified host. In the **`purehost setattr`** command, this option replaces all WWNs previously associated with the specified host with those in the list.

Options that control display format:

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Description

The host is the abstraction used by Purity//FA to organize the storage network addresses (iSCSI Qualified Names, NVMe qualified names, or Fibre Channel world wide names) of client computers and to control communications between clients and volumes. The host's only attributes are lists of one or more iSCSI Qualified Names (IQNs), NVMe Qualified Names (NQNs), or Fibre Channel World Wide Names (WWNs) that identify the host's initiators.

Creating Hosts

The `purehost create` command creates a host and optionally associates one or more IQNs, NQNs, WWNs, VLAN, or preferred arrays with it.

To add IQNs, NQNs, WWNs, VLAN, or preferred arrays to a host during host creation, include the respective **`--iqnlist`**, **`--nqnlist`**, **`--wwnlist`**, **`--vlan`**, or **`--preferred-array`** option with the `purehost create` command.

Associating IQNs, NQNs, or WWNs with Hosts

The host cannot communicate with the array until at least one IQN, NQN, WWN, or VLAN has been associated with it.

IQNs, NQNs, or WWNs can be added to or removed from a host at any time.

To add IQNs, NQNs, or WWNs to an existing host, include the respective **`--addiqnlist`**, **`--addnqnlist`**, or **`--addwwnlist`** option with the `purehost setattr` command. IQNs, NQNs, and WWNs that are already associated with the host are not affected.

To add IQNs, NQNs, or WWNs to an existing host, completely replacing the ones that are currently associated with the host, include the respective **`--iqnlist`**, **`--nqnlist`**, or **`--wwnlist`** option with the `purehost setattr` command.

To remove IQNs, NQNs, WWNs, or VLANs from an existing host, include the respective **`--remiqnlist`**, **`--remnqnlist`**, **`--remwwnlist`**, or **`--vlan`** option with the `purehost setattr` command.

Once the IQNs, NQNs, or WWNs have been specified, enable communication between the host and volumes by establishing connections, either private or shared, between the two.

Host to VLAN Binding

To add a VLAN to a host during host creation, include the **`--vlan`** option with the **`purehost create`** command. To add a VLAN to an existing host include the **`--vlan`** option with the **`purehost setattr`** command. Hosts with active connections cannot have their VLAN ID changed.

Run the **`purehost list --vlan`** command to display the VLAN IDs associated with the host.

Creating Host-Volume Connections

After a host has been created and the IQNs, NQNs, or WWNs have been specified, establish a connection, either private or shared, between the host and volumes. Host-volume connections can be established or broken at any time.

To establish a private connection, run the **`purehost connect`** or **`purevol connect --host`** command. To establish a shared connection, run the **`purehgroup connect`** or **`purevol connect --hgroup`** command.

For more information about private connections, refer to purehost-connect. For more information about shared connections, refer to purehgroup-connect.

Deleting Hosts

The **`purehost delete`** command removes hosts that are no longer required. A host cannot be deleted while it has connections to volumes, either private or shared.

Renaming Hosts

Run the **`purehost rename`** command to change the current name (**`OLD-NAME`**) of a host to the new name (**`NEW-NAME`**). The name change is effective immediately and the old name is no longer recognized in CLI, GUI, or REST interactions. In the Purity//FA GUI, the new name appears upon page refresh.

Adding Hosts to Protection Groups

The `purehost add` command adds existing hosts to existing protection groups. Hosts can only be added to protection groups that replicate to target arrays through asynchronous replication. Hosts and host groups are not supported for replication to offload targets.

Multiple hosts can be added to multiple protection groups. Enter multiple protection groups in comma-separated format.

If a protection group already includes other hosts, those hosts are unaffected.

Only members of the same object type can belong to a protection group. For example, a host cannot be added to a protection group that already contains host groups or volumes.

The `purehost add` command only accepts hosts that do not already belong to any of the specified protection groups. If any of the hosts already belong to any of the protection groups, the entire command fails.

Run the `purehost list --protect` command to see a list of all protected hosts and their associated protection groups.

The `purehost remove` command removes one or more hosts from one or more protection groups. All of the specified hosts must belong to all of the specified protection groups in the command; otherwise, the command fails.

Hosts can also be added to and removed from protection groups through the `purepgroup setattr` command.

Preferred Arrays

For ActiveCluster replication configurations, Purity//FA supports both non-uniform and uniform storage access. In a non-uniform storage access configuration, a connected host has access to the array that is local to the host. In a uniform storage access configuration a connected host has access to both arrays within a pod.

In a uniform storage access configuration, when accessing one array over another is more efficient due to access characteristics, such as a shorter distance and lower latency, you might consider configuring a preferred array for each host (also known as Asymmetric Logical Unit Access or ALUA). To configure a preferred array (or a list of arrays) for an active and optimized path to a host, use the **`purehost create -preferred-array`** command when you create the host. To modify a preferred-array configuration, use the **`purehost setattr --preferred-array`** command.

If a preferred array is set for a host, then the other arrays in the same pod will expose active/non-optimized paths to that host. A host can have more than one preferred array.

Host I/O Performance and I/O Balance Monitoring

The `purehost monitor` command displays real-time I/O performance and I/O balance information for all or specified hosts. The output includes I/O performance information, including the following data about bandwidth, IOPS, and latency:

- **Name**: Object name.
- **Time**: Current time.
- **B/s**: Bandwidth. Number of bytes read/written.
- **op/s**: IOPS. Number of read, write, or mirrored write requests processed per second.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve a read, write, or mirrored write I/O request. Latency does not include SAN time or QoS rate limit time.
- **B/op**: IOPS. Average I/O size per read, write, mirrored write, and both read and write (all) operations. Must include the `--size` option to see the B/op columns.

Include the `--interval` option to specify the number of seconds between each point-in-time update. Include the `--repeat` option to specify the number of times to repeat the update. The `--interval` and `--repeat` options can be combined.

Include the `--balance` option to displays the I/O counts on each path between the host and the controller. Ideally, I/O counts on all the host paths should be as close to each other as possible. The `purehost monitor --balance` output includes the following I/O balance data:

- **Name**: Host name.
- **Time**: Current time.
- **Initiator WWN**: Fibre Channel initiator port name.
- **Initiator IQN**: iSCSI initiator port name.
- **Initiator NQN**: NVMe initiator port name.
- **Target**:
  - Target Fibre Channel component name for the Fibre Channel initiator (WWN).
  - Target network interface names (both primary and secondary controllers) for the iSCSI initiator (IQN).
  - NVMe storage target device name for the NVMe initiator (NQN).
- **Target**: For WWN, target Fibre Channel component name. For IQN, target primary or secondary controller. For NQN, NVMe storage target device.

- **Target WWN**: Target WWN port name. Only applies to WWN.
- **Failover**: Port to which this array port has failed over. The port name only appears if the array port has failed over.
- **I/O Count**: I/O count for the host path.
- **I/O Relative to Max**: Percentage of I/O counts for this path relative to the path with highest number of I/O counts. The path with the highest number of I/O counts is displayed with an I/O Relative to Max percentage of 100%. The percentage values of all other paths in the host are then calculated relative to the path with the highest number of I/O counts.

The `--interval` and `--repeat` options can be used with the `--balance` option. Include the `--interval` option to set the interval at which I/O balance details are logged. Each I/O balance update consists of the I/O statistics collected during the entire interval. Include the `--repeat` option to specify the number of times to repeat the update.

In an ActiveCluster replication configuration, include the `--array` option to break down performance data by the array to which the I/O is directed. Include the `--mirrored` option to display performance data for mirrored writes (i.e., writes to volumes in stretched pods). If one array of a stretched pod is offline, the writes processed on the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again.

The `purehost monitor --latency` command displays real-time and historical I/O latency information for volumes connected to hosts. The `purehost monitor --latency` output includes the following data:

- **SAN us/op**: SAN time. Average time, measured in microseconds, required to transfer data between the initiator and the array. Slow data transfers will result in higher SAN times.
- **Queue us/op**: Queue time. Average time, measured in microseconds, that an I/O request spends in the array waiting to be served. The time is averaged across all I/Os of the selected types.
- **QoS Rate Limit us/op**: QoS rate limit time. Average time, measured in microseconds, that reads, writes, or mirrored writes spend in queue as a result of bandwidth limits reached on one or more volumes.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve a read, write, or mirrored write I/O request. Latency does not include SAN time or QoS rate limit time.

Include the `--mirrored` option to display latency data for mirrored writes (i.e., writes to volumes in stretched pods).

Exceptions

Purity//FA will not create a host if:

- The specified name is already associated with another host in the array.
- Any of the specified IQNs, NQNs, or WWNs is already associated with an existing host in the array.
- The creation of the host would exceed the limit of concurrent hosts, or the creation of the IQN, NQN or WWN would exceed the limit of concurrent initiators.

Purity//FA will not delete a host while it has connections to volumes, either private or shared.

Purity//FA will not associate an IQN, NQN or WWN with a host if:

- The creation of the IQN, NQN or WWN would exceed the maximum number of concurrent initiators.
- The specified IQN, NQN or WWN is already associated with another host on the array.

Hosts are configured through the Purity//FA GUI (**Storage** tab) and Purity//FA CLI (`purehost` command).

The following command usage is not supported on Cloud Block Store:

- The `purehost create --wwnlist` command
- The `purehost setattr --addwwnlist` command
- The `purehost setattr --remwwnlist` command
- The `purehost setattr --wwnlist` command

Examples

Example 1

```
purehost create HOST1
```

Creates a host called **`HOST1`**. **`HOST1`** cannot be connected to volumes or associated with a host group until at least one WWN has been associated with it.

Example 2

```
purehost create --wwnlist 0123456789abcde1,0123456789abcde2 HOSTFC
purehost create --iqnlist iqn.2001-04.com.example:diskarrays-sn-a8675309 HOSTISCSI
purehost create --nqnlist nqn.2001-04.com.example:diskarrays-sn-a8675309 HOSTNVME
```

Creates a host called **`HOSTFC`** and associates WWNs 01:23:45:67:89:ab:cd:e1 and 01:23:45:67:89:ab:cd:e2 with it.

Creates a second host called **`HOSTISCSI`** and associates IQN iqn.2001-04.com.example:diskarrays-sn-a8675309 with it.

Creates a third host called **`HOSTNVME`** and associates NQN nqn.2001-04.com.example:diskarrays-sn-a8675309 with it.

Example 3

```
purehost create --vlan 47 --iqnlist iqn.2001-04.com.example:diskarrays-sn-1 HOST1
```

Creates a host called **`HOST1`** and associates IQN iqn.2001-04.com.example:diskarrays-sn-1 with it, while adding a VLAN to **`HOST1`**.

Example 4

```
purehost delete HOST5
```

Deletes host **`HOST5.`** **`HOST5`** private connections and host group association (if any) must previously have been broken.

Example 5

```
purehost add --pgroup PGROUP1,PGROUP2 HOST1 HOST2
```

Adds hosts **`HOST1`** and **`HOST2`** to protection groups **`PGROUP1`** and **`PGROUP2`**, respectively.

Example 6

```
purehost monitor --repeat 60 --interval 10 HOST1
```

Displays real-time performance data for host **`HOST1`**. Sixty (60) point-in-time updates are displayed, with each update taken every ten (10) seconds.

Example 7

```
purehost monitor --balance
```

Displays I/O balance details for all hosts.

Example 8

```
purehost monitor --balance --repeat 2 --interval 2 HOST3
```

Displays I/O balance details for host **HOST3** containing 2 seconds' worth of data with 2 seconds between each update.

Example 9

```
purehost setattr --personality esxi HOST3
```

Sets the host personality of host **HOST3** to the **esxi** virtual memory system to ensure the array works optimally with the host.

Example 10

```
purehost setattr --wwnlist 0123456789abcdef,01:23:45:67:89:ab:cd:ee HOST3
```

Replaces all WWN previously associated with **HOST3** with the two specified. This example also illustrates the two formats for entering WWNs.

Example 11

```
purehost setattr --remwwnlist 01:23:45:67:89:ab:cd:ed HOST4
purehost setattr --addwwnlist 0123456789abcdec,0123456789abcdeb HOST4
```

Disassociates WWN 01:23:45:67:89:ab:cd:ed from **HOST4** and replaces it with 01:23:45:67:89:-ab:cd:ec and 01:23:45:67:89:ab:cd:eb. Other WWNs associated with HOST4 are unaffected.

Example 12

```
purehost setattr --preferred-array ARRAY1 HOST1
```

Sets the preferred array for **HOST1** to **ARRAY1**. Once the preferred array is set, **ARRAY1** exposes active/optimized paths to **HOST1**.

Example 13

```
purehost setattr --host-user example_user --host-password h1 h2
```

Sets the host password and the host username for CHAP authentication. This will configure CHAP on multiple hosts at one time. The CLI prompts for the password interactively.

Example 14

```
purehost setattr host1 --vlan any
```

Adds VLAN to an existing host.

See Also

purehost-connect, purehost-list
purearray, purehgroup, purepod, purevol

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purehost-connect

purehost-connect, purehost-disconnect, purevol-connect, purevol-disconnect – manage private connections between hosts and volumes

Synopsis

purehost connect --vol ***VOL*** [--lun ***LUN***] [--use-protocol-endpoint ***PROTOCOLENDPOINT***] [--context ***REMOTE***] ***HOST...***

purehost disconnect --vol ***VOL*** [--context ***REMOTE***] ***HOST...***

purevol connect {--host ***HOST*** | --hgroup ***HGROUP***} [--lun ***LUN***] [--use-protocolendpoint ***PROTOCOL-ENDPOINT***] [--context ***REMOTE***] ***VOL...***

purevol disconnect {--host ***HOST*** | --hgroup ***HGROUP***} [--context ***REMOTE***] ***VOL...***

Arguments

HOST

Name of a host to be connected to or disconnected from a volume.

VOL

Name of a volume to be connected to or disconnected from a host.

Options

-h | --help

Can be used with any command or subcommand to display a brief syntax description.

`--context` ***`REMOTE`***

Used to specify the remote array on which a command is processed. ***`REMOTE`*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--hgroup`

Host group with which shared connections to the specified volumes are to be made or broken by **`purevol connect`** and **`purevol disconnect`** commands. Exactly one host group must be specified by this option. The option is mutually exclusive with the **`--host`** option.

`--host`

Host with which private connections to the specified volumes are to be made or broken by **`purevol connect`** and **`purevol disconnect`** commands. Exactly one host must be specified by this option. The option is mutually exclusive with the **`--hgroup`** option.

`--lun`

Logical unit number (LUN) by which hosts associated with the host group are to address the volume. If not specified, Purity//FA automatically assigns a LUN to the connection. To automatically assign a LUN to a shared connection, Purity//FA starts at `254` and counts down to the minimum `LUN 1`, assigning the first available LUN to the connection. If all LUNs in the `[1...254]` range are taken, Purity//FA starts at `LUN 255` and counts up to the maximum `LUN 4095`, assigning the first available LUN to the connection.

`--use-protocol-endpoint`

Connects the FlashArray virtual volume to VMware ESXi host groups.

For more information about virtual volumes, including configuration steps, refer to the Pure Storage vSphere Web Client Plugin for vSphere User Guide on the Knowledge site at `https://support.purestorage.com`.

`--vol`

Name of the volume to be connected to or disconnected from the specified hosts host groups. Exactly one volume must be specified.

Description

Makes and breaks private connections between hosts and volumes and shared connections between hosts associated with host groups and volumes.

For private connections:

- The `purehost connect` and `purehost disconnect` commands are used to manage private connections between a single volume and one or more hosts.
- The `purevol connect` and `purevol disconnect` commands are used with the `--host` option to manage private connections between a single host and one or more volumes.

For shared connections:

- The `purevol connect` and `purevol disconnect` commands are used with the `--hgroup` option to manage shared connections between a single host group and one or more volumes. See purehgroup-connect for additional commands used to manage shared connections.

Private vs. Shared Host-Volume Connections

Purity//FA supports two types of host-volume connections:

`Private`

Connects one volume to one host. Private connections are independent of one another. For example, the sequence:

```
purevol connect --host HOST1 VOL1 VOL2
purehost disconnect --vol VOL1 HOST1
```

connects `HOST1` to `VOL1` and `HOST1` to `VOL2`, and then disconnects `HOST1` and `VOL1`, leaving `HOST1` connected to `VOL2`.

Private connections are typically used for boot volumes and for stand-alone (non-clustered) host applications.

To establish a private connection, run the `purevol connect --host` or `purehost connect --vol` command. Both commands are functionally identical.

To break a private connection that is no longer required, run the `purevol disconnect --host` or `purehost disconnect --vol` command. Both commands are functionally identical. When a connection has been broken, its LUN is available for reuse.

`Shared`

Connects a designated set of hosts (via a host group) to a designated set of volumes, providing the hosts with a consistent "view" of the volumes. All associated hosts use the same LUN to address a given associated volume. All hosts and volumes associated with a host group are automatically connected to each other by virtue of their associations with the group.

To establish a shared connection (assuming the host belongs to a host group), run the **`purevol connect --hgroup`** or **`purehgroup connect --vol`** command. Both commands are functionally identical. For example, the command:

```
purevol connect --hgroup HGROUP1 VOL1 VOL2
```

is equivalent to the sequence:

```
purehgroup connect --vol VOL1 HGROUP1
purehgroup connect --vol VOL2 HGROUP1
```

Both commands establish shared connections between the hosts associated with `HGROUP1` and `VOL1` and `VOL2`.

Shared connections are useful for cluster applications in which several related hosts require consistent (same LUN) connectivity to a set of storage volumes.

To break a shared connection that is no longer required, run the **`purevol disconnect --hgroup or purehgroup disconnect --vol`** command. Both commands are functionally identical. When a connection has been broken, its LUN is available for reuse.

For more information about private connections and the **`purehgroup connect`** command, refer to purehgroup-connect.

App Hosts

The Purity Run platform extends array functionality by integrating add-on services into the Purity//FA operating system. Each service that runs on the platform is provided by an app.

Each app has a dedicated host, known as an app host. The app host is used to connect FlashArray volumes to the app.

Run the **`purehost connect --vol`** or **`purevol connect --host`** command to connect FlashArray volumes to an app host, and thereby its associated app. Likewise, run the **`purehost disconnect --vol`** or **`purevol disconnect --host`** command to break the connection between the FlashArray volumes and the app.

For more information about Pure Apps and the **`pureapp`** command, refer to pureapp.

LUN Management

When a volume is connected to a host, it is assigned a logical unit number (LUN) ID, which the host uses to communicate with the volume. A volume can be connected to individual hosts as well as to multiple host groups simultaneously. Include the **`--lun`** option with the **`purehost connect`** or **`purevol connect --host`** command to manually assign a LUN ID anywhere in the `[1...4095]` range.

Purity//FA automatically assigns a LUN to the private connection. To do this, Purity//FA starts at LUN `1` and counts up to the maximum LUN `4095`, assigning the first available LUN to the connection.

If multiple hosts are specified in a single **`purehost connect`** command, there is no guarantee that the same LUN will be associated with each connection established. When **`--lun`** and **`--host`** are both specified in a **`purehost connect`** or **`purevol connect`** command, exactly one host and one volume must be specified.

To change the LUN associated with a private connection, the connection must first be broken and then recreated by **`purehost connect`** or **`purevol connect`**.

Exceptions

Purity//FA will not establish a (private) connection between a volume and a host if:

- An unavailable LUN was specified.
- The volume is already connected to the host, either through a private or shared connection.

Examples

Example 1

```
purevol connect --host HOST1 VOL1 VOL2
```

Establishes private connections between `HOST1` and `VOL1` and between `HOST1` and `VOL2`.

Purity//FA assigns a LUN to each connection. If `HOST1` is associated with a host group, Purity//FA assigns a LUN according to the LUN assignment guidelines for shared connections. If `HOST1` is not associated with a host group, Purity//FA assigns a LUN according to the LUN assignment guidelines for private connections.

Example 2

```
purehost connect --vol VOL3 --lun 4 HOST2
```

```
purevol connect --host HOST2 --lun 5 VOL4
```

Establishes private connections between `HOST2` and `VOL3` and between `HOST2` and `VOL4`. If LUN 4 or LUN 5 is already in use by another connection to `HOST2`, the corresponding connection fails.

Example 3

```
purevol connect --host @linux app_vol001
```

Connects FlashArray `volume app_vol001` to the `linux` app via app host `@linux`.

Example 4

```
purehost disconnect --vol app_vol002 @linux
```

Breaks the connection between FlashArray volume `app_vol002` and the `linux` app.

See Also

purehost, purehost-list, pureapp, purehgroup, purevol

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purehost-list

purehost-list, purehost-listobj – display information about Purity//FA host objects, host-volume connections, and storage provisioning and consumption

Synopsis

purehost list [--all | --chap | --connect | --preferred-array | --personality | --protect | --space | --vlan] [--cli | --csv | --nvp] [--filter ***FILTER***] [--limit ***LIMIT***] [--notitle] [--page] [--private | --shared] [--raw] [--sort ***SORT***] [--remote] [--context ***REMOTES***] [***HOST***...]

purehost listobj [--csv] [--remote] [--type {host | iqn | nqn | vol | wwn}] [***HOST***...]

Arguments

HOST
: Host object for which the information specified by options is to be displayed.

Options

Options that control information displayed:

`-h | --help`
: Can be used with any command or subcommand to display a brief syntax description.

`default` (no content option specified with **`purehost list`**)
: Displays associated world wide names and host groups for the specified hosts.

`--all` (**`purehost list`** only)

Displays all visible attributes of the specified hosts.

`--chap` (**`purehost list`** only)

Displays CHAP authentication settings.

`--connect` (**`purehost list`** only)

Displays volumes connected to the specified hosts and the LUNs used to address them.

`--context` ***`REMOTES`***

Used to specify the remote array or arrays on which a command is processed. ***`REMOTES`*** is a comma-separated list of names of arrays within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--personality` (**`purehost list`** only)

Displays host personality settings. The host personality determines how the Purity//FA system tunes the protocol used between the array and the initiator. To ensure the array works optimally with the host, the host personality should be the name of the host operating or virtual memory system. The host personality is set through the **`purehost setattr --personality`** command.

`--preferred-array` (**`purehost list`** only)

Displays the preferred-array setting for each host. A dash (–) in the `Preferred Array` column indicates that a preferred array has not been set.

`--private` (**`purehost list connect`** only)

Restricts the list or display of volumes connected to specified hosts to those with private connections. Invalid when combined with other options.

`--protect` (**`purehost list`** only)

Displays all protected hosts and their associated protection groups.

`--remote`

Includes a list of remote hosts.

`--shared` (**`purehost list connect`** only)

Restricts the display of volumes connected to specified hosts to those with shared connections. Invalid when combined with other options.

`--space` (**`purehost list`** only)

On purchased arrays, displays size and space consumption information for the volumes connected to each host.
Not supported on subscription storage, where effective used capacity information is available in the GUI Analysis > Capacity > Array and Analysis > Capacity > Volumes tabs.

`--type` (**`purehost listobj`** only)

Specifies the type of information about specified hosts that is to be produced in whitespace-separated format suitable for scripting.

`--vlan` (**`purehost list`** only)

Displays the VLAN ID associated with each host.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Description

The **`purehost list`** command displays a list of hosts on the array. The list includes host attributes such as host group association, interface, connected volumes (both shared and private), provisioned size, and storage consumption.

The information to be displayed is specified by including one of the following options:

- If no options are specified, displays names, associated world wide names, and host groups for specified hosts.
- If the **`--all`** option is specified, displays all visible attributes of the specified hosts. Display includes associated IQNs, NQNs or WWNs, host groups, and the connected volumes and the LUNs used to address them.
- If the **`--chap`** option is specified, displays host and target usernames. Also indicates whether host and target passwords have been set.
- If the **`--connect`** option is specified, displays volumes associated with the specified hosts, and the LUNs used to address them.
- If the **`--personality`** option is specified, displays the personality setting associated with the specified hosts.
- If the **`--preferred-array`** option is specified, displays the preferred-array setting associated with each host.
- If the **`--protect`** option is specified, displays all protected hosts and their associated protection groups.
- If the **`--remote`** option is specified, the display includes a list of remote hosts, meaning a list of hosts that have been created on remote arrays. Hosts on remote arrays have the naming convention `ARRAY:HOST`, where `ARRAY` represents the name of the remote array, and `HOST` represents the name of the host on the remote array.
- If the **`--vlan`** option is specified, the display lists the VLAN ID associated with each host.
- If the **`--space`** option is specified (purchased arrays only), the display lists the following size and space consumption information for the volumes connected to each host:

  `Size`

  Total provisioned size of all volumes connected to the host. Represents storage capacity reported to hosts.

`Thin Provisioning`

Percentage of volume sectors that do not contain host-written data because the hosts have not written data to them or the sectors have been explicitly trimmed.

`Data Reduction`

Ratio of mapped sectors within a volume versus the amount of physical space the data occupies after data compression and deduplication. The data reduction ratio does not include thin provisioning savings.

For example, a data reduction ratio of 5:1 means that for every 5 MB the host writes to the array, 1 MB is stored on the array's flash modules.

`Total Reduction`

Ratio of provisioned sectors within a volume versus the amount of physical space the data occupies after reduction via data compression and deduplication and with thin provisioning savings. Total reduction is data reduction with thin provisioning savings.

For example, a total reduction ratio of 10:1 means that for every 10 MB of provisioned space, 1 MB is stored on the array's flash modules.

`Volumes`

Physical space occupied by volume data that is not shared between volumes, excluding array metadata and snapshots.

`Snapshots`

Physical space occupied by data unique to one or more snapshots.

`Total`

Total physical space occupied by system, shared space, volume, and snapshot data.

By default, space consumption for all connected volumes, both private and shared, is displayed. The display can be restricted to volumes with private or shared connections by specifying the **`--private`** or the **`--shared`** option.

The **`purehost listobj`** command produces lists of certain attributes of specified hosts in either whitespace or comma-separated form, suitable for scripting.

If the **`--remote`** option is specified, the display includes a list of remote hosts, meaning a list of hosts that have been created on remote arrays. Hosts on remote arrays have the naming convention `ARRAY:HOST`, where `ARRAY` represents the name of the remote array, and `HOST` represents the name of the host on the remote array.

Include the **`--type`** option to display one of the following types of lists:

`--type host` (default if **`--type`** option not specified)

List contains the specified host names. If no host names are specified, the list contains the names of all host objects.

`--type iqn`

List contains the IQNs associated with each specified host. If no hosts are specified, the list contains all IQNs (administratively assigned and discovered) known to the array.

`--type nqn`

List contains the NQNs associated with each specified host. If no hosts are specified, the list contains all NQNs (administratively assigned and discovered) known to the array.

`--type vol`

List contains the volumes connected to the specified hosts. If no hosts are specified, list contains names of all volumes connected to any host. List can be restricted to show only private connections by specifying the **`--private`** option.

`--type wwn`

List contains the world wide names associated with each specified host. If no hosts are specified, the list contains all world wide names (administratively assigned and discovered) known to the array.

Lists are whitespace-separated by default. Specify the **`--csv`** option to produce a comma-separated list.

App Hosts

App hosts are used to connect FlashArray volumes to Pure apps. If an app is installed on the array, its app host will appear in the **`purehost list`** output.

App host names begin with a distinctive `@` symbol. The naming convention for app hosts is `@APP`, where `APP` denotes the app name.

The following example displays the app host for the `linux` app.

```
$ purehost list @linux*
Name   WWN IQN NQN Host Group
@linux -   -   -   -
```

Include the **`--connect`** option to display all of the volumes associated with the app host, and thereby connected to the app. In the following example, five FlashArray volumes, along with the `linux` boot and data volumes, are associated with the `@linux` app host, and thereby connected to the `@linux` app.

```
$ purehost list --connect @linux*
```

```
Name   LUN   Vol          Host Group
@linux 1     @linux_boot  -
@linux 2     @linux_data  -
@linux 3     app_vol001   -
@linux 4     app_vol002   -
@linux 5     app_vol003   -
@linux 6     app_vol004   -
@linux 7     app_vol005   -
```

For more information about Pure apps and the `pureapp` command, refer to pureapp.

Exceptions

None.

Examples

Example 1

```
purehost list
```

Displays names, associated world wide names, and associated host groups (if any) for all hosts.

Example 2

```
purehost list --connect
```

Displays names, connected volumes and logical units for all hosts. Both private and shared volume connections are displayed.

Example 3

```
purehost list --space --private --csv --notitle HOST1 HOST2 HOST3
```

On purchased arrays, displays the above mentioned virtual and physical space consumption for volumes associated with each of `HOST1`, `HOST2`, and `HOST3`. Not supported on subscription storage.

Example 4

```
purehost list --remote
```

Displays a list of all hosts on both the local and remote arrays, their associated IQNs, NQNs, or WWNs and their associated host groups (if any).

Example 5

```
purehost listobj --type vol --private HOST1 HOST2
```

Lists all volumes with private connections to `HOST1` and `HOST2`.

Example 6

```
purehost list --connect @linux
```

Displays the names of all volumes associated with app host `linux`, and thereby connected to the `linux` app.

See Also

purehost, purehost-connect, pureapp, purehgroup, purevol

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purehw

purehw, purehw-list, purehw-setattr – displays information about and controls visual identification of FlashArray hardware components

Synopsis

```
purehw list [--csv | --nvp] [--notitle] [--page] [--raw] [--sfp] [--
spec] [--static] [--type COMPONENT-TYPE] [COMPONENT...]
purehw setattr [--identify {off | on} | --index INDEX] COMPONENT...
```

Arguments

COMPONENT

Hardware component whose information is to be displayed or whose attribute is to be set to the specified value.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--identify`

Turns a visual identifier for the component on or off. Valid for controllers, NVRAM bays, storage bays, and storage shelves. Used with the **`purehw setattr`** command to set front panel LED identifiers on or off.

`--index`

In multi-shelf arrays, sets the top-level shelf component to a unique number. Once set, the number becomes part of the name of each component in the shelf.

`--sfp`

Displays SFP and QSFP diagnostic information such as temperature, voltage, power, and fault statistics for both Fibre Channel and Ethernet ports, when available.

`--spec`

Displays hardware component model names, part numbers, and serial numbers.

`--static`

Displays SFP static information such as connector type, rate, wavelength, thresholds, and vendor name and numbers. Requires the `--sfp` option.

`--type`

Type of component for which information is to be displayed. When this option is specified, information is displayed for all components of the specified type. Valid values for **`--type`** are: `bay` (flash module bay), `ch` (chassis), `ct` (controller), `dca` (direct compress accelerator), `eth` (Ethernet port), `fan` (fan), `fc` (Fibre Channel port), `ib` (InfiniBand port), `iom` (I/O module), `nvb` (NVRAM module bay), `pwr` (power supply), `sas` (SAS port), `sh` (storage shelf), and `tmp` (temperature sensor).

Supported **`--type`** values vary by FlashArray model.

Options that control display format:

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Product Names and Conventions

FlashArray refers to any product in the FlashArray series, including FlashArray//X, FlashArray//XR2, FlashArray//XR3, FlashArray//M, FlashArray//C, etc.

Description

Most FlashArray hardware components report their operational status. The **`purehw list`** command displays information about the FlashArray hardware components.

The **`purehw list --sfp`** command displays dynamic temperature voltage, power, and fault information for Fibre Channel and Ethernet ports, when available. The following example displays SFP information for ports CT0.FC2 and CT1.FC2:

```
purehw list --sfp CT0.FC2 CT1.FC2
Name     Temperature  Voltage     Tx Bias      Tx Power     Rx Power     Tx Fault Rx Los
CT0.FC2  ok (47.32 C) ok (3.31 V) ok (7.29 mA) ok (0.60 mW) ok (0.50 mW) False    False
CT1.FC2  ok (38.65 C) ok (3.33 V) ok (5.33 mA) ok (0.69 mW) ok (0.50 mW) False    False
```

The **`purehw list --sfp --static`** command displays static port information such as connector type, rate, wavelength, vendor numbers, and thresholds for Fibre Channel and Ethernet ports, when available.

The **`purehw setattr`** command controls the visual identification of certain hardware components, and sets numerical values of top-level shelf components.

FlashArray Hardware Components

Hardware components and their naming vary by FlashArray series. To see the hardware technical specifications for each FlashArray model, refer to the FlashArray page at https://support.purestorage.com .

Hardware Components in a FlashArray

The FlashArray chassis, controller and storage shelf names have the form `XXm`. The names of other components have the form `XXm.YYn` or `XXm.YYYn`, where:

`XX`

Denotes the type of component:

- `CH` - //X or //M chassis.
- `CT` - Controller.
- `SH` - Storage shelf.

`m`

Identifies the specific controller or storage shelf:

- For an //X or //M chassis, `m` has a value of 0. For example, `CH0`.
- For controllers, `m` has a value of 0 or 1. For example, `CT0`, `CT1`.
- For storage shelves, `m` represents the shelf number, starting at 0. For example, `SH0`, `SH1`.
- The assigned number can be changed on the shelf front panel or by running the `purehw setattr --id` command.

`YY or YYY`

Denotes the type of component. For example, `FAN` for cooling device, `FC` for Fibre Channel port.

`n`

Identifies the specific component by its index (its relative position within the //X or //M chassis, controller, or storage shelf), starting at 0.

The following table lists hardware components that report status, grouped by their location on the array. The Identify Light column shows which components have an LED light on the physical component that can be turned on and off.

Table 1. Chassis (CH0)

Component Name	Identify Light	Component Type
CH0	Yes	Chassis
CH0.BAYn	Yes	Storage bay
CH0.NVBn	Yes	NVRAM bay
CH0.PWRn	--	Power module

Table 2. Controller (CTm)

Component Name	Identify Light	Component Type
CTm	Yes	Controller
CTm.ETHn	--	Ethernet port
CTm.FANn	--	Cooling fan
CTm.FCn	--	Fibre Channel port
CTm.IBn	--	InfiniBand port (included only with certain upgrade kits)
CTm.SASn	--	SAS port
CTm.TMPn	--	Temperature sensor

Table 3. Storage Shelf (SHm)

Component Name	Identify Light	Component Type
SHm	Yes	Storage shelf
SHm.BAYn	Yes	Storage bay
SHm.FANn	--	Cooling fan
SHm.IOMn	--	I/O module
SHm.PWRn	--	Power module
SHm.SASn	--	SAS port
SHm.TMPn	--	Temperature sensor

FlashArray Chassis Components (CH0)

In a FlashArray chassis (`CH0`) contains the controllers and the bays that host the Flash Modules and NVRAM modules. The controller component names use `CT0`. The component names for the bays use `CH0`. The chassis status is generated from a combination of software conditions and the status of its components.

The components in a FlashArray chassis include:

- **Storage Bay** (`CH0.BAYn`):

  Each storage shelf contains a row of hot-swappable storage bays, which contain flash modules used for the array's data. Storage bays are indexed from left to right starting at 0.

- **NVRAM Bay** (`CH0.NVBn`):

  The chassis contains multiple NVRAM bays, which are hot-swappable and house NVRAM modules (temporary holding areas for host-written data). Bays are indexed from left to right, facing the front of the shelf.

- **Power Supply** (`CH0.PWRn`):

  The internal shelf contains redundant power supplies that report status. Power supply replacement is a service technician operation.

FlashArray Controller Components (CTm)

Purity//FA reports an overall controller (`CTm`) status. The controller status is generated from a combination of software conditions and the status of its components. For example, the status `ok` is reported for properly functioning controllers. Controllers contain 2- or 4-port Fibre Channel interface cards, 2-port Ethernet I/O interface cards, or a mix of both types.

The components in a FlashArray controller include:

- **Ethernet Port** (`CTm.ETHn`):

  Ethernet ports report status and link speed. Each controller contains at least 1x1GbE port for management (including one used for remote array administration) and 2x10GbE ports for replication.

- **Fan** (`CTm.FANn`):

  Each controller contains multiple cooling fans that report status. Fan replacement is a service technician operation.

- **Fibre Channel Port** (`CTm.FCn`):

  Fibre Channel ports report status and link speed. Most Fibre Channel-based controllers contain redundant 2- or 4-port Fibre Channel host bus adapters. Some controllers support a single Fibre Channel adapter. To see the Fibre Channel port details for each FlashArray model, refer to the Products page at https://support.purestorage.com.

- **InfiniBand Port** (`CTm.IBn`):

  InfiniBand ports are shipped only in upgrade kits for FA-4xx-to-FlashArray//X or //M upgrades. FlashArray//X and //M use InfiniBand ports only for controller upgrades from FA-4xx systems. InfiniBand ports report status and communication speed.

- **SAS Ports** (`CTm.SAS`):

  Each controller contains multiple dual-port SAS interface cards that communicate with flash modules. SAS ports report status and communication speed.

- **Temperature Sensor** (`CTm.TMPn`):

  Each controller chassis contains multiple temperature sensors that report their status and the ambient temperature at various points within the chassis. In steady-state operation, Purity//FA generates alerts when a temperature sensor reports a value outside of normal operating range.

Expansion Shelf Components (SHm)

Purity//FA reports an overall storage shelf (`SHm`) status generated from a combination of software conditions and the status of its components.

The optional expansion shelf is available with most FlashArray models. Refer to the associated hardware documentation for a given FlashArray model for more information.

The components in an expansion shelf include:

- **Storage Bay** (`SHm.BAYn`):

  Each storage shelf contains a row of hot-swappable storage bays, which contain flash modules used for the array's data. Storage bays are indexed from left to right starting at 0.

- **Fan** (`SHm.FANn`):

  Each of a shelf's power and cooling modules (PCMs) reports cooling fan status.

- **IOM** (`SHm.IOMn`):

  Redundant I/O Modules (IOMs) accessed from the rear of the shelf chassis contain the shelf's SAS ports.

- **Power Supply** (`SHm.PWRn`):

  The power supply sub-component within each of a shelf's power and cooling modules (PCMs) reports its operating status. Power supply replacement is a service technician operation.

- **SAS Ports** (`SHm.SASn`):

  Multiple SAS ports handle communication between the array's controllers and the shelf's flash modules.

- **Temperature Sensor** (`SHm.TMPn`):

  Each controller chassis contains multiple temperature sensors that report their status and the ambient temperature at various points within the chassis. In steady-state operation, Purity//FA generates alerts when a temperature sensor reports a value outside of normal operating range.

Viewing Hardware Component Details

The `purehw list` command displays information about array hardware components that are capable of reporting their status. The display is primarily useful for diagnosing hardware-related problems.

If one or more component names are specified in the command line, information is displayed for those components only. If component names are not specified in the command line, information is displayed for all components.

Include the `--spec` option to display the hardware specifications for each component wherever applicable. Hardware specifications, all of which are unique to Pure Storage, include FlashArray hardware model names, part numbers, and serial numbers. Component part numbers and serial numbers and are also printed on the physical hardware.

Include the `--type` option to display information for all components of the specified type.

The `purehw list` output includes the following columns:

Status

Each component reports its status as either:

`ok`
: Functioning properly at full capacity.

`critical`
: Not functioning or requiring immediate attention.

`degraded`
: Functioning, but not at full capability due to a non-fatal failure.

`device_off`
: Installed, but powered off.

`identifying`
: Functioning, but is not yet initialized.

`not_installed`
: Does not appear to be installed.

`unknown`
: Insufficient information to determine a state for this device.

Identify

State (on or off) of an LED used to visually identify the component. (Relevant only for controllers, NVRAM bays, storage bays, and storage shelves.)

Slot

Slot number occupied by the PCI Express card that hosts the component. (Relevant only for ports hosted by PCI Express cards in controller chassis.)

Index

Number that identifies the relative position of a hardware component within the array.

Speed

Speed at which the component is operating. (Relevant only for interface ports [bits/second].)

Temperature

Temperature reported by the component. (Relevant only for temperature sensors.)

Voltage

Input voltage (VIN) of the power supplies in the chassis. Only applies to //M models.

Changing Hardware Component Attributes

FlashArray controllers, storage shelves, NVRAM bays, and storage bays are equipped with LEDs that are illuminated to identify a physical component. The **`purehw setattr --identify`** command turns a component's identifying light on or off. The Identify column in the **`purehw list`** output displays the current state of the identifying light for each component that has one.

Storage shelf control panels contain numeric displays in which numbers can be set to uniquely identify shelves in multi-shelf arrays. Run the **`purehw setattr --index`** command to set the top-level storage shelf number. Once the top-level storage shelf number has been set, the number becomes part of the name of each component in the shelf. For example, the **`purehw setattr --index 2 SH0`** command sets shelf `SH0` to `SH2`, and all of the components in the shelf automatically assume names in the form `SH2.COMPONENT-NAME` (e.g., `SH2.BAY0`, `SH2.FAN1`, etc.).

Exceptions

The `purehw setattr --identify` command is not supported on Cloud Block Store.

Examples

Example 1

```
purehw list SH0.BAY11 SH0.BAY12
```

Displays information for flash modules in slots `11` and `12` of storage shelf `0`.

Example 2

```
purehw list --type fan
```

Displays information for all cooling fans in the array.

Example 3

```
purehw setattr --identify on CH0.BAY1
```

Illuminates the identifying LED for storage bay `1` in shelf `CH0`.

Example 4

```
purehw setattr --index 2 SH0
```

Sets shelf SH0 and all of its components to `SH2`. For example, sets `SH0` to `SH2`, `SH0.BAY0` to `SH2.BAY0`, `SH0.FAN1` to `SH2.FAN1`, and so on).

See Also

puredrive

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purekmip

purekmip, purekmip-create, purekmip-delete – creates and deletes KMIP server objects, and imports CA certificates

purekmip-list – displays the attributes of KMIP server objects and CA certificates

purekmip-setattr – changes the attributes of a KMIP server object or imports CA certificates

purekmip-test – tests the KMIP server connectivity

Synopsis

```
purekmip create [--ca-certificate] [--certificate CERTIFICATE]
[--uri URI-LIST] KMIP...
purekmip delete KMIP...
purekmip list [--ca-certificate KMIP] [KMIP...]
purekmip setattr [--ca-certificate] [--certificate CERTIFICATE]
[--uri URI-LIST] KMIP...
purekmip test KMIP...
```

Arguments

KMIP

KMIP server object to be created, deleted, modified, tested, or displayed.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--ca-certificate`

Imports the CA certificate generated by a certificate authority (CA) as the KMIP server communication certificate.

`--ca-certificate` ***`KMIP`***

Displays the CA certificate of a certificate authority (CA) as the KMIP server communication certificate for the specified KMIP server object.

`--certificate` ***`CERTIFICATE`***

CSR or self-signed certificate used to establish a mutually authenticated connection between the FlashArray array and the KMIP server.

`--uri` ***`URI-LIST`***

Comma-separated list of URIs assigned to the KMIP server object. Specify the URI in the format **`PROTOCOL://HOSTNAME:PORT`**. For example, **`tls://192.0.2.100:80,tls://192.0.2.101:80`**.

Description

The Key Management Interoperability Protocol (KMIP) server is used in combination with the Pure Storage Rapid Data Locking (RDL) feature to further secure the FlashArray array's encrypted data.

For more information on how to configure the KMIP server with the Rapid Data Locking, refer to the Enhanced Data Security Guide on the Knowledge site at https://support.purestorage.com.

Exceptions

The following commands are not supported on Cloud Block Store:

- The **`purekmip create`** command
- The **`purekmip delete`** command
- The **`purekmip setattr`** command

Examples

Example 1

```
purekmip create Vault_kmip --uri hashicorp-vault.dev.purestorage.com:5696 --certificate
Vault_cert --ca-certificate
```

Creates the `Vault_kmip` KMIP object of the remote HashiCorp Vault KMIP server with the `Vault_cert` certificate for communication with the KMIP server. Before running this command, you should obtain the CA certificate from your certificate authority.

Example 2

```
purekmip test Vault_kmip
```

Tests the communication between the FlashArray and the KMIP server using the `Vault_kmip` object created in Example 1.

Example 3

```
purekmip list --ca-certificate Vault_kmip
```

Displays the CA certificate of the `Vault_kmip` KMIP object.

Example 4

```
purekmip create CTM_kmip --uri thalesctm.dev.purestorage.com:5696 --certificate CTM_
cert --ca-certificate
```

Creates the `CTM_kmip` KMIP object of the remote CipherTrust Manager (CTM) KMIP server with the `CTM_cert` certificate for communication with the KMIP server. Before running this command, you should obtain the CA certificate from your certificate authority.

Example 5

```
purekmip test CTM_kmip
```

Tests the communication between the FlashArray and the KMIP server using the `CTM_kmip` object created in Example 4.

Example 6

```
purekmip list --ca-certificate CTM_kmip
```

Displays the CA certificate of the `CTM_kmip` KMIP object.

Example 7

```
purekmip create CTM_kmip --uri thalesctm.dev.purestorage.com:5696,thalesctm2.dev.purest
orage.com:5696 --certificate CTM_cert --ca-certificate
```

Creates the `CTM_kmip` KMIP object of two remote CTM KMIP servers with the `CTM_cert` certificate.

See Also

purearray, purecert

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purelog

purelog, purelog-file, purelog-list, purelog-setattr, purelog-syslog, purelog-test – manages event logging and connections to audit log targets and syslog servers

Synopsis

purelog delete ***NAME***...

purelog file create --dir ***DIR*** [--keep-for ***KEEP_FOR***] [--keep-size ***KEEP_SIZE***] ***NAME***

purelog file delete ***NAME***...

purelog file list [--cli | --csv | --nvp] [--notitle] [--raw] [--filter ***FILTER***] [--page] [--sort SORT] [***NAME***...]

purelog file setattr [--dir ***DIR***] [--keep-for ***KEEP_FOR***] [--keep-size ***KEEP_SIZE***] ***NAME***

purelog list [--cli | --csv | --nvp] [--notitle | --raw] [***NAME***...]

purelog syslog create --uri **URI** [--sourcelist ***SOURCELIST***] [--services ***SERVICES***] ***NAME***

purelog syslog delete ***NAME***...

purelog syslog global [enable | disable] --tls-audit

purelog syslog global list [--cli | --csv | --nvp] [--notitle] [--raw] [--page] [--ca-certificate]

purelog syslog global setattr {--ca-certificate | --logging-severity {debug,info,notice}}

purelog syslog list [--cli | --csv | --nvp] [--notitle] [--raw] [--filter ***FILTER***] [--page] [--sort SORT] [***NAME***...]

purelog syslog setattr [--uri ***URI***] [--sourcelist ***SOURCELIST***] [--services ***SERVICES***] ***NAME***

purelog syslog test

Arguments

INTERFACE

Controller and port of an interface.

NAME

Name used by Purity//FA to identify a log target, such as a syslog server or local directory.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--ca-certificate`

Import or display the Certificate Authority's (CA) certificate.

Purity//FA's syslog agent uses the CA's certificate in communicating with syslog servers using the TLS protocol.

`--dir DIR`

The name of a managed directory where audit logs will be stored.

`--keep-for` ***KEEP_FOR***

Option for time-based log file retention. The duration is specified as an integer, followed by one of the suffix letters `s` (seconds), `m` (minutes), `h` (hours), `d` (days), or `w` (weeks). For example, `2d`, `3w`, `4y`. To remove time-based retention, set the option to `0` (zero).

`--keep-size` ***KEEP_SIZE***

Option for size-based log file retention, the maximum size of each target log file (minimum 100 MiB). The size is specified as an integer, followed by one of the suffix letters `M` (MiB), `G` (GiB), or `T` (TiB). For example, `100M`, `10G`, `1T`. To remove size-based retention, set the option to `0` (zero).

`--logging-severity` ***LEVEL***

Sets the logging level for both the event log and the remote syslog (if configured).

LEVEL is one of the following:

- **notice**. Logs events that are unusual, including warnings and errors.
- **info**. Normal operations that require no action. Default.
- **debug**. Verbose information useful for debugging and auditing.

`--services` ***`SERVICES`***

Services that are used in the target. Valid values: **`management`**, **`data-audit`**. When omitted, the default value is **`management`**.

`--sourcelist` ***`INTERFACE...`***

A comma-separated list of network interface sources (one for each controller) for the syslog server. For example, **`ct0.eth0,ct1.eth0`**. The same interface name should be specified on each controller. Set ***INTERFACE*** to an empty string (**`""`**) to unset the interface list.

`--tls-audit`

Forward messages needed for TLS auditing.

`--`*`uri`* ***`URI`***

Creates or sets the URI of the remote syslog server. For example, `tcp://MyHost.com`.

Specify the URI in the format **`PROTOCOL://HOSTNAME:PORT`**.

`PROTOCOL` is `tcp`, `tls`, or `udp`.

`HOSTNAME` is the syslog server hostname or IP address. If specifying an IP address, for IPv4, specify the IP address in the form **`ddd.ddd.ddd.ddd`**, where `ddd` is a number ranging from `0` to `255` representing a group of 8 bits.

For IPv6, specify the IP address in the form **`[xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx]`**, where `xxxx` is a hexadecimal number representing a group of 16 bits. Enclose the entire address in square brackets (**`[]`**). Consecutive fields of zeros can be shortened by replacing the zeros with a double colon (**`::`**).

`PORT` is the port at which the server is listening. If a port number is specified, append it to the end of the address. If the port is not specified, it defaults to `514`.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed

both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Description

The **`purelog`** command configures the logging level for Purity//FA's event log, and configures Purity//FA's syslog agent to forward syslog messages to log targets such as local managed file directories or syslog servers. When a local managed directory is used as a target, the directory must be empty at start and it is advisable to not place the target below the directory being audited.

The **`purelog syslog global`** command configures the CA's certificate for authenticating syslog servers and displays the event log logging level. The **`enable`** and **`disable`** subcommands enable or disable global options. The **`list`** subcommand displays global configuration. Output includes the current TLS audit status and logging level. The **`setattr`** subcommand sets global attributes. For example, run the following command to set the event log logging level to notice, which is the least verbose setting:

```
purelog syslog global setattr --logging-severity notice
```

Syslog messages can be forwarded using the TLS protocol. SSL certificate installed using the purecert command is used by Purity//FA's syslog agent to authenticate itself to syslog servers. Run the **`purelog syslog global setattr --ca-certificate`** command to import the CA's certificate that Purity//FA's syslog agent will use for authenticating the syslog servers.

The **`test`** subcommand sends a test message to syslog servers. Verification of successful test message transmission is done at the destination.

The **`purelog file`** and **`purelog syslog`** commands configure, display, and set attributes of file auditing targets for file and syslog servers, respectively, or management logging to syslog servers. The **`create`** subcommand configures a log target.

For **`syslog`**, use the **`--sourcelist`** option to specify the controller ports to be used for logging traffic to the syslog server. The **`--sourcelist`** option takes a comma-separated list of symmetrical (each controller using the same port name) ports, one for each controller. The **`--uri`** option sets the URI of the remote syslog server. Specify the URI in the format `protocol://hostname:port`.

Valid options for **`file`** include **`--dir`** to specify the managed directory where the log files are stored, and **`--keep-for`** and **`--keep-size`** to specify time-based and size-based log file retention, respectively.

The **`delete`** subcommand deletes the log target configuration and stops forwarding messages, the **`list`** subcommand displays the configured log targets, and the **`setattr`** subcommand sets or resets target attributes.

Examples

Example 1

```
purelog syslog create --uri tcp://MyHost.com LOGSERVER1
```

Configures a syslog server named `LOGSERVER1` running on host `MyHost.com`, using `TCP` and at the default port `514`.

This immediately enables transmission of all future syslog messages to `LOGSERVER1`.

Example 2

```
purelog syslog create --uri tcp://[2001:0db8:85a3::ae26:8a2e:0370:7334]:614 LOGSERVER2
```

Configures a syslog server named `LOGSERVER2` running on host `2001:0db8:85a3::ae26:8a2e:0370:7334`, using `TCP` and at port `614`.

This immediately enables transmissions of all future syslog messages to `LOGSERVER2`.

Example 3

```
purelog syslog delete LOGSERVER2
```

Stops transmission of future syslog messages to `LOGSERVER2` and deletes the configured syslog server.

Example 4

```
purelog list LOGSERVER3
```

Displays the URI for `LOGSERVER3`.

Example 5

```
purelog syslog global disable --tls-audit
```

Disables TLS audit for all syslog servers.

Example 6

```
purelog syslog global enable --tls-audit
```

Enables TLS audit for all syslog servers.

Syslog messages needed for TLS auditing are forwarded to all configured syslog servers.

Example 7

```
purelog syslog global list
```

Displays global options.

Example 8

```
purelog syslog global list --ca-certificate
```

Displays the imported CA certificate.

Example 9

```
purelog syslog global setattr --ca-certificate
```

Import CA certificate.

Example 10

```
purelog syslog setattr --uri tcp://MyHost.com LOGSERVER1
```

Set URI for `LOGSERVER1` to `tcp://MyHost.com`.

Example 11

```
purelog syslog global setattr --logging-severity debug
```

Sets the event logging level to `debug`.

Example 12

```
purelog syslog create --uri tcp://1.1.1.2 --sourcelist ct0.eth0,ct1.eth0 server01
```

Specifies `ct0.eth0` and `ct1.eth0` as the network interface sources for the syslog server `server01`.

See Also

purearray, purepolicy

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puremaintenance

puremaintenance, puremaintenance-list, puremaintenance-schedule, puremaintenance-unschedule — management of maintenance windows

Synopsis

```
puremaintenance list { --cli | --csv | --nvp } [--notitle] [--page] [--raw]
puremaintenance schedule [--timeout PERIOD] MAINTENANCE...
puremaintenance unschedule MAINTENANCE...
```

Arguments

MAINTENANCE

Name of the maintenance window, which must be `array` or `environment`.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that control maintenance window duration:

`--timeout` ***`PERIOD`***

Sets the duration of the maintenance window. The duration is specified as an integer, followed by one of the suffix letters `s` (seconds), `m` (minutes), `h` (hours), or `d` (days).

The default maintenance period is 60 minutes. If no **`--timeout`** is specified, the duration of the maintenance window defaults to this value. The minimum maintenance period is 60 seconds, specified as `60s` or `1m`. The maximum maintenance period is 24 hours, specified as `24h` or `1d`.

Description

The puremaintenance CLI command sets array and environment maintenance windows, which suppress a set of alerts related to connections, paths, ports, and other resources that are down during maintenance. An array maintenance window should be used when performing maintenance on an array. An environment maintenance window should be used when performing maintenance in your environment.

The **puremaintenance list** command displays a list of currently running maintenance windows. To display only maintenance windows of a particular type, include the list type argument in the command, which may be `array` or `environment`.

The **puremaintenance list** command displays the following information:

`Name`
: The name of the maintenance window, which may be either `array` or `environment`.

`Created`
: Date and time that the maintenance window began.

`Expires`
: Date and time that the maintenance window ends.

The **puremaintenance schedule** command schedules a maintenance window which starts at the moment the command is issued and lasts for the specified duration. Two types of maintenance windows may be scheduled: `array` and `environment`. Users with array administrator or higher privileges can schedule an `environment` maintenance window. Only administrators with system engineer or higher privileges can schedule an `array` maintenance window.

The **puremaintenance schedule** command displays the following information:

`Name`
: The name of the maintenance window, which may be either `array` or `environment`.

`Created`
: Date and time that the maintenance window began.

`Expires`
: Date and time that the maintenance window ends.

The **puremaintenance unschedule** command immediately causes a designated maintenance window to expire. The maintenance window type must be specified: `array` or `environment`. Users with array administrator or higher privileges can unschedule a maintenance window.

Examples

Example 1

```
puremaintenance list
Name          Created                  Expires
```

```
environment  2020-01-03 22:42:33  2020-01-04 00:42:33
```

Lists the currently running maintenance windows of all types, including the times they were created and they times they expire. If there are no maintenance windows running currently, no output is generated.

Example 2

```
puremaintenance list environment
Name         Created              Expires
environment  2020-01-03 22:42:33  2020-01-04 00:42:33
```

Lists the currently running environment maintenance window, including the time it was created and when it expires. If there is no environment maintenance window running currently, no output is generated.

Example 3

```
puremaintenance list --csv
Name,Created,Expires
environment,2020-01-06 18:59:57,2020-01-06 21:59:57
```

Lists the currently running maintenance windows of all types, including the times they were created and they times they expire in CSV format. If there are no maintenance windows running currently, no output is generated.

Example 4

```
puremaintenance schedule --timeout 2h environment
Name         Created              Expires
environment  2020-01-03 22:42:33  2020-01-04 00:42:33
```

Schedules a maintenance window of the specified duration and type. The maintenance window begins after the command is entered. The output lists the maintenance window type, the time it was created, and when it expires.

Example 5

```
puremaintenance unschedule environment
Name         Created  Expires
environment  -        -
```

Immediately cancels the designated maintenance window.

See Also

purealert

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

pureman

pureman – displays man pages for Purity//FA commands

Synopsis

pureman {puread | pureadmin | purealert | pureapiclient | pureapp | purearray | pureaudit | purecert | pureconfig | puredir | puredns | puredrive | pureds | purefile | purefs | purehgroup | purehgroup-connect | purehgroup-list | purehost | purehost-connect | purehost-list | purehw | purekmip | purelog | puremaintenance | pureman | puremessage | puremultifactor | purenetwork | pureoffload | purepgroup | purepgroup-list | purepgroup-setattr | pureplugin | purepod | purepolicy | pureport | puresession | puresmis | puresmtp | puresnmp | puresso | puresubnet | puresupport | puresw | purevchost | purevgroup | purevol | purevol-list | purevol-setattr}

Description

The `pureman` command displays extensive help for each Purity//FA CLI command. To display the man page for a Purity//FA CLI command, run the `pureman` command with the Purity//FA CLI command name. For example, run the following command to display the man page for the **purevol** command:

```
pureman purevol
```

Help on certain Purity//FA subcommands is also supported by running the subcommand preceded with a hyphen. For example, run the following command to display the man page for the **purevol list** subcommand:

```
pureman purevol-list
```

CLI Conventions

The following list describes the conventions used in CLI help documentation:

- Text in fixed-width (Courier) font must be entered exactly as shown. For example, `purehost list`.
- Text not enclosed in brackets represents mandatory text.
- Text inside square brackets (“[ ]”) represents optional items. Do not type the brackets.
- Text inside curly braces (“{ }”) represents text, where one (and only one) item must be specified. Do not type the braces.
- The vertical bar (|) separates mutually exclusive items.
- Uppercase italic text represents entered text whose value is based on the nature of the subcommand or option. For example, `--size` ***`SIZE`***, where SIZE represents the value to be entered, such as `100g`.

CLI Command Syntax

PuriPurity//FA CLI commands have the general form:

```
command subcommand --options OBJECT-LIST
```

The parts of a command are:

COMMAND

Type of FlashArray object to be acted upon, prefixed by "pure". For example, the `purevol` command acts on Purity//FA virtual storage volumes.

Run the `pureman` command to see a list of Purity//FA CLI commands.

SUBCOMMAND

Action to be performed on the specified object. Most CLI subcommands are common to some or all object types.

For example, `purehost list` lists all hosts on the array, while `purevol list` lists all volumes on the array.

The following subcommands are common to most or all object types:

`create`

Creates and names one or more FlashArray objects.

`delete`

Deletes one or more specified objects.

`list`

Lists information about one or more objects. To list information for all objects, do not specify the object in the command.

`listobj`

Creates whitespace-separated lists of objects or attributes related to one or more objects.

For example, **`purevol listobj --type host`** creates a list of the hosts to which volumes have connections. The **`listobj`** subcommand is primarily used to create lists of object and attribute names for scripting purposes.

`rename`

Changes the name of the specified object. Purity//FA identifies the object name in administrative operations and displays. The new name is effective immediately and the old name is no longer recognized in Purity//FA GUI and CLI interactions. In the Purity//FA GUI, the new name appears upon page refresh. Hardware object names cannot be changed.

`setattr`

Changes the attribute values of the specified objects.

`OPTIONS`

Options that specify attribute values or modify the action performed by the subcommand.

For example, in the following command, the **`--addvollist`** option adds volumes `VOL1`, `VOL2`, and `VOL3` to protection group PGROUP1:

```
purepgroup setattr --addvollist VOL1,VOL2,VOL3 PGROUP1
```

Some options, such as **`--hostlist`**, inherently apply only to a single object. Other options, such as **`--size`**, can be set for multiple objects in a single command.

Some options can be multi-valued. For example, in the following command, the **`--hostlist`** option associates multiple hosts (`HOST1`, `HOST2`, and `HOST3`) with the host group `HGROUP1`.

```
purehgroup setattr --hostlist HOST1,HOST2,HOST3 HGROUP1
```

`Object-List`

Object or list of objects upon which the command is to be operated.

If a subcommand changes the object state, then at least one object must be specified. Examples of subcommands that change the object state include **`create`**, **`delete`**, and **`setattr`**. For example, **`purehost create HOST1`** creates host `HOST1`. In the command synopses, OBJECT specifications that are not enclosed in square brackets (for example, "***`HOST`***") represent ones that are required.

Passive subcommands, such as **`list`**, which do not change object state, do not require object specification. Leaving out the object is equivalent to specifying all objects of the type. For example, **purevol list** with no volumes specified displays information about all volumes in an array. In the command synopses, OBJECT specifications enclosed in square brackets (for example, "***[HOST]***") represent ones that are optional.

Most subcommands act on a single object. For example, in the following command, the **setattr** subcommand can only be run on a single host group to change the attributes of that host group.

```
purehgroup setattr --addhostlist HOST1 HGROUP1
```

Certain subcommands can operate on multiple objects. For example, the following command connects volume `VOL1` to host groups `HGROUP1` and `HGROUP2`.

```
purehgroup connect --vol VOL1 HGROUP1 HGROUP2
```

In the command synopses, OBJECT specifications that are followed by ellipses (for example, ***HGROUP...***) indicate that multiple objects can be entered.

CLI Output

Various Purity//FA CLI subcommands, including **list** and **monitor**, generate list outputs. With the ability to create and manage hundreds of volumes and snapshots, list outputs can become very long.

The Purity//FA CLI includes options to control the way a list output is displayed and formatted. The CLI also includes sort, filter, and limit options to control the results you see and the order in which you see them.

Interactive Paging

Pagination divides a large output into discrete pages. Pagination is disabled by default and is only in effect if the **--page** option is specified and the number of lines in the list output exceeds the size of the window.

To interactively move through a paginated list output:

- Press the [**Right Arrow**] key or space bar to move to the next page.
- Press the [**Left Arrow**] key to move to the previous page.

To quit interactive paging and exit the list view, press **q**.

With pagination, each page of the CLI list output begins with the column titles. To suppress the column titles, run the command with the **--notitle** option.

Using Wildcards

When performing a list or monitor operation, include the asterisk in the name argument to expand the list results. For example, the **purehost list *cat*** command displays a list of all hosts that contain "cat", such as `cat`, `catnap`, `happycats`, and `lolcat`. If the asterisks were not included, only the host named `cat` would be returned. As another example, the **purevol list *vol*** command displays a list of all volumes that contain "vol", including ones that begin or end with "vol".

```
$ purevol list *vol*
Name      Size   Source   Created
Myvol     100G   -        2016-04-11 10:18:07 PDT
MyVol-01  100G   -        2016-04-11 10:18:07 PDT
vol       1G     -        2016-04-11 11:19:19 PDT
vol01     100G   -        2016-04-11 10:17:23 PDT
Volume    100G   -        2016-04-11 10:17:23 PDT
```

If Purity//FA cannot find matches for the wildcard argument specified, an error message appears.

Asterisks are also allowed in Purity//FA CLI options that take a list of object names. For example, the **purevol list --snap --pgrouplist *pg*** command displays a list of all volume snapshots that are created as part of a protection group snapshot with "pg" in the protection group name. As another example, the following command displays a list of volume snapshots for all volumes that end with "01" and are created as part of a protection group snapshot with "pg" in the protection group name.

```
$ purevol list --snap --pgrouplist *pg* *01
Name                    Size   Source   Created
pgroup01.001.vol01      100G   vol01    2016-04-13 11:44:46 PDT
pgroup01.123.vol01      100G   vol01    2016-04-13 11:44:46 PDT
pgroup01.abc.vol01      100G   vol01    2016-04-13 11:44:46 PDT
pgroup01.backup.vol01   100G   vol01    2016-04-13 11:44:45 PDT
pgroup01.snap001.vol01  100G   vol01    2016-04-13 11:44:45 PDT
```

```
pgroup01.snap002.vol01   100G   vol01    2016-04-13 11:44:42 PDT
pgroup01.suffix.vol01    100G   vol01    2016-04-13 11:44:46 PDT
```

Formatting

Format options control the way list outputs are displayed. The following format options are common to most `list` and `monitor` subcommands:

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Here is a comparison between the `purehost list` output with formatted column titles and data, and the same output with unformatted column titles and data:

```
$ purehost list --space
Name   Size  Thin Provisioning   Data Reduction ...
H001   60T   57%                 1.5 to 1 ...
$ purehost list --space --raw
```

```
name  size            thin_provisioning  data_reduction ...
H001  65970697666560  0.5712341          1.5123 ...
```

Sorting

When running a list or monitor operation, include the `--sort` option to sort a column of the output in ascending or descending order. The following commands support the `--sort` option: purehgroup, purehost, purepgroup, pureport, and purevol.

The sort option adheres to the following syntax:

`--sort SORT`

`SORT` represents a comma-separated list of columns to sort. Use the unformatted title name (`--raw`) of the column to represent each column listed. To sort a column in descending order, append the minus (`-`) sign to the column name.

For example, run the following commands to get the unformatted column titles in the `purehost list` output, and then sort the same output by host group in descending order:

```
$ purehost list --raw
name                     ... hgroup
ESXi-GRP-Cluster02-H0001 ... ESXi-GRP-Cluster02-HG003
ESXi-GRP-Cluster02-H0002 ... ESXi-GRP-Cluster02-HG003
ESXi-IT-Cluster01-H0001  ... ESXi-IT-Cluster01-HG001
ESXi-IT-Cluster02-H0001  ... ESXi-IT-Cluster02-HG002
ESXi-STG-Cluster03-H0001 ... ESXi-STG-Cluster03-HG005
$ purehost list --sort hgroup-
Name                     ... Host Group
ESXi-STG-Cluster03-H0001 ... ESXi-STG-Cluster03-HG005
ESXi-IT-Cluster02-H0001  ... ESXi-IT-Cluster02-HG002
ESXi-IT-Cluster01-H0001  ... ESXi-IT-Cluster01-HG001
ESXi-GRP-Cluster02-H0002 ... ESXi-GRP-Cluster02-HG003
ESXi-GRP-Cluster02-H0001 ... ESXi-GRP-Cluster02-HG003
```

If multiple columns are specified, the output is sorted by the order of the columns listed.

If a sorted column contains non-unique values and a secondary sort argument is not specified, Purity//FA automatically performs a secondary sort using the default sort criteria (typically by `Name`).

Examples

Example 1: Display a list of volumes sorted in descending order by physical space occupied.

```
$ purevol list --space --sort "total-"
```

Example 2: Display a list of protection groups sorted in ascending order by source array name.

```
$ purepgroup list --sort "source"
```

Example 3: Display a list of volumes that are sorted in descending order by volume size. If any of volumes are identical in size, sort those volumes in descending order by physical space occupied.

```
$ purevol list --space --sort size-,total-
```

Filtering

When running a list or monitor operation, include the **`--filter`** option to narrow the results of the output to only the rows that meet the filter criteria. The following commands support the **`--filter`** option: purehgroup, purehost, purepgroup, pureport, and purevol. Filtering can be performed on any column of the list output.

Filtering with Operators

When filtering with operators, use the following syntax:

`--filter "RAW_TITLE OPERATOR VALUE"`

`RAW_TITLE` represents the unformatted title name of the column by which to filter. Run the list or monitor operation with the **`--raw`** option to get the unformatted title name.

`OPERATOR` represents the type of filter match (**=**, **!=**, **<**, **>**, **<=**, or **>=**) used to compare `RAW_TITLE` to `VALUE`.

`VALUE` represents the value (number, date, or string) that determines the results to be included in the list output. Literal strings must be wrapped in quotes.

Filtering supports the following operators:

Operator	Description

=	Equals
!=	Does not equal
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to

Filtering supports the asterisk wildcard character. For example, the value "`*esx*`" in the `purepgroup list --filter "hosts=*esx*"` command displays a list of all protection groups with hosts members that contain "esx" in the host name.

The AND and OR operators can be used to further refine your filter. The AND operator displays only the results that meet all of the filter criteria in the command. The OR operator displays the results that meet at least one of the filter criteria in the command.

Examples

Example 1: Display a list of protection groups configured on source array `pure-001`.

```
$ purepgroup list --filter "source = 'pure-001'"
```

Example 2: Display a list of volumes that were created on or before `2016-05-23 13:09:39`.

```
$ purevol list --filter "created <= '2016-05-23 13:09:39 PDT'"
```

Example 3: Display a list of volume snapshots that are greater than 2 gigabytes in size.

```
$ purevol list --space --snap --filter "snapshots > '2G'"
```

Example 4: Display a list of volumes that are currently inactive.

```
$ purevol monitor --filter "output_per_sec=0 and input_per_sec=0"
```

Example 5: Display a list of volumes that begin with "vol10" or are greater than 100 gigabytes in size.

```
$ purevol list --filter "name = 'vol10*' or size > '100G'"
```

Filtering with Functions

The filter option supports the CONTAINS and NOT functions.

`--filter FUNCTION(PARAMETERS)`

Function	Description
`contains(raw_title,string)`	Contains the enclosed string.

	Takes exactly two parameters, where **`raw_title`** represents the unformatted title name of the column by which to filter and **`string`** represents the string to search within the column. Run the list or monitor operation with the **`--raw`** option to get the unformatted title name.
`not(expression)`	Inverse of the enclosed expression.

Examples

Example 1: Get a list of all volumes that include "cluster03" in the volume name.

```
$ purevol list --filter "contains(name, 'cluster03')"
```

Example 2: Get a list of all hosts that are associated with host groups with "PROD" in the host group name.

```
$ purehost list --filter "not(contains(hgroup, 'PROD'))"
```

Example 3: Get a list of all hosts that are not associated with host groups with "PROD" in the host group name.

```
$ purehost list --filter "not(contains(hgroup, 'GRP'))"
```

Existence Checks

The Purity//FA CLI supports existence checks. For example, the **`purevol list --filter "name"`** command checks to see if the "name" column exists in the volume list output.

Examples

Example 1: Get a list of all hosts associated with a host group.

```
$ purehost list --filter "hgroup"
```

Example 2: Get a list of all volumes that were not created from another source.

```
$ purevol list --filter "not(source)"
```

Example 3: Get a list of all hosts that are not associated with Fibre Channel WWNs.

```
$ purehost list --filter "not(wwn)"
```

Setting Limits

When running a list or monitor operation, include the **--limit** option to restrict the result size to the limit specified. The following commands support the **--limit** option: purehgroup, purehost, purepgroup, purepod, pureport, and purevol.

The limit option adheres to the following syntax:

```
--limit ROWS
```

ROWS represents the maximum number of rows to return.

For example, run the following command to list only the first 5 rows of the **purevol list** output:

```
$ purevol list --limit 5
Name                  Size  Source  Created                  Serial
ESXi-Cluster01-vol001 100G  -       2016-05-23 13:09:40 PDT B1438B8ACE487B00011021
ESXi-Cluster01-vol002 100G  -       2016-05-23 13:09:40 PDT B1438B8ACE487B0001101D
ESXi-Cluster01-vol003 100G  -       2016-05-23 13:09:40 PDT B1438B8ACE487B00011024
ESXi-Cluster01-vol004 100G  -       2016-05-23 13:09:40 PDT B1438B8ACE487B00011018
ESXi-Cluster01-vol005 100G  -       2016-05-23 13:09:40 PDT B1438B8ACE487B00011025
```

Combining Sorting, Filtering, and Limit Options

The **--sort**, **--filter**, and **--limit** options can all be combined together.

Examples

Example 1

```
$ purevol list --space --sort "total-" --limit 10 --filter "name = '*esx*'"
```

Displays the top 10 volumes that occupy the largest amount of physical space and include "esx" in the volume name.

Example 2

```
purevol list --space --sort "snapshots-" --limit 10
```

Displays the top 10 volumes that have the largest physical space occupied by data unique to one or more snapshots.

Example 3

```
$ purevol list --space --sort "data_reduction-" --limit 10
```

Displays the top 10 volumes with the highest data reduction ratios.

Example 4

```
$ purevol monitor --sort "output_per_sec-" --limit 10
```

Displays the top 10 volumes that use the most read bandwidth.

Example 5

```
$ purevol list --connect --filter "hgroup = '*PROD*'" --sort size- --limit 10
```

Displays the top 10 volumes with the largest provisioned sizes and have shared connections to host groups with "PROD" in the host group name.

Example 6

```
pureman purehost
```

Displays the Purity//FA man page for the `purehost` command.

Example 7

```
pureman purehost-connect
```

Displays the Purity//FA man page for the `purehost connect` subcommand.

Related Commands

The `pureman` command displays a list of the available Purity//FA commands.

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puremultifactor

puremultifactor, puremultifactor-list, puremultifactor-securid-am-disable, puremultifactor-securid-am-enable, puremultifactor-securid-am-list, puremultifactor-securid-am-setattr, puremultifactor-securid-am-test – manages multi-factor authentication with the RSA SecurID® Authentication Manager

Synopsis

puremultifactor list [--cli | --csv | --nvp] [--notitle] [--page] [--raw]

puremultifactor securid-am disable

puremultifactor securid-am enable

puremultifactor securid-am list [--cli | --csv | --nvp] [--notitle] [--page] [--raw]

puremultifactor securid-am setattr [--rest-server ***REST_SERVER***] [--endpoint-path ***ENDPOINT_PATH***] [--access-key] [--client-name ***CLIENT_NAME***] [--certificate]

puremultifactor securid-am test

Options

With the **puremultifactor securid-am setattr** command, the **--rest-server**, **--endpoint-path**, **--access-key**, and **--client-name** options may be entered singly or in any combination. The **--certificate** option must be entered after the RSA SecurID® Authentication Manager server is configured with the other `setattr` options.

With the **--certificate** and **--access-key** options, you are prompted to enter the certificate or access key information.

    -h | --help

Can be used with any command or subcommand to display a brief syntax description.

`--rest-server REST_SERVER`

The authentication server host (for instance, the server running RSA SecurID Authentication Manager), including the port number.

For example: `https://rsa-am.company.com:5555`.

List replica servers separated by commas, without any spaces around the commas.

`--endpoint-path ENDPOINT_PATH`

The API endpoint path.

The default endpoint path is `/mfa/v1_1/authn`.

`--access-key`

The access key (client key) for the RSA REST API. The key is entered through interactive prompt.

`--client-name`

The Accessible Agent for the FlashArray, as configured in the RSA Security Console.

The client name entered with **`puremultifactor securid-am setattr`** must match the name of an RSA Accessible Agent.

`--certificate`

Imports the RSA SecurID Authentication Manager's certificate data. The certificate data is entered through interactive prompt. The certificate data must be PEM formatted (Base64 encoded) and include the `"-----BEGIN CERTIFICATE-----"` and `"-----END CERTIFICATE-----"` lines. The certificate cannot exceed 3000 characters in total length. Press **`Enter`** and then **`Control-d`** to exit the interactive session. To clear the certificate, enter blank lines at the prompt.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed

both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Description

The **`puremultifactor`** command manages configuring and enabling multi-factor authentication for a FlashArray. This release supports the RSA SecurID® Authentication Manager.

Note: For multi-factor authentication with SAML2 single-sign on (SSO) through a third-party identity provider, instead see the **`puresso`** command.

The RSA SecurID Authentication Manager provides a different method of granting users access to the array in place of authentication through user passwords. When multi-factor authentication is enabled on a FlashArray, user passwords are no longer accepted to login to the array.

With multi-factor authentication, the array leverages third-party software for authentication (to gain access to the array). Authorization and group membership (to determine the user's role in the array) are still handled by the array. (User roles are defined with the **`--role`** option to the **`pureadmin create`** command.)

Important notes and warnings:

- The accounts used to log into the FlashArray must be local users with matching user accounts in the RSA Security Console. Each FlashArray user name must match its corresponding user name in the RSA Security Console.
- The **`puremultifactor securid-am test`** command verifies that the array can communicate with the RSA SecurID server and is ***required*** before enabling multi-factor authentication on the array.
- Only the **`pureuser`** account can be used to configure multi-factor authentication.

- When multi-factor authentication is enabled, array users who do not have accounts created in the RSA SecurID server (with matching user names) are not allowed to log into the array.
- Multi-factor authentication can only be configured and enabled with the `puremultifactor` CLI command. The Purity GUI does not support these operations.

Note that if any users are already logged into the array when multi-factor authentication is enabled, those user sessions are not affected (not terminated). Multi-factor authentication is used on their subsequent login attempts.

Configuration on the RSA SecurID Authentication Manager

The following must be configured in the RSA Security Console:

`The Accessible Agent for the FlashArray`

The value provided for the `puremultifactor securid-am setattr` command's **`--client-name`** option must match the array's Accessible Agent name in the RSA Security Console.

`Each FlashArray user`

Each user to be allowed access to the array must have an account in the RSA Security Console.

The user names configured in the RSA Security Console must exactly match the user names used on the FlashArray. If there is a typo or other mismatch, the user is denied authentication.

Configuration on the FlashArray

The following RSA SecurID information is required. This information is typically supplied by your RSA SecurID administrators.

- The URL of your RSA SecurID Authentication Manager server, including port number, and the URLs of any replica servers, including their port numbers.
- The RSA SecurID access key (not the access ID).
- The Accessible Agent name for the FlashArray.

- The certificate used by the RSA SecurID Authentication Manager.

  The certificate is required only if your RSA SecurID Authentication Manager does not have a certificate from a well-known, trusted root certificate authority.
- Instructions to obtain your RSA tokencode and your personal PIN.

Description of puremultifactor

Run the `puremultifactor securid-am setattr` command to configure the FlashArray to use the RSA SecurID Authentication Manager to authenticate array users. Use this command to set or clear the REST server, endpoint path, access key, client name, and certificate values.

The **`--rest-server`** ***`REST_SERVER`*** option specifies a combination of the RSA SecurID Authentication Manager server and the RSA REST API port. List additional replica servers (if any) as a comma-separated list.

`--endpoint-path` is optional and by default is set to `/mfa/v1_1/authn`. The endpoint path does not need to be changed unless the RSA Authentication API version has changed.

The **`--certificate`** option cannot be combined with the **`--access-key`**, **`--client-name`**, or **`--rest-server`** option.

The `puremultifactor securid-am test` command runs a series of tests to verify that the FlashArray can successfully contact the RSA REST server or servers.

Multi-factor authentication cannot be enabled until the test passes. The `puremultifactor securid-am test` command can be run at any time.

The `puremultifactor securid-am enable` command enables multi-factor authentication for the FlashArray. The RSA SecurID Authentication Manager then determines if users are allowed to log into the array.

Enable multi-factor authentication *after* you have completed these steps:

- Achieved a successful run of the `puremultifactor securid-am test` command.
- Created matching user accounts for FlashArray users in the RSA Security Console.

The `puremultifactor securid-am disable command` disables multi-factor authentication for the array. Authentication is then handled by user passwords. (The multi-factor configuration remains and can be re-enabled.)

The **`puremultifactor securid-am list`** command displays the current RSA SecurID configuration, including whether or not multi-factor authentication is enabled. Include the `--certificate` option to display the imported certificate (if any).

The **`puremultifactor list`** command displays the configured authentication manager, if any, and whether or not multi-factor authentication is currently enabled.

Certificate Requirements

The certificate must be PEM formatted (Base64 encoded) and include the `"-----BEGIN CERTIFICATE-----"` and `"-----END CERTIFICATE-----"` lines. The certificate cannot exceed 3000 characters in total length.

Trust Chain Error Message

The following error message is issued when an account other then `pureuser` performs the configuration steps:

```
puremultifactor: error: Unable to determine trust chain.
```

Examples

Examples 2 through 7 show the configuration steps in order.

⚠ **Important:** **`puremultifactor`** commands must be run by the ***`pureuser`*** account.

Example 1

```
puremultifactor list
```

Lists the type of multi-factor authentication configured for the array and whether or not multi-factor authentication is currently enabled.

If multi-factor authentication has never been configured on the array, returns with no message (as shown above).

This example shows RSA SecurID is configured and is enabled:

```
puremultifactor list
Protocol                            Enabled
RSA SecurID Authentication Manager  True
```

Example 2

```
puremultifactor securid-am setattr --rest-server https://rsa1.ps.com:5555 --client
       -name vm-az --access-key
Enter access key (client key):
Retype access key (client key):
REST Server               Endpoint Path   Enabled Access Key Client Name
https://rsa1.ps.com:5555 /mfa/v1_1/authn False   ****       vm-az
```

Sets the URL for the RSA SecurID Authentication Manager server and sets the client name (`vm-az` in this example).

Provide the RSA SecurID Authentication Manager access key at the prompt. Note that the cursor does not move when you type or paste the access key. Be careful to paste it only once.

This command also lists the resulting multi-factor authentication configuration.

This command is the first step of the configuration process and does not enable multi-factor authentication.

Example 3

```
puremultifactor securid-am setattr --certificate
Please enter certificate data, then press Enter followed by ^D:
```

Imports the RSA SecurID Authentication Manager server certificate.

Enter the certificate at the prompt, including the `-----BEGIN CERTIFICATE-----` and `-----END CERTIFICATE-----` lines, then select **Enter** and select **Ctrl-D**.

This command is part of the configuration process and does not enable multi-factor authentication.

Example 4

```
puremultifactor securid-am list
REST Server               Endpoint Path   Enabled Access Key Client Name
https://rsa1.ps.com:5555 /mfa/v1_1/authn False   ****       vm-az
```

Lists the configuration information for the RSA SecurID Authentication Manager server.

Example 5

```
puremultifactor securid-am test
REST Server                 Success
https://rsa1.ps.com:5555  True
```

Tests the RSA SecurID Authentication Manager configuration. This example shows a successful test.

Multi-factor authentication cannot be enabled until this test passes.

Example 6

```
puremultifactor securid-am enable
REST Server                Endpoint Path   Enabled Access Key Client Name
https://rsa1.ps.com:5555 /mfa/v1_1/authn True    ****       vm-az
```

Enables multi-factor authentication using the RSA SecurID Authentication Manager.

From this point on, all new login attempts are authenticated by the RSA SecurID Authentication Manager. Any existing user sessions are not affected and remain open.

Example 7

```
puremultifactor securid-am setattr --rest-server https://rsa1.ps.com:5555,https:
      //rsa2.ps.com:5556,https://rsa3.ps.com:5557 --client-name vm-az --access-key
Enter access key (client key):
Retype access key (client key):
REST Server                Endpoint Path   Enabled Access Key Client Name
https://rsa1.ps.com:5555 /mfa/v1_1/authn False   ****   vm-az
https://rsa2.ps.com:5556
https://rsa3.ps.com:5557
```

A configuration example that includes two replica servers.

Example 8

```
puremultifactor securid-am list --certificate
-----BEGIN CERTIFICATE-----
MIIDyTCCArGgAwIBAgIQextHsKSPmwRnjZNbfCZWBzANBgkqhkiG9w0BAQsFADCB
gDEzMDEGA1UEAwwqU1NBIHJvb3QgQ0EgZm9yIHJzYS1hbS5kZXYucHVyZXN0b3Jh
```

```
   ...
MXtySY0RxkCdm3v4Q6NSSznz16HUj79rQ4sDMVvZAf+oUu0qwk9JPc5yBQMrCpvN
r7CYDIv9yOi2ni+Gaw==
-----END CERTIFICATE-----
```

Lists the certificate information for the RSA SecurID Authentication Manager.

Example 9

```
puremultifactor securid-am disable
REST Server              Endpoint Path    Enabled Access Key Client Name
https://rsa1.ps.com:5555 /mfa/v1_1/authn  False   ****       vm-az
```

Disables multi-factor authentication with the RSA SecurID Authentication Manager.

From this point on, all new login attempts are authenticated by user passwords. Any existing user sessions are not affected and remain open.

RSA SecurID Resources

The RSA SecurID Access Knowledge Base [https://community.rsa.com/community/products/securid/knowledge-base] has several relevant articles, including the following:

- 000035143 - How to set up the REST RSA SecurID Authentication API for Authentication Manager 8.2 SP1 [https://community.rsa.com/docs/DOC-76573}
- RSA SecurID Tokens [https://community.rsa.com/docs/DOC-77049]

See Also

pureadmin

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purenetwork

purenetwork, purenetwork-eth-create, purenetwork-eth-delete, purenetwork-eth-disable, purenetwork-eth-enable, purenetwork-eth-list, purenetwork-eth-monitor, purenetwork-eth-setattr – manage, monitor, and display the Ethernet interfaces used to connect a FlashArray system to an Ethernet network

purenetwork-eth-neighbor – display information about devices connected to the array

purenetwork-eth-ping, purenetwork-eth-trace – ping remote destinations and traces routes on the Ethernet network

purenetwork-fc-disable, purenetwork-fc-enable, purenetwork-fc-list, purenetwork-fc-monitor, purenetwork-fc-setattr – manage, monitor, and display the Fibre Channel interfaces used to connect a FlashArray system to a Fibre Channel network

Synopsis

```
purenetwork eth create lacpbond [--subinterfacelist SUBINTERFACELIST]
INTERFACE

purenetwork eth create vif [--address ADDRESS] { --subnet SUBNET | --
subinterfacelist SUBINTERFACELIST } INTERFACE...

purenetwork eth delete INTERFACE...

purenetwork eth disable INTERFACE...

purenetwork eth enable INTERFACE...

purenetwork eth list [ --cli | --csv | --nvp ] [--notitle] [--page] [-
-raw] [--service SERVICE] [--subnet SUBNET] [--vlan VLAN]
[INTERFACE...]

purenetwork eth monitor [--error] [--flow-control] [--interval
INTERVAL][--rdma-error] [--repeat REPEAT] [--total] [--historical { 1h
| 3h | 24h | 7d | 30d | 90d | 1y } ] [--csv] [--notitle] [--raw] [--
filter FILTER] [--limit LIMIT] [--page] [--sort SORT] [INTERFACE...]
```

purenetwork eth neighbor list [--csv | --nvp] [--notitle] [--raw] [***INTERFACE...***]

purenetwork eth ping [--count ***COUNT***] [--interface ***INTERFACE***] [--no-hostname] [--packet-size ***PACKET-SIZE***] [--user-to-user-latency] ***DEST***

purenetwork eth setattr [--address ***ADDRESS***] [--gateway ***GATEWAY***] [--mtu ***MTUSIZE***] [--netmask ***NETMASK***] [--servicelist {ds,file,iscsi,management,nvme-roce,nvme-tcp,offload,replication,system} | --addservicelist {ds,file,iscsi,management,nvme-roce,nvme-tcp,offload,replication,system} | --remservicelist {ds,file,iscsi,management,nvme-roce,nvme-tcp,offload,replication,system}] ***INTERFACE...***] [--subinterfacelist ***SUBINTERFACELIST*** | --addsubinterfacelist ***SUBINTERFACELIST*** | --remsubinterfacelist ***SUBINTERFACELIST***] [--subnet ***SUBNET***] ***INTERFACE...***

purenetwork eth trace [--dont-fragment] [--interface ***INTERFACE***] [--method ***METHOD***] [--mtu] [--no-hostname] [--port ***PORT***] [--source ***SOURCE***] ***DEST***

purenetwork fc disable [--override-npiv-check] ***INTERFACE...***

purenetwork fc enable [--override-npiv-check] ***INTERFACE...***

purenetwork fc list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--service ***SERVICE***] [***INTERFACE...***]

purenetwork fc monitor [--error] [--interval ***INTERVAL***] [--repeat ***REPEAT***] [--total] [--historical { 1h | 3h | 24h | 7d | 30d | 90d | 1y }] [--csv] [--notitle] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--page] [--sort ***SORT***] [***INTERFACE...***]

purenetwork fc setattr [--servicelist {scsi-fc,replication,nvme-fc}] ***INTERFACE...***

purenetwork list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--service ***SERVICE***] [***INTERFACE...***]

Arguments

DEST

Internet Protocol (IP) address or full hostname of a ping target or of a remote computer to which the network route is to be determined.

`INTERFACE`

Network interface name.

Ethernet interface names are in the form CT*x*.ETH*y*, where *x* denotes the controller (0 or 1) and *y* denotes the Ethernet port.

App interface names are in the form *APP*.data*y* and *APP*.mgmt*y*, where *APP* denotes the app name and *y* denotes the interface (0 or 1).

Bond interface names are comprised of alphanumeric characters.

VLAN interface names include a VLAN ID number, which is appended to the name of the physical interface.

Fibre Channel interface names are in the form CTx.FC*y*, where x denotes the controller (0 or 1) and *y* denotes the Fibre Channel port.

In CLI commands, interface names are case-insensitive. For example, *CT0.ETH0*, *Ct0.Eth0*, *CT0.eth0*, and *ct0.eth0* refer to the same Ethernet interface name. Likewise, *bond1*, *Bond1*, and *BOND1* refer to the same bond interface name.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--address` ***`ADDRESS`***

IP address to be associated with the specified Ethernet interface.

For IPv4, enter the address in CIDR notation `ddd.ddd.ddd.ddd/dd`. For example, `10.20.20.210/24`. Alternatively, specify the address `ddd.ddd.ddd.ddd` with a netmask (`--netmask`).

For IPv6, enter the address and prefix length in the form `xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx/xxx`. For example, `2001:0db8:85a3::ae26:8a2e:0370:7334/64`. Consecutive fields of zeros can be shortened by replacing the zeros with a double colon (`::`). Alternatively, specify the address `xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx` with a prefix length (**`--netmask`**).

`--addservicelist`

Adds one or more services to an interface service list. For multiple services, use a comma-separated list.

`--addsubinterfacelist` ***SUBINTERFACELIST***

Comma-separated list of one or more additional child devices to be added to the specified bond interface.

Include the controller name **`ctX.`** prefix for LACP and VLAN interfaces. For floating virtual interfaces, only include the interface name, **`ethX`**, without the controller name prefix.

Purity//FA disables the interface before it is added as a child device to a bond interface. If the child device that you are adding to the bond interface is an administrative interface, it will lose SSH connection and no longer be able to connect to the controller.

`--count` ***COUNT***

Number of Internet Control Message Protocol (ICMP) ping messages to send in sequence.

`--dont-fragment`

Do not fragment the probe packets. For IPv4 and IPv6, it also sets the Don't Fragment (DF) bit. By default, IP allows the packets to be fragmented when it goes through a segment with a smaller maximum transmission unit (MTU).

`--error`

When used with the **`purenetwork eth monitor`** command, displays error statistics for all or the specified Ethernet interfaces, including CRC, frame, carrier, and dropped errors.

When used with the **`purenetwork fc monitor`** command, displays error statistics for all or the specified Fibre Channel interfaces, including CRC, link failure, loss of signal, loss of synchronization, and invalid word errors.

`--flow-control`

This option is used with the **`purenetwork eth monitor`** command. It displays flow-control errors per second.

`--gateway` ***GATEWAY***

IP address of the gateway through which the specified interface is to communicate with the network. For IPv4, specify the gateway IP address in the form `ddd.ddd.ddd.ddd`. For IPv6, specify the gateway IP address in the form `xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx`. When specifying an IPv6 address, consecutive fields of zeros can be shortened by replacing the zeros with a double colon (`::`).

To remove the gateway specification, set to a null value (`""`).

`--historical` ***TIME***

Displays historical data over the specified range of time. Valid time ranges include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year. This option is mutually exclusive with the **`--total`** option.

`--interface` ***INTERFACE***

For the **purenetwork eth ping** command, specifies an interface name using the physical Ethernet port name, such as `eth0` or `eth1`, or the logical name. You can run the **purenetwork eth list** command to obtain the logical name of an interface, such as `ct0.eth0`, `ct1.eth5`, `replbond`, or `vir0`. If you specify a physical Ethernet port name, the default controller is the current controller. Alternatively, you can specify an IP address to which the source address is set.

For the **purenetwork eth trace** command, specifies either the physical Ethernet port name or the logical name.

`--interval` ***INTERVAL***

Sets the number of seconds between displays of real-time updates. If omitted, the interval defaults to every 30 seconds.

`--method` ***METHOD***

Method for trace operations. By default, the trace operation uses the UDP method. Trace operations can also be specified as ICMP (icmp) or TCP (tcp).

`--mtu (`**purenetwork trace**` only)`

Maximum transmission unit (MTU) along the path being traced between two Ethernet hosts.

`--mtu` ***MTU-SIZE*** `(`**purenetwork eth setattr**` only)`

Specifies the maximum message transfer unit (packet) size for the interface in bytes. Valid values are integers between 568 and 9000 (inclusive). The default value is 1500.

`--netmask` ***NETMASK***

Defines the range of IP addresses that make up a group of IP addresses on the same network. For IPv4, if the address is not entered in CIDR notation, enter the subnet mask in the form **ddd.ddd.ddd.ddd**. For example, **255.255.255.0**. For IPv6, if the address entered did not include a prefix length, specify the prefix length. For example, **64**.

`--no-hostname`

Do not map IP addresses to hostnames when displaying them.

`--override-npiv-check`

Specifies this option with the **purenetwork fc disable** or **purenetwork fc enable** command to disable or enable a Fibre Channel interface configured for the **scsi-fc** service when N-Port ID Virtualization (NPIV) is in automatic mode.

`--packet-size` ***PACKET-SIZE***

The number of data bytes to be sent. The default packet size is 56. For ICMP tracing, the default packet size is combined with 8 bytes of ICMP header data to equal 64 ICMP data bytes.

`--port` ***PORT***

Initial destination port. The default destination port is 33434.

For UDP tracing, specifies the destination port base used by **`purenetwork trace`**. The destination port number will be incremented by each probe. For ICMP tracing, specifies the initial ICMP sequence value incremented by each probe. For TCP tracing, specifies the destination port to connect.

`--rdma-error`

This option is used with the **`purenetwork eth monitor`** command. It displays remote direct memory access (RDMA) errors per second.

`--remservicelist`

Removes one or more services to an interface service list. For multiple services, use a comma-separated list.

`--remsubinterfacelist` ***SUBINTERFACELIST***

Comma-separated list of one or more child devices to be removed from the specified bond interface.

`--repeat` ***REPEAT***

Sets the number of times to display real-time statistics. If omitted, the repeat count defaults to 1.

`--service`

Lists only the interfaces configured with the specified service.

Supported services include (case-sensitive) `app`, `ds`, `file`, `iscsi`, `management`, `nvme-roce`, `nvme-tcp`, `offload`, `replication`, `scsi-fc`, and `system`.

A `system` interface is only used on Cloud Block Store arrays.

`--servicelist`

Assigns the specified (comma-separated) service list to one or more specified interfaces. Replaces the previous service list. Service names are case sensitive. Supported service lists depend on whether the network interface is Ethernet or Fibre Channel.

Supported Ethernet services include `ds`, `file`, `iscsi`, `management`, `nvme-roce`, and `nvme-tcp`.

Supported Fibre Channel services include `nvme-fc` and `scsi-fc`.

Contact Pure Storage Technical Services to configure replication ports.

`--source` ***SOURCE***

Alternative source address. The source address must be the IP address of one of the Ethernet interfaces. By default, the **`purenetwork eth trace`** command uses the address of the outgoing interface.

`--subinterfacelist` ***SUBINTERFACELIST***

Comma-separated list of one or more child devices to be added to the specified bond interface. In the **`purenetwork eth setattr`** command, this option replaces the entire list of child devices that are currently associated with the specified bonding interface.

Include the controller name **ctX.** prefix for LACP and VLAN interfaces. For floating virtual interfaces, only include the interface name, **ethX**, without the controller name prefix.

Purity//FA disables the interface before it is added as a child device to a bond interface. If the child device that you are adding to the bond interface is an administrative interface, it will lose SSH connection and no longer be able to connect to the controller.

`--subnet` ***SUBNET***

For **purenetwork eth create vif** and **purenetwork eth setattr**, name of the subnet which the physical, virtual, bond, or VLAN interface is to be attached. To detach a physical, virtual, or bond interface from a subnet, set to an empty string (`""`).

For **purenetwork eth list**, lists only the interfaces that belong to the specified subnet.

`--total`

Follows output lines with a single line containing column totals in columns where they are meaningful. This option is mutually exclusive with the **--historical** option.

`--user-to-user-latency`

Prints full user-to-user latency. By default, **purenetwork ping** prints network round-trip times.

`--vlan`

Lists only the interfaces configured with the specified VLAN ID.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **--cli** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **--csv** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **ITEMNAME=VALUE**. Argument names and information items are displayed flush left. The **--nvp** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--page`

Turns on interactive paging.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Description

Manages the interfaces and the network connection attributes of the array.

Each FlashArray controller is equipped with two 1-gigabit Ethernet (1GbE) interfaces that connect to a data center network for array administration. The controller is also equipped with Fibre Channel interfaces, which can be configured to support host initiator traffic and replication traffic. Note that each traffic requires a dedicated Fibre Channel interface. In addition to the two 1GbE Ethernet interfaces, the FlashArray controller is also equipped with additional Ethernet interfaces. For more information about additional Ethernet interfaces, search for the FlashArray//XR3 Port Usage and Definition page on the Knowledge site at **`https://support.purestorage.com`**.

The Ethernet interfaces are called CTx.ETH0 and CTx.ETH1, where x denotes the array controller number. Physical interface ports are located on controller rear bulkheads, and are labeled ETH1 (left) and ETH0 (right). The Fibre Channel interfaces are called CTx.FCn, where x denotes the array controller number with a value of 0 or 1 and n denotes the Fibre Channel port number starting at 0.

To enable or disable an Ethernet interface, use the **`purenetwork eth enable`** or **`purenetwork eth disable`** command, respectively. To enable or disable a Fibre Channel interface, use the `purenetwork fc enable` or `purenetwork fc disable` command, respectively.

The `purenetwork list` command lists all the network interfaces for all controllers on the array.

App interfaces only appear for the apps that have been installed. For bond interfaces, the list also includes the bond interface child devices, if any. For interfaces that belong to subnets, the list also includes the name of the subnet to which they belong. Specify the **`--service`** or **`--vlan`** option with the `purenetwork eth list` command to display only the interfaces that are configured with the specified service type or VLAN ID, respectively.

The `purenetwork eth neighbor list` command lists information about other devices connected either to the array or to specified array ports. See "Neighboring Devices" on page 308.

Configuring Ethernet Interfaces

To list only Ethernet interfaces, use the `purenetwork eth list` command.

Use the `purenetwork eth setattr` command to change the IP address, netmask, gateway, and MTU attributes of the specified physical, virtual, or bond interface. If the interface belongs to a subnet, you can only change the IP address.

The prefix length can either be specified with the **`--address`** option, or as a netmask (for IPv4) or prefix length (for IPv6) with the **`--netmask`** option.

In the following example, an IPv4 address is being associated with interface **`ct0.eth1`** (both commands are equivalent):

```
purenetwork eth setattr --address 192.168.0.25/24 ct0.eth1
purenetwork eth setattr --address 192.168.0.25 --netmask 255.255.255.0 ct0.eth1
```

In the following example, an IPv6 address is being associated with interface **`ct0.eth1`** (both commands are equivalent):

```
purenetwork eth setattr --address 2001:0db8:85a3::ae26:8a2e:0370:7334/64 ct0.eth1
purenetwork eth setattr --address 2001:0db8:85a3::ae26:8a2e:0370:7334 --netmask 64
ct0.eth1
```

Network interface IP addresses and netmasks are set explicitly (DHCP is not supported), along with the corresponding netmasks.

Gateways are specified by IP address. To remove a port's gateway specification, set the **`--gateway`** option to a null value (**`""`**). In the following example, a null value is specified to remove the port's gateway specification that was associated with **`ct0.eth0`**:

```
purenetwork eth setattr --gateway "" ct0.eth1
```

Change the maximum transmission unit (MTU) of an interface by specifying **`--mtu`**. If you are changing the MTU of a physical interface that is associated with a VLAN, verify the MTU of the physical interface is greater than or equal to (>=) the MTU of the VLAN interface.

Configure a service on a specified network interface by using the **`--servicelist`** option. Services include `ds`, `file`, `iscsi`, `management`, `nvme-roce`, `nvme-tcp`, `replication`, and `system`. (`system` is supported only on Cloud Block Store.)

In the following example, port **`ct0.eth1`** is configured as a management port:

```
purenetwork eth setattr --servicelist management ct0.eth1
```

Binding CloudSnap to Offload

Binding CloudSnap to Offload interface uses "offload" network service. To add "offload" network service to the selected Offload interface run the **`purenetwork eth setattr --addservicelist`** command. Offload service can be added to the interface before or after Offload installation. Once offload service has been added, the Offload state will turn to **`healthy`**. For more information about installing the offload app, refer to "puresw" on page 542 or the CloudSnap Install and Upgrade Guide.

```
purenetwork eth setattr --addservicelist offload replbond
Name     Enabled Type Subnet ... Services     Subinterfaces
replbond True    bond -          replication  ct1.eth2
                                 offload      ct0.eth2
                                              ct1.eth3
                                              ct0.eth3
```

Fibre Channel Interfaces

To list only Fibre Channel interfaces, use the **`purenetwork fc list`** command.

For example, the following command lists only Fibre Channel interfaces on the FlashArray:

```
purenetwork fc list
Name Enabled WWN Speed Services
CT0.FC0 True 50:01:50:01:50:01:50:00 8.00 Gb/s scsi-fc
CT0.FC1 True 50:01:50:01:50:01:50:01 8.00 Gb/s scsi-fc
CT0.FC2 True 50:01:50:01:50:01:50:02 8.00 Gb/s scsi-fc
CT0.FC3 True 50:01:50:01:50:01:50:03 8.00 Gb/s scsi-fc
```

By default, Fibre Channel ports are configured for SCSI over Fibre Channel service. Contact Pure Storage Technical Services for other service configurations, including NVMe over Fibre Channel and replication.

Configure a service on a specified network interface by using the **`--servicelist`** option. Services include `nvme-fc` and `scsi-fc`. In the following example, port **`ct0.fc0`** is configured as an nvme-fc port:

```
purenetwork fc setattr --servicelist nvme-fc ct0.fc0
```

Apps

The Purity Run platform extends array functionality by integrating add-on services into the Purity//FA operating system. Each service that runs on the platform is provided by an app.

For each app that is installed, one app management interface is created per array management interface. An app data interface may also be created for high-speed data transfers.

The naming convention for app interfaces is ***`APP`*`.datay`** for the app data interface, and ***`APP`*`.mgmty`** for the app management interface, where ***`APP`*** denotes the app name, and *y* denotes the interface.

The following example lists three app interfaces for the **`linux`** app. The **`linux.mgmt0`** interface has been enabled and configured with an IP address to give **`pureuser`** the ability to log into the app.

```
$ purenetwork eth list
Name Enabled Address ... Speed Services
...
linux.data0 False - ... 10.00 Gb/s app
linux.mgmt0 True 10.8.102.96 ... 1.00 Gb/s app
linux.mgmt1 False - ... 1.00 Gb/s app
...
```

For more information about Apps, refer to pureapp.

Network Bonding

A bond interface combines two or more similar Ethernet interfaces, such as two or more 1GbE or two or more 10GbE interfaces, to form a single virtual "bonded" interface. A bond interface provides link redundancy, higher data transfer rates, load balancing, and link redundancy.

The `purenetwork eth setattr --addsubinterfacelist` command adds Ethernet interfaces to a bond interface. When you add an Ethernet interface to a bond interface, it becomes a subinterface of the bond interface. An Ethernet interface can only belong to one bond interface at a time. Administrative subinterfaces cannot be added as subinterfaces to bond interfaces. Furthermore, bond interfaces cannot be subinterfaces to other bond interfaces.

The `purenetwork eth setattr --remsubinterfacelist` command removes subinterfaces from a bond interface. When a subinterface is removed from its bond interface, it becomes an Ethernet interface.

The `purenetwork eth delete` command deletes a bond interface. Before a bond interface can be deleted, it must be empty of all subinterfaces.

Subnets

Interfaces with common attributes can be organized into subnetworks, or subnets, to enhance the efficiency of app, data (file, iSCSI, NVMe-RoCE, or NVMe-TCP), management, and replication traffic.

In Purity//FA, subnets can include physical, virtual, bond, and VLAN interfaces.

Physical, virtual, and bond interfaces can belong to the same subnet. To configure a subnet with physical, virtual, or bond interfaces:

1. Create the subnet. Run `puresubnet eth create` to create the subnet.
2. Add the physical, virtual, and bond interfaces to the subnet. Run `purenetwork eth setattr --subnet` to add a physical, virtual, or bond interface to a subnet.

To remove a physical, virtual, or bond interface from a subnet, set the **`--subnet`** option to a null value.

VLAN interfaces can belong to subnets, but they cannot be mixed with other interface types. All of the VLAN interfaces within a subnet must be in the same VLAN.

To configure a subnet with VLAN interfaces:

1. Create a subnet, assigning a VLAN ID to the subnet. Run **`puresubnet eth create --vlan`** to create the subnet.
2. Run **`purenetwork eth create vif --subnet`** to create a VLAN interface. The VLAN ID of the VLAN interface must match the VLAN ID of the subnet. Run the command for each corresponding physical network interface to be associated with the VLAN. To remove a VLAN interface from a subnet, run **`purenetwork eth delete`**.

For more information about subnets, refer to puresubnet.

Subnets and VLAN Interfaces

VLAN tagging allows customers to isolate traffic through multiple virtual local area networks (VLANs), ensuring data routes to and from the appropriate networks.

The **`purenetwork eth create vif`** command creates VLAN interfaces.

Create a VLAN interface for each of the corresponding physical network interfaces you want to associate with a VLAN.

In Purity//FA, VLAN interfaces have the naming structure **`ct`*`x`*`.ETH`*`y`*`.`*`z`***, where ***x*** denotes the controller (0 or 1), ***y*** denotes the interface (0 or 1), and ***z*** denotes the VLAN ID number. For example, **`ct0.eth1.500`**.

The **`purenetwork eth create vif`** command requires the **`--subnet`** option, which attaches the new VLAN interface to the appropriate subnet. Run **`puresubnet list`** to display a list of all subnets on the array. Run **`puresubnet create`** to create a new subnet.

All interfaces, including VLAN interfaces, inherit the MTU value of its subnet. Note that the MTU of a VLAN interface cannot exceed the MTU of the corresponding physical interface.

VLAN tagging is supported for the following service types: file, iSCSI, NVMe-RoCE, and NVMe-TCP. Before creating a VLAN interface, verify that one or more of these are configured on the physical interface.

When VLAN tagging is used for file, VLAN IDs must be mirrored for two controllers. For example, if a subnet with VLAN ID 50 is assigned to `ct0.eth5`, the same subnet must be assigned to `ct1.eth5`.

When creating a VLAN interface, include the **`--address`** option to create a reachable VLAN interface.

After a VLAN interface has been created and attached to a subnet, the interface inherits the mask, gateway, MTU, and VLAN ID from the subnet. Likewise, the subnet inherits the service type (for example, iSCSI) from its interfaces.

For more information about subnets and VLAN tagging, refer to puresubnet.

LACP

Link Aggregation Control Protocol (LACP) is an IEEE standard that allows individual Ethernet links to be aggregated into a single logical Ethernet link. Depending on your scenario, it can be used to increase bandwidth utilization, increase availability, or simplify network configurations. In order for LACP to work with the FlashArray, the network switch must be configured for LACP as well.

LACP (IEEE 802.3ad) is supported on the following FlashArray Ethernet ports:

- iSCSI
- File VIFs
- NVMe-TCP
- Replication (ActiveCluster only)

Prior to configuring LACP on the FlashArray, LACP must be configured on the network switch according to the network switch vendor's best practices. LACP can only be configured between Ethernet ports on the same controller. LACP is not supported on ports across controllers. Subinterfaces added to an LACP interface must have the same speed, MTU, and service.

To create the two LACP bond interfaces, use the **`purenetwork eth create lacpbond`** command. Specify the **`--subinterfacelist`** option with a comma separated list of physical interfaces, and a name to be used for the bond. Repeat for the second bond. Enable the interfaces with the **`purenetwork eth enable`** command. LACP interface names must be alphanumeric characters only.

The following example creates a File VIF using two LACP bonds as subinterfaces, one for each controller, and enables the interfaces.

In this example:

- `ct0.eth2` and `ct0.eth3` are the desired Ethernet interfaces to be aggregated on controller 0.
- `ct1.eth2` and `ct1.eth3` are the desired Ethernet interfaces to be aggregated on controller 1.

- `lacp0` and `lacp1` are the desired names of the LACP interfaces.
- `filevif` is the name of the File VIF.

1. Create and enable the LACP bonds using physical ports on controller 0 and 1:

```
purenetwork eth create lacpbond --subinterfacelist ct0.eth2,ct0.eth3 lacp0
purenetwork eth create lacpbond --subinterfacelist ct1.eth2,ct1.eth3 lacp1
purenetwork eth enable lacp0
purenetwork eth enable lacp1
```

2. Create the File VIF using the two LACP bonds as subinterfaces, setting an IP address to be associated with the File VIF interface, then enable the File VIF:

```
purenetwork eth create vif --address 192.168.10.211/24 --subinterfacelist
lacp0,lacp1 filevif
purenetwork eth enable filevif
```

Network Ping and Trace

Use the **`purenetwork eth ping`** command to determine whether a remote computer can be accessed by the FlashArray, provided that ICMP is enabled on the remote computer. When using the command, specify the target device by the IP address. If a DNS service is available and configured for the FlashArray, you can also specify the target device by the hostname.

In the following example, each of the three commands sends an ICMP echo request packet to the target host with the IP address `8.8.8.8` and then waits for the target host to respond with an ICMP echo reply packet. The receipt of corresponding echo reply packets and round-trip times are displayed to indicate the availability of the target device.

```
purenetwork eth ping --interface ct0.eth0 8.8.8.8
```

Sends an echo request packet from controller `ct0` and waits for a reply.

```
purenetwork eth ping --interface ct1.eth0 8.8.8.8
```

Sends an echo request packet from controller `ct1` and waits for a reply.

```
purenetwork eth ping --interface eth0 8.8.8.8
```

Sends an echo request packet from the controller of the outgoing interface and waits for a reply.

Use the **purenetwork eth trace** command to trace and display the route to a remote computer identified by the IP address or host name. To specify a host name, a DNS service must be available and configured for the FlashArray.

To find the available Ethernet interface names and the associated IP addresses, run the **purenetwork eth list** command.

Monitoring Network Interfaces

You can display network statistics, historical bandwidth, and error reporting for all or the specified network and Fibre Channel interfaces by using the **purenetwork eth monitor** and **purenetwork fc monitor** commands, respectively. The following example displays the network statistics for interfaces `ct0.eth0` and `ct0.eth1`.

```
$ purenetwork eth monitor
Name      Time                    B/s(rx)  B/s(tx)  Packets/s(rx)  Packets/s(tx)  Total
Errors/s
ct0.eth0  2019-06-25 13:34:11 PDT 8913     634      105            5              0
ct0.eth1  2019-06-25 13:34:11 PDT 7781     162      97             1              0
```

To show the aggregated data for each statistic of all network interfaces, specify the **--total** option, as shown in the following example.

```
$ purenetwork eth monitor --total
Name      Time                    B/s(rx)  B/s(tx)  Packets/s(rx)  Packets/s(tx)  Total
Errors/s
ct0.eth0 2019-06-25 13:34:11 PDT  8913     634      105            5              0
ct0.eth1 2019-06-25 13:34:11 PDT  7781     162      97             1              0
(total)  2019-06-25 13:34:11 PDT  16694    796      202            6              0
```

To report error statistics of network interfaces, run the **purenetwork eth monitor --error** command.

The following types of errors are reported:

- **CRC errors**: Indicate that packets have incorrect checksums. A cyclic redundancy check (CRC) is an error-detecting code for data transmission.
- **Frame errors**: Indicate that the received packets have misaligned Ethernet frames.
- **Carrier errors**: Indicate duplex mismatch or faulty hardware issues.

- **Dropped errors**: Indicate network congestion. For example, the packets are dropped because the connected switch ports cannot process packets fast enough.

To report RDMA errors, run the `purenetwork eth monitor --rdma-error` command.

- **Req CQE Errors/s (rx)**: The number of times the port detected completion queue entries (CQEs) completed with errors per second.
- **Sequence Errors/s (rx)**: The number of packet sequence errors detected per second during the reception of packets.
- **Local ACK Timeout Error/s (tx)**: The number of times the acknowledgment (ACK) timer expired for queue pairs (QPs) per second during transmission.

To report flow control errors, run the `purenetwork eth monitor --flow-control` command.

- **Lossless B/s (rx):** The number of bytes received per second with lossless flow control settings.
- **Lossless B/s (tx)**: The number of bytes transmitted per second with lossless flow control settings.
- **Pause Frames/s (rx)**: The number of pause frames received per second by a network interface.
- **Pause Frames/s (tx)**: The number of pause frames transmitted per second by a network interface.
- **Congestion Packets/s (rx)**: The number of congestion control packets received per second by a network interface.
- **Congestion Packets/s (tx)**: The number of congestion control packets transmitted per second by a network interface.
- **Discarded Packets/s (rx)**: The number of received packets that are discarded per second by a network interface.
- **Discarded Packest/s (tx)**: The number of transmitted packets that are discarded per second by a network interface.

Neighboring Devices

The `purenetwork eth neighbor list` command uses information from Link Layer Discovery Protocol (LLDP) to list information about other devices connected to the array or connected to specified ports. This information can be helpful to determine how the array is cabled.

By default, the information is displayed with key names in one column and values in another. `Local Port` refers to the array controller and port. `Key` refers to the type of information, such as ID, name, component, address, and capability. The `Description` key displays the software running on the neighboring device.

In the following truncated example, each section shows a device connected to the `eth0` port on controller `ct0`.

```
# purenetwork eth neighbor list ct0.eth0
Local Port    Key                     Neighbor Value
ct0.eth0      Chassis Id              mac 00:1c:73:50:58:d7
              Chassis Name            eth122-d
              Addresses               10.7.122.5
              Description             Arista Networks EOS version 4.15.10M running on an
              Repeater                Supported: False, Enabled: False
              Bridge                  Supported: True, Enabled: True
              WLAN                    Supported: False, Enabled: False
              Router                  Supported: True, Enabled: False
              Telephone               Supported: False, Enabled: False
              Docsis                  Supported: False, Enabled: False
              Station                 Supported: False, Enabled: False
              Port Description        -
              Port Id                 ifname Ethernet16
              Initial TTL In Seconds  120
ct0.eth0      Chassis Id              mac 00:50:52:a5:42:63
              Chassis Name            vm-sa3-ct0
              Description             FlashArray VMware vm-sa3-ct0 202303182227-64x
              Repeater                Supported: False, Enabled: False
              Bridge                  Supported: True, Enabled: True
              WLAN                    Supported: True, Enabled: False
              Router                  Supported: True, Enabled: False
              Telephone               Supported: False, Enabled: False
              Docsis                  Supported: False, Enabled: False
              Station                 Supported: True, Enabled: False
              Port Description        eth1
```

```
             Port Id                mac 00:50:56:a5:62:63
             Initial TTL In Seconds 4
ct0.eth0     Chassis Id             mac 00:50:54:a5:c1:06
             Chassis Name           vm-tdub-ct0
             Description            FlashArray VMware vm-tdub-ct0 202209171844-62x
             Repeater               Supported: False, Enabled: False
             Bridge                 Supported: True, Enabled: True
             WLAN                   Supported: True, Enabled: False
             Router                 Supported: True, Enabled: False
             Telephone              Supported: False, Enabled: False
             Docsis                 Supported: False, Enabled: False
             Station                Supported: True, Enabled: False
             Port Description       eth0
             Port Id                mac 00:50:54:a5:1c:8c
             Initial TTL In Seconds 4
ct0.eth0     Chassis Id             mac 00:50:54:a5:a8:64
             Chassis Name           vm-vpathi2
             Description            FlashArray VMware vm-vgpathi2 202106012355-hm
             Repeater               Supported: False, Enabled: False
             Bridge                 Supported: True, Enabled: True
             WLAN                   Supported: True, Enabled: False
             Router                 Supported: True, Enabled: False
             Telephone              Supported: False, Enabled: False
             Docsis                 Supported: False, Enabled: False
             Station                Supported: True, Enabled: False
             Port Description       eth0
             Port Id                mac 00:50:54:a5:79:8c
             Initial TTL In Seconds 4
```

Exceptions

The following commands are not supported on Cloud Block Store:

- The `purenetwork eth create` command
- The `purenetwork eth delete` command
- The `purenetwork eth disable` command
- The `purenetwork eth enable` command
- The `purenetwork eth setattr` command

Examples

Example 1

```
purenetwork eth enable ct0.eth1
```

Enables controller `ct0`'s administrative network interface `eth1` to communicate with the administrative network.

Example 2

```
purenetwork list
```

Lists the attributes of the all network interfaces on the array.

Example 3

```
purenetwork eth setattr ct0.eth1 --address 192.168.0.24 --netmask 255.255.255.0
purenetwork eth setattr ct0.eth1 --address 192.168.0.24/24
```

Assigns IPv4 address `192.168.0.24` to administrative Ethernet interface `ct0.eth1`. Both commands are equivalent.

Example 4

```
purenetwork eth setattr ct0.eth1 --address 2001:db8:85a3::8a2e:370:7334 --netmask 64
```

Assigns IPv6 address `2001:db8:85a3::8a2e:370:7334` to administrative Ethernet interface `ct0.eth1` with the prefix length of 64 bits.

Example 5

```
purenetwork eth setattr --addsubinterfacelist eth2,eth3 bond009
```

Adds Ethernet interfaces `eth2` and `eth3` as child devices to bond interface `bond009`.

Example 6

```
purenetwork eth setattr --address 10.8.108.165 linux.mgmt0
purenetwork eth enable linux.mgmt0
```

Assigns IP address `10.8.108.165` to app management interface `linux.mgmt0` and enables the interface, giving `pureuser` the ability to log into the `linux` app.

Example 7

```
purenetwork eth setattr --servicelist management ct0.eth1
Name       Enabled  Type     Subnet  Address  Mask  Gateway  MTU  MAC
ct0.eth1   True     physical -       -        -     -        1500 00:50:56:a5:2a:3b
Speed       Services    Subinterfaces
10.00 Gb/s management -
```

Configures the service on port `ct0.eth1` to `management`.

Example 8

```
purenetwork eth create vif --address 192.168.0.24 --subnet ESXHost001 ct0.eth5.500
```

Creates VLAN interface `ct0.eth5.500` with IP address `192.168.0.24`, and adds it to existing subnet `ESXHost001`. The new VLAN interface inherits the mask, gateway, MTU, and VLAN ID from subnet `ESXHost001`.

Example 9

```
purenetwork eth create vif --subinterfacelist ct0.eth4,ct0.eth5,ct1.eth4,ct1.eth5 file-
vip
```

Creates a floating virtual interface named `filevip`, using four interfaces on two controllers, automatically adding the file service to the interface.

Example 10

```
purenetwork eth delete ct0.eth5.500
```

Removes VLAN interface `ct0.eth5.500` from its subnet and then deletes it.

Example 11

```
purenetwork eth setattr --subnet "" ct0.eth5
```

Removes physical interface `ct0.eth5` from its subnet. The interface is not deleted.

Example 12

```
purenetwork eth disable replbond
purenetwork eth setattr --remchildlist eth2,eth3 --address "" --netmask "" --gateway ""
replbond
purenetwork eth delete replbond
```

Disables the `replbond` bond interface, removes child devices `eth2` and `eth3` from the bond interface and clears the bond interface of all settings, and deletes the `replbond` bond interface.

Example 13

```
purenetwork eth ping --count 10 myhost.mydomain.com
```

Sends 10 ICMP ping messages to host `myhost.mydomain.com` and displays the response received for each one.

Example 14

```
purenetwork eth ping --interface eth0 8.8.8.8
```

Sends an ICMP ping message from Ethernet port `eth0` of the current controller to the host with IP address `8.8.8.8`, and displays the response received.

Example 15

```
purenetwork eth trace 192.168.0.1
```

Displays the route taken by a ping request to network address `192.168.0.1`.

Example 16

```
purenetwork eth monitor --historical 1h
```

Displays the network statistics of all Ethernet interfaces over the past 1 hour.

Example 17

```
purenetwork eth monitor --error
```

Displays the error statistics of all Ethernet interfaces.

Example 18

```
purenetwork fc monitor --error
```

Displays the error statistics of all Fibre Channel interfaces.

Example 19

```
purenetwork fc enable ct0.fc0
```

Enables the Fibre Channel interfaces `ct0.fc0`.

Example 20

```
purenetwork fc list
```

Displays the attributes of all the Fibre Channel interfaces.

Example 21

```
purenetwork fc setattr --servicelist scsi-fc ct0.fc0 ct0.fc1 ct1.fc0 ct1.fc1
Name       Enabled  WWN  Speed        Active Services   Services
ct0.fc0    True     -    8.00 Gb/s    scsi-fc           scsi-fc
ct0.fc1    True     -    8.00 Gb/s    scsi-fc           scsi-fc
ct1.fc0    True     -    8.00 Gb/s    scsi-fc           scsi-fc
ct1.fc1    True     -    8.00 Gb/s    scsi-fc           scsi-fc
```

Configures the service on ports `ct0.fc0`, `ct0.fc1`, `ct1.fc0`, and `ct1.fc1` to `scsi-fc`.

Example 22

```
purenetwork eth neighbor list
```

Displays Link Layer Discovery Protocol (LLDP) information about other devices connected to the array.

Example 23

```
purenetwork eth neighbor list ct1.eth0,ct1.eth1
```

Displays LLDP information about devices connected to ports `ct1.eth0` and `ct1.eth1`.

See Also

puread, pureapp, purearray, puredns, pureds, puresubnet, purevol-list

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

pureoffload

pureoffload, pureoffload-list, pureoffload-azure, pureoffload-s3 – manages offload targets

Synopsis

pureoffload list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--raw] [--space] [--total] [--context ***REMOTES***] ***NAME...***

pureoffload azure connect --account-name ***ACCOUNT-NAME*** [--container-name ***CONTAINER-NAME***] [--initialize] [--profile **PROFILE**] [--context ***REMOTE***] ***NAME***

pureoffload azure disconnect [--context ***REMOTE***] ***NAME***

pureoffload azure list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--page-with-token] [--token ***TOKEN***] [--context ***REMOTE***] [***NAME...***]

pureoffload s3 connect --access-key-id ***ACCESS-KEY-ID*** --bucket ***BUCKET*** [--uri ***URI***] [-- initialize] [--placement-strategy { aws-standard-class | retention-based }] [--profile **PROFILE**] [--context ***REMOTE***] ***NAME***

pureoffload s3 disconnect [--context ***REMOTE***] ***NAME***

pureoffload s3 list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--page-with-token] [--token ***TOKEN***] [--context ***REMOTE***] [***NAME...***]

Arguments

NAME

Name used by Purity//FA to identify an offload target.

PROFILE

The CloudSnap target specified by the --profile option. With s3, the s3-aws, s3-flashblade, s3-other, s3-wasabi-pay-as-you-go, and the s3-wasabi-rcs targets are supported.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--access-key-id` ***ACCESS-KEY-ID***

Access key ID of the AWS account.

A secret access key of the AWS account is also required to authenticate requests between the array and S3 bucket. The secret access key is 40 characters in length, and is entered through interactive prompt during connection.

`--bucket` ***BUCKET***

Name of the S3 bucket.

`--container-name` ***CONTAINER-NAME***

Name of the Microsoft Azure Blob container.

`--context` ***REMOTE***

Used to specify the remote array on which a command is processed. ***REMOTE*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

If used with **`pureoffload list`**, `--context` can specify multiple arrays and ***REMOTES*** is a comma-separated list of array names.

`--initialize`

Prepares an Azure Blob container or S3 bucket for offloading. Only required if this is the first time a FlashArray array is connecting to the Azure Blob container or S3 bucket. The array will only initialize an Azure Blob container or S3 bucket if it is empty.

`--placement-strategy`

Specifies the placement strategy for offloaded data. The valid values are as follows:

- `aws-standard-class`: selects the S3 Standard storage class.
- `retention-based`: selects the S3 storage class based on the retention period of the protection group.

If you do not specify this option, the default is the `retention-based` placement strategy when the array is initially connected to an S3 offload target.

`--profile`

Specifies the type of CloudSnap target. With Azure, the azure target is supported.

`--space`

Displays the space consumption in bytes of individual offload targets.

`--total`

Used with the **`--space`** option to display the total space occupied by all offload targets.

`--uri` ***`URI`***

Address or Hostname of the S3 server. For example `--uri http://10.59.245.10`.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--page-with-token`

Used to paginate output with a continuation token (**`--token TOKEN`**). Results are printed in `stdout` and the token is printed in `stderr`.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Description

The offload target feature enables system administrators to replicate point-in-time volume snapshots from the array to an external storage system for long-term retention. Each snapshot is an immutable image of the volume data at that instance in time. The data is transmitted securely and stored unencrypted on the storage system.

Supported offload targets include FlashBlade S3, Amazon S3, Wasabi S3, and Azure Blob.

Each offload target represents an external storage system, such as an Azure Blob container or S3 bucket, to where Purity//FA volume snapshots (generated via protection group snapshots) can be replicated.

Before you can connect to, manage, and replicate to an offload target, the respective Purity//FA app must be installed. For example, to connect to an Azure Blob container or S3 bucket, the CloudSnap offload app must be installed. To determine if apps are installed on your array, run the `pureapp list` command. To install the CloudSnap offload app, contact Pure Storage Technical Services.

To replicate volume snapshots to an offload target, the array must be able to connect to and reach the external storage system. Before you configure an offload target on the array, perform the following steps to prepare the offload target and verify that the network is set up to support the offload process:

1. Verify that at least one interface with the replication service is configured on the array. The `purenetwork` command lists and manages network interfaces. Assign an IP address to the port; this will be the interface that will be used to connect to the target device, such as an Azure Blob container or an S3 bucket. For optimum performance, an Ethernet interface of at least 10GbE is recommended.
2. Prepare the offload target.
   - For Azure Blob, create a Microsoft Azure Blob container and set the storage account to the hot access tier. Grant basic read and write ACL permissions, and verify that the container contains no blobs. By default, server-side encryption is enabled for the container and cannot be disabled.
   - For S3, create an S3 bucket. Grant basic read and write ACL permissions, and enable default (server-side) encryption for the bucket. Also verify that the bucket is empty of all objects and does not have any lifecycle policies.
3. Verify that the array can reach the offload target.

The **pureoffload** command displays and manages the connections between the array and offload targets.

The **pureoffload list** command displays a list of all offload targets that are connected to the array. Each offload target represents an external storage system, such as an Azure Blob container or S3 bucket, to where Purity//FA volume snapshots (generated via protection group snapshots) can be replicated. Offload targets that are disconnected from the array do not appear in the list of offload targets.

Specify the **--space** option with the **pureoffload list** command to display the space consumption in bytes of individual offload targets. Include the **--total** option to display the total space consumption of all offload targets, as shown in the following example:

```
$ pureoffload list --space
Name         Snapshots
myOffload1   33.47G


$ pureoffload list --space --total
Name         Snapshots
myOffload1   33.47G
(total)      33.47G
```

Azure Blob Offload

The **pureoffload azure list** command displays the connection details for each Azure Blob offload target that is connected to the array. Details include connection status, protocol type, container name, account name, and secret access key (displayed in masked form). Please note that Azure has only a single Azure profile available as of now.

To configure an offload target:

Connect the array to the offload target. For Azure, creating the connection to the Microsoft Azure Blob container requires the container account name and secret access key, both of which are created through the Microsoft Azure storage website.

Create a protection group, adding volume members and the offload target to the group.

Define the protection group snapshot and replication schedule.

Enable the protection group schedule.

The **pureoffload azure connect** command connects the array to an Azure Blob offload target. An array can be connected to one offload target at a time, while multiple arrays can be connected to the same offload target. Include the required **--account-name** option to specify the Microsoft Azure Blob account name when creating a connection between the array and an Azure Blob offload target. The secret access key of the Azure Blob account is required to authenticate requests between the array and Azure Blob container. The secret access key is entered twice through interactive prompt during connection. Specify the name of the Azure Blob container using the **--container-name** option. If not specified, the default is `offload`. If this is the first time any FlashArray array is connecting to the Azure Blob container, include the **--initialize** option to prepare the Azure Blob container for offloading. Use the **--profile** option to specify the type of CloudSnap target for the offload.

The **purepgroup** command creates protection groups and generates the protection group snapshots through which Azure Blob offload snapshots are generated. The **purepgroup** command also generates the volume snapshots and replicates them, either on a scheduled basis or on demand, to the offload target via protection group snapshots. For more information about creating protection groups and generating protection group snapshots, refer to purepgroup.

The **purevol** command displays and manages the individual volume snapshots on the offload target that are generated through the protection group snapshots. Through the **purevol** command, view a list of volume snapshots on the offload target and restore volume snapshots from an offload target to create a brand new volume. For more information about viewing and restoring volume snapshots, refer to purevol.

The **pureoffload azure disconnect** command disconnects the array from an Azure Blob offload target. Whenever an array is disconnected from an offload target, any data transfer processes that are in progress are suspended. The processes resume when the connection is re-established.

In the following example, the first command connects the array to Azure Blob offload target `mytarget`, which will offload volume snapshots to Azure Blob container `mybackup`. Because this is the first time any FlashArray array is connecting to Azure Blob container `mybackup`, the **--initialize** option is included. The secret access key is entered interactively. The second command displays the connection details of each Azure Blob offload target that is connected to the array.

```
$ pureoffload azure connect --initialize
                            --account-name mytest
                            --container-name mybackup mytarget
Enter secret access key:
Name      Status     Protocol  Container Name  Account Name  Secret Access Key
```

```
mytarget connected  azure     mybackup        mytest         ****

$ pureoffload azure list
Name      Status      Protocol  Container Name  Account Name   Secret Access Key
mytarget connected  azure     mybackup        mytest         ****
```

S3 Offload

The **pureoffload s3 list** command displays the connection details for each S3 offload target that is connected to the array. Details include connection status, protocol type, bucket name, access key ID, secret access key (displayed in masked form), and placement strategy.

To configure an offload target:

Connect the array to the offload target. For S3, creating the connection to the S3 bucket requires the bucket's access key ID and secret access key, both of which are created through your vendor’s services.

Create a protection group, adding volume members and the offload target to the group.

Define the protection group snapshot and replication schedule.

Enable the protection group schedule.

The **pureoffload s3 connect** command connects the array to an S3 offload target. An array can be connected to one offload target at a time, while multiple arrays can be connected to the same offload target. Include the required **--access-key-id** and **--bucket** options when creating a connection between the array and an S3 offload target. The access key ID is of the AWS account. The 40-character secret access key of the AWS account is also required to authenticate requests between the array and S3 bucket. The secret access key is entered twice through interactive prompt during connection. The bucket represents the name of the S3 bucket. If this is the first time any FlashArray array is connecting to the S3 bucket, include the **--initialize** option to prepare the S3 bucket for offloading. Use the **--profile** option to specify the type of CloudSnap target for the offload.

The **purepgroup** command creates protection groups and generates the protection group snapshots through which S3 offload snapshots are generated. The **purepgroup** command also generates the volume snapshots and replicates them, either on a scheduled basis or on demand, to the offload target via protection group snapshots. For more information about creating protection groups and generating protection group snapshots, refer to purepgroup.

The `purevol` command displays and manages the individual volume snapshots on the offload target that are generated through the protection group snapshots. Through the `purevol` command, view a list of volume snapshots on the offload target and restore volume snapshots from an offload target to create a brand new volume. For more information about viewing and restoring volume snapshots, refer to purevol.

The `pureoffload s3 disconnect` command disconnects the array from an S3 offload target. Whenever an array is disconnected from an offload target, any data transfer processes that are in progress are suspended. The processes resume when the connection is re-established.

Dynamic Placement

When you connect the array to an S3 offload target using the `pureoffload s3 connect` command, the command chooses a cost-effective storage class using dynamic placement. Dynamic placement stores objects automatically with an optimized storage class depending on the retention period of the protection groups. For data that is less frequently accessed, dynamic placement provides a more cost-effective use of object storage.

By default, dynamic placement chooses the retention-based placement strategy when the array is initially connected to an S3 offload target. To manually select a placement strategy for an offload target connection, specify the **`--placement-strategy`** option as follows:

`--placement-strategy retention-based`

Specifies the retention-based placement strategy to configure the S3 storage class based on the retention period of the protection group.

This is the default placement strategy if you do not specify the **`--placement-strategy`** option.

`--placement-strategy aws-standard-class`

Specifies this placement strategy to select the S3 standard storage class to offload volume snapshots. This storage class is suitable for general-purpose storage of frequently accessed data.

When an S3 offload target is disconnected and then reconnected, the placement strategy is determined as follows:

- If no placement strategy is specified, the existing placement strategy is used.
- If a different placement strategy is specified, the system will move objects to the storage class selected by the placement strategy for the offload target connection regardless of storage costs.

Examples

Example 1

```
pureoffload list
```

Displays a list of all offload targets that are connected to the array, their connection statuses and their protocol types.

Example 2

```
pureoffload list --space
```

Displays the space consumption in bytes of individual offload targets.

Example 3

```
pureoffload s3 connect --initialize --bucket purebucket1
                       --access-key-id ABCDEFGHIJKLMNOPQRST S3-TARGET01
Enter secret access key:
```

Connects the array to S3 offload target `S3-TARGET01`, which will offload volume snapshots to Amazon S3 bucket `purebucket1`. Since this is the first time any FlashArray array is connecting to S3 bucket `purebucket1`, the **`--initialize`** option is included. The secret access key is entered interactively.

Example 4

```
pureoffload s3 connect --bucket purebucket2
                       --access-key-id ZYXWVUTSRQPOMNLKJIHG S3-TARGET02
Enter secret access key:
```

Connects the array to AWS offload target `S3-TARGET02`, which will offload volume snapshots to Amazon S3 bucket `purebucket1`. A FlashArray array has already initialized S3 bucket `purebucket1` from a previous S3 offload connection, so the bucket does not need to be initialized again. The secret access key is entered interactively.

Example 5

```
pureoffload s3 connect --access-key-id PSFBSAZRLAFBIPHHDKOMBDBGNIOFNINIBBOMLGKBM --
bucket purebucket3 --initialize --profile s3-flashblade --uri http://10.59.245.10 S3-
FB-TARGET
```

```
Enter secret access key:
```

Connects the array to S3 FlashBlade offload target.

Example 6

```
pureoffload s3 connect --bucket purebucket1 --access-key-id ABCDEFGHIJKLMNOPQRST S3-
TARGET03
                       --placement-strategy aws-standard-class
Enter secret access key:
```

Connects the array to S3 offload target `S3-TARGET03` and sets the placement strategy to the `aws-standard-class` storage class to offload volume snapshots to Amazon S3 bucket `prebucket1`. The secret access key is entered interactively.

Example 7

```
pureoffload s3 disconnect S3-TARGET
```

Disconnects the array from S3 offload target `S3-TARGET`.

Example 8

```
pureoffload azure disconnect AZCONN
```

Disconnects the array from Azure Blob offload target `AZCONN`.

Example 9

```
purepgroup create --vollist vol01,vol02 --targetlist S3-TARGET01 pgroup01
purepgroup schedule --replicate-frequency 4h pgroup01
purepgroup enable --replicate pgroup01
```

Creates protection group pgroup01 with volumes `vol01` and `vol02` and offload target `S3-TARGET01`.

Configures the protection group schedule to replicate a snapshot of all volumes in `pgroup01` to offload target `S3-TARGET01` every four hours.

Enables the replication schedule for protection `pgroup01` to immediately begin replicating protection group snapshots to offload target `S3-TARGET01`.

Example 10

```
purevol list --snap --on S3-TARGET01
purevol get --on S3-TARGET01 pure-001:pgroup01.25.vol03
purevol copy pure-001:vol03.restore-of.pure-001-pgroup01-25-vol03 restore1
```

Displays a list of volume snapshots (generated from protection group snapshots) both local and remote to the array that have been replicated and retained on offload target `S3-TARGET01`.

Restores volume snapshot `pure-001:pgroup01.25.vol03` from offload target `S3-TARGET01`, resulting in a volume snapshot with the name `pure-001:vol03.restore-of.pure-001-pgroup01-25-vol03`, which is then copied to create a new volume named `restore1`.

See Also

purepgroup, purevol

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purepgroup

purepgroup, purepgroup-copy, purepgroup-create, purepgroup-destroy, purepgroup-eradicate, purepgroup-recover, purepgroup-rename, purepgroup-send, purepgroup-snap – manage the creation and deletion of the Purity//FA protection group (pgroup) objects.

purepgroup-retention-lock-ratchet – manages the configuration of SafeMode.

Synopsis

```
purepgroup copy [ --overwrite ] [--context REMOTE]  SOURCE DEST

purepgroup create [ --hgrouplist HGROUPS | --hostlist HOSTS | --
vollist VOLS ] [--targetlist TARGETS] [--context REMOTE] PGROUP...

purepgroup destroy [--on REMOTE] [--context REMOTE] PGROUP...

purepgroup eradicate [--on REMOTE] [--context REMOTE] PGROUP...

purepgroup recover [--on REMOTE] [--context REMOTE] PGROUP...

purepgroup rename [--context REMOTE]  OLD-NAME NEW-NAME

purepgroup retention-lock ratchet [--context REMOTE] PGROUP...

purepgroup send [--convert-source-to-baseline] [--to ARRAYS] [--con-
text REMOTE]

PGROUP.NNN...

purepgroup snap [--apply-retention] [--replicate-now | --replicate] [-
-allow-throttle] [--dry-run][--suffix SUFFIX] [--context REMOTE]
PGROUP...
```

For **`purepgroup list`**, see purepgroup-list.

For **`purepgroup setattr`**, see purepgroup-setattr.

Arguments

PGROUP

Protection group or protection group snapshot to be created, destroyed, eradicated, or recovered. For `purepgroup snap`, protection group to be snapped.

SOURCE

Protection group or protection group snapshot on the target array from where data is copied. The data is copied to the ***DEST*** protection group.

DEST

Protection group to where data is copied. The data is copied from the SOURCE protection group or protection group snapshot on the target array.

OLD-NAME

Name of the protection group to be renamed.

NEW-NAME

Name by which the protection group is to be known after the command executes.

PGROUP.NNN

Name of the protection group snapshot.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--allow-throttle`

Used with `purepgroup snap` command to check the health of the array. If the array health is not optimal, the snapshot will fail.

`--apply-retention`

Applies the retention schedule settings to the snapshot.

`--context` ***REMOTE***

Used to specify the remote array on which a command is processed. ***REMOTE*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--convert-source-to-baseline`

Used with `purepgroup send` to make it so the snapshot would be eradicated automatically when it is no longer baseline on the source.

`--dry-run`

Simulates taking a snapshot on the array.

`--for-replication`

Specifies the snapshot that will be used for manual replication requests at a later time.

`--hgrouplist`

Comma-separated list of one or more host groups to be included in the specified protection groups.

`--hostlist`

Comma-separated list of one or more hosts to be included in the specified protection groups.

`--on` ***ON***

Specifies the source of the protection group when used with `purepgroup snap`.

`--on` ***REMOTE***

Used with `purepgroup destroy` to destroy protection groups on an offload target.

Used with `purepgroup eradicate` to eradicate protection groups or snapshots on an offload target.

Used with `purepgroup recover` to recover destroyed protection groups on an offload target.

`--overwrite`

Allows `purepgroup copy` to overwrite existing volumes. Without this option, if the DEST protection group already contains volumes with matching names, then the command fails. With this option, if the DEST protection group exists, then the SOURCE and DEST protection groups must contain the same volumes (volume names).

`--replicate`, `--replicate-now`

Replicates the snapshot to all targets. For asynchronous replication, only replicates snapshots to target arrays or target pods that have allowed replication.

`--suffix`

Specifies a name suffix for the snapshots created. Specifying this option causes snapshots to have names of the form ***PGROUP.SUFFIX*** rather than the default ***PGROUP.NNN*** form. The names of all snapshots created by a single command that specifies this option have the same suffix.

`--targetlist`

Comma-separated list of one or more target arrays or target pods (for asynchronous replication) or offload targets (for replication to an offload target) to be included in the specified protection groups. The targets receive the replicated snapshots.

`--to`

Used with **`purepgroup send`** to specify a comma-separated list of arrays or pods that will receive the snapshot.

`--vollist`

Comma-separated list of one or more volumes to be included in the specified protection groups.

Conventions

Object Names

Valid characters are letters (A-Z and a-z), digits (0-9), and the hyphen (-) character. The first and last characters of the name must be alphanumeric, and the name must contain at least one letter or '-'.

Most objects in Purity//FA that can be named, including host groups, hosts, volumes, protection groups, volume and protection group suffixes, SNMP managers, and subnets, can be 1-63 characters in length.

Array names can be 1-56 characters in length. The array name length is limited to 56 characters so that the names of the individual controllers, which are assigned by Purity//FA based on the array name, do not exceed the maximum allowed by DNS.

Names are case-insensitive on input. For example, **`vol1`**, **`Vol1`**, and **`VOL1`** all represent the same volume. Purity//FA displays names in the case in which they were specified when created or renamed.

All objects in a pod have a fully qualified name that include the pod name and object name. The fully qualified name of a volume in a pod is **`POD::VOLUME`**, with double colons (**`::`**) separating the pod name and volume name. The fully qualified name of a protection group in a pod is **`POD::PGROUP`**, with double colons (**`::`**) separating the pod name and protection group name. For example, the fully qualified name of a volume named **`vol01`** in a pod named **`pod01`** is **`pod01::vol01`**, and the fully qualified name of a protection group named **`pgroup01`** in a pod named **`pod01`** is **`pod01::pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target array, the fully qualified name of the protection group on the target array is **`POD:PGROUP`**, with single colons (`:`) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to target array **`array02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod01:pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target pod, the fully qualified name of the protection group on the target pod is **`POD:PGROUP`**, with single colons (`:`) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to a target pod **`pod02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod02::pod01:pgroup01`**.

All objects in a volume group have a fully qualified name that includes the volume group name and the object name, separated by a forward slash (`/`). For example, the fully qualified name of a volume named **`vol01`** in a volume group named **`vgroup01`** is **`vgroup01/vol01`**.

Realms, pods, and volume groups provide a namespace with unique naming conventions. All objects in a realm have a fully qualified name that include the realm name and object name or realm, pod, and object names. The fully qualified name of a pod in a realm is **`REALM::POD`**, with double colons (`::`) separating the realm name and pod name. The fully qualified name of a volume in a pod in a realm is **`REALM::POD::VOLUME`**, with double colons (`::`) separating the realm, pod, and volume names. The fully qualified name of a protection group in a pod in a realm is **`REALM::POD::PGROUP`**, with double colons (`::`) separating the realm, pod, and protection group names. The fully qualified name of an object in a realm but not in a pod is **`REALM::HOST`**, using host as an example.

Protection Group Snapshot Names

The protection group snapshot naming convention is **`PGROUP.NNN`**, where:

- **`PGROUP`** is the name of the protection group.
- **`NNN`** is a unique monotonically increasing number or a manually-assigned protection group snapshot suffix name.

The protection group volume snapshot naming convention is **`PGROUP.NNN.VOL`**, where:

- **`PGROUP`** is the name of the protection group.
- **`NNN`** is a unique monotonically increasing number or a manually-assigned protection group snapshot suffix name.
- **`VOL`** is name of the volume member.

If you are viewing replicated snapshots on a target array or offload target, the snapshot name begins with the name of the source array from where the snapshot was taken.

Protection Groups in Pods

When administering a protection group that is in a pod, always include the pod namespace in the protection group name.

Each protection group in a pod consists of the pod namespace identifier and the protection group name, separated by a double colon (`::`). The naming convention for a protection group inside a pod is **`POD::PGROUP`**, where:

- **`POD`** is the name of the container pod.
- **`PGROUP`** is the name of the protection group inside the pod.

For example, the fully qualified name for a protection group named pgroup01 inside a pod named pod01 is **`pod01::pgroup01`**. The following command creates the protection group pgroup01 in pod01:

```
purepgroup create pod01::pgroup01
```

Description

A protection group defines a set of volumes, hosts, or host groups (called members) that are protected together through snapshots with point-in-time consistency across the member volumes. The members within the protection group have common data protection requirements and the same snapshot, replication, and retention schedules.

Protection group snapshots capture the content of all volumes on the source array for the specified protection group at a single point in time. The snapshot is an immutable image of the volume data at that instance in time. The volumes are either direct members of the protection group or connected to any of its hosts or host groups within a protection group.

Volumes are protected through protection group snapshots that are retained locally, replicated to other arrays or storage systems, or both. Purity//FA supports the following types of replication:

- Asynchronous replication, where volume snapshots are replicated from the current (source) array to a target array or user pod. For asynchronous replication, a single protection group can consist of multiple hosts, host groups, and volumes. Likewise, hosts, host groups, and volumes can be associated with multiple protection groups.

- ActiveCluster replication, where volume snapshots are synchronously replicated between two FlashArrays through stretched pods. For more information about ActiveCluster replication, refer to purepod.
- ActiveDR replication, where volume snapshots are continuously replicated between two FlashArrays through linked pods. For more information about ActiveDR replication, refer to purepod.
- Replication to an offload target, where volume snapshots are replicated from the current array to an external storage system such as an S3 bucket. For replication to offload targets, only volume members can be added to protection groups. For more information about offload targets, refer to pureoffload.

Protection group snapshots cannot be connected to hosts or host groups for writing and reading. However, the volumes within a protection group snapshot, which are visible in both the Storage and Protection tabs, can be copied to new or existing live, host-accessible volumes.

Creating Protection Groups

Run **`purepgroup create`** on the source array (source) to create a new protection group.

For asynchronous replication, include the **`--hostlist`**, **`--hgrouplist`**, or **`--vollist`** option to add hosts, host groups, or volumes, respectively, as members to the protection group. Only members of the same object type can belong to a protection group. For example, hosts or host groups cannot be added to a protection group that contains volumes.

To replicate snapshots, the protection group must include at least one target array or offload (target), or user pod target to where the replicated data is written. Include the **`--targetlist`** option to add the targets to the protection group. Once the targets have been added to the protection group, the source array must connect to each of the targets. To connect to a target array, run the command **`purearray connect`**. For more information about connecting arrays, refer to purepgroup-setattr. To connect to an S3 offload target, run the command **`pureoffload s3 connect`**. For more information about connecting to an NFS or S3 offload target, refer to pureoffload.

Once a protection group has been created with members and targets (for replication only), it is ready to generate and replicate snapshots.

Renaming Protection Groups

Run **`purepgroup rename`** to change the current name (OLD-NAME) of the protection group to the new name (NEW-NAME). The name change is effective immediately and the old name is no longer recognized in CLI, GUI, or REST interactions. In the Purity//FA GUI, the new name appears upon page refresh.

Protection group snapshots cannot be renamed.

Destroying Protection Groups

Protection groups can be destroyed, eradicated, and recovered by the respective **`purepgroup`** command.

The **`purepgroup destroy`** command destroys the specified protection group and, implicitly, all of its snapshots. Once a protection group has been destroyed, the replication process for the protection group stops. Destroyed protection groups and their snapshots undergo an eradication pending period after which time the protection groups and their snapshots are completely eradicated and unrecoverable. The length of eradication pending period is set by the enabled delay value if the protection group is not protected by SafeMode, or by the disabled delay value if the protection group is protected by SafeMode.

On connected arrays, use the **`--on`** option to destroy protection groups on an offload target. If the source and target arrays are disconnected, run the **`purepgroup destroy`** command on the target array.

Run **`purepgroup list --pending`** to display a list of protection groups and snapshots, including those that are pending eradication. Run **`purepgroup list --pending-only`** to only display a list of protection groups and their snapshots that are pending eradication.

During the eradication pending period, the destruction of the protection group and snapshots can be canceled by running **`purepgroup recover`**, returning the protection group, including its replication schedule and snapshots, to its original state. On connected arrays, use the **`--on`** option to recover destroyed protection groups on an offload target. If the source and target arrays are disconnected, run the **`purepgroup recover`** command on the target array.

During the eradication pending period, the destroyed protection group and its snapshots can be eradicated by running **`purepgroup eradicate`** (unless the SafeMode manual eradication prevention feature is enabled). This command terminates the eradication pending period and immediately begins reclamation of physical storage occupied by data "charged" to the snapshots of the destroyed protection group. Once eradication has begun, a protection group and its snapshots can no longer be recovered.

On connected arrays, use the `--on` option to eradicate protection groups or snapshots on an offload target. If the source and target arrays are disconnected, run the **`purepgroup eradicate`** command on the target array.

Generating Protection Group Snapshots On Demand

There are two ways to generate and replicate protection group snapshots:

- **Scheduled.** Schedule the protection group to automatically generate local snapshots and/or replicate snapshot data to its targets on a regular, pre-defined basis.

  Configure protection group schedules through the **`purepgroup schedule`** command. For steps on how to configure a protection group snapshot and replication schedule, refer to purepgroup-setattr.

  View protection group schedules through the **`purepgroup list --schedule`** command. For more information about viewing protection group attributes and schedules, refer to purepgroup-list.

- **On Demand.**Manually generate local protection group snapshots or replicate protection group snapshot data to its targets.

An on-demand snapshot represents a snapshot that is manually generated by running **`purepgroup snap`**. If the **`--replicate-now`** option is used, the on-demand snapshot is replicated immediately to the targets. Alternatively, if the **`--replicate`** option is used, the on-demand snapshot is queued for replication. Purity//FA will begin replicating data to each target only when all earlier replication sessions for the same protection group have been completed to that target, excluding those started with **`--replicate-now`**.

By default, an on-demand snapshot is retained indefinitely or until it is manually destroyed. Include the **`--apply-retention`** option to apply the scheduled snapshot or replication retention policy to the on-demand snapshot. For example, when generating a local snapshot on demand, include the **`--apply-retention`** option to keep the snapshot on the source array for the length of time based on the snapshot retention schedule. Likewise, when replicating a snapshot on demand, include the **`--apply-retention`** option to keep the snapshot on the target for the length of time based on the replication retention schedule.

The retention limits for protection group snapshots are as follows:

- The maximum amount of time to keep all snapshots is 2147483647 seconds.
- The maximum number of snapshots to keep per day is 2147483647 snapshots.

- The number of days to keep a set of snapshots after the initial retention period is 2147483647 days.

When an on-demand snapshot is replicated with the `--replicate` or `--replicate-now` options, and no retention policy is applied, the snapshot is retained on both the source and target arrays. However, when a retention policy is applied with the `--apply-retention` option, the snapshot will not be retained on the source after replication, although one snapshot may be kept as a baseline. Therefore, to keep the snapshot on the source after replication, with retention applied, take another on-demand snapshot without replication.

To replace the NNN snapshot number in the snapshot name with a custom string, include the `--suffix` option in the `purepgroup snap` command.

An array may become overloaded when taking an on-demand snapshot. Use the `purepgroup snap --allow-throttle` command to check the internal health of the array to determine if an on-demand snapshot is recommended at that point in time. This allows for throttling of snapshots to lessen the impact on the array performance.

Before taking an on-demand snapshot, use the `--dry-run` option to simulate taking a snapshot. The `purepgroup snap --dry-run` option will check the internal health of the array and return an error if unsuccessful or return successfully without creating a snapshot. The `--dry-run` option can be used simultaneously with any existing options, such as `--allow-throttle`.

Destroying Protection Group Snapshots

Destroy a protection group snapshot if it is no longer required. Run `purepgroup destroy` to destroy the specified protection group snapshot.

Destroying a protection group snapshot destroys all of its protection group volume snapshots, thereby reclaiming the physical storage space occupied by its data.

Destroyed protection group snapshots follow the same eradication pending behavior as destroyed protection groups. When a protection group snapshot is destroyed, Purity//FA automatically takes an undo snapshot. The undo snapshot enters an eradication pending period, after which time the snapshot is eradicated. During the eradication pending period, the undo snapshot can be viewed, recovered, or permanently eradicated. The length of eradication pending period is set by the enabled delay value if the protection group is not protected by SafeMode, or by the disabled delay value if the protection group is protected by SafeMode. (See "Eradication Delays" on page 88 in the `purearray` chapter for information on the disabled delay and enabled delay.)

Protection group volume snapshots cannot be destroyed individually. A protection group volume snapshot can only be destroyed by destroying the protection group snapshot to which it belongs.

Sending Protection Group Snapshots To Other Arrays

Send an existing protection group snapshot from a source array to one or more target arrays or user pods with the `purepgroup send` command.

A snapshot can be sent to all targets that have the allowed protection group, or it can be sent to a specific array using the `--to` option.

A snapshot can be sent from one target array or user pod to another using the `--to` option, presuming that snapshot is already on the array from which it is being sent. That snapshot could have been replicated from other arrays by any commands (e.g., `puregroup send`, `puregroup snap --replicate-now`) or replication schedule.

Protection group snapshots cannot be sent from an array to itself.

The `--convert-source-to-baseline` option will set the snapshot to be eradicated automatically when it is no longer the baseline on the source array. This option cannot be applied in the same line as the `--to` option.

Restoring Volumes from a Target Array

The `purepgroup copy` command restores the state of the volumes within a protection group that resides on a target array to a previous protection group snapshot. The restored volumes are added as real volumes to a new or existing protection group. To restore volume snapshots that reside on an offload target, run the `purevol get --on` command. For more information about restoring volumes from an offload target, refer to purevol.

The `purepgroup copy` command requires source (***SOURCE***) and destination (***DEST***) arguments.

The source argument represents the protection group snapshot to be restored. To restore a specific protection group snapshot, include the full protection group snapshot name in the source argument. For example, if source is specified as `pgroup1.5213`, Purity//FA restores the volumes from protection snapshot `pgroup1.5213`. To restore volumes from the most recent, fully generated protection group snapshot (either locally generated or replicated from another array), specify the protection group name only. For example, if source is specified as `pgroup1`, Purity//FA restores the volumes from the latest, fully generated protection group snapshot of `pgroup1`.

The source argument also determines whether the protection group snapshot to be restored was generated locally or replicated from another array. If the protection group snapshot was replicated from another array, include the name of the array in the source argument. For example, if source is specified as **`array1:pgroup1.5213`**, Purity//FA restores the volumes from protection group snapshot **`pgroup1.5213`**, which was replicated over from **`array1`**.

The destination argument represents the name of the protection group to where the volumes will be restored. If the destination protection group and all of its volumes already exist, include the **`--overwrite`** option to overwrite all of the existing volumes with the snapshot contents. When using the **`--overwrite`** option, the names of the volumes that are being overwritten must match the names of the volumes that are being restored.

The following restrictions apply to the **`purepgroup copy`** command:

- You cannot restore a volume if it already exists on the array unless you include the **`--overwrite`** option.
- You cannot overwrite destination protection groups that contain hosts, host groups, or connected volumes.
- You cannot run the **`purepgroup copy --overwrite`** command to create multiple clones of the protection group or its volume snapshots.

Restoring volumes from a protection group snapshot does not automatically expose the restored volumes to hosts and host groups. Run the **`purehgroup connect`**, **`purehost connect`** or **`purevol connect`** command to establish connections.

If the protection group snapshot being restored is one that was replicated from another array, include the target group in the **`purepgroup copy`** command to include the original source array as a target array in the new protection group.

After the volumes are restored, the created date of the volumes is set to the date of the **`purepgroup copy`** action.

Enabling SafeMode

SafeMode provides ransomware protection per protection group through retention lock. By ratchet enabling the retention lock, all of the following are disallowed for a non-empty protection group:

- Destroying the protection group
- Manual eradication of the protection group and its container
- Member and target removal
- Decreasing the disabled delay (thereby decreasing the eradication pending period for protected objects)

- Disabling snapshot or replication schedule
- Decreasing snapshot or replication retention or frequency
- Changing the blackout period, only clear blackout period is allowed
- Disallow on the target side

Once the protection group retention lock is ratcheted, it cannot be unlocked by the user. Contact Pure Storage Technical Services for further assistance. Enrollment is required with at least two administrators and pin codes.

Retention lock is not supported for and cannot be ratcheted on protection groups with host or host group members, or groups with offload targets. To use retention lock in one of these situations, create another protection group that includes the volumes to be protected, not including the offload targets, on which retention lock can be enabled. The same applies when trying to add host or host group members, or adding offload targets, to a protection group that is ratcheted.

To enable SafeMode for one or more protection groups, use the `purepgroup retention-lock ratchet` command. For example, the following command enables retention lock for the protection group named **`pgroup1`**. Retention Lock becomes ratcheted but Manual Eradication will be disabled only when the protection group is not empty.

```
$ purepgroup retention-lock ratchet pgroup1
Name    Retention Lock  Manual Eradication

pgroup1  ratcheted       disabled
```

The `purepgroup retention-lock list` command displays the SafeMode status for each protection group. Valid options include **`--pending`** which includes destroyed protection groups that are in the eradication pending state, or **`--pending-only`** which only displays destroyed protection groups that are in the eradication pending state. Manual Eradication is “disabled” if the non-empty protection group is protected with SafeMode retention lock, otherwise “enabled”.

Examples

Example 1

```
purepgroup create --vollist vol1,vol2,vol3 pgroup1
```

Creates protection group **`pgroup1`** and associates volumes **`vol1`**,**`vol2`**,**`vol3`** with it.

Example 2

```
purepgroup create --vollist vol5,vol6 --targetlist nfs-target pgroup2
```

Creates protection group **`pgroup2`** with volumes **`vol5`** and **`vol6`** and offload target **`nfs-target`**.

Example 3

```
purepgroup destroy pgroup12
```

... *less than the eradication pending period passes*...

```
purepgroup recover pgroup12
```

Destroys protection group **`pgroup12`** and its snapshots and then recovers the destroyed protection group and its snapshots.

Example 4

```
purepgroup destroy pgroup5.10
```

... *less than the eradication pending period passes*...

```
purepgroup eradicate pgroup5.10
```

Destroys protection group snapshot **`pgroup5.10`** and then eradicates the destroyed snapshot. Purity//FA immediately begins reclaiming the physical storage space occupied by the snapshot. Protection group snapshot **`pgroup5.10`** can no longer be recovered.

Destroy and manual eradication is not supported if the SafeMode manual eradication prevention feature is enabled.

Example 5

```
purepgroup snap --replicate-now pgroup10
```

Generates an on-demand snapshot of all volumes, hosts, or host groups in protection group **`pgroup10`** and replicates the snapshot to all of the **`pgroup10`** targets.

Example 6

```
purepgroup copy pgroup1 pgroup6
purevol connect --host host01 vol01 vol02
```

Restores the volumes from the most recent, fully and locally generated protection group snapshot of protection group **pgroup1** to a new protection group named **pgroup6** on the same array, and then connects newly restored volumes **vol01** and **vol02** to host **host01**.

Example 7

```
purepgroup copy array1:pgroup2.5213 pgroup25
```

Restores the volumes from protection group snapshot **pgroup2.5213**, which was replicated over from source array **array1**. The restored volumes are saved as real volumes to new protection group **pgroup25** on the current array. To replicate new protection group **pgroup25** and its restored volumes from the current array back to the original source array, enable replication and then allow replication from the source array.

Example 8

```
purepgroup copy --overwrite pgroup5.4765 pgroup5
```

Restores the volumes from locally generated protection group snapshot **pgroup1.123** to existing protection group **pgroup1**, overwriting all of the existing volumes with the snapshot contents.

Example 9

```
purepgroup list --snap
```

Lists all the snapshots on the array.

Example 10

```
purepgroup send snapshot.name
```

Sends the snapshot **snapshot.name** to every array in the allowed protection group.

Example 11

```
purepgroup send --to array1 snapshot.name
```

Sends the snapshot **snapshot.name** to the array **array1**.

Example 12

```
purepgroup --allow-throttle --dry-run pgroup2
```

Assesses the internal health of the array and simulates taking an on-demand snapshot of all volumes, hosts, or host groups in protection group **`pgroup2`** without generating one.

See Also

purepgroup-list, purepgroup-setattr, purehgroup, purehost, purevol, purepod

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purepgroup-list

purepgroup-list, purepgroup-listobj – display protection group attributes and storage consumption.

purepgroup-retention-lock-list – displays SafeMode status.

Synopsis

purepgroup list [--on ***REMOTE***] [--pending | --pending-only] [--retention | --retention-lock | --schedule | --space | --transfer] [--snap] [--sort ***SORT***] [--source | --target] [--total] [--transfer] [--cli | --csv | --nvp] [--filter ***FILTER***] [--limit ***LIMIT***] [--notitle] [--page] [--raw] [--workload ***WORKLOAD***] [--context ***REMOTES***] [***PGROUP***...]

purepgroup listobj [--pending | --pending-only] [--source | --target] [--type { --hgroup | --host | --pgroup | --snap | --source | --target | --vol }] [--csv] [***PGROUP***...]

purepgroup retention-lock list [--pending | --pending-only] [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--context ***REMOTES***] [***PGROUP***...]

Arguments

PGROUP

Protection group for which the information specified by options is to be displayed.

Options

Options that control information displayed:

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--context` ***`REMOTES`***

Used to specify the remote array or arrays on which a command is processed. ***`REMOTES`*** is a comma-separated list of names of arrays within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--on` ***`REMOTE`***

Displays a list of protection groups or protection group snapshots on an offload target.

`--pending`

Displays destroyed protection groups or snapshots that have been destroyed and are in the eradication pending state.

`--pending-only`

Only display destroyed protection groups or snapshots that are in the eradication pending state.

`--retention`

Displays the retention schedule for each protection group.

`--retention-lock`

Displays retention-lock status for the selected protection groups. Equal to the **`purepgroup retention-lock list`** command.

`--schedule`

Displays the snapshot and asynchronous replication schedule for each protection group.

`--snap`

Displays a list of protection group snapshots taken in each source and target protection group.

When used with **`--on`**, displays a list of protection group snapshots, both local and remote to the array, that have been replicated and retained on the offload target.

`--source`

Displays a list of protection groups that were created on this array. When used with **`--snap`**, displays a list of protection group snapshots that were created and retained on this array.

`--space`

Displays the size and space consumption or effective used capacity details for all snapshots in each protection group.

`--target`

Displays a list of protection groups or snapshots that were replicated to this array.

`--total`

Used with **`--space`** to display total physical space or effective used capacity occupied by snapshot data for each protection group.

`--transfer`

Used with **`--snap`** to display asynchronous replication data transfer statistics, including data transfer start time, data transfer end time, data transfer progress, and amount of logical/physical data transferred.

`--type`

Displays a list of hosts, host groups, protection groups, snapshots, sources, targets or volumes associated with the specified protection groups.

`--workload` ***`WORKLOAD`***

The name of the workload. Include the **`--workload`** option to display the workload the protection group is in. Include the **`WORKLOAD`** name to display the protection groups that are part of the specified workload.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Conventions

The protection group snapshot naming convention is **`PGROUP.NNN`**, where:

- **`PGROUP`** is the name of the protection group.
- **`NNN`** is a unique monotonically increasing number or a manually-assigned protection group snapshot suffix name.

The protection group volume snapshot naming convention is **`PGROUP.NNN.VOL`**, where:

- **`PGROUP`** is the name of the protection group.
- **`NNN`** is a unique monotonically increasing number or a manually-assigned protection group snapshot suffix name.
- **`VOL`** is name of the volume member.

If you are viewing replicated snapshots on a target array, the snapshot name begins with the name of the source array from where the snapshot was taken.

Description

The **`purepgroup list`** command displays information about each protection group, including their associated source arrays, replication targets, hosts, host groups, user pod targets, and volumes. Optionally add the protection group argument to any of the commands to view details for the specified protection groups. The list includes protection groups that were created on the current array to replicate snapshot data to other arrays or offload targets, created on a remote array and replicated asynchronously to this array, or created inside a pod on a remote array and stretched to the current array.

The following **`purepgroup list`** sample output generated on array **`array1`** illustrates the different ways in which protection groups can be listed:

- Protection group **`pgroup05`** was created on array **`array5`**, and the current array was added to the protection group as a target array for asynchronous replication.
- Protection group **`pgroup1`** was created on the current array, and **`array2`** was added as a target array for asynchronous replication.
- Protection group **`pgroup10`** was created on the current array and **`array3`** was added as a target array for asynchronous replication. However, **`array3`** has disallowed asynchronous replication.
- Protection group **`pgroup20`** was created on the current array, and **`nfs-target`** was added as an offload target.
- Protection group **`pod05::pgroup01`** is on the current array. The protection group is in pod **`pod05`**, which was either created on the current array or created on another array and stretched to this array. The pod may or may not be stretched across another array.
- Protection group **`pod05::pgroup02`** is on the current array. The protection group is in pod **`pod05`**, which was either created on the current array or created on another array and stretched to this array. The pod may or may not be stretched across another array. Array **`array4`** was added to the protection group as a target array for asynchronous replication. If the pod is stretched across another array and both arrays in the stretched pod are connected to the target array, both arrays will share the load of replicating data to the target array.
- Protection group **`pod25::pgroup100`** is on a remote array. The current array and a pod on the current array were added to the protection group as targets for asynchronous replication. On the target array, the fully qualified name of protection groups inside pods is **`SOURCE-POD:PGROUP`** for targeted arrays and **`TARGETPOD::SOURCE-POD:PGROUP`** for targeted pods. Single colons (**`:`**) separate the source pod name and volume name. The protection group is inside **`pod25`**, which may or may not be stretched across another array. If the pod is stretched across another array and both arrays in the stretched pod are connected to the target array,

both arrays will share the load of replicating data to the current (target) array.

- Protection group **`pod06::pgroup3`** is on the current array. The protection group is in pod **`pod06`**, which was either created on the current array or created on another array and stretched to this array. The pod may or may not be stretched across another array. **`pod26`** was added as a target user pod for asynchronous replication.

```
//From array1
$ purepgroup list
Name                   Source  Targets              Host Groups  Hosts  Volumes
array5:pgroup05        array5  -                    -            -      array:vol1
pgroup1                array1  array2               -            -      v1
                                                                        v2
                                                                        v3
pgroup10               array1  array3 (disallowed)  hg1          -      -
pgroup20               array1  nfs-target           -            -      v17
                                                                        v18
pod05::pgroup01        pod05   -                    -            -      v60
pod05::pgroup02        pod05   array4               -            -      v50
                                                                 -      v51
                                                                 -      v52
pod25:pgroup100        pod25   array1               -            -      v85
                                                    -            -      v92
pod06::pgroup03        pod06   pod26                -            -      pod06:vol1
pod07:pod25:pgroup100  pod25   pod07                -            -      pod25:v85
```

The **`purepgroup list`** command includes the mutually exclusive **`--schedule`**, **`--retention`**, and **`--space`** options.

The **`--schedule`** option displays the snapshot and asynchronous replication schedule for each protection group.

In the following sample output, the snapshot and asynchronous replication schedules have been enabled for protection group **`pgroup1`**. The protection group has been configured to enable snapshot generation every hour and replication every 2 hours. Replication is suspended during the blackout period between 9am and 5pm. Replication from the source to targets will not occur during the blackout period, but hourly snapshots will continue to be taken on the source array.

The snapshot schedule for protection group **pgroup05** on source array **array5** has also been configured to enable snapshot generation every hour. However, replication has been disabled. Once enabled, snapshot replication will occur every 2 days at or soon after 6pm.

```
$ purepgroup list --schedule
Name              Schedule    Enabled    Frequency    At    Blackout
array5:pgroup05   snap        True       6h           -     -
                   replicate   False      2d           6pm   -
pgroup1           snap        True       1h           -     -
replicate   True       2h           6pm   9am-5pm
```

The **--retention** option displays the retention schedule for each protection group.

In the following sample output, the retention schedule has been set to keep all snapshots on the source array for 1 day. After that, Purity//FA will keep 2 of the snapshots for an additional 7 days and eradicate the others. Seven days later, Purity//FA will eradicate the 2 snapshots.

The retention schedule has also been set to keep all replicated snapshots on the targets for 7 days. After that, Purity//FA will keep 4 of the replicated snapshots for an additional 7 days and eradicate the others. Seven days later, Purity//FA will eradicate the 4 snapshots.

```
$ purepgroup list --retention
Name      Array    All For    Per Day    Days
pgroup1   source   1d         2          7
target   7d         4          7
```

The **--space** option displays the size and space consumption or effective used capacity details for all snapshots in each protection group. Include the **--total** option to display the total space consumption or effective used capacity for all protection groups.

The **--snap** option displays a list of scheduled and on-demand snapshots generated or replicated of each protection group. The list also displays the snapshot creation date. Include the –**-transfer** option to display asynchronous replication data transfer statistics, including data transfer start time, data transfer end time, data transfer progress, and amount of logical/physical data transferred.

Include the **--on** option to display a list of protection groups both local and remote to the array that are connected to the offload target. Add the **--snap** option to display a list of protection group snapshots, both local and remote to the array, that have been replicated and retained on the offload target.

The **`--pending`** option includes protection groups or snapshots that have been destroyed and are in the eradication pending state in the output list. The **`--pending`** option can also be used in conjunction with the **`--schedule`**, **`--retention`**, **`--space`**, or **`--snap`** options to include a list of pending protection groups or snapshots.

The **`--pending-only`** option only displays protection groups or snapshots that have been destroyed and are in the eradication pending state in the output list. The **`--pending-only`** option can also be used in conjunction with the **`--schedule`**, **`--retention`**, **`--space`**, or **`--snap`** options to include a list of pending protection groups or snapshots.

The **`purepgroup listobj`** command displays a list of protection groups. Include the protection group argument to view the list for the specified protection group.

Include the **`--type`** option to view a list of hosts, host groups, snapshots, sources, targets or volumes associated with a protection group. The **`--type`** option produces the following types of lists:

`--type hgroup`

Displays a list of host groups that belong to a protection group.

`--type host`

Displays a list of hosts that belong to a protection group.

`--type pgroup (default if the` **`--type`** `option not specified)`

Displays a list of protection groups.

`--type snap`

Displays a list of snapshots that were taken for a protection group.

`--type source`

Displays a list of replication source arrays.

`--type target`

Displays a list of replication target arrays and offload targets.

`--type vol`

Displays a list of volumes that belong to a protection group.

The **`--source`** option displays a list of protection groups or snapshots that were created on this array.

The **`--target`** option displays a list of protection groups or snapshots that were replicated to this array from another array.

Include the **`--workload`** option to display the protection groups that are part of a workload.

Exceptions

None.

Examples

Example 1

```
purepgroup list
```

Displays information about each protection group, including their source arrays, targets, hosts, host groups, and volumes.

Example 2

```
purepgroup list --pending --snap
```

Displays a list of snapshots, both on-demand and scheduled, including those which have been destroyed and are in the eradication pending state.

Example 3

```
purepgroup list --pending-only --snap
```

Displays a list of snapshots, both on-demand and scheduled, that have been destroyed and are in the eradication pending state.

Example 4

```
purepgroup list --schedule pgroup4
```

Displays the snapshot and asynchronous replication schedule for protection group **`pgroup4`**.

Example 5

```
purepgroup list --on offload-target
```

Displays a list of protection groups both local and remote to the array that are connected to off-load target **`offload-target`**.

Example 6

```
purepgroup list --on s3-target --snap
```

Displays a list of snapshots, both local and remote to the array, that have been replicated and retained on offload target **`s3-target`**.

Example 7

```
purepgroup listobj
```

Displays a list of protection groups.

Example 8

```
purepgroup listobj --type vol pgroup7 pgroup10
```

Displays volumes that belong to protection group **`pgroup7`**, **`pgroup10`**, or both.

Example 9

```
purepgroup listobj --type snap --target
```

Displays protection group snapshots replicated to this array.

See Also

pureoffload, purepgroup, purepgroup-setattr, purepod

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purepgroup-setattr

purepgroup-allow, purepgroup-disable, purepgroup-disallow, purepgroup-enable, purepgroup-monitor, purepgroup-schedule, purepgroup-setattr – manage the attributes of the Purity//FA protection group (pgroup).

Synopsis

purepgroup allow [--context ***REMOTE***] ***PGROUP***...

purepgroup disable [--replicate] [--snap] [--context ***REMOTE***] ***PGROUP***...

purepgroup disallow [--context ***REMOTE***] ***PGROUP***...

purepgroup enable [--replicate] [--snap] [--context ***REMOTE***] ***PGROUP***...

purepgroup monitor [--csv] [--notitle] [--raw] --replication [--array] [--historical { 1h | 1y | 24h | 30d | 3h | 7d | 90d }] [**PGROUP...**]

purepgroup retain [--all-for ***PERIOD***] [--per-day ***SNAPS-PER-DAY***] [--days ***COUNT***] [--target-all-for ***PERIOD***] [--target-per-day ***SNAPS-PER-DAY***] [--target-days ***COUNT***] [--context ***REMOTE***] ***PGROUP***...

purepgroup schedule [--snap-frequency ***FREQUENCY***] [--snap-at ***TIME***] [--replicate-frequency ***FREQUENCY***] [--replicate-at ***TIME***] [--replicate-blackout ***WINDOW***] [--context ***REMOTE***] ***PGROUP***...

purepgroup setattr [--addhgrouplist ***HGROUPS***] [--hgrouplist ***HGROUPS***] [--remhgrouplist ***HGROUPS***] [--addhostlist ***HOSTS***] [--hostlist ***HOSTS***] [--remhostlist ***HOSTS***] [--addtargetlist ***TARGETS***] [--targetlist ***TARGETS***] [--remtargetlist ***TARGETS***] [--addvollist ***VOLS***] [--vollist ***VOLS***] [--remvollist ***VOLS***] [--context ***REMOTE***] ***PGROUP***...

Arguments

PGROUP

Protection group name. For `purepgroup allow` and `purepgroup disallow`, array name and protection group name of the source array and protection group from which to allow or disallow asynchronous replication.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--addhgrouplist`

Comma-separated list of one or more additional host groups to be included in the specified protection groups. Has no effect on host groups already associated with the group. Hosts and host groups are not supported for replication to offload targets.

`--addhostlist`

Comma-separated list of one or more additional hosts to be included in the specified protection groups. Has no effect on hosts already associated with the group. Hosts and host groups are not supported for replication to offload targets.

`--addtargetlist`

Comma-separated list of one or more additional targets to be included in the specified protection groups. Has no effect on targets already associated with the group.

`--addvollist`

Comma-separated list of one or more additional volumes to be included in the specified protection groups. Has no effect on volumes already associated with the group.

`--all-for`

Length of time to keep the snapshots on the source array before they are eradicated.

`--array`

Breaks down perfomance data by the array to which the I/O is directed.

`--context` ***REMOTE***

Used to specify the remote array on which a command is processed. ***REMOTE*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--days`

Number of days to keep the **`--per-day`** snapshots beyond the **`--all-for`** period before they are eradicated.

`--hgrouplist`

Comma-separated list of one or more host groups to be included in the specified protection groups, replacing any host groups that currently belong to the protection group. Hosts and host groups are not supported for replication to offload targets.

`--historical`

Displays historical performance data over the specified range of time. Valid time ranges include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year

`--hostlist`

Comma-separated list of one or more hosts to be included in the specified protection groups, replacing any hosts that currently belong to the protection group. Hosts and host groups are not supported for replication to offload targets.

`--per-day`

Number of per-day snapshots to keep beyond the **`--all-for`** period.

`--remhgrouplist`

Comma-separated list of one or more host groups to be removed from the specified protection group.

`--remhostlist`

Comma-separated list of one or more hosts to be removed from the specified protection group.

`--remtargetlist`

Comma-separated list of one or more targets whose associations with the specified protection group are to be removed. Removing a target array from a protection group immediately deletes all of the data on the target array that was sent from its source array. As a result, replicating data again from the source array to the target array would cause another baseline transfer. This action cannot be performed on a protection group that is ratcheted.

`--remvollist`

Comma-separated list of one or more volumes to be removed from the specified protection group.

`--replicate`

Enables or disables replication.

`--replicate-at`

Preferred time, on the hour, at which to replicate the snapshots. To clear the preferred time, set to an empty string (`""`).

`--replicate-blackout`

Range of time at which to suspend replication. To clear the blackout period, set to an empty string (`""`).

`--replicate-frequency`

Replication frequency.

`--replication`

Displays replication performance. When the **`--array`** option is specified, the display breaks down performance data by the array to which the I/O is directed.

`--snap`

Enables or disables snapshot creation on the source array.

`--snap-at`

Preferred time, on the hour, at which to generate the snapshot. To clear the preferred time, set to an empty string (`""`).

`--snap-frequency`

Snapshot frequency.

`--target-all-for`

Length of time to keep the replicated snapshots on the targets.

`--target-days`

Number of days to keep the **`--target-per-day`** snapshots beyond the **`--target-all-for`** period before they are eradicated.

`--target-per-day`

Number of per-day snapshots to keep beyond the **`--target-all-for`** period.

`--targetlist`

Comma-separated list of one or more targets to be associated with the protection group. When specified in the purepgroup setattr command, replaces the existing targets of the protection group.

`--vollist`

Comma-separated list of one or more volumes to be included in the specified protection groups, replacing any volumes that currently belong to the protection group.

Conventions

The protection group snapshot naming convention is `PGROUP.NNN`, where:

- `PGROUP` is the name of the protection group.
- `NNN` is a unique monotonically increasing number or a manually-assigned protection group snapshot suffix name.

The protection group volume snapshot naming convention is `PGROUP.NNN.VOL`, where:

- `PGROUP` is the name of the protection group.
- `NNN` is a unique monotonically increasing number or a manually-assigned protection group snapshot suffix name.
- `VOL` is name of the volume member.

If you are viewing replicated snapshots on a target array, the snapshot name begins with the name of the source array from where the snapshot was taken.

Description

After a protection group has been created, configure the schedules to generate and replicate snapshots to another FlashArray array or to an external storage system. For more information about creating protection groups, refer to purepgroup.

To configure the protection group snapshot schedule:

1. Define the snapshot schedule.
2. Define the retention schedule.
3. Enable protection group snapshots.

To configure the protection group replication schedule:

1. Connect the source and targets.

   To connect the source array to a target array, run the **`purearray connect`** command. If the protection group is in a stretched pod, for high availability, connect both arrays in the stretched pod to the target array. For more information about array connections, refer to purearray.

   To connect the source array to a storage system, such as an S3 bucket, run the **`pureoffload s3 connect`** command. If the protection group is in a stretched pod, for high availability, connect both arrays in the stretched pod to the offload target. For more information about offload targets, refer to pureoffload.

2. For asynchronous replication, verify that each target array has allowed replication.
3. Define the replication schedule.
4. Define the retention schedule.
5. Enable protection group replication.

The snapshot and replication schedules can be enabled and disabled at any time.

Note: Protection group snapshots can also be generated and replicated on demand. For more information about generating on-demand snapshots, refer to "purepgroup" on page 327.

Connecting Arrays for Asynchronous Replication

Replication between a source and target array can only occur if the arrays are connected and the target array has allowed replication.

To connect a source array to a target array, log in to the target array and run the **`purearray connect --connection-key`** command to obtain a connection key, and then log in to the source array and run the **`purearray connect`** command to connect the target array to the source array. A source array can connect to multiple target arrays. For more information about connecting arrays, refer to purearray.

Once two arrays are connected, run the **`purepgroup list`** command on either array to verify that the target array has allowed data to be replicated to it. If an array has disallowed replication, **`(disallowed)`** appears next to its name. To allow replication, run the **`purepgroup allow`** command on the target array.

The **`purepgroup disallow`** command stops replication to the target array. Allowing and disallowing replication on a target array will not impact the replication process between the source array and other target arrays. If you disallow replication while a replication session is in progress, Purity//FA will wait until the session is complete and then stop any new replication sessions from being created.

When specifying the **`purepgroup allow`** and **`purepgroup disallow`** commands, include the source array name and protection group name in the argument, separating the two names with a colon (:).

In the following example, the array is allowing data to be replicated from protection group **`PGROUP1`** on array **`ARRAY1`**.

```
purepgroup allow ARRAY1:PGROUP1
```

Defining the Protection Group Snapshot and Replication Schedules

Once a protection group has been created and connections between the source and targets have been established (for replication only), define the protection group snapshot and replication schedules.

Each protection group includes two schedules:

- **Snapshot Schedule.**

  Defines when and how frequently snapshots are taken and saved on the local (source) array. Run **`purepgroup schedule --snap-frequency`** to configure the snapshot schedule.
- **Replication Schedule.**

  Defines when and how frequently snapshots are taken and immediately replicated to its targets. Run **`purepgroup schedule --replicate-frequency`** to configure the replication schedule.

The snapshot schedule is independent of the replication schedule, meaning you can enable one schedule without enabling the other. You can also enable or disable both schedules at any time. The schedules are by default disabled.

The **`purepgroup schedule`** command configures the snapshot and replication schedules for the protection group. By default, the protection group schedules are configured to generate snapshots every hour and replicate snapshots to the targets every 4 hours.

Snapshot Schedule

The **`purepgroup schedule --snap-frequency`** command specifies how frequently Purity//FA generates snapshots on the source array.

The frequency is specified as an integer, followed by one of the suffix letters **`s`** (seconds), **`m`** (minutes), **`h`** (hours), **`d`** (days), or **`w`** (weeks).

For example, run the following command to generate snapshots every 2 hours for protection group **pgroup1**.

```
purepgroup schedule --snap-frequency 2h pgroup1
```

If the **--snap-frequency** option is set to one or more days, set the **--snap-at** option to specify the preferred time, on the hour, at which to generate the snapshot on the source array. If the preferred time is set and you want to cancel it, meaning that you no longer want to specify a preferred time, set the **--snap-at** option to an empty string (**""**).

The time can be specified in 12-hour or 24-hour format. To specify the time in 12-hour format, enter the time integer followed by the suffix **am** or **pm**.

For example, run the following command to set the preferred snapshot time for protection group **pgroup3** to 6:00pm. A snapshot will be generated every 2 days at or soon after 6pm.

```
purepgroup schedule --snap-frequency 2d --snap-at 18 pgroup3
```

Replication Schedule

The **purepgroup schedule --replicate-frequency** command specifies how frequently Purity//FA replicates the snapshots from the source array to the targets.

The frequency is specified as an integer, followed by one of the suffix letters **s** (seconds), **m** (minutes), **h** (hours), **d** (days), or **w** (weeks).

For example, run the following command to replicate snapshots to the targets every 6 hours for protection group **pgroup2**.

```
purepgroup schedule --replication-frequency 6h pgroup2
```

If the **--replicate-frequency** option is set to one or more days, set the **--replicate-at** option to specify the preferred time, on the hour, at which to replicate the snapshots to the targets. If the preferred time is set and you want to cancel it, meaning that you no longer want to specify a preferred time, set the **--replicate-at** option to an empty string (**""**).

The time can be specified in 12-hour or 24-hour format. To specify the time in 12-hour format, enter the time integer followed by the suffix **am** or **pm**.

For example, run the following command to set the preferred replication time for protection group **pgroup3** to 6:00pm. Snapshot replication will occur every 2 days at or soon after 6pm.

```
purepgroup schedule --replicate-frequency 2d --replicate-at 18 pgroup3
```

The **--replicate-blackout** option (blackout period) specifies the range of time in which to suspend replication. Replication from the source to targets will not occur during the blackout period.

The start and end times for the blackout window must be set on the hour. The time can be specified in 12-hour or 24-hour format. To specify the time in 12-hour format, enter the time integer followed by the suffix **am** or **pm**.

For example, run the following command to suspend replication between 9:00am and 5:00pm for protection group **pgroup4**.

```
purepgroup schedule --replicate-blackout 9am-5pm pgroup4
```

Blackout periods only apply to scheduled replications. On-demand replications (**purepgroup snap --replicate-now**) do not observe the blackout period.

If the blackout period is set and you want to cancel it, meaning that you no longer want to specify a blackout period, set the **--replicate-blackout** option to an empty string (**""**).

Defining the Protection Group Retention Schedule

The **purepgroup retain** command configures the snapshot retention schedule for the specified protection group.

By default, protection groups are scheduled to retain all snapshots for 1 day. After that, Purity//FA keeps 4 snapshots for an additional 7 days and eradicates the others. Seven days later, Purity//FA eradicates the 4 snapshots.

The following sample retention schedule output reflects the protection group schedule with the default schedule settings:

```
$ purepgroup list --retention
Name      Array   All For   Per Day   Days
pgroup1   source  1d        4         7
          target  1d        4         7
```

The **--all-for** option specifies the length of time to keep the snapshots on the source array before they are eradicated. The length of time is specified as an integer, followed by one of the suffix letters **s** (seconds), **m** (minutes), **h** (hours), **d** (days), or **w** (weeks).

For example, run the following command to keep all snapshots on the source array for 5 days.

```
purepgroup retain --all-for 5d pgroup5
```

The **`--per-day`** option specifies the number of per-day snapshots to keep beyond the **`--all-for`** period, while the **`--days`** option specifies the number of days to keep the **`--per-day`** snapshots beyond the **`--all-for`** period. After the **`--days`** period, the snapshots are eradicated.

For example, run the following command to keep 8 snapshots per day for an additional 5 days on the source array *after* the **`--all-for`** period.

```
purepgroup retain --per-day 8 --days 5 pgroup6
```

The **`--target-all-for`** option specifies the length of time to keep the replicated snapshots on the targets before they are eradicated.

The length of time is specified as an integer, followed by one of the suffix letters **`s`** (seconds), **`m`** (minutes), **`h`** (hours), **`d`** (days), or **`w`** (weeks).

For example, run the following command to keep all replicated snapshots on the targets for 7 days.

```
purepgroup retain --target-all-for 7d pgroup7
```

The **`--target-per-day`** option specifies the number of per-day snapshots to keep beyond the **`--target-all-for`** period, while the **`--target-days`** option specifies the number of days to keep the **`--target-per-day`** snapshots beyond the **`--target-all-for`** period. After the **`--target-days`** period, the replicated snapshots are eradicated.

For example, run the following command to keep 4 replicated snapshots per day for an additional 3 days on the target *after* the **`--target-all-for`** period.

```
purepgroup retain --target-per-day 4 --target-days 3 pgroup8
```

Enabling Protection Group Snapshots and Replication

Once the protection group snapshot and/or replication schedules have been defined, run the **`purepgroup enable`** command to enable snapshots and replications on the source array for the specified protection groups. Specify the **`--snap`** option to enable snapshots only. Specify the **`--replicate`** option to enable replication only. If the command is run without either option, Purity//FA enables both snapshots and replication.

Once you enable the snapshot schedule, Purity//FA immediately starts the snapshot process, with the following exception:

- If you are enabling the snapshot schedule and the **`--snap-at`** time is specified, Purity//FA starts the snapshot process at the specified **`--snap-at`** time.

Once you enable the replication schedule, Purity//FA immediately starts the replication process, with the following exceptions:

- If you are enabling the replication schedule during the blackout period, Purity//FA waits for the blackout period to end before it begins the snapshot and/or replication process.
- and the **`--replicate-at`** time is specified, If you are enabling the replication schedule Purity//FA starts the replication process at the specified **`--replicate-at`** time.

The **`purepgroup disable`** command disables snapshots and replication for the specified protection group. Specify the **`--snap`** option to disable snapshots only. Specify the **`--replicate`** option to disable replication only. If the command is run without either option, Purity//FA disables both snapshots and replication.

Space Consumption Considerations

Consider space consumption when you configure the snapshot, replication, and retention schedules. The amount of space consumed on the source array depends on how many snapshots you want to generate, how frequently you want to generate the snapshots, how many snapshots you want to retain, and how long you want to retain the snapshots.

Likewise, the amount of space consumed on the target depends how many snapshots you want to replicate, how frequently you want to replicate the snapshots, how many replicated snapshots you want to retain, and how long you want to retain them.

In the following sample snapshot, replication, and retention schedules, the settings in Scenario 1 will consume more space than the settings in Scenario 2:

```
SCENARIO 1
$ purepgroup list --schedule
Name          Schedule    Enabled   Frequency   At    Blackout
array1:pg1    snap        False     15m         -     -
              replicate   False     1h          6pm   -

$ purepgroup list --retention
Name       Array    All For    Per Day    Days
pgroup1    source   2w         24         30
           target   2w         24         30
```

```
SCENARIO 2
$ purepgroup list --schedule
Name          Schedule     Enabled    Frequency    At    Blackout
array1:pg1    snap         False      1h           -     -
              replicate    False      4h           6pm   -

$ purepgroup list --retention
Name       Array    All For    Per Day    Days
pgroup1    source   1d         4          7
           target   1d         4          7
```

Adding and Removing Protection Group Members

The `purepgroup setattr` command adds members to a protection group or removes members from a protection group.

For asynchronous replication, only members of the same object type can belong to a protection group. For example, hosts or host groups cannot be added to a protection group that contains volumes. The **`--addhgrouplist`**, **`--addhostlist`**, and **`--addvollist`** options add one or more members to a protection group. Existing members in the protection group are unaffected. The **`--hgrouplist`**, **`--hostlist`**, and **`--vollist`** options add one or more members to a protection group, replacing all members that are currently in the protection group. The **`--remhgrouplist`**, **`--remhostlist`**, and **`--remvollist`** options remove members from a protection group.

For replication to offload targets, only volume members can be added to protection groups. The **`--addvollist`** option adds one or more volume members to a protection group. Existing members in the protection group are unaffected. The **`--vollist`** option adds one or more members to a protection group, replacing all volume members that are currently in the protection group. The **`--remvollist`** option removes volume members from a protection group.

Note that an alternate (and preferred!) method to add and remove protection group members is through the **`add`** and **`remove`** subcommands. Run the **`add`** subcommand with the respective `purehgroup`, `purehost`, or `purevol` command to add one or more members to one or more protection groups. For example, run **`purehgroup add --pgroup PGROUP HGROUP`** to add host groups to protection groups. Run the **`remove`** subcommand with the respective

purehgroup, **purehost**, or **purevol** command to remove one or more members from one or more protection groups For example, run **purevol remove --pgroup PGROUP VOL** to remove volumes from protection groups.

Adding and Removing Targets

The **purepgroup setattr** command adds target arrays and target user pods to protection groups, and removes target arrays and user pods from protection groups.

The **--addtargetlist** option adds one or more target arrays or user pods to a protection group. Existing target arrays in the protection group are unaffected. Only arrays that are connected to the current array can be added to a protection group as target arrays. Array connections are created through the **purearray connect** command.

If you add a target array to a protection group that is in a stretched pod, for high availability, make sure the peer array of the stretched pod is also connected to the target array. If only one of the arrays is connected to the target array, Purity//FA generates an alert notifying users of this misconfiguration.

The **--targetlist** option adds one or more target arrays or target user pods to a protection group, replacing all target arrays that are currently in the protection group.

The **--remtargetlist** option removes target arrays and user pods from a protection group. Removing a target array from a protection group immediately deletes all of the data on the target array that was sent from its source array. As a result, replicating data again from the source array to the target array or user pod would cause another baseline transfer. This action cannot be performed on a protection group that is ratcheted.

Monitoring Protection Groups

By default, the **purepgroup monitor** command displays real-time performance data. The **purepgroup monitor --historical** command displays historical performance data over any of the following ranges of time: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, or 1 year. Use the **--replication** option to display replication performance. When the **--array** option is specified, the display breaks down performance data by the array to which the I/O is directed.

Examples

Example 1

```
purepgroup enable pgroup4
```

(Run from the source array)...

Enables snapshots and replication for protection group **`pgroup4`**.

Example 2

```
purepgroup allow array1:pgroup4
```

(Run from the target array)...

Allows replication from source array **`array1`** to the target array for protection group **`pgroup4`**.

Example 3

```
purepgroup schedule --snap --hourly 3 pgroup2
```

Of the snapshots generated on the **`pgroup2`** source array based on the snapshot frequency, keeps the 3 most recent hourly snapshots.

Example 4

```
purepgroup schedule --replicate --blackout 9:00am-5:00pm pgroup10
```

Stops replicating snapshots from the source array to all targets between 9:00am and 5:00pm every day for protection group **`pgroup10`**.

Example 5

```
purepgroup setattr --addhostlist host4,host7,host10 pgroup42
```

Adds hosts **`host4`**, **`host7`**, and **`host10`** to protection group **`pgroup42`**. The existing hosts in pgroup42 are not affected.

Example 6

```
purepgroup monitor --replication
```

Displays replication performance.

See Also

pureoffload, purepgroup, purepgroup-list, purepod

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

pureplugin

pureplugin, pureplugin-check, pureplugin-list, pureplugin-uninstall – manages the Pure Storage FlashArray plugins.

Synopsis

```
pureplugin check --host HOST --user USER PLUGIN
pureplugin list [--csv | --nvp] [--notitle] [--page] [--raw]
[PLUGIN...]
pureplugin uninstall --host HOST --user USER PLUGIN
```

Arguments

PLUGIN

Pure Storage FlashArray plugin name.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--host`

The target host on which to install the plugin.

`--user`

User name with which to log in to the target host.

Options that control display format:

`--csv`
: Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`
: Lists information without column titles.

`--nvp`
: Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`
: Turns on interactive paging.

`--raw`
: Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Description

The **`pureplugin check`** command checks the plugin version that is currently installed on the target host. Values must be specified for the **`--host`** and **`--user`** options. If the plugin is not installed, the version displays a dash.

The **`pureplugin list`** command displays a list of plugins on the array that can be installed on the target host. Use **`--csv`** to display the output as a comma-separated list of values. Use **`--nvp`** to display the output as name-value pairs. Use **`--notitle`** to display the output without column titles. To display a list of specific plugins, type the list of plugin names at the end of the command.

The **`pureplugin uninstall`** command uninstalls the specified plugin. Values must be specified for the **`--host`** and **`--user`** options.

Please note that the pureplugin command has been deprecated, and will be removed in future releases. See the Deprecated Commands chapter for more details.

Examples

Example 1

```
pureplugin check --host vCenterHost --user vCenterAdmin vsphere
```

Checks the vSphere plugin version that is currently installed on vCenterHost.

Example 2

```
pureplugin list vsphere
```

Displays the available version of the vSphere plugin that can be installed on vCenter.

Example 3

```
pureplugin uninstall --host vCenterHost --user vCenterAdmin vsphere
```

Uninstalls the vSphere plugin.

See Also

purearray

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purepod

purepod, purepod-add, purepod-clone, purepod-create, purepod-demote, purepod-destroy, purepod-eradicate, purepod-move, purepod-promote, purepod-recover, purepod-remove, purepod-rename, purepod-replica-link, purepod-setattr – manage the Purity//FA pod

purepod-demote, purepod-promote – manage the promotion status of the Purity//FA pod

purepod-eradication-config-list – displays SafeMode configuration status

purepod-list, purepod-listobj – display pod attributes and storage consumption

purepod-monitor – monitors pod I/O performance and replication bandwidth

purepod-replica-link – manages replica links for ActiveDR replication

Synopsis

purepod add --array ***ARRAYLIST*** [--context ***REMOTE***] ***POD***

purepod clone [--allow-throttle] [--dry-run] [--context ***REMOTE***] ***SOURCE TARGET***

purepod create [--failover-preference ***ARRAY-LIST***] [--quota-limit ***SIZE***] [--context ***REMOTE***] ***POD...***

purepod demote [--quiesce | --skip-quiesce] [--context ***REMOTE***] ***POD...***

purepod destroy [--destroy-contents] [--context ***REMOTE***] ***POD...***

purepod eradicate [--eradicate-contents] [--context ***REMOTE***] ***POD...***

purepod eradication-config list [--pending | --pending-only] [--csv | --nvp] [--notitle] [--raw] [--page] [--page-with-token] [--token ***TOKEN***] [***POD...***]

purepod list [--failover-preference] | --footprint | --mediator | --on ***REMOTE*** |--space [--historical {1h | 3h | 24h | 7d | 30d | 90d | 1y}] [--pending | --pending-only] [--total] [--cli | --csv | --nvp] [--notitle] [--raw] [--filter ***FILTER***] [--page] [--page-with-token] [--token ***TOKEN***] [--limit ***LIMIT***] [--sort ***SORT***] [--context ***REMOTE***] [***POD...***]

purepod listobj [--pending | --pending-only] [--type { array | pgroup | pod | vol }] [--csv] [***POD...***]

purepod monitor[--csv] [--notitle] [--raw] [--filter ***FILTER***] [--page] [--limit LIMIT] [--sort SORT] [--mirrored] [--array] [--latency] [--replication] [--total] [--total-only] [--continuous | --resync |--sync | --periodic] [--protocol {nfs, smb} | --protocol-group {block, file}] [--interval INTERVAL] [--repeat REPEAT] [--size] [--historical { 1h | 3h | 24h | 7d | 30d | 90d | 1y }] [***POD...***]

purepod move --from ***FROM_MEMBER_NAMES*** --to ***FROM_MEMBER_NAMES*** [--with-hosts ***HOST_NAMES***] [--with-hgroups ***HGROUP_NAMES***] [--context ***REMOTE***] ***POD...***

purepod promote [--abort-quiesce] [--promote-from ***PROMOTE-FROM***] ***POD***

purepod recover [--context ***REMOTE***] ***POD...***

purepod remove --array ***ARRAYLIST*** [--with-unknown] ***POD***

purepod rename [--context ***REMOTE***] ***OLD-NAME NEW-NAME***

purepod replica-link {create | pause | resume} --remote-pod ***REMOTE-POD*** [--remote ***REMOTE***] [--context ***REMOTE***] ***POD***

purepod replica-link delete --remote-pod ***REMOTE-POD*** [--context ***REMOTE***] ***POD***

purepod replica-link list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--lag] [--historical {1h | 3h | 24h | 7d | 30d | 90d | 1y}] [--context ***REMOTES***] [***POD...***]

purepod replica-link mapping policy connect --remote-policy ***REMOTE_POLICY*** [--context ***REMOTE***] ***POD***

purepod replica-link mapping policy disconnect --remote-policy ***REMOTE_POLICY*** [--context ***REMOTE***] ***POD***

purepod replica-link mapping policy list [--cli | --csv | --nvp] [--notitle] [--raw] [--filter ***FILTER***] [--page] [--page-with-token] [--token ***TOKEN***] [--limit ***LIMIT***] [--sort ***SORT***] [--context ***REMOTES***] [***POD*** ...]

purepod replica-link monitor [--csv] [--notitle] [--page] [--raw] [--filter ***FILTER***] [--limit LIMIT] [--sort SORT] --replication [--historical { 1h | 3h | 24h | 7d | 30d | 90d | 1y }] [POD...]

```
purepod setattr { --failover-preference ARRAYLIST | --mediator
MEDIATOR} [--quota-limit SIZE [--ignore-usage]] [--context REMOTE] POD
```

Arguments

POD
: Pod to be created, stretched/unstretched, destroyed, eradicated, or recovered. Also the pod for which the mediator is to be configured.

OLD-NAME
: Name of the pod to be renamed.

NEW-NAME
: Name by which the pod is to be known after the command executes.

SOURCE
: Pod from where data is cloned. The data is cloned to the ***TARGET*** pod.

TARGET
: Pod to where data is cloned. The data is cloned from the ***SOURCE*** pod.

Options

`-h | --help`
: Can be used with any command or subcommand to display a brief syntax description.

`--abort-quiesce`
: Specifies to promote the pod while the replica-link status is `quiescing` without waiting for the quiesce operation to complete.

`--allow-throttle`
: Used with **`purepod clone`** to check the internal health of the array to determine if the pod clone operation would incur too much overhead on the array. If conditions are not optimal at the current time, the pod clone is disallowed (throttled).

`--array`
: When used with **`purepod add`**, stretches the pod between two arrays for ActiveCluster replication.

When used with **`purepod remove`**, unstretches a stretched pod.

When used with **`purepod monitor`**, breaks down performance data by the array to which the I/O is directed.

`--context` ***`REMOTE`***

Used to specify the remote array on which a command is processed. ***`REMOTE`*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

If used with **`purepod list`**, **`purepod replica-link list`**, or **`purepod replica-link mapping policy list`**, `--context` can specify multiple arrays and ***`REMOTE`*** is a comma-separated list of array names.

`--continuous`

Must be used with the **`--replication`** option to display the bandwidth information of ActiveDR replication of the Purity//FA pods. When the **`--array`** option is specified, the display breaks down the information by each linked array (local and remote).

`--destroy-contents`

Allows the **`purepod destroy`** command to destroy pod contents.

`--dry-run`

Used with **`purepod clone`**, simulates the operation without actually executing it. The dry-run checks all conditions the operation would require, and returns a result indicating whether the pod clone operation would succeed or not.

`--eradicate-contents`

Allows the **`purepod eradicate`** command to eradicate pod contents.

`--failover-preference` ***`ARRAY-LIST`***

Displays or determines which array within a stretched pod should be given priority to stay online, should the arrays ever lose contact with each other. The current array and any peer arrays that are connected to the current array for ActiveCluster replication can be added to a pod for failover preference. A failover preference of `(`**`auto`**`)` means that no arrays have been configured for failover preference. To clear the list of failover preferences, set to an empty string (`""`).

`--footprint`

Displays the maximum amount of physical space or effective used capacity the pod or pods would take up on any array, measured in bytes. Footprint represents the amount of physical space or effective used capacity the pod or pods would require if moved or copied to a different array. Footprint is an estimate, as differences in deduplication rates and in average compression rates between two arrays could affect the actual physical space or effective used capacity requirement.

`--from` ***`FROM_MEMBER_NAMES`***

Comma-separated list of one or more locations (realm or array) a pod is to be moved from.

`--historical` ***TIME***

When used with **`purepod list --space`**, displays historical size and space consumption or effective used capacity over the specified range of time.

When used with **`purepod monitor`**, displays historical performance data over the specified range of time.

When used with **`purepod replica-link`**, you must specify the **`--lag`** option to display the lag history and replica-link information over the specified range of time.

Valid time ranges include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year.

`--interval` ***INTERVAL***

Sets the number of seconds between displays of real-time performance data. At each interval, the system displays a point-in-time snapshot of the performance data. If omitted, the interval defaults to every 5 seconds.

`--lag`

Displays the amount of time in seconds that the replication target is behind the source. This is the time difference between the current time and the recovery point.

`--latency`

Displays real-time and historical I/O latency information.

`--mediator`

When used with **`purepod list`**, includes mediator details for the local array and all peer arrays. Mediator details include name, version and status.

When used with **`purepod setattr`**, replaces the current mediator with the specified one. By default, the Pure1 Cloud Mediator (**`purestorage`**) serves as the mediator.

`--mirrored`

Includes performance data for mirrored writes (i.e., writes to volumes in stretched pods). If one array of a stretched pod is offline, the writes processed on the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again.

Include the **`--periodic`** option to display pod periodic replication bandwidth information.

`--on` ***REMOTE***

Displays a list of pods that are on the specified remote array but not stretched to this array. Pods that are stretched to this array will not appear in the list. ***REMOTE*** can be set to (*) to represent all remote arrays.

`--pending`

Includes destroyed pods that are in the eradication pending state. If not specified, pods that are pending eradication are not shown.

`--pending-only`

Only displays destroyed pods that are in the eradication pending state.

`--periodic`

Displays the pod's periodic replication bandwidth, which includes bandwidth used by periodic snapshots, protection groups, and offload replication.

`--promote-from` ***PROMOTE-FROM***

Promotes the pod using the specified undo-demote pod as the source. When the promotion process is complete, the pod contains the same configuration and data as the undo-demote pod. The undo-demote pod will be eradicated.

`--protocol`

Displays the pod's I/O performance for the **`smb`** or **`nfs`** protocol.

`--protocol-group`

Displays the pod's I/O performance for the **`block`** or **`file`** metrics.

`--quiesce`

Specifies to demote the pod and allow it to become a target pod after the replica-link status changes to `quiesced`. Using this setting ensures that all local data has been replicated to the remote pod before the demotion process.

`--quota-limit`

Sets an upper limit on the amount of data a pod can contain. Quota limits can be removed by setting the value to empty quotes " ".

When this option is set, new volumes cannot be created that take more space than the quota limit.

It is by default impossible to set a quota limit lower than the existing usage. To set a quota limit lower than the existing usage, use the **`--ignore-usage`** option. When this option is set, new volumes cannot be created within the pod until the existing usage drops below the quota limit. Destroyed objects are not charged against the quota limit.

`--remote-policy` ***REMOTE_POLICY***

Name of the remote policy, for connecting through a replica-link.

`--repeat` ***REPEAT***

Sets the number of times to display real-time performance data. If omitted, the repeat count defaults to 1.

`--replication`

Displays the bandwidth information of all replication types and the total bandwidth of the Purity//FA pod. When the **`--array`** option is specified, the display breaks down the bandwidth information by the array (local and remote). To display the bandwidth information of each replication type, specify the **`--continuous`**, **`--resync`**, or **`--sync`** option.

`--resync`

Must be used with the **`--replication`** option to display the bandwidth information of resync replication of the Purity//FA pod. When the **`--array`** option is specified, the display breaks down the information by the array to which the I/O is directed.

`--skip-quiesce`

Specifies to demote the pod and allow it to become a target pod without waiting for the **`quiesced`** replica-link status. Using this setting loses all pending data to be replicated to the remote pod.

`--sync`

Must be used with the **`--replication`** option to display the bandwidth information of synchronous replication of the Purity//FA pod. When the **`--array`** option is specified, the display breaks down the information by the array to which the I/O is directed.

`--space`

Displays size and space consumption or effective used capacity information for the volumes and snapshots inside the pod. The space metrics are local to the array.

`--to` ***`TO_MEMBER_NAMES`***

Comma-separated list of one or more locations (realm or array) a pod is to be moved to.

`--total`

Follows output lines with a single line containing column totals in columns where they are meaningful.

`--total-only`

Display only a line (or lines, if required) containing aggregated column data.

`--type`

Specifies the type of information (arrays, pods, protection groups, or volumes) about specified pods to be produced in whitespace-separated list format suitable for scripting.

`--with-hgroups` ***`HGROUPS_NAMES`***

Comma-separated list of one or more host groups to be moved with a pod.

`--with-hosts` ***`HOST_NAMES`***

Comma-separated list of one or more hosts to be moved with a pod.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--page-with-token`

Used to paginate output with a continuation token (**`--token TOKEN`**). Results are printed in `stdout` and the token is printed in `stderr`.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Object Names

Valid characters are letters (A-Z and a-z), digits (0-9), and the hyphen (-) character. The first and last characters of the name must be alphanumeric, and the name must contain at least one letter or '-'.

Most objects in Purity//FA that can be named, including host groups, hosts, volumes, protection groups, volume and protection group suffixes, SNMP managers, and subnets, can be 1-63 characters in length.

Array names can be 1-56 characters in length. The array name length is limited to 56 characters so that the names of the individual controllers, which are assigned by Purity//FA based on the array name, do not exceed the maximum allowed by DNS.

Names are case-insensitive on input. For example, **`vol1`**, **`Vol1`**, and **`VOL1`** all represent the same volume. Purity//FA displays names in the case in which they were specified when created or renamed.

All objects in a pod have a fully qualified name that include the pod name and object name. The fully qualified name of a volume in a pod is **`POD::VOLUME`**, with double colons (**`::`**) separating the pod name and volume name. The fully qualified name of a protection group in a pod is **`POD::PGROUP`**, with double colons (**`::`**) separating the pod name and protection group name. For example, the fully qualified name of a volume named **`vol01`** in a pod named **`pod01`** is **`pod01::vol01`**, and the fully qualified name of a protection group named **`pgroup01`** in a pod named **`pod01`** is **`pod01::pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target array, the fully qualified name of the protection group on the target array is **`POD:PGROUP`**, with single colons (**`:`**) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to target array **`array02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod01:pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target pod, the fully qualified name of the protection group on the target pod is **`POD:PGROUP`**, with single colons (**`:`**) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to a target pod **`pod02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod02::pod01:pgroup01`**.

All objects in a volume group have a fully qualified name that includes the volume group name and the object name, separated by a forward slash (`/`). For example, the fully qualified name of a volume named **`vol01`** in a volume group named **`vgroup01`** is **`vgroup01/vol01`**.

Realms, pods, and volume groups provide a namespace with unique naming conventions. All objects in a realm have a fully qualified name that include the realm name and object name or realm, pod, and object names. The fully qualified name of a pod in a realm is **`REALM::POD`**, with double colons (`::`) separating the realm name and pod name. The fully qualified name of a volume in a pod in a realm is **`REALM::POD::VOLUME`**, with double colons (`::`) separating the realm, pod, and volume names. The fully qualified name of a protection group in a pod in a realm is **`REALM::POD::PGROUP`**, with double colons (`::`) separating the realm, pod, and protection group names. The fully qualified name of an object in a realm but not in a pod is **`REALM::HOST`**, using host as an example.

Description

Organizations replicate critical data for various reasons, such as for off-host backup, software development and testing, data distribution and consolidation, and perhaps most importantly, disaster protection.

Purity//FA offers three types of replication: asynchronous replication, ActiveDR replication, and ActiveCluster replication.

Asynchronous replication is managed through protection groups consisting of member hosts, host groups, or volumes. Replication policies are created for each protection group to define when Purity//FA should take snapshots of the protection group's data set and send it off to its target arrays, where they are saved as replicas. Asynchronous replication is configured through the **`purepgroup`** command.

ActiveDR replication allows pod-to-pod, continuous replication of compressed data from a source array at the production site to a target array at the recovery site, providing a near-zero Recovery Point Objective (RPO). ActiveDR replication is configured through the **`purepod`** command. For more information about ActiveDR replication, see ActiveDR Replication.

ActiveCluster replication is managed through pods. A pod representing a collection of protection groups and volumes is created on one array and stretched to another array, resulting in fully synchronized writes between the two arrays. A pod can contain a mix of volumes, and protection groups with member volumes. Writes to the pod coming into either array are immediately synchronized and seen on both arrays. ActiveCluster replication is configured through the **purepod** command.

Purity supports multiple connections between FlashArrays for a hub-and-spoke topology for stretched pods. This way, a single FlashArray can participate as a consolidator, synchronously replicating the desired volumes for FlashArrays dedicated for specific workloads. IP supports up to five synchronous connections between FlashArrays. Fibre Channel (FC) supports one synchronous connection.

When stretching pods between FlashArrays, make sure not to exceed the limits for stretched objects like pods, volumes, volume snapshots, and protection group snapshots. For information about the ActiveCluster synchronous IP or FC replication limits, see one of the FlashArray Model Limits articles, as applicable to the given model, on the Knowledge site at `https://support.purestorage.com`.

For information about Purity//FA replication requirements and interoperability details, see the Purity Replication Requirements and Interoperability Matrix article on the Knowledge site at `https://support.purestorage.com`.

Configuring ActiveCluster Replication

The process of creating and stretching a pod so that volume data is synchronously replicated between two arrays involves the following broad steps:

1. Verify that you have four physical interfaces configured with the replication service. For more information on how to configure the interfaces for ActiveCluster replication, see purenetwork.
2. Create a connection between the two arrays over which the pod will be stretched. For more information on how to connect arrays, see purearray.
3. From the array with the volumes you want to synchronously replicate, create a pod (**purepod create**), and then add volumes and protection groups to the pod.

   Volumes are created in pods through the **purevol create** command. Volumes are moved into pods and protection groups through the **purevol move** command. For more information on how to create and move volumes, see purevol.

   Protection groups are created through the **purepgroup create** command. For more information on how to create protection groups, see purepgroup.
4. Create hosts, if necessary, and connect them to the volumes in the pod. If you have uniform storage access, meaning each host is connected to both arrays, but accessing one array is more efficient for reasons such as shorter distance and lower latency, then you may want to configure a preferred array for each host (also known as Asymmetric Logical Unit Access, or

ALUA, preferred paths). Hosts and preferred arrays are created and configured through the **`purehost`** command. For more information on how to create and configure hosts, see pure-host.

5 Stretch the pod (**`purepod add`**) to the remote array.

Once stretched, the pod immediately starts synchronizing the data across the two arrays. Once fully synchronized, all of the data in the pod is mirrored between the two arrays. For example, when a volume is resized or renamed, the change will be seen on both arrays instantaneously.

Creating Pods

The **`purepod create`** command creates a pod on the local array.

Each pod must be given a name that is unique across the arrays to which they are stretched, so a pod cannot be stretched to an array that already contains a pod with the same name.

After a pod has been created, add file systems and policies, and volumes and protection groups to the pod, and then stretch the pod to another (connected) array. Volumes are created inside pods through the **`purevol create`** command. Volumes can be moved into or out of pods and protection groups through the **`purevol move`** command. Protection groups are created inside pods through the **`purepgroup create`** command. Protection groups cannot be moved into or out of pods.

Include the **`--failover-preference`** option to specify which array within a stretched pod should be given priority to stay online, should the arrays ever lose contact with each other.

Include the **`--quota-limit`** option to set an upper limit on the data a pod can contain. Quota limits can be removed by setting the value to empty quotes " ".

Pod Naming Conventions

Each pod is a separate namespace for the volumes and protection groups it contains.

Each volume in a pod consists of the pod namespace identifier and the volume name, separated by a double colon (**`::`**). The naming convention for a volume inside a pod is **`POD::VOL`**, where:

`POD` is the name of the container pod.

`VOL` is the name of the volume inside the pod.

For example, the fully qualified name of a volume named vol01 inside a pod named pod01 is **`pod01::vol01`**.

In the following example, an array has three volumes named **vol01** that are completely independent of one another - one on the array named **vol01**, another volume in pod01 named **pod01::vol01**, and a third volume in pod02 named **pod02::vol01**.

```
$ purepod list *vol01
Name         Size Source Created                 Serial
pod01::vol01 1T   -      2017-06-12 16:05:37 PDT C54E7DD1D6F6408600011814
pod02::vol01 1T   -      2017-06-12 16:05:37 PDT C54E7DD1D6F6405305839572
vol01        1T   -      2017-06-08 14:48:02 PDT C54E7DD1D6F6408600011010
```

When administering a volume that is in a pod, always include the pod namespace in the volume name. For example, to create a volume named vol02 in pod01 that is 1T in size, run the command **purepod create --size 1T pod01::vol02**. To increase the virtual size of volume vol11 in pod pod02 to 2 GB, run the command **purepod setattr --size 2G pod02::vol11**.

Each protection group in a pod consists of the pod namespace identifier and the protection group name, separated by a double colon (**::**). The naming convention for a protection group inside a pod is **POD::PGROUP**, where:

POD is the name of the container pod.

PGROUP is the name of the protection group inside the pod.

For example, the fully qualified name for a protection group named pgroup02 inside a pod named pod09 is **pod09::pgroup02**.

When administering a protection group that is in a pod, always include the pod namespace in the protection group name. For example, to create a protection group named pgroup02 in pod01, run the command **purepgroup create pod01::pgroup02**.

Automatic Default Protection Group Lists for Pods

The automatic default protection group lists mechanism ensures that every volume created in a pod or copied to a pod automatically becomes a member of one or more protection groups. When a new pod is created, a new default protection group list is created for the pod by copying the default protection group list for the root of the array.

Initially, the array root default protection group list contains one protection group named **pgroup-auto**, as shown by the **purearray default-protection list ""** command. The empty list **""** in the command denotes the root of the array, as opposed to a pod. In the command output, **"-"** denotes the root of the array.

```
$ purearray default-protection list ""
```

```
Container Name Container Type Default Protection Name Default Protection Type
-              -              pgroup-auto             protection_group
```

When a new pod is then created, the pod is initially configured with a default protection group list copied from the array root list, with pod protection group names modified with the pod name. A pod protection group is created for each protection group named in its list, as shown in the example below. With an array root default protection group list containing **`pg1`** and **`pg2`**, the pod default protection group list initially contains **`pod1::pg1`** and **`pod1::pg2`**, and protection groups **`pod1::pg1`** and **`pod1::pg2`** are created in the pod. Any volume created in the pod, or copied to the pod, by default is made a member of each pod protection group in the current pod default protection group list.

```
$ purepod create pod1
Name Source Array Status Frozen At Promotion Status Link Count
pod1 -      vm-3  online -         promoted         0


$ purearray default-protection list
Container Name Container Type Default Protection Name Default Protection Type
-              -              pg1                     protection_group
-              -              pg2                     protection_group
pod1           pod            pod1::pg1               protection_group
                              pod1::pg2               protection_group
```

After the initial creation of the pod protection group list, the pod default protection group list is completely independent of the array root default protection group list and the default protection group lists of other pods, if any.

If the array root default protection group list is an empty list, no pod default protection group list is created, no new pod protection groups are automatically created, and the pod initially has no default protection. A pod default protection group list can later be configured.

Default protection group lists are configured with the **`purearray default-protection`** command. The **`purearray default-protection set`** command supports enabling default protection for a specific pod or pods whether or not the array root or other pods have default protection group lists. See "Automatic SafeMode™ Default Protection for Volumes" on page 86, in the `purearray` section, for information about default protection group lists.

Configuring Hosts and Preferred Arrays

For a host to read and write data on a volume within a pod, a connection between the two must be established. Verify that each volume in the pod is connected to a host.

If you have uniform storage access, meaning each host is connected to both arrays, but accessing one array is more efficient for reasons such as shorter distance and lower latency, then you may want to configure a preferred array for each host (also known as Asymmetric Logical Unit Access, or ALUA, preferred paths). If a preferred array is configured for a host, then other arrays will expose active/non-optimized paths to that host.

Hosts are created and preferred arrays are configured through the **`purehost`** command. For more information on how to create and configure hosts, see purehost.

Stretching Pods

The **`purepod add --array`** command stretches a pod to another array. When a pod is stretched, the volume data inside the pod is synchronously replicated between the two arrays. The arrays of a stretched pod are considered peer to one another - there is no concept of "source" or "target" array.

When the pod is first stretched, the pod data is asynchronously replicated to the other array before transitioning into synchronous mode. An array can host multiple stretched pods, each one replicated to a different peer.

Once synchronized, the peer arrays of a stretched pod present the pod volumes, including their contents, volume serial numbers, and snapshots, identically to all connected hosts at all times, and pod volume reads always return the same data regardless of which array receives and executes them.

Array administrators are also peers with respect to managing the pod and the volumes, protection groups, and snapshots it contains. For example, either administrator can change the size of a volume in the pod, create and destroy volumes in the pod, move volumes into and out of the pod, connect hosts to the volumes in the pod, add volumes to protection groups in the pod, schedule, take, and destroy snapshots of volumes and protection groups in the pod, and create clone volumes from snapshots of volumes in the pod. These operations take effect immediately and are visible on both arrays.

Before a pod can be stretched to another array, the two arrays must first be connected. Arrays are connected by running the command **`purearray connect`**.

After the arrays are connected, run the **`purepod add --array`** command to stretch the pod to the other array. For example, run the following command from array **`array01`** to stretch pod **`pod01`** on array **`array01`** to remote array **`array02`**,

```
$ purepod add --array array02 pod01
```

A pod can be stretched across no more than two arrays.

Stretched pods cannot be destroyed.

Volumes cannot be added to or removed from a pod that is stretched. To move volumes into or out of a stretched pod, unstretch the pod, move the volumes into or out of the unstretched pod, and then restretch the pod.

Protection groups cannot be created in a pod that is stretched. To create protection groups in a stretched pod, unstretch the pod, create the protection groups inside the unstretched pod, and then restretch the pod.

If an unstretched pod contains a protection group that is asynchronously replicating to another array, stretching the pod to any other array will not disrupt the asynchronous replication connection. If an unstretched pod contains a protection group that is asynchronously replicating to another array and the pod is stretched to that same array, any in-progress data transfer processes stop, the target array disallows asynchronous replication, and the ActiveCluster replication process begins.

Unstretching Pods

When the volumes within a stretched pod no longer need to replicate synchronously, unstretch the pod to collapse it to a single array.

Run the **`purepod remove --array`** command to unstretch a pod, removing it from the specified array.

After a pod has been unstretched, ActiveCluster replication stops. A destroyed version of the pod with "restretch" appended to the pod name is created on the array that no longer has the pod. The restretch pod represents a point-in-time snapshot of the pod, just before it was unstretched. Run the **`purepod list --pending-only`** command to display the details of the restretch pod.

The restretch pod enters an eradication pending period starting from the time that the pod was unstretched.

If within the eradication pending period the same pod is stretched to the array again, the pod is restored by first recovering the destroyed (*.restretch) pod, and then replicating the new pod data until the stretched pod is in sync. Once fully synchronized, pod activity resumes on both arrays.

The length of eradication pending period is set by the enabled delay value if the pod is not protected by SafeMode, or by the disabled delay value if the pod is protected by SafeMode. (See "Eradication Delays" on page 88 in the `purearray` chapter for information on the disabled delay and enabled delay.)

After the eradication pending period has elapsed, the restretch pod is permanently eradicated. If the pod is restretched after the eradication pending period, the arrays synchronize the pod data as if it were the first time the pod was stretched.

A restretch pod can be cloned or destroyed, but it cannot be explicitly recovered.

In the event that a local array goes offline while a pod is still stretched across two arrays, the status of the remote array becomes unknown and there is no guarantee that the pod is online elsewhere. In such cases, you will only be able to unstretch the pod from the local array by including the **`--with-unknown`** option to force the unstretch.

In the following example, pod01 is stretched across arrays array01 and array02. Local array01 has gone offline. As a result, the status of its peer array (array02) is unknown. To unstretch pod01 while it is in offline/unknown status, you must include the **`--with-unknown`** option to force the unstretch.

```
$ purepod list
Name   Source  Array    Status   Frozen At                Promotion Status Link Count
pod01  -       array01  offline  2017-11-14 17:25:40 PST  promoted  0
-      array02  unknown  -
```

Note that by using the **`--with-unknown`** option, you risk losing data if there is a problem with the pod on the remote array.

Moving Pods

Use the **`purepod move`** command to move one or more pods between a realm and an array. This example moves a pod from `realm-1` to the array `pure01`:

```
$ purepod move --to pure01 --from realm-1 realm-1::pod4
Name Source Array  Status Frozen At Member Promotion Status Link Count Quota Limit
pod4 -      pure01 online -         pure01 promoted  0 -
```

This example moves the pod back to `realm-1`:

```
$ purepod move --from pure01 --to realm-1::pod4
Name          Source Array  Status Frozen At Member Promotion Status Link Count
realm-1::pod4 -      pure01 online -         pure01 promoted  0      -
```

Include the **`--with-hosts`** ***`HOST_NAMES`*** option to include the specified pod hosts in the move.

Include the **`--with-hgroups`** ***`HGROUPS_NAMES`*** option to include the specified pod host groups in the move.

A pod with active connections cannot be moved unless the **`--with-hosts`** and/or **`--with-hgroups`** options are included.

Cloning Pods

In the event of an extreme disaster where one array of a stretched pod is in an offline status while the other array is in an online status, and then the online array becomes permanently disabled, the pod or restretch pod in the offline array can be cloned to create a point-in-time consistent copy of the pod and its contents.

Run the **`purepod clone`** command to clone an existing pod (**`SOURCE`**) to create a new pod (**`TARGET`**).

When cloned, the new pod takes on the entire history and configuration of the source pod, including its protection groups, protection group snapshot and replication policies, volumes, volume snapshots, policies and file systems, exports and quotas, and historical performance metrics. The new pod does not take on the ActiveCluster replication configuration of the source pod, and the serial numbers of its volumes differ from those of the source pod's volumes.

Once cloned, the new pod can be configured to resume ActiveCluster replication and the source pod can be deleted from the arrays over which it was stretched.

After cloning a pod, run the **`purepod list`** command to display the new pod and its details, including and the name of the source pod from which the new pod was cloned. In the following example, pod **`pod01`**, which is currently in "offline" status, has been cloned to create a new pod named **`clonedpod01`**.

```
$ purepod clone pod01 clonedpod01
Name        Source  Array    Status  Frozen At               Promotion Status Link Count
clonedpod01 pod01   pure-001 online  -                       promoted  0
$ purepod list
Name         Source Array    Status  Frozen At               Promotion Status Link Count
pod01        -      pure-001 offline 2017-11-14 17:25:40 PST promoted  0
```

```
clonedpod01  pod01  pure-001 online  -
```

Notice that the source pod for `clonedpod01` is `pod01`. If the source pod is eradicated, the name of the source pod no longer appears.

Clone Throttling

An array may become overloaded when cloning a pod. Use the **`purepod clone --allow-throttle`** command to check the internal health of the array to determine if a clone is recommended at that point in time. This allows for throttling of cloning operation to lessen the impact on the array performance.

```
$ purepod clone --allow-throttle pod01 clonedpod02
Name        Source  Array    Status  Frozen At                Promotion Status Link Count
clonedpod02 pod01   pure-001 online  -                        promoted   0
```

Simulating a Cloning Operation

The **`purepod clone --dry-run`** option checks all the conditions the pod cloning operation would require. Use this command to determine if the command would return an error or return successfully (without actually creating the clone). This example returns success:

```
$ purepod clone --dry-run pod01 clonedpod03
Success   Details
True      Pod can be cloned.
```

This example returns an error:

```
$ purepod clone --dry-run pod01 clonedpod02
Success   Details
False     Error on clonedpod02: Cannot create pod due to pod with same name already exists.
```

Destroying Pods

When a pod is no longer needed, it can be destroyed. Run the **`purepod destroy`** command to destroy a pod.

When run with no options, the **`purepod destroy`** command can only destroy a pod if it is empty, so before destroying a pod, ensure all volumes and protection groups inside the pod have been either moved out of the pod or destroyed. Or add the **`--destroy-contents`** option to destroy both the pod and its contents, including all volumes, snapshots, and protection groups inside the pod.

You cannot destroy a stretched pod. To destroy a stretched pod, unstretch the pod before destroying it.

The destruction of pods and their contents is not immediate. Instead, the pod is placed in an eradication pending period. The pod can be manually recovered or permanently eradicated within the eradication pending period. After the eradication pending period, the pod is completely eradicated and not recoverable. The length of eradication pending period is set by the enabled delay value if the protection group is not protected by SafeMode, or by the disabled delay value if the protection group is protected by SafeMode. (See "Eradication Delays" on page 88 in the `purearray` chapter for information on the disabled delay and enabled delay.)

Recovering Pods

A destroyed pod can be recovered at any time during the eradication pending period. Run the **`purepod recover`** command to recover a pod.

Once a pod has been recovered, its destroyed volumes and protection groups can also be recovered to bring the pod and its contents back to their previous state.

Eradicating Pods

A destroyed pod can be permanently eradicated at any time during the eradication pending period (unless the SafeMode manual eradication prevention feature is enabled). Run the **`purepod eradicate`** command to eradicate a pod.

The **`purepod eradicate`** command only eradicates a pod if its destroyed volumes and protection groups are already eradicated. Therefore, before eradicating a pod, first eradicate all its destroyed volumes and protection groups. Or add the **`--eradicate-contents`** option to also eradicate the pod's destroyed volumes and protection groups.

Once a pod has been eradicated, it can no longer be recovered.

Renaming Pods

Run the **`purepod rename`** subcommand to change the current name (**`OLD-NAME`**) of the pod to the new name (**`NEW-NAME`**). The name change is effective immediately and the old name is no longer recognized in CLI, GUI or REST interactions. All volumes, volume snapshots, and protection groups with the pod namespace are also renamed. In the Purity//FA GUI, the new name appears upon page refresh.

Mediator

Pure Storage provides a passive mediation solution for ActiveCluster replication that guarantees that a pod remains online on one array when the other array is offline.

During regular operation, each array periodically pings the mediator to confirm its status. In the event of an outage in the environment and two arrays lose contact with each other, the mediator takes control to determine which array remains online and continues data services.

When the arrays lose contact with each other, the first array to reach the mediator becomes the array that stays online and continues serving I/O while its peer array goes offline. If failover preference is configured for the pod, any of the arrays that are listed in the failover preference list are given priority to stay online. At this point, the stretched pod falls out of sync. To avoid inconsistent data between the two arrays (also known as "split brain"), the pod data in the offline array becomes frozen. The Frozen At time, also known as the recovery point, represents the time at which the data on the pod was frozen when the array went offline.

When the peer arrays re-establish contact with each other, the array that was online throughout the outage starts replicating pod data to its peer array until the pod is once again in sync, at which time pod activity resumes on both arrays.

The mediation process happens per pod, so if two arrays share multiple pods and the arrays lose contact with each other, one array could be online for some pods but offline for other pods, and vice versa for the other array.

View the mediator status for each pod by running the **`purepod list --mediator`** command. Possible mediator statuses include:

- **Online.** The array is successfully communicating with the mediator.
- **Unreachable.** The array cannot reach the mediator, either due to network issues or because the mediator is down. When a mediator is unreachable, ActiveCluster replication continues to function provided all arrays are healthy and communicating, but a high availability event without mediator access can result in an outage.

If an array cannot successfully communicate with the mediator, an alert is generated. If after verifying your network connections the mediator status is still unreachable, contact Pure Storage Technical Services.

By default, the mediator is set to **purestorage**, representing the Pure1© Cloud Mediator. The Pure1 Cloud Mediator requires no special configuration to work. View the mediator configuration for each pod by running the **purepod list --mediator** command.

The mediator service can be configured to use a different mediator, such as one that is hosted on premise, by running the **purepod setattr --mediator** command. Before you configure the mediator, contact Pure Storage Technical Services to obtain the package and steps on how to install an alternate mediator.

To set the mediator service back to the default Pure1 Cloud Mediator, use an empty string ("").

Failover Preference

Failover preference determines which array within a stretched pod should be given priority to stay online, should the arrays ever lose contact with each other. If both arrays are online and can reach the mediator, failover preference determines which of those two arrays should be given priority to stay online. If, however, only one of the two arrays can reach the mediator, that array stays online regardless of failover preference. Failover preference is set through the **--failover-preference** option.

The current array and any peer arrays that are connected to the current array for ActiveCluster replication can be added to a pod for failover preference. A pod can be configured with multiple arrays for failover preference, even if the pod is not stretched to those arrays. For example, failover preference for a pod can be configured to include all of the arrays of a data center.

In the following example, pod **pod01** on current array **array01** has been configured with three arrays with failover preference: **arrayA**, **arrayB**, and **arrayC**. All three of the arrays are already connected to the current array for ActiveCluster replication. The pod is then stretched to **arrayA**.

```
$ purepod list pod01 --failover-preference
Name  Source Array   Status Frozen At Promotion Status  Link Count  Failover Preference
pod01 -      array01 online -         promoted          0           arrayA
arrayB
arrayC
$ purepod add --array arrayA pod01
Name  Arrays
pod01 array01
arrayA
```

If the arrays lose contact with each other, **arrayA** will be given priority over **array01** to reach the mediator first. If **arrayA** cannot reach the mediator, **array01** will take over.

In the following example, pod **pod02** on current array **array02** has been configured with just one array - its own - with failover preference. The pod is then stretched to array **array04**.

```
$ purepod list pod02 --failover-preference
Name   Source Array   Status Frozen At Promotion Status  Link Count  Failover Preference
pod02  -       array02 online -         promoted          0           array02
$ purepod add --array array04 pod02
Name   Arrays
pod02 array02
array04
```

If the arrays lose contact with each other, **array02** will be given priority over **array04** to reach the mediator first. If **array02** cannot reach the mediator, **array04** will take over.

The arrays listed for failover preference do not appear in any priority order, so if a pod is stretched over two arrays that are configured for failover preference, either array will have an opportunity to stay online for the given pod.

A failover preference of (**auto**) means that no arrays have been configured for failover preference. In such cases, if both arrays within a stretched pod can reach the mediator after losing contact with each other, either array will have an opportunity to stay online for the given pod.

To clear the list of failover preferences, set to an empty string (`""`).

Pre-Election

In addition to passive mediation and failover preference, Purity provides the pre-election behavior to further ensure a stretched pod remains online. With pre-election, an array within a stretched pod is chosen by Purity to keep a pod online when other failures occur in the environment.

The pre-election behavior elects one array of the stretched pod to remain online in the rare event that:

- The mediator is inaccessible on both arrays within the stretched pod, preventing the arrays from racing to the mediator to determine which one keeps the pod online.

... and then later ...

- The arrays within the stretched pod become disconnected from each other.

When the mediator becomes inaccessible on both arrays, Purity pre-elects an array per pod to keep the pod online. Then, if the arrays lose contact with each other, the pre-elected array for that pod takes over to keep the pod online while its peer array takes the pod offline.

If either array reconnects to the mediator before they lose contact with each other, the pre-election result is cancelled. The array with access to the mediator will race to the mediator and keep the pod online if its peer array fails or the arrays become disconnected from each other.

Run the **`purepod list --mediator`** command to display the pre-election status of an array. If the array is pre-elected to keep the pod online in the event of a communication failure between the arrays, the term (**`pre-elected`**) is appended to the mediator status in the Mediator Status column.

In the following example, pod **`pod01`** is stretched over arrays **`array01`** and **`array02`**. The arrays cannot reach the mediator, so Purity has pre-elected array **`array01`** to keep the pod online in the event of a communication failure between the arrays.

```
purepod list --mediator
Name  Source  Mediator    Mediator Version Array   Status ... Mediator Status
pod01 -       purestorage 1.0              array01 online ... unavailable (pre-elected)
                                           array02 online ... unavailable
```

Then, if the arrays lose contact with each other, the pre-elected array for that pod takes over to keep the pod online while its peer array takes the pod offline.

One and only one array within each pod is pre-elected at a given point in time, so while a pre-elected array is keeping the pod online, the pod on its non-elected peer array remains offline during the communication failure.

Users cannot pre-elect arrays. Purity uses various factors, including the following ones (listed in order of precedence), to determine which array is pre-elected:

- If a pod has a failover preference set, then the array that is preferred will be pre-elected.
- If one of the arrays has no hosts connected to volumes in the pod, then the other array will be pre-elected.
- If neither of the above factors applies, one of the arrays is selected by Purity.

If the pre-elected array goes down while pre-election is in effect, the non-elected peer array will not bring the pod online.

If the non-elected array reconnects to the mediator while it is still disconnected from the pre-elected array, it is ignored and will still keep the pod offline. If the data in the non-elected pod must be accessed, clone it to create a point-in-time consistent copy of the pod and its contents, including its volumes and snapshot history. After the pod has been cloned, disconnect the hosts from the original volumes and reconnect the hosts to the volumes within the cloned pod.

If the arrays re-establish contact with each other but the mediator is still inaccessible, the array that was online throughout the outage starts replicating pod data to its peer array until the pod is once again in sync and both arrays serve I/O. One array will still be pre-elected, as indicated by the term (**`pre-elected`**), in case both arrays lose contact with each other again.

When the peer arrays re-establish contact with each other *and* can access the mediator, the array that was online throughout the outage starts replicating pod data to its peer array until the pod is once again in sync and both arrays serve I/O, at which time pod activity returns to normal.

ActiveDR Replication

ActiveDR is a Purity//FA data protection feature that enables active-passive, continuous replication by linking pods across two FlashArrays, providing a low Recovery Point Objective (RPO).

ActiveDR replication streams pod-to-pod transfer of compressed data from a source FlashArray at the production site to a target FlashArray at the recovery site. If the source FlashArray becomes unavailable due to events such as a disaster or workload migration, you can immediately fail over to the target FlashArray.

A low RPO allows you to recover at the target site with less data loss compared to scheduled snapshot replication. Because ActiveDR replication constantly replicates data to the target FlashArray and does not wait for the write acknowledgement from the target FlashArray, no additional host write latency is incurred when the distance between the two FlashArrays increases.

For information about ActiveDR and how to use ActiveDR replication to provide fast recovery, see the following topics:

- "Setting Up ActiveDR Replication" on the next page
- "Promotion Status of a Pod" on page 401
- "Adding Data to a Pod on a Source FlashArray" on page 397
- "Replica Links" on page 404
- "Performing a Failover for Fast Recovery" on page 407
- "Performing a Reprotect Process after a Failover" on page 409
- "Performing a Failback Process after a Failover" on page 410
- "Performing a Planned Failover" on page 412
- "Performing a Test Recovery Process" on page 413

Setting Up ActiveDR Replication

As part of a disaster recovery strategy, you can protect data on the source FlashArray at the production site by setting up ActiveDR replication to a target FlashArray at the disaster recovery (DR) site.

Setting up ActiveDR replication includes these steps:

1. Connecting the Source and Target FlashArrays
2. Setting Up a Source Pod
3. Setting Up a Pod on the Target FlashArray
4. Demoting the Pod on the Target FlashArray
5. Creating a Replica Link to Initiate ActiveDR Replication

Connecting the Source and Target FlashArrays

Before you configure ActiveDR replication, you must connect a source FlashArray at the production site to a target FlashArray at the disaster recovery site. For more information about how to connect two FlashArrays, see "Connected Arrays" on page 1.

Setting Up a Source Pod

To set up a source pod for ActiveDR replication,

1. Create a source pod on the source FlashArray by using the `purepod create` command. For more information, see "Creating Pods" on page 1.
2. Move existing volumes into the pod by using the `purevol move` command.

Note: You cannot move volumes into the source pod after the replica link is created.

For more information, see "Moving Volumes" on page 601.

Setting Up a Pod on the Target FlashArray

Set up a pod on the target FlashArray at the disaster recovery (DR) site for ActiveDR replication by using the `purepod create` command. For more information, see "Creating Pods" on page 382.

Demoting the Pod on the Target FlashArray

Demote the pod created for ActiveDR replication on the target FlashArray at the disaster recovery (DR) site to allow it to become a target pod.

When you demote a pod,

- The pod promotion status changes.

  When you create a pod initially, its promotion status is **`promoted`**, which allows the pod to provide read/write access to the host. When the demotion process is complete, the pod promotion status changes to **`demoted`**, which allows read-only access.
- An undo-demote pod is created.

  When the pod status changes to **`demoted`**, an undo-demote pod is created and placed in an eradication pending period. An undo-demote pod preserves the pod configuration and data in the state before the demotion process. Therefore, you can retrieve the data not being updated during the demotion process by using the undo-demote pod. An undo-demote pod is automatically eradicated at the end of the eradication pending period.

Note: If an undo-demote pod already exists, the demotion process fails with an error.

For more information, see "Promotion Status of a Pod" on page 401.

To demote a pod at the recovery site, use the **`purepod demote`** command. The promotion status of the pod shows **`demoting`** initially and then transitions to **`demoted`** when the demotion process is complete.

You can cancel the demotion process when the pod is in the **`demoting`** status by using the **`purepod promote`** command.

The following example demotes pod **`pod2`** and then displays only the undo-demote pod.

```
$ purepod demote pod2
Name Source Array  Status Frozen At Promotion Status Link Count
pod2 None   arrayB online -         demoted          0

$ purepod list --pending-only
Name             Source Time Remaining Array Status Frozen at Promotion Status Link Count
pod2.undo-demote pod2   23:59:53       arrayB online -        promoted         0
```

Adding Data to a Pod on a Source FlashArray

Once a source pod is created on the FlashArray, file data or volumes can be added. You can move existing file systems or volumes into the pod with the **`purefs move`** or **`purevol move`** commands, respectively, create new file systems or volumes, and create policies and snapshot policies. Volumes cannot be added to pods with file systems or vice versa. Directory snapshots managed by a snapshot policy will become unmanaged when the parent file system is moved to a different pod.

Note: Volumes cannot be added to pods with file systems and file systems cannot be added to pods with volumes.

With the file systems or volumes in place, a replica link can be created to initiate ActiveDR replication. File systems and volumes cannot be moved into a pod when the replication is initiated; only after deletion of the replica link. However, you may create a new volume or file system in the pod without deleting or pausing the link.

A replica link can only be created when the Purity//FA on the target array is of the same version as the source, or newer. Hence, if the source array is being upgraded to a newer version, a link cannot be created or recreated after deletion, without first upgrading the target array. Only file policy or block features that are supported on both the source and target arrays, can be used; unsupported operations will fail. The recommendation is therefore to use the same version of Purity//FA on both arrays, source and target.

Creating a Replica Link to Initiate ActiveDR Replication

To initiate ActiveDR replication, create a replica link from the source pod at the production site to the demoted pod at the target site by using the **`purepod replica-link create`** command.

When you link a source pod with a demoted pod using a replica link, the demoted pod becomes the target pod of the source pod. The target pod serves as a replica pod to track the changes of the source pod, including volumes, snapshots, protection groups, and protection group snapshots.

When the local and remote FlashArrays are connected, the replica link starts the baseline process between the source and target pods. When the baseline process is complete, the source pod starts to replicate data to the target pod, changing the replica-link status to **`replicating`**. For more information about replica links, see "Replica Links" on page 404.

The following example demotes pod **`pod2`** on FlashArray **`arrayB`** and then associates pod **`pod1`** (source) to pod **`pod2`** by creating a replica link to start ActiveDR replication. Pod **`pod2`** serves as a replica pod for pod **`pod1`** on FlashArray **`arrayA`**.

```
$ purepod demote pod2
Name Source Array  Status Frozen At Promotion Status Link Count
pod2 -      arrayB online -         demoted          0

$ purepod replica-link create pod1 --remote-pod pod2 --remote arrayB
Name  Direction  Remote Pod  Remote  Status       Recovery Point           Lag
pod1  -->        pod2        arrayB  replicating  2020-01-04 10:47:24 PDT  1s
```

The following example creates two replica links from FlashArray **`arrayA`**:

First command creates a replica link from pod **podA** on FlashArray **arrayA** to pod **podB** on FlashArray **arrayB**. The replica-link status shows **replicating** with a one-second lag.

Second command creates a replica link from pod **podC** on FlashArray **arrayA** to pod **podD** on FlashArray **arrayC**. The replica-link status shows **replicating** with a one-second lag.

```
//From arrayA
$ purepod replica-link create podA --remote-pod podB --remote arrayB
Name  Direction  Remote Pod  Remote  Status       Recovery Point           Lag
podA  -->        podB        arrayB  replicating  2020-01-04 10:47:24 PDT  1s

//From arrayA
$ purepod replica-link create podC --remote-pod podD --remote arrayC
Name  Direction  Remote Pod  Remote  Status       Recovery Point           Lag
podC  -->        podD        arrayC  replicating  2020-01-04 10:47:24 PDT  1s
```

The purepod replica-link Subcommands

Use the following subcommands with the **purepod replica-link** command to create, remove, display, suspend, resume, and monitor replica links. You can use the subcommands from either the source or target FlashArray, but the **create** subcommand is only allowed on the source FlashArray.

- **create**

  Creates a replica link to connect a local FlashArray to a remote FlashArray. The local FlashArray must be the source FlashArray.
- **delete**

  Deletes a replica link between a local FlashArray and a remote FlashArray.
- **list**

  Displays the replica-link status and lag information on the local FlashArray regardless whether the local pod is a source or target pod.
- **mapping**

  Manage mappings across the replica-link.
- **pause**

  Pauses ActiveDR replication by stopping the replica-link connection between a local FlashArray and a remote FlashArray. The write streams continue in the background. To continue the replication, resume the replica link.
- **resume**

  Resumes ActiveDR replication after the pause operation.

This example shows the replica-link information:

- From FlashArray **arrayA** where pod **podA** and pod **podC** are replicating data to pod **podB** on FlashArray **arrayB** and pod **podD** on FlashArray **arrayC**, respectively.
- From FlashArray **arrayB** where pod **podB** is receiving incoming replication from pod **podA** on FlashArray **arrayA**.

```
//From arrayA
$ purepod replica-link list
Name  Direction  Remote Pod  Remote  Status       Recovery Point           Lag
podA  -->        podB        arrayB  replicating  2019-10-04 10:47:24 PDT  1s
podC --> podD arrayC replicating 2019-10-04 10:47:24 PDT 1s

$ purepod replica-link list
//From arrayB
Name  Direction  Remote Pod  Remote  Status       Recovery Point           Lag
podB  <--        podA        arrayA  replicating  2019-10-04 10:47:24 PDT  1s
```

This example suspends ActiveDR replication between pod **podA** on array **arrayA** and pod **podB** on **arrayB**.

```
// From arrayA
$ purepod replica-link pause podA --remote-array arrayB
Name  Direction  Remote Pod  Remote  Status       Recovery Point           Lag
podA  -->        podB        arrayB  paused       2020-02-24 10:47:24 PDT  1s
```

This example resumes the replica link between pod **podA** on array **arrayA** and pod **podB** on **arrayB**.

```
// From arrayA
$ purepod replica-link resume podA --remote-array arrayB
Name  Direction  Remote Pod  Remote  Status       Recovery Point           Lag
podA  -->        podB        arrayB  replicating  2020-02-24 10:47:24 PDT  1s
```

To delete a replica link, specify the **delete** subcommand with the **purepod replica-link** command. The following example deletes the replica link between pod **podA** and pod **podB**.

```
// From arrayA
$ purepod replica-link delete podA --remote-pod podB
Name  Remote Pod
podA  podB
```

Replica-link Mapping for File Policies

Policy mappings are created when a policy is created on the source pod of an existing file rep-lica-link, or when a replica link is created between two file pods and there are pre-existing policies on the source or target pod. To view these policy mappings, use the **`purepod replica-link mapping policy list`** command. After the mapping is created and the policies are replicated to the target, policy mappings can be managed from the target array only. To manage the mapping, use the `purepod replica-link mapping` command from the target array.

The following example, executed on the target array, connects the source remote policy `policy01` to the target local pod named `pod02`. From then on, `policy01` along with all of its exports will be replicated to `pod02`.

```
purepod replica-link mapping policy connect --remote-policy policy01 pod02
```

To disconnect a policy, use the **`purepod replica-link mapping policy disconnect`** command. The previously connected source policy will no longer be replicated to the target pod, and the export policy along with its exports on the target will remain.

If the name of a replicated policy becomes in conflict with another existing policy on the target array, a digit postfix will be appended to the new export name on the target pod. For example, if the export named `policy01` already exists on the target array, the replicated policy will be named `policy01.0`.

Promotion Status of a Pod

The promotion status of a pod indicates whether a pod is promoted or demoted in the pod-to-pod, ActiveDR replication process. A promoted pod provides read/write access to the host, while a demoted pod provides read-only access. By default, the promotion status of a pod is **`promoted`** when it is initially created.

To see the promotion status of a pod, use the **`purepod list`** command. A pod can have one of the following promotion statuses:

- **`promoting`** - The promotion process of the pod is under way.
- **`promoted`** - The promotion process is complete, and the pod has been promoted. This is the default status of a pod when it is initially created.
- **`demoting`** - The demotion process of the pod is under way.
- **`demoted`** - The demotion process is complete, and the pod has been demoted.

To manage the promotion status of a pod, see

- "Demoting Pods" on page 403
- "Promoting Pods" on page 404

When you demote a pod by using the `purepod demote` command,

- The promotion status of the pod changes.

  When you create a pod initially, its promotion status is **`promoted`**, which allows the pod to provide read/write access to the host. During the demotion process, the promotion status changes to **`demoting`** initially before transitioning to **`demoted`** when the demotion process is complete. A pod with the **`demoting`** or **`demoted`** status stops receiving new writes and configuration changes from the host; that is, it allows read-only access to the host.
- An undo-demote pod is created if the demoted pod has no undo-demote pod.

  When the pod status changes to **`demoted`**, an undo-demote pod is created and placed in an eradication pending period. An undo-demote pod preserves the pod configuration and data in the state before the demotion process. Therefore, you can retrieve the data not being updated during the demotion process by using the undo-demote pod. An undo-remote pod is automatically eradicated after the eradication pending period expires.

Note: If an undo-demote pod already exists, the demotion process fails with an error.

To revert to the state before the demotion process, use the **`purepod clone`** command or the **`purepod promote`** command with the **`--promote-from`** ***`PROMOTE-FROM`*** option.

A pod can have only one undo-demote pod named *pod_name*.undo-demote. You cannot demote a pod that already has an undo-demote pod. To demote such a pod, you must first eradicate the undo-demote pod. You cannot rename an undo-demote pod; however, when you rename a demoted pod, the associated undo-demote pod automatically inherits the new pod name. For example, renaming a demoted pod podA to podB automatically changes the undo-demote pod name from podA.undo-demote to podB.undo-demote.

When you demote a pod that is the source of a replica link, you must specify either the **`--quiesce`** or **`--skip-quiesce`** option to restrict the promotion status transitions.

The **`--quiesce`** option

Demotes a pod to allow it to become a target pod after the replica-link status changes to **`quiesced`**. Setting this option ensures that all local data has been replicated to the remote pod before the pod is demoted.

You should specify this option when performing a planned failover.

The **`--skip-quiesce`** option

Demotes a pod to allow it to become a target pod without waiting for the **`quiesced`** status of the replica link. Using this option loses any data that has not been replicated to the remote pod.

When you promote a pod that was demoted, note the following conditions:

- The promotion status of the pod initially shows the **`promoting`** status, indicating the promotion process is in progress. When the promotion process is complete, the promotion status transitions to **`promoted`**.

Note: You must wait for the promotion status to transition to **`promoted`** before accessing the data in the pod.

- Promoting a pod is restricted if the replica-link status is in **`quiescing`**.

  To override this restriction, specify the **`--abort-quiesce`** option with the **`purepod promote`** command to force promotion without waiting for the quiesce operation to complete replicating data from the source. Using this option loses any data that has not been replicated and reverts the pod to its most recent recovery point.

Demoting Pods

By default, the promotion status of a pod is **`promoted`** when it is initially created. You can demote a pod to allow it to become a target pod for ActiveDR replication. To demote a pod, use the **`purepod demote`** command.

In this example, pod **`podA`** is initially created with the default promotion status of **`promoted`**, and then the **`purepod demote`** command demotes pod **`podA`**, changing the promotion status to **`demoted`**.

```
$ purepod create podA
Name Source Array   Status  Frozen At  Promotion Status  Link Count
podA -      arrayA  online  -          promoted          0
$ purepod demote podA
Name Source Array   Status  Frozen At  Promotion Status  Link Count
podA -      arrayA  online  -          demoted           0
```

If the pod is the source of a replica link, specify the **`--quiesce`** or **`--skip-quiesce`** option to restrict the promotion status transitions.

The **`--quiesce`** setting

> Demotes a pod to allow it to become a target pod after the replica-link status changes to **`quiesced`**. Using this setting ensures that all local data has been replicated to the remote pod before the demotion process.

The **`--skip-quiesce`** setting

> Demotes a pod to allow it to become a target pod without waiting for the **`quiesced`** status of the replica link. Using this setting loses all pending data to be replicated to the remote pod.

For more information, see "Promotion Status of a Pod" on page 401.

Promoting Pods

You can promote a pod that was previously demoted to allow read/write access to the host. If the pod is the target of a replica link, the pod will be updated with the latest replicated data from the journal.

To promote a pod, use the **`purepod promote`** command. You can optionally specify the following options:

`--abort-quiesce`

Specifies this option to promote the pod while the replica-link status is in the **`quiescing`** state without waiting for the quiesce operation to complete.

`--promote-from` ***`PROMOTE-FROM`***

Specifies this option to promote the pod from the specified undo-demote pod. When the promotion process is complete, the pod contains the same configuration and data as the undo-demote pod. The undo-demote pod will be eradicated.

When you promote a pod that was demoted, the promotion status of the pod initially shows the **`promoting`** status indicating the promotion process is in progress. When the promotion process is complete, the promotion status changes to **`promoted`**, as shown in the following example:

```
$ purepod promote pod
Name Source Array   Status  Frozen At  Promotion Status  Link Count
podA - arrayA online - promoting 0
$ purepod list
Name Source Array Status Frozen At Promotion Status Link Count
podA - arrayA online - promoted 0
```

You must wait for the promotion status to transition to **`promoted`** before accessing the data in the pod.

For more information, see "Promotion Status of a Pod" on page 401.

Replica Links

When you associate a source pod with a demoted pod by creating a replica link, the demoted pod becomes the target pod of the source pod. The direction of the replica link is always from the source pod to the target pod. You can create replica links in either direction between the same two FlashArrays. The target pod of a replica link cannot be on the same FlashArray as the source pod.

The target pod of a replica link tracks the data and configuration changes of the source pod, including changes to volumes, snapshots, protection groups, and protection group snapshots. Changes to the source pod are continuously replicated to the target FlashArray where they are stored in the background in a journal. When the target pod is demoted, it is updated with the latest changes from the journal every few minutes.

This form of replication does not have an impact on front-end write latency because host writes on the source are not required to wait for acknowledgment from the target FlashArray as they would with ActiveCluster replication. Therefore, writes on the source are not affected by latency on the replication network or the distance between the source and target FlashArrays.

Note the following configuration differences between a source pod and the associated target pod:

- The replicated volumes in the target pod have different serial numbers from the same volumes in the source pod.
- The target pod has different hosts and host group connections.

Replica-Link Status

Replica-link status includes the following values:

- **`baselining`**

  Indicates that the source pod is sending the initial data set.

Note: During the baseline process, promoting a target pod in the `demoted` status is not allowed.

- **`idle`**

  Indicates that write streams stop because the source pod is being demoted with the **`purepod demote --skip-quiesce`** command.
- **`paused`**

  Indicates that ActiveDR replication between the source and target pods has been paused.

  For information on how to resume the replication, see "The purepod replica-link Sub-commands" on page 399.
- **`quiescing`**

  Indicates that the source pod is not accepting new writes and the most recent writes to the source pod are currently being replicated to the target pod.

- **`quiesced`**

  Indicates that the source pod is demoted and all the new writes have been replicated to the target pod.
- **`replicating`**

  Indicates that the source pod is replicating data to the target pod.
- **`unhealthy`**

  Indicates that the current replica link is unhealthy. You should check the connection.

Lag and Recovery Point

In addition to the replica-link status, you can analyze the replication status by comparing different lags. In the case of a long lag time, you can use the recovery point to determine the most recent snapshot to recover.

- **Lag**

  The amount of time measured in seconds (CLI) or in milliseconds (REST API) that the replication target is behind the source. This is the time difference between the current time and the recovery point.
- **Recovery point**
  - Timestamp of the most recent changes that have been successfully replicated in seconds (CLI) or in milliseconds (REST API) since the UNIX epoch.
  - The value represents recovery point if the pod is promoted.
  - The value is null if the replica link is baselining.

Note: The lag and recovery point refer to the data that is successfully replicated to the journal on the target and can be recovered by promoting the pod. The current contents of the target pod might not reflect the reported recovery point. The reason is that the target pod is updated periodically only when it is demoted.

To display the current replica-link status and lag information, use the **`--lag`** option with the **`purepod replica-link list`** command, as shown in the following example. To display the history of replica-link status and lag information, specify the **`--historical`** option.

```
// From arrayA
$ purepod replica-link list --lag
Name Direction Remote Pod Remote Status      Average Lag Maximum Lag Recovery Point
podA -->       podB       arrayB replicating 2s          2s      2020-02-24 14:09:49 PDT
```

Bandwidth Requirements

There are no bandwidth requirements to maintain the near-zero RPO of ActiveDR replication. However, when the front-end data transfer rate exceeds the available bandwidth in your environment, RPO increases and ActiveDR replication automatically transitions to asynchronous mode to minimize lag.

To display the current replica-link bandwidth information, use the **`--replication`** option with the **`purepod replica-link monitor`** command. To display the replica-link bandwidth history, specify the **`--historical`** option.

The following example displays the replica-link bandwidth information from FlashArray **`arrayA`**.

```
// From arrayA
$ purepod replica-link monitor --replication
Name  Direction  Remote Pod  Remote  Time  B/s(transmitted)  B/s(received)
podA  -->        podB        arrayB  ...   2.00M             0.00M
```

Unlink Operation

ActiveDR replication associates a source pod with a target pod using a replica link. When you unlink the two pods by deleting the replica link, the data in the target journal is automatically transferred to an undo-demote pod. You can retrieve the data using the undo-demote pod; therefore, the pods may be relinked without transferring a complete baseline of all data. An undo-demote pod is automatically eradicated after the eradication pending period expires.

Performing a Failover for Fast Recovery

In the event of a disaster, the production site fails over to the disaster recovery (DR) site to minimize the downtime and to mitigate data loss associated with the disaster. To initiate a failover, promote the target pod at the recovery site to be the new source pod for your production operations.

Note: Before a failover process, you should configure ActiveDR replication by linking the source FlashArray at the production side with a target FlashArray at the recovery site to protect your mission-critical workloads. For more information, see "Setting Up ActiveDR Replication" on page 396.

To perform a failover process,

1. To speed up the failover process, you may connect the hosts to the volumes in the target pod at the recovery site before a failover.

   For more information, see "Failover Preparation" below.
2. Promote the target pod at the recovery site to be the new source pod by using the **purepod promote** command.

   For more information, see "Promoting Pods" on page 404.

   The promotion status of the target pod changes to **promoting** while the target pod is being updated with the latest changes. When the promotion process is complete, the promotion status changes to **promoted** and the target pod can now allow write access in addition to read access to the host.

   To force an immediate failover while the replica link is in the **quiescing** state without waiting for replication to complete, promote the target pod by using the **purepod promote** command with the **--abort-quiesce** option.

   The following example promotes pod **pod2** to allow read/write access to the host.

```
$ purepod promote pod2
Name Source Array Status Frozen At Promotion Status Link Count

pod2 - arrayB online - promoted 1
```

3. Shut down the hosts that are associated with the unavailable FlashArray at the original production site.
4. Start the hosts that are associated with the target pod at the recovery site.

   The recovery site becomes the new production site.

After the unavailable FlashArray is restored, you can perform the following processes:

- "Performing a Reprotect Process after a Failover" on page 1
- "Performing a Failback Process after a Failover" on page 410

Failover Preparation

To speed up and simplify a failover process, you can connect the hosts to the volumes in the target pod at the recovery site before a disaster occurs. After this connection, these replica volumes provide only read access, while the target pod is in a passive state.

In a disaster event, the source pod fails over to a designated target pod that is promoted to allow read/write access to the host. Before you start a host application, you should remount the file systems through the host OS. This refreshing process ensures that the host OS or applications have been cleared and do not contain stale or invalid data from the previous state of the volumes.

Note: The capability to access volumes in a read-only state depends on the host operating system and applications to mount and read a read-only volume. This capability varies by operating systems and versions.

Performing a Reprotect Process after a Failover

After a failover, the recovery site becomes the new production site. You can reprotect the data at the new production site by performing a reprotect process when the unavailable FlashArray at the original production site is restored. A reprotect process reverses the direction of replication after a failover so that the original production site becomes the target of replication and the new production site is protected.

To perform a reprotect process,

1. Restore the unavailable FlashArray at the original production site.
2. Demote the pod on the restored FlashArray by using the `--skip-quiesce` option with the `purepod demote` command.

   Using the `--skip-quiesce` option demotes the pod to allow it to become a target pod without waiting for the `quiescing` replica-link status. The replica link automatically reverses its direction. Note that any data that has not been replicated is preserved in the undo-demote pod for the length of the eradication pending period.

   For more information about demoting pods, see "Demoting Pods" on page 403.

During the demotion process, if the network is disconnected, the promotion status of the pod transitions to `demoting`. The reprotect process is complete when the network is restored and then the pod promotion status transitions to `demoted`.

In the following example, the target pod `pod2` on FlashArray `arrayB` becomes the new source. When the unavailable source pod `pod1` on FlashArray `arrayA` is restored and then demoted with the `--skip-quiesce` option, the replica-link reverses its direction from `pod2` to `pod1`.

```
// From arrayB
$ purepod list
Name  Source  Array   Status  Frozen At  Promotion Status  Link Count
pod2  -       arrayB  online  -          promoted          1


// From arrayA
$ purepod demote --skip-quiesce pod1
Name  Source  Array   Status  Frozen At  Promotion Status  Link Count
```

```
pod1   -       arrayA  online  -          demoted          1

$ purepod replica-link list
Name  Direction  Remote Pod  Remote  Status      Recovery Point  Lag
pod1  <--        pod2        arrayB  replicating -               0s
```

Performing a Failback Process after a Failover

After a failover, the disaster recovery (DR) site becomes the new production site. You can perform a failback process to switch the production operations back to the original production site when the unavailable FlashArray is restored. A failback and a planned failover are both scheduled failover processes. The only difference is that a failback process switches the production operations from the recovery site back to the original production site for data protection.

To fail back to the original production site,

1. Restore the unavailable FlashArray at the original production site.
2. Quiesce the applications on the hosts of the FlashArray at the new production site (recovery site).
3. Demote the pod at the new production site by using the **`--quiesce`** option with the **`purepod demote`** command.

   The promotion status of the pod changes to **`demoting`**, and the replica-link status transitions to **`quiescing`**.

   Using the **`--quiesce`** option demotes the pod to allow it to become a target pod after the replica-link status changes to **`quiesced`**. Setting this option ensures that all local data has been replicated to the remote pod before the pod is demoted.

   For more information about demoting a pod, see "Demoting Pods" on page 403.
4. Wait for the replication from the new production site to complete by monitoring the replica-link status, using the **`purepod replica-link list`** command.

   When no more data writes occur, the replica-link status changes to `quiesced` and the promotion status of the pod changes to `demoted`. Alternatively, you can monitor the lag or the recovery point to determine when the last data writes occurred.

   For more information about replica links, see "Replica Links" on page 404.
5. Promote the source pod at the original production site by using the **`purepod promote`** command.

The promotion status of the pod changes to **`promoting`** and eventually transitions to **`promoted`**. When the promotion status transitions to **`promoted`**, the replica-link reverses its direction. The command fails to run if the replica-link status is **`quiescing`**.

6 Start your production applications on the new source FlashArray.

In the following example, pod **`pod2`** on FlashArray **`arrayB`** is demoted with the **`--quiesce`** option to ensure all local data has been replicated before the demotion process. When the demotion process is complete, the promotion status of pod **`pod2`** transitions from **`demoting`** to **`demoted`**. The unavailable FlashArray **`arrayA`** is restored, and pod **`pod1`** is promoted to be the source pod. The replica link changes its direction from **`pod1`** to **`pod2`**.

```
// From arrayB
$ purepod list
Name  Source  Array   Status  Frozen At  Promotion Status  Link Count
pod2  -  arrayB       online  -          promoted          1

$ purepod demote --quiesce pod2
Name  Source  Array   Status  Frozen At  Promotion Status  Link Count
pod2  -       arrayB  online  -          demoting          1

$ purepod list
Name  Source  Array   Status  Frozen At  Promotion Status  Link Count
pod2  -       arrayB  online  -          demoted           1

// From arrayA
$ purepod promote pod1
Name  Source  Array   Status  Frozen At  Promotion Status  Link Count
pod1  -       arrayA  online  -          promoted          1

$ purepod replica-link list
Name  Direction  Remote Pod  Remote  Status       Recovery Point  Lag
pod1  -->        pod2        arrayB  replicating  -               0s
```

Performing a Planned Failover

You can perform a planned failover to transition workloads or applications from one site to another. A planned failover and a failback are both scheduled failover processes. The only difference is that a failback process switches the operation from the recovery site back to the original production site after the unavailable FlashArray has been restored.

To perform a planned failover,

1. Quiesce the applications on the hosts of the source FlashArray at the production side.
2. Demote the source pod at the production site by using `--quiesce` option with the `purepod demote` command.

   The promotion status of the pod changes to `demoting`, and the replica-link status transitions to `quiescing`.

   Setting the `--quiesce` option demotes the pod to allow it to become a target pod after the replica-link status changes to `quiesced`. Using this setting ensures that all local data has been replicated to the remote pod before the pod is demoted.
3. Wait for the replication to complete by monitoring the replica-link status.

   When no more data writes occur, the replica-link status changes to `quiesced` and the promotion status of the pod changes to `demoted`. Alternatively, you can monitor the lag or the recovery point to determine when the last data writes occurred.
4. Promote the target pod to be the new source pod by using the `purepod promote` command.

   The promotion status of the target pod changes to `promoting` while the target pod is being updated with the most recent write. When the promotion process is complete, the promotion status changes to `promoted` and the target pod can now allow write access to the host. As soon as the status transitions to `promoted`, the replica-link reverses its direction.

   Note: The command fails to run if the replica-link status is `quiescing`.
5. Start your production applications on the new source FlashArray.

Recovery Strategies for Planned Failovers

During a planned failover, if the target pod or the source pod goes offline unexpectedly, proceed with these recovery strategies as appropriate.

During the demotion process of the source pod (status `demoting`)

- If the target pod goes offline, you can promote the source pod by using the `purepod promote` command.

  The replica-link status changes to **`unhealthy`** because the target pod is unavailable. When the target pod is back online, the replica-link status changes to **`replicating`**.
- If the source pod goes offline, you can force the promotion of the target pod by using the `purepod promote` command with the **`--abort-quiesce`** option.

  The replica-link status changes to **`unhealthy`** because the source pod is unavailable. When the source pod becomes available and re-connects to the target pod, the promotion status of the source pod transitions from **`demoting`** to **`demoted`**. The replica link reverses its direction and the target pod becomes the new source pod.

After the demotion process of the source pod (status **`demoted`**)

- The replica-link status is **`quiesced`**, but the replica link has not reversed its direction. If the target pod goes offline, you can promote the source pod by using the `purepod promote` command.

  The replica-link status changes to **`unhealthy`** because the target pod is unavailable. When the target pod comes back online, the replica-link status transitions to **`replicating`**.

Performing a Test Recovery Process

You can run a test recovery process for testing purposes such as evaluating your disaster recovery strategy or checking your applications. Simulating a failover in the test environment allows you to assess if the recovery procedure achieves the designated RPO and RTO values in the event of a disaster. This feature allows failover testing and promotion of a target pod without disrupting replication to maintain the RPO.

1. Configure ActiveDR replication by associating a source pod with a target pod for data protection as described in "Setting Up ActiveDR Replication" on page 396.
2. Promote the target pod by using the **`purepod promote`** command.

   The promotion status of the target pod changes to **`promoting`**. When the promotion process is complete, the promotion status changes to **`promoted`**. After being promoted, the target pod can now provide read/write access to the host. The source pod continues replicating data in the background in a journal without periodically updating the promoted target pod.
3. Bring up the host on the target pod.

   The data presented to the host will be the point in time when the last data was replicated and before the target pod was promoted.

4 Perform your tests on the target pod.

In the meantime, replication continues streaming writes in the background in a journal without periodically updating the target pod. Therefore, you maintain the RPO without losing any data.

5 When the test is complete, terminate the test recovery process by demoting the target pod.

When you demote the target pod,

- The test data written to the target pod will be discarded. However, the data will be saved in an undo-demote pod that is placed in an eradication pending period.
- ActiveDR replication resumes streaming writes to the target pod.

During an actual failover, when the source FlashArray is offline, ActiveDR replication is disrupted so that no new writes are replicated from the source pod to the target pod. However, if both the source and target FlashArrays are still online and connected as in the test recovery process, ActiveDR replication streams new writes in the background in a journal without periodically updating the target pod to maintain the RPO. You can optionally choose to stop ActiveDR replication from the source pod to the target pod.

In the following example, the target pod **`pod2`** is promoted for test recovery purpose. After the test is finished, pod **`pod2`** is demoted, its promotion status changes to **`demoted`**, and the pod **`pod2.undo-demote`** is automatically created for storing the test data for the length of the eradication pending period.

```
$ purepod promote pod2
Name  Source  Array   Status  Frozen At  Promotion Status  Link Count
pod2 - arrayB online - promoted 1

$ purepod demote pod2
Name  Source  Array   Status  Frozen At  Promotion Status  Link Count
pod2 - arrayB online - demoted 1

$ purepod list —pending-only
Name             Source  Array  Time Remaining Status  Promotion Status  Link Count
pod2.undo-demote -       arrayB 23:59:53       online  promoted          0
```

Listing Pod Details

The **`purepod list`** command displays a list of all pods on the local array. If a pod is stretched, the details of its peer array are also displayed.

The status of the array (Status) determines the ability of a pod to synchronously replicate data. The status details are unique to the local array and will differ from the status details when viewed from a peer array, so what one array knows about its own status and the status of its peer arrays will differ from what its peer arrays know.

Possible statuses on the local array include:

`Online`
: The array is online and has the latest data for the pod. The array can handle I/O to the pod and take over during a high availability event.

`Offline`
: The array is experiencing problems and may not have the latest data for the pod. The array cannot handle I/O to the pod and cannot take over during a high availability event.

`Resyncing`
: The array is actively getting the latest data for the pod so that it becomes fully synchronized with its peer array. During the resyncing process, the array cannot handle I/O to the pod. Once the arrays are fully synchronized, the array changes to Online status.

`Unknown`
: The status of the peer array is unknown because this array is offline and cannot determine the state of the pod on the peer array. Only the peer array can ever be in unknown status; this unknown status is unique to the local array and will differ when viewed from its peer array.

During normal operation, the status of each pod is **`online`**, meaning the arrays are online and have the latest data for the pod. If the pod is stretched over two arrays, the data is synchronously replicating as expected.

In the following example, local array **`array01`** has two pods. Pod **`pod01`** resides locally and is unstretched, while pod **`pod02`** is stretched between the local array and **`array02`**. Both arrays are online, which means that they are serving I/O and operating as expected.

```
$ purepod list
Name    Source  Array   Status  Frozen At
pod01   -       array01 online  -
                array02 online  -
pod02   -       array01 online  -
```

When a pod is stretched to a peer array, the array goes into **`resyncing`** state as it actively gets the latest pod data. Once the peer array becomes synchronized with the local array, the status on the peer array changes to **`online`**. Writes to the pod coming into either array are immediately synchronized and seen on both arrays.

When the arrays lose contact with each other, they race to the mediator. The first array to reach the mediator becomes the array that stays **`online`** to continue serving I/O while its peer array goes **`offline`**. At this point, any stretched pods fall out of sync. The data in the offline array becomes frozen. Since the offline array has no contact with its peer array, it does not know its peer's status, so it gives its peer a status of unknown.

The mediation process happens per pod, so if two arrays share multiple pods and the arrays lose contact with each other, one array could be online for some pods but offline for other pods, and vice versa for the other array.

In the following example, arrays **`array01`** and **`array02`** lose contact with each other. Both arrays race to the mediator, where **`array02`** wins and stays online. **`array01`** goes offline and becomes frozen at 2017-11-14 17:25:40 PST. Since **`array01`** is offline, it cannot determine the status of peer **`array02`**, so its status is **`unknown`**. Here are the status details for both arrays, as seen from the offline array **`array01`**:

```
$ purepod list
Name     Source  Array   Status  Frozen At
pod01    -       array01 offline 2017-11-14 17:25:40 PST
         -       array02 unknown -
```

When the peer arrays re-establish contact with each other, the array that was **`online`** throughout the outage starts replicating pod data to its peer array until the pod is once again in sync, at which time the **`offline`** array returns to **`online`** status and pod activity resumes across both arrays.

Here are the status details for both arrays after array01 comes back online:

```
$ purepod list
Name     Source  Array   Status  Frozen At
pod01    -       array01 online  -
         -       array02 online  -
```

Include the **`--mediator`** option to display the name and version number of the mediator that is used to determine which array will continue data services should an outage occur in the environment.

By default, the mediator is set to **`purestorage`**, representing the Pure1© Cloud Mediator.

The mediator status indicates if the mediator is available to mediate a high availability event. Possible mediator statuses include:

`Online`

The array is successfully communicating with the mediator, and the mediator is available to mediate a high availability event.

`Unreachable`

The array cannot reach the mediator, either due to network issues or because the mediator is down. When a mediator is unreachable, ActiveCluster replication continues to function provided all arrays are healthy and communicating, but a high availability event without mediator access can result in an outage. If an array cannot successfully communicate with the mediator, an alert is generated. If after verifying your network connections the mediator status is still unreachable, contact Pure Storage Technical Services.

If an array cannot successfully communicate with the mediator, an alert is generated.

In the following example, local array **`array01`** has two pods. Pod **`pod01`** is unstretched, while pod **`pod02`** is stretched to peer array **`array02`**. All of the arrays over which the pods are stretched are online and serving I/O. Since pod **`pod02`** is stretched to peer array **`array02`**, any I/O that comes into one of the arrays is replicated to the other array so that the pod is always in sync.

```
$ purepod list --mediator
Name  Source  Mediator     Mediator Version  Mediator Status  Array    Status  Frozen At
pod01 -       purestorage  1.0               online           array01  online  -
pod02 -       purestorage  1.0               online           array01  online  -
      -       purestorage  1.0               online           array02  online  -
```

Include the **`--on`** option to display a list of pods that are on the specified remote array but not stretched to this array. Pods that are stretched to this array will not appear in the list. REMOTE can be set to (*) to represent all remote arrays.

In the following example, the current array, array **`pure-abc`**, is connected to three arrays (**`pure-001`**, **`pure-021`**, and **`pure-050`**) with some pods that are not stretched to the current array. Pod **`pod01`** is stretched over arrays **`pure-001`** and **`pure-050`**. Pod **`pod02`** is on array **`pure-001`** and not stretched to any other arrays. Pod **`pod03`** is stretched over arrays **`pure-001`** and **`pure-021`**.

```
//From array pure-abc
purepod list --on *

Name Array

pod01 pure-001

pure-050

pod02 pure-001

pod03 pure-001
```

```
pure-021
```

Include the **`--space`** option to display size and space consumption details for all volumes and snapshots in each pod on the local array. The following details are displayed for a purchased array:

Provisioned Size

The sum of the provisioned sizes of objects in the pod. The value represents the storage capacity reported to hosts.

Size is capped by the container’s quota limit. If the pod is above the quota limit, this size reports the amount of logical space the pod has consumed.

Virtual

The amount of data that the host has written to the pod as perceived by the array, before any data deduplication or compression.

Thin Provisioning

Percentage of volume sectors that do not contain host-written data because the hosts have not written data to them or the sectors have been explicitly trimmed.

Data Reduction

Ratio of mapped sectors within a volume versus the amount of physical space the data occupies after data compression and deduplication. The data reduction ratio does not include thin provisioning savings.

For example, a data reduction ratio of 5:1 means that for every 5 MB the host writes to the array, 1 MB is stored on the array's flash modules.

Total Reduction

Ratio of provisioned sectors within a volume versus the amount of physical space the data occupies after reduction via data compression and deduplication *and* with thin provisioning savings. Total reduction is data reduction with thin provisioning savings.

For example, a total reduction ratio of 10:1 means that for every 10 MB of provisioned space, 1 MB is stored on the array's flash modules.

Unique

Physical space that is occupied by data of both volumes and file systems in the pod after data reduction and deduplication, but excluding metadata and snapshots.

Snapshots

Physical space occupied by data unique to one or more snapshots.

Shared

Physical space occupied by deduplicated data, meaning that the space is shared with other volumes and snapshots as a result of data deduplication occurring within a particular pod.

Replication

Physical system space used to accommodate pod-based replication features, including failovers, resync, and disaster recovery testing.

Total

Total physical space occupied by system, shared space, volume, and snapshot data.

The following details are displayed with the `--space` option on subscription storage:

Name

Name of the pod.

Provisioned Size

The sum of the sizes of all volumes and file systems in the pod.

Displays a '-' sign for arrays when a file system in the pod has unlimited provisioned size.

Virtual

The amount of data that the host has written to the pod as perceived by the array, before any data deduplication or compression.

Unique

Effective used capacity data of both volumes and file systems in the pod after removing clones, but excluding metadata and snapshots.

Snapshots

Storage consumed by data unique to one or more snapshots.

Shared

Storage consumed by cloned data, meaning that the space is shared with cloned volumes and snapshots as a result of data deduplication.

Replication

Storage consumed for either of the ActiveDR or ActiveCluster pod-based replication features.

Total

Total effective used capacity containing user data, including Shared, Snapshots, and Unique storage.

In the following example (on a purchased array), the Shared column for the `purepod list --space` command displays the amount of space shared between volumes within a particular pod.

```
$purevol list --space
Name  Size   Thin Provisioning   Data Reduction    Total Reduction    Unique
podA  200M   0%                  8.2 to 1          8.2 to 1           0.00
Snapshots    Shared   Replication   Total
0.00         36.48M   0.00          36.48M
```

In the following example, pod **`pod01`** is stretched across arrays **`array01`** and **`array02`**. Both arrays are online and the pod data between the two arrays is in sync, taking up space across both arrays.

Note that space consumption differs between the two arrays due to reasons such as compression, deduplication, hardware differences, resources, and so on.

```
//From array01
$ purevol list --space
Name          Provisioned Size   Used Provisioned   Thin Provisioning
pod01::vol01  1T                 53G                100%
pod01::vol02  1T                 47G                100%
pod01::vol03  1T                 47G                100%

Data Reduction  Total Reduction  Unique ... Total
7.6 to 1        >100 to 1        34.43M ... 34.43M
6.3 to 1        >100 to 1        28.64M ... 28.64M
6.3 to 1        >100 to 1        28.07M ... 28.07M

//From array02
$ purevol list --space
Name          Provisioned Size   Used Provisioned   Thin Provisioning
pod01::vol01  1T                 60G                100%
pod01::vol02  1T                 52G                100%
pod01::vol03  1T                 48G                100%

Data Reduction  Total Reduction  Unique ... Total
5.6 to 1        >100 to 1        46.65M ... 46.65M
5.6 to 1        >100 to 1        32.50M ... 32.50M
6.0 to 1        >100 to 1        29.67M ... 29.67M
```

Include the **`--total`** option with the **`--space`** option to display the total size and space consumption or effective used capacity for all pods.

Include the **--footprint** option to display the maximum amount of physical space or effective used capacity the pod would take up if moved or copied to a different array. The footprint metric is mostly used for capacity planning.

In the following example on a purchased array, if pod **pod01** was stretched or migrated to an empty array, it would take up 2 gigabytes of (physical) space.

```
purepod list --footprint
Name Footprint
pod01 2.00G
```

Include the **--total** option with the **--footprint** option to add a summary line that displays the combined physical space or effective used capacity of all listed pods.

The **purepod list --pending** command displays a list of all pods, including ones that have been destroyed and are in the eradication pending state. The **purepod list --pending-only** command only displays a list of pods that have been destroyed and are in the eradication pending state.

The **purepod listobj** command displays a list of pod attributes, either in whitespace or comma-separated form, suitable for scripting. Include the **--type** option to list FlashArray objects that are associated with one or more pods. The **--type** option accepts the following arguments:

`--type array`

Display a list of arrays over which the specified pods reside, either stretched or unstretched. If no pods are specified, the list contains the names of all arrays that contain pods.

`--type pgroup`

Displays a list of protection groups that reside in the specified pods. If no pods are specified, the list contains the names of all protection groups that reside in pods.

`--type pod` (default if **--type** option not specified)

Displays a list of pods that have been created on the array or stretched to this array from another array. If no pods are specified, the list contains the names of all pods that reside on this array.

`--type vol`

Displays a list of volumes that reside in the specified pods. If no pods are specified, the list contains the names of all volumes that reside in pods.

The **purepod eradication-config list** command displays the SafeMode configuration status for each pod. Valid options include **--pending** which includes destroyed pods that are in the eradication pending state, or **--pending-only** which only displays destroyed pods that are in the eradication pending state. Manual Eradication is “all-disabled” if SafeMode is enabled array-wide, “partially-disabled” if one or more non-empty protection groups are protected with SafeMode retention lock. Otherwise, Manual Eradication is “all-enabled”.

Monitoring Pod I/O Performance

The `purepod monitor` command displays real-time and historical I/O performance information for all of the specified pods. The output includes the following bandwidth, IOPS, and latency data:

- **Name**: Object name.
- **Time**: Current time.
- **B/s**: Bandwidth. Number of bytes read/written.
- **op/s**: IOPS. Number of read, write, or mirrored write requests processed per second.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve a read, write, or mirrored write I/O request. Latency does not include SAN time or QoS rate limit time.
- **B/op**: IOPS. Average I/O size per read, write, mirrored write, and both read and write (all) operations. Must include the `--size` option to see the B/op columns.

Include the `--interval` option to specify the number of seconds between each real-time update. Include the `--repeat` option to specify the number of times to repeat the real-time update. The `--interval` and `--repeat` options can be combined.

Include the `--array` option to break down performance data by the array to which the I/O is directed.

Include the `--latency` option to display real-time and historical I/O latency information for all of the volumes in each pod. The `purepod monitor --latency` output includes the following data:

- **SAN us/op**: SAN time. Average time, measured in microseconds, required to transfer data between the initiator and the array. Slow data transfers will result in higher SAN times.
- **Queue us/op**: Queue time. Average time, measured in microseconds, that an I/O request spends in the array waiting to be served. The time is averaged across all I/Os of the selected types.

- **QoS Rate Limit us/op**: QoS rate limit time. Average time, measured in microseconds, that reads, writes, or mirrored writes spend in queue as a result of bandwidth limits reached on one or more volumes.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve a read, write, or mirrored write I/O request. Latency does not include SAN time or QoS rate limit time.

Include the **`--mirrored`** option to display performance or latency data for mirrored writes (i.e., writes to volumes in stretched pods). If one array of a stretched pod is offline, the writes processed on the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again.

By default, the `purepod monitor` command displays real-time performance data. The `purepod monitor --historical` command displays historical performance data over any of the following ranges of time: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, or 1 year.

Include the **`--protocol`** option to display the pod's I/O performance for the specific protocol. Specify the protocol type as **`smb`** or **`nfs`**. The **`--protocol-group`** option displays the pod's I/O performance for the specific protocol group. Specify the protocol group type as **`block`** or **`file`**.

To display the bandwidth information of each replication type (continuous, resync, and synchronous) and the total bandwidth, use the **`--replication`** option with the `purepod monitor` command, as shown in this example.

```
$ purepod monitor --replication
Name  Time                     B/s (continuous)  B/s (resync)  B/s (sync)  B/s (total)
pod1  2020-02-06 16:29:38 PST  186.00B           0.00          0.00        186.00B
pod3  2020-02-06 16:29:38 PST  186.00B           0.00          0.00        186.00B
```

To display the bandwidth information of each replication type and the total bandwidth by the array (both local and remote), use the **`--array`** option, as shown in this example.

```
$ purepod monitor --replication --array
Name  Array   B/s(continuous)  B/s(resync)  B/s(sync)  B/s(total)
podA  arrayA  7.00M            3.00M        4.00m      14.00M
podA  arrayB  7.00M            3.00M        4.00m      14.00M
```

To display the bandwidth information for a specific replication type such as continuous, resync, or synchronous, use the **`--continuous`**, **`--resync`**, or **`--sync`** option, respectively, with the **`--replication`** option. The following example shows the bandwidth information of ActiveDR replication.

```
$ purepod monitor --replication --continuous
```

```
Name  B/s(toRemote)  B/s(fromRemote)  B/s(total)
podA  2.00M          5.00M            7.00M
```

Examples

Example 1

```
purepod create POD01
```

Creates a pod named **`POD01`**.

Example 2

```
purearray connect --management-address ARRAY01 --type sync-replication --connection-key
purepod add --array ARRAY01 POD01
```

Connects the current array to remote array **`ARRAY01`**, and then stretches **`POD01`** to array **`ARRAY01`**.

Example 3

```
purearray connect --management-address ARRAYDC1 --type sync-replication
--connection-key
purepod create --failover-preference ARRAYDC1 POD02
purepod add --array ARRAYDC1 POD02
```

Connects the current array to remote array **`ARRAYDC1`**, creates a pod named **`POD02`** with fail-over preference given to array **`ARRAYDC1`**, and then stretches **`POD02`** to array **`ARRAYDC1`**.

Example 4

```
purepod remove --array ARRAY01 POD01
```

Unstretches **`POD01`** from array **`ARRAY01`**, creating a destroyed pod with name **`POD01.restretch`** on array **`ARRAY01`**.

Example 5

```
purepod clone POD10 POD10-CLONE
```

Clones pod **`POD10`** to create a new pod named **`POD10-CLONE`**.

Example 6

```
purepod clone --dry-run POD10 POD10-CLONE2
```

Checks array health to determine the pod clone operation can succeed. Does not actually create the clone.

Example 7

```
purepod clone --allow-throttle POD10 POD10-CLONE2
```

First, checks array health to determine the pod clone operation can succeed. If array health is optimal, clones pod **POD10** to create a new pod named **POD10-CLONE2**. If array health would not support the clone operation, the clone is not attempted.

Example 8

```
purepod destroy POD10::VOL20
purepgroup destroy POD10::PGROUP05 POD10::PGROUP06
purepod destroy POD10
```

Destroys volume **VOL20** and protection groups **PGROUP05** and **PGROUP06** in **POD10**, clearing **POD10** of all volumes and protection groups. Destroys empty pod **POD10**.

Example 9

```
purepod list --mediator
Name Source Mediator Mediator Version Array Status Frozen At Mediator Status
pod01 - purestorage 1.0 array01 online - unavailable
array02 online - unavailable
```

Displays a list of all pods on the local array and the local array's mediator details. The local array **array01** has one pod named **pod01**. The pod is stretched, so the details of peer array **array02** are also displayed.

Example 10

```
purepod list --on ARRAY-001
Name Array
pod01 ARRAY-001
ARRAY-050
pod02 ARRAY-001
```

```
pod03 ARRAY-001
ARRAY-021
```

Displays a list of pods that are on remote array **ARRAY-001**, but not stretched to this array.

Example 11

```
purepod monitor --replication --array --historical 1h
```

Displays the historical bandwidth information of each replication type and the total by array (both local and remote arrays) over the past one hour.

Example 12

```
purepod demote --quiesce pod2
```

Demotes pod **pod2** to allow it to become a target pod after the replica-link status changes to **quiesced**. Setting this option ensures that all local data has been replicated to the remote pod before pod **pod2** is demoted.

Example 13

```
purepod promote --abort-quiesce pod1
```

Promotes pod **pod1** without waiting for the **quiesced** replica-link state.

Example 14

```
purepod replica-link list --lag --historical 1h
```

Displays the history of replica-link status and lag information over the past one hour from the local array.

Example 15

```
purepod replica-link monitor --replication --historical 1h
```

Displays the history of replica-link bandwidth over the past one hour from the local array.

Example 16

```
purepod list --space
```

Displays the size and space consumption or effective used capacity details for all volumes, file systems, and snapshots in each pod on the local array.

Example 17

```
purepod create --quota-limit 10G pod10
```

Creates pod **pod10** with an upper limit on the amount of data that pod can contain set to 10G.

Example 18

```
purepod setattr --quota-limit " " pod10
```

Removes the quota limit for **pod10**.

Example 19

```
purepod setattr --quota-limit 10G --ignore-usage pod10
```

Sets a quota limit at 10G, and allows the quota limit to be set even if there is more than 10G worth of usage in the pod already.

Example 20

```
purepod move --to pure01 --from realm-1 --with-hosts host1,host2 realm-1::pod1
```

Moves `pod1` from `realm-1` to the array `pure01` and also moves `host1` and `host2`.

See Also

purehgroup, purehgroup-connect, purehost, purehost-connect, purepgroup, purevol

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purepolicy

purepolicy – manages the creation, naming, and listing of policies

Synopsis

purepolicy alert-watcher create [--context ***REMOTE***] ***ALERT-POLICY***

purepolicy alert-watcher delete [--context ***REMOTE***] ***ALERT-POLICY***

purepolicy alert-watcher disable [--context ***REMOTE***] ***ALERT-POLICY***

purepolicy alert-watcher enable [--context ***REMOTE***] ***ALERT-POLICY***

purepolicy alert-watcher list [--cli | --csv | --nvp] [--notitle] [--raw] [--filter ***FILTER***] [--page] [--page-with-token] [--token ***TOKEN***] [--limit ***LIMIT***] [--sort ***SORT***] [--context ***REMOTES***] [***ALERT-POLICY***]

purepolicy alert-watcher rename [--context ***REMOTE***] ***OLD-NAME NEW-NAME***

purepolicy alert-watcher rule add [--context ***REMOTE***] [--address ***EMAIL-ADDRESS***] [--minimum-notification-severity ***SEVERITY***] [--include-codes ***CODES***] [--exclude-codes ***CODES***] ***ALERT-POLICY***

purepolicy alert-watcher rule list [--page-with-token] [--token ***TOKEN***] [--context ***REMOTES***]

purepolicy alert-watcher rule remove [--address **EMAIL-ADDRESS**] [--context ***REMOTE***] ***ALERT-POLICY***

purepolicy alert-watcher rule setattr [--context ***REMOTE***] [--address ***EMAIL-ADDRESS***] [--minimum-notification-severity ***SEVERITY***] [--include-codes ***CODES*** | --exclude-codes ***CODES***] ***ALERT-POLICY***

purepolicy alert-watcher rule test ***EMAIL-ADDRESS***

purepolicy audit file list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--member] [--pending | --

pending-only] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--context ***REMOTES***] [***NAME*** ...]

purepolicy audit file create [--context ***REMOTE***] --targets ***LOG_TARGETS*** ***NAME***...

purepolicy audit file copy [--context ***REMOTE***] ***SOURCE*** ***TARGETPOLICY***...

purepolicy audit file delete [--context ***REMOTE***] ***NAME***...

purepolicy audit file add --dir ***DIR*** [--context ***REMOTE***] ***POLICY***

purepolicy audit file remove --dir ***DIR*** [--context ***REMOTE***] ***POLICY***...

purepolicy audit file enable [--context ***REMOTE***] ***NAME***...

purepolicy audit file disable [--context ***REMOTE***] ***NAME***...

purepolicy audit file rename [--context ***REMOTE***] ***OLD-NAME*** ***NEW-NAME***

purepolicy audit file setattr [--context ***REMOTE***] [--targets ***TARGETS***] ***NAME***...

purepolicy autodir add --dir ***DIR*** [--context ***REMOTE***] ***POLICY***

purepolicy autodir copy [--context ***REMOTE***] ***SOURCE*** ***TARGETPOLICY***...

purepolicy autodir create[--context ***REMOTE***] ***NAME***...

purepolicy autodir delete [--context ***REMOTE***] ***NAME***...

purepolicy autodir disable [--context ***REMOTE***] ***NAME***...

purepolicy autodir enable [--context ***REMOTE***] ***NAME***...

purepolicy autodir list [--cli | --csv | --nvp] [--notitle] [--raw] [--filter ***FILTER***] [--page] [--page-with-token] [--token ***TOKEN***] [--limit ***LIMIT***] [--sort ***SORT***] [--member] [--pending | --pending-only] [--context ***REMOTES***] [***NAME***...]

purepolicy autodir remove --dir ***DIR*** [--context ***REMOTE***] ***POLICY***...

purepolicy autodir rename [--context ***REMOTE***] ***OLD-NAME*** ***NEW-NAME***

purepolicy list [--csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--member] [--pending | --pending-only] [--context ***REMOTES***] [***NAME***...]

```
purepolicy management-access add {--admin ADMIN | --ds-mapping DS_
MAPPING} [--context REMOTE] POLICY
```

```
purepolicy management-access copy SOURCE [--context REMOTE]
TARGETPOLICY...
```

```
purepolicy management-access create [--context REMOTE] {--array ARRAY
| --realm REALM} --role {admin | storage | support | viewer} [--
aggregation-strategy {all-permissions | least-common -permissions}]
NAME...
```

```
purepolicy management-access delete [--context REMOTE] NAME...
```

```
purepolicy management-access disable [--context REMOTE] NAME...
```

```
purepolicy management-access enable [--context REMOTE] NAME...
```

```
purepolicy management-access list [--cli | --csv | --nvp] [--notitle]
[--raw] [--filter FILTER] [--page] [--page-with-token] [--token TOKEN]
[--limit LIMIT] [--sort SORT] [--member] [--pending | --pending-only]
[--context REMOTES] [NAME...
```

```
purepolicy management-access remove (--admin ADMIN | --ds-mapping DS_
MAPPING) [--context REMOTE] POLICY...
```

```
purepolicy management-access rename [--context REMOTE] OLD-NAME NEW-
NAME
```

```
purepolicy management-access setattr [--context REMOTE] [--array ARRAY
| --realm REALM] [--role {admin | storage | support | viewer}] [--
aggregation-strategy {all-permissions | least-common-permissions}]
NAME...
```

```
purepolicy nfs add --dir DIR --export-name EXPORT_NAME [--context
REMOTE] POLICY
```

```
purepolicy nfs copy [--context REMOTE] SOURCE TARGETPOLICY...
```

```
purepolicy nfs create [--context REMOTE] [--connect | --disconnect] [-
-disable-user-mapping] NAME...
```

```
purepolicy nfs delete [--context REMOTE] NAME...
```

```
purepolicy nfs disable [--context REMOTE] [--user-mapping] NAME...
```

```
purepolicy nfs enable [--context REMOTE] [--user-mapping] NAME...
```

purepolicy nfs list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--member] [--pending | --pending-only] [--mapping] [--context ***REMOTES***] [***NAME...***]

purepolicy nfs remove --dir ***DIR*** [--context ***REMOTE***] ***POLICY...***

purepolicy nfs rename [--context ***REMOTE***] ***OLD-NAME NEW-NAME***

purepolicy nfs rule add [--context ***REMOTE***] [--client ***CLIENT***] [--all-squash | --no-root-squash | --root-squash] [--rw | --ro] [--anonuid ***ANONUID***] [--anongid ***ANONGID***] [--version ***VERSION***] [--security ***SECURITY***] ***POLICY...***

purepolicy nfs rule list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--pending | --pending-only] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--context ***REMOTES***] [***POLICY...***]

purepolicy nfs rule remove --rule-name ***RULE*** [--context ***REMOTE***] ***POLICY***

purepolicy nfs setattr [--context ***REMOTE***] [--version ***VERSION***] [--security ***SECURITY***] ***NAME...***

purepolicy password disable [--context ***REMOTE***] [--username-check] [--dictionary-check] ***NAME...***

purepolicy password enable [--context ***REMOTE***] [--username-check] [--dictionary-check] ***NAME...***

purepolicy password list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--context ***REMOTES***] [***NAME...***]

purepolicy password setattr [--context ***REMOTE***] [--min-password-length ***LENGTH***] [--max-login-attempts ***ATTEMPT***] [--lockout-duration ***DURATION***] [--min-characters-per-group ***COUNT***] [--min-character-groups ***COUNT***] [--password-history ***COUNT***] [--min-password-age ***DURATION***] ***NAME...***

purepolicy smb add --dir ***DIR*** --export-name ***EXPORT_NAME*** [--context ***REMOTE***] ***POLICY***

purepolicy smb copy [--context ***REMOTE***] ***SOURCE TARGETPOLICY...***

purepolicy smb create [--context ***REMOTE***] [--connect | --disconnect] [--access-based-enumeration] ***NAME...***

purepolicy smb delete [--context ***REMOTE***] ***NAME...***

purepolicy smb disable [--context ***REMOTE***] [--access-based-enumeration] ***NAME...***

purepolicy smb enable [--context ***REMOTE***] [--access-based-enumeration] ***NAME...***

purepolicy smb list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--member] [--pending | --pending-only] [--mapping] [--context ***REMOTES***] [***NAME...***]

purepolicy smb remove --dir ***DIR*** [--context ***REMOTE***] ***POLICY...***

purepolicy smb rename [--context ***REMOTE***] ***OLD-NAME NEW-NAME***

purepolicy smb rule add [--context ***REMOTE***] [--client ***CLIENT***] [--anonymous-access-allowed] [--smb-encryption-required] ***POLICY...***

purepolicy smb rule list [--cli | --csv | --nvp] [--notitle] [--page | --pending | --pending-only] [--page-with-token] [--token ***TOKEN***] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--context ***REMOTES***] [***POLICY...***]

purepolicy smb rule remove --rule-name ***RULE*** [--context ***REMOTE***] ***POLICY***

purepolicy quota add --dir ***DIR*** [--ignore-usage] [--context ***REMOTE***] ***POLICY***

purepolicy quota copy [--context ***REMOTE***] ***SOURCE TARGETPOLICY...***

purepolicy quota create [--context ***REMOTE***] ***NAME...***

purepolicy quota delete [--context ***REMOTE***] ***NAME...***

purepolicy quota disable [--context ***REMOTE***] ***NAME...***

purepolicy quota enable [--context ***REMOTE***] [--ignore-usage] ***NAME...***

purepolicy quota list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--member] [--pending | --pending-only] [--context ***REMOTES***] [***NAME...***]

purepolicy quota remove --dir ***DIR*** [--context ***REMOTE***] ***POLICY...***

purepolicy quota rename [--context ***REMOTE***] ***OLD-NAME NEW-NAME***

purepolicy quota rule add [--context ***REMOTE***] [--quota-limit ***QUOTA_ LIMIT***] [--notifications ***NOTIFICATIONS***] [--enforced] [--ignore-usage] ***POLICY...***

purepolicy quota rule disable [--context ***REMOTE***] --enforced --rule-name ***RULE POLICY...***

purepolicy quota rule enable [--context ***REMOTE***] --enforced --rule-name ***RULE*** [--ignore-usage] ***POLICY...***

purepolicy quota rule list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--pending | --pending-only] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--context ***REMOTES***] [***POLICY...***]

purepolicy quota rule remove --rule-name ***RULE*** [--context ***REMOTE***] ***POLICY***

purepolicy quota rule setattr [--context ***REMOTE***] [--quota-limit ***QUOTA_ LIMIT***] [--notifications ***NOTIFICATIONS***] [--ignore-usage] --rule-name ***RULE POLICY...***

purepolicy snapshot add --dir ***DIR*** [--context ***REMOTE***] ***POLICY***

purepolicy snapshot copy [--context ***REMOTE***] ***SOURCE TARGETPOLICY...***

purepolicy snapshot create [--context ***REMOTE***] ***NAME...***

purepolicy snapshot delete [--context ***REMOTE***] ***NAME...***

purepolicy snapshot disable [--context ***REMOTE***] ***NAME...***

purepolicy snapshot enable [--context ***REMOTE***] ***NAME...***

purepolicy snapshot list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--member] [--pending | --pending-only] [--context ***REMOTES***] [***NAME...***]

purepolicy snapshot remove --dir ***DIR*** [--context ***REMOTE***] ***POLICY...***

purepolicy snapshot rename [--context ***REMOTE***] ***OLD-NAME NEW-NAME***

purepolicy snapshot rule add [--context ***REMOTE***] --every ***EVERY*** [--at ***AT***] --keep-for ***KEEP_FOR*** --client-name ***CLIENT_NAME*** [--suffix ***SUFFIX***] ***POLICY...***

purepolicy snapshot rule list [--cli | --csv | --nvp] [--notitle] [--page] [--page-with-token] [--token ***TOKEN***] [--pending | --pending-only] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--context ***REMOTES***] [***POLICY...***]

purepolicy snapshot rule remove --rule-name ***RULE*** [--context ***REMOTE***] ***POLICY***

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--access-based-enumeration`

Access-Based Enumeration (ABE) is disabled by default when an SMB policy is created. Use the option to enable ABE when creating a new SMB policy, or with the **`enable`** or **`disable`** subcommands for existing SMB policies.

`--admin` ***ADMIN***

A comma-separated list of one or more local administrator names. Used with `purepolicy management-access add` and `purepolicy management-access remove` to add or remove access policies to or from local administrators.

`--aggregation-strategy`

Defines the behavior when multiple access policies are attached to a user or group. Optional, defaults to all-permissions. The following values are supported:

`least-common-permissions`: The user or group is assigned only the permissions of most restrictive access policy.

`all-permissions`: The user or group is assigned all permissions of all access policies, combined. Default.

`--all-squash`

All users mounting this export are mapped to anon (UID/GID anonuid/anongid). This option is mutually exclusive with the **`--root-squash`** and **`--no-root-squash`** options.

`--anongid` ***ANONGID***

Specifies the GID to use when squashing groups, when NFS user mapping is disabled. This argument is ignored when NFS user mapping is enabled, as GID is provided by the directory services, defaulting to the user's primary group's GID.

`--anonuid` ***ANONUID***

Specifies the UID to use when squashing user UID.

`--anonymous-access-allowed`

Can be used when adding an SMB rule to a policy, to allow anonymous access. Clients that do not provide credentials are allowed access to the export. If omitted, anonymous users are restricted access.

`--array` ***ARRAY***

With `purepolicy management-access create` and `purepolicy management-access setattr`, sets the scope of the access policy to the specified array. In this initial release, only the local array is supported.

`--at` ***AT***

Optionally, when adding a snapshot rule, specifies the time of day a snapshot is taken, specified in the format **`HH[am|pm]`** for example **`11pm`** or **`23`**. If the **`--at`** option is specified then **`--every`** must be a multiple of day (i.e., a 24-hour period).

If the time was previously set, specify **`""`** to clear the time.

`--client` ***CLIENT***

Used when adding an SMB or NFS rule. Allows only clients that match the specified hostname, IPv4, or IPv6. For example, **`*.cs.foo.edu`**, **`192.168.10.2`**, **`192.168.10.0/24`**, **`2001:db8::7873`**, or **`2001:db8::/32`**. If omitted, the default value is “*****”, which means no clients are restricted.

A client's access is granted only if the value satisfies one of the following: Matches the IP or IP CIDR, if the item can be converted into an IP (CIDR). Matches the full hostname (either with or without wildcard characters **`*`** and **`?`**) that the client IP belongs to, if the item cannot be converted into an IP (CIDR).

Valid IP (CIDR)

A valid IP (CIDR) item is an IPv4 or IPv6 address with or without prefix length. Examples of each type are shown in this table.

Table 4. Example IP Addresses

Address Type	Example
IPv4 with prefix	`192.168.0.0/16`
IPv4 without prefix	`192.168.0.1`
IPv6 with prefix	`fd01::1:0/112`
IPv6 without prefix	`fd01::123`

Valid Hostname

A valid hostname is one of the following: A fully qualified domain name (FQDN), for example `mycomputer.mydomain`.A hostname with wildcard characters, for example `mycomputer*`, where `*` matches zero or more characters, or `mycomputer.m?domain`, where `?` matches one character.

Table 5. Examples of CLIENT Values

To Grant Access To ...	Client Value to Use
All IPs	*
All IPv4 addresses (and only IPv4 addresses)	`0.0.0.0/0`
All IPv6 addresses (and only IPv6 addresses)	`::/0`
A single IPv4 address	The IP by itself or with a prefix length 32. Examples: `192.168.0.1`, `192.168.0.1/32`
A single IPv6 address	The IP by itself or with a prelix length 128. Examples: `fd01::123`, `fd01::123/128`
A range of IPv4 addresses	IP with prefix length (any IP inside the range but typically the subnet IP). Example for the range of IPs from `192.168.0.0` to `192.168.255.255`: `192.168.0.0/16`
A range of IPv6 addresses	IP with prefix length (any IP inside the range but typically the subnet IP). Example for the range of IPs from `fd01::1:0` to `fd01::1:ffff`: `fd01::1:0/112`
A single host	The host's FQDN. Example: `mycomputer@mydomain` (The FQDN must match the DNS information provided to the FlashArray.)
All hosts within a domain	Example: `*.mydomain`
A set of hosts within a domain	Example for all computers with names beginning with `pure0` in the domain `mydomain`: `pure0*.mydomain`

.

`--client-name` ***CLIENT_NAME***

When adding a snapshot rule, sets the customizable part of the snapshot name displayed to clients.

`--connect`

Create a policy with replica-link connected, for use with ActiveDR replication. Disconnected is the default.

`--context` ***REMOTE***

Used to specify the remote array on which a command is processed. ***REMOTE*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

If used with **purepolicy alert-watcher list**, **purepolicy alert-watcher rule list**, **purepolicy audit file list**, **purepolicy autodir list**, **purepolicy list**, **purepolicy management-access list**, **purepolicy nfs list**, **purepolicy nfs rule list**, **purepolicy password list**, **purepolicy smb list**, **purepolicy smb rule list**, **purepolicy quota list**, **purepolicy quota rule list**, **purepolicy snapshot list**, or **purepolicy snapshot rule list**, `--context` can specify multiple arrays and ***REMOTES*** is a comma-separated list of array names.

`--dictionary-check`

Verify if passwords exceeding 6 or more characters have been compromised. Skip verification by using the **purepolicy password disable** command.

`--dir` ***DIR***

The name of a managed directory. When adding or removing directories to or from a policy, a comma-separated list of names can be used. When adding SMB, NFS, or autodir policies to directories, specify one directory name for each command line.

`--disable-user-mapping`

Disables NFS user mapping to allow file services without using directory services. User mapping is enabled by default. Disabling user mapping for existing files or directories might cause accessibility issues.

`--disconnect`

Create a policy with replica-link not connected. This is the default.

`--ds-mapping` ***DS_MAPPING***

A comma-separated list of one or more mapping entry names, used with mapping to directory service roles. Used with `purepolicy management-access add` (or `remove`) to add (or remove) one or more access policies to the directory service role mapping.

`--enforced`

For use with directory quota rules. Enforces the directory quota rule limit. If omitted, the limit is unenforced and only the notification is sent when the limit is reached. When modifying an existing quota rule, the rule can be set to enforced or unenforced with the subcommands **enable** or **disable**, respectively.

`--every` ***EVERY***

When adding a snapshot rule, specifies the snapshot interval in the format `N`**[m|h|d|w]** for example **15m**, **1h**, **2d**, **3w**. The minimum value is five minutes (5m) and the maximum value is one year (1y). If **--at** is specified then **--every** must be a multiple of day (i.e., a 24-hour period).

`--export-name` ***EXPORT_NAME***

Sets the export/share name when adding an SMB or NFS policy to a target directory.

`--ignore-usage`

For use with directory quota rules. Adds the rule even if the quota is already exceeded.

`--keep-for` ***KEEP_FOR***

When adding a snapshot rule, specifies the period in which snapshots are retained before they are eradicated, specified in the format **`N[m|h|d|w|y]`** for example **`15m`**, **`1h`**, **`2d`**, **`3w`**, or **`4y`**. There is no maximum value, the minimum value is five minutes (**`5m`**).

`--mapping`

Display ActiveDR replica-link policy mapping.

`--max-login-attempts` ***ATTEMPT***

Locks the user account after a maximum number of unsuccessful login attempts. To disable this feature set the value to 0 or an empty string.

`--member`

Directories that are members of a policy.

`--min-characters-per-group` ***COUNT***

Sets the global minimum characters each character group is allowed to have for local account passwords. New passwords must meet the COUNT requirement for each of the character groups they are enforced to have. The default value is 1 character. Minimum characters per group count changes do not apply to existing passwords.

`--min-character-groups` ***COUNT***

Sets the global minimum character groups a user is allowed to have for local account passwords. New passwords must meet the COUNT requirement for each of the character groups they are enforced to have. The default value is 1 character. The character groups are lower case letters, uppercase [a-z], [A-Z], [0-9], other. If the number of required groups is set, it does not matter which of the character groups are used within the password as long as the amount of groups is equal or greater than then COUNT. Minimum characters group count changes do not apply to existing passwords.

`--min-password-age` ***DURATION***

Sets the minimum time duration before a password change can be performed. Specify the duration in the format [s|m|h|d|w], for example 15m, 1h, 2d, 3w.

`--min-password-length` ***LENGTH***

Sets the minimum password length for local accounts ranging from 1 to 100 characters. Minimum password length changes do not apply to existing passwords.

`--minimum-notification-severity`

For use with purepolicy alert-watcher. Enables you to set a threshold notification severity (i.e. critical, info, or warning) to affect which alerts result in email notifications to the specified array admins.

`--no-root-squash`

Performs no squashing of incoming UID or GID. Allows root users and groups to access the file system with root privilege. This option is mutually exclusive with the **`--root-squash`** and **`--all-squash`** options.

`--notifications` ***`NOTIFICATIONS`***

Notification recipients, for use with directory quota rules. A comma-separated list of notification recipients. Valid attributes are: **`user`**, **`group`**, or, for both: **`user,group`**. The actual recipient address is looked up for each instance of notifications. If omitted, no notifications are sent.

`--pending`

Includes destroyed objects pending eradication: policies, rules, or snapshots. If not specified, items that are pending eradication are not included.

`--pending-only`

Only includes destroyed objects pending eradication: policies, rules, or snapshots.

`--quota-limit` ***`QUOTA_LIMIT`***

For use with directory quota rules. Quota limit sizes are specified as an integer, followed by one of the suffix letters **`K`**, **`M`**, **`G`**, **`T`**, **`P`**, **`E`**, representing KiB, MiB, GiB, TiB, PiB, and EiB, respectively, where "Ki" denotes 2^10, "Mi" denotes 2^20, and so on.

`--realm` ***`REALM`***

With `purepolicy management-access create` and `purepolicy management-access setattr`, sets the scope of the access policy to the specified realm.

`--ro`

Allows read only access and disallows any request that changes the file system. The file access timestamp will still be updated even for read only access. This option is mutually exclusive with the ***`--rw`*** option. If omitted, both read and write requests are allowed.

`--role`

Specifies the role defined in an access policy. One of `admin`, `storage`, `support`, or `viewer` must be specified.

`--root-squash`

Prevents client users with root privilege from mapping their root privilege to a file system. This is the default. All users with UID 0 will have their UID mapped to anon. All users with GID 0 will have their GID mapped to anon. This option is mutually exclusive with the **`--no-root-squash`** and **`--all-squash`** options.

`--rule-name` ***`RULE`***

Comma-separated list of one or more rule names.

`--rw`

Allows both read and write requests, which is the default for NFS exports. This option is mutually exclusive with the ***`--ro`*** option.

`--security` ***`SECURITY`***

Comma-separated list of allowed NFS security protocols. Valid values include **`auth_sys`**, **`krb5`**, **`krb5i`**, **`krb5p`** for AUTH_SYS, Kerberos 5, Kerberos 5i, and Kerberos 5p, respectively. If not specified, the default is AUTH_SYS. Kerberos is supported for NFS version 4.1 only and requires User Mapping to be enabled. Any combinations of krb5, krb5i, krb5p and AUTH_SYS can be configured.

`--smb-encryption-required`

Enables SMB encryption and turns on data encryption for the export. Require the remote client to use SMB encryption. Clients that do not support encryption will be denied access. By default, when SMB encryption is enabled, only SMB 3.0 clients are allowed access.

If omitted, negotiation of encryption is enabled but data encryption is not turned on for this export.

`--suffix` ***`SUFFIX`***

Optionally, when adding a snapshot rule, specifies a suffix string for the directory snapshot. Suffix can only be specified for a rule that keeps only one snapshot (that is, with the same retention period as the snapshot interval). When omitted, Purity//FA creates a unique number for the directory snapshot.

`--targets` ***`LOG_TARGETS`***

The name of the audit log target.

`--user-mapping`

Use this option to enable or disable user-mapping for an existing NFS policy. User-mapping is enabled by default. Disabling user mapping for existing files or directories might cause accessibility issues.

`--username-check`

Verifies if passwords contain the username, if username is 4 or more characters. Skip the verification by using the **`purepolicy password disable`** command.

`--version` ***`VERSION`***

Comma-separated list of allowed NFS protocol versions for the NFS rule. Valid values include **`nfsv3`** and **`nfsv4`** for NFS version 3 and 4.1 respectively. If not specified, the default is nfsv3.

To upgrade or add NFS version 4.1 to existing NFS policies, use the **`purepolicy nfs setattr`** command to modify the attributes. Should you instead remove existing rules, client mounts may disconnect or not function correctly.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--page-with-token`

Used to paginate output with a continuation token (**`--token TOKEN`**). Results are printed in `stdout` and the token is printed in `stderr`.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Arguments

ALERT-POLICY

The name of the alert policy.

CODES

The alert code. (E.g. 40)

EMAIL-ADDRESS

The email address that will start or stop receiving alert messages.

NAME

The name of the policy when creating or deleting a policy, enabling or disabling. If used with listing, only policies that match the policy name will be displayed.

NEW-NAME

Name by which the policy is to be known after the command executes.

OLD-NAME

Current name of the existing policy to be renamed.

POLICY

The name of the policy when adding or removing a policy to source audited directories. When used with listing, only rules that are assigned to the specified one or more policies will be displayed.

SEVERITY

Utilized with the `--minimum-notification-severity` option, this argument represents a threshold value. If an alert is of a lower severity than the indicated value, an email about the alert will not be sent to the email addresses for which this rule is applied. Values include `critical`, `info`, and `warning`.

SOURCE

The name of the policy to be copied.

TARGETPOLICY

The name of the target policy to where the ***SOURCE*** policy will be copied to.

With **`purepolicy management-access copy`**, ***TARGETPOLICY*** cannot already exist.

Description

The **purepolicy** command manages policies used for creating exports (i.e., shares), directory quotas, scheduled snapshots, and management access policies. After creation, and when rules are added, policies can be used for creating exports, directory quotas, and scheduled snapshots, and for granting access permissions. File-related policies are also available through the **puredir** command. Management access policies are also attached to users or groups with the **pureadmin** and **pureds** commands.

The **purepolicy list** command lists policies of any type.

SMB and NFS Policies

To create an SMB or NFS policy, use the **purepolicy smb create** or **purepolicy nfs create** command, with a unique policy name. The file policy can be created inside a pod by using the fully qualified name of pod and policy, for example `pod01::policy01`.

The **--connect** and **--disconnect** options are used with ActiveDR to define replication mapping between arrays upon policy creation in the source pod of a file replica-link. By default, without specifying these options, the policy you create is disconnected. When an export policy is connected, deletion of the source policy will automatically delete the target policy, while deletion of the target policy is disallowed. When an export policy is disconnected, deletion of the source and target policy is allowed.

To view these policy mappings, use the **--mapping** option with the **list** subcommand.

Access-Based Enumeration (ABE) allows SMB exports to hide directories or files from clients that have less than generic read permissions. Note that this feature does not apply to users that belong to the Administrators and Backup Operators groups. The feature should not be regarded as a security measure, and for performance reasons, it should only be considered if it significantly benefits the user. ABE is disabled by default when an SMB policy is created. Use the **--access-based-enumeration** option to enable, as in the following example:

```
$ purepolicy smb enable --access-based-enumeration smb-simple
Name          Type  Enabled  Access Based Enumeration Enabled
smb-simple    smb   True     True
```

The feature can later be disabled or enabled with the **`disable`** and **`enable`** subcommands, respectively. Changes affect connected clients only after they refresh their view or after they reconnect.

SMB or NFS policies can be created in a pod that will be used for ActiveDR. You can copy policies from pod-to-pod by using the **`purepolicy smb copy`** or **`purepolicy nfs copy`** commands, respectively.

Define and add a rule for one or more SMB or NFS policies with the **`purepolicy smb rule add`** and **`purepolicy nfs rule add`** commands, respectively. All rules are automatically named, for example **`r_1`**, **`r_2`**, and so on. These rule names are globally unique.

To set or change the allowed versions of NFS, use the **`--version`** option. To set or change the allowed security features, use the **`--security`** option. These attributes are set for the NFS rules by using the **`purepolicy nfs rule add`** command. To change any of these attributes for an existing NFS policy, use the **`purepolicy nfs setattr`** command. This is preferable to replacing an existing rule since, if you remove a rule for an export that is currently in use, the client mounts might be disconnected.

NFS squashing is controlled with one of three options: **`--root-squash`** (the default), **`--no-root-squash`**, or **`--all-squash`**. These can be used in combination with **`--anonuid`** and **`--anongid`** to specify the resulting UID and GID, otherwise the default is 65534 (nfsnobody). Note that when NFS squashing is used in combination with user mapping, the **`--anongid`** option is ignored and the GID is provided by the directory services.

The NFS **`--rw`** option allows both read and write requests, which is the default. With the **`--ro`** option, the exports that use the policy provide read-only access and any request that changes the file system is denied. The file access timestamp is updated for read-only access as well as for read and write.

Each policy can be reused, creating exports for a number of managed directories. To create an export, add a policy to a managed directory by using the **`purepolicy smb add`** or **`purepolicy nfs add`** commands. The result is the same as creating an export using the **`puredir export create`** command.

To display a list of SMB or NFS policies, use the **`purepolicy smb list`** or **`purepolicy nfs list`** commands. Include the **`--member`** option for a list of directories that are members of each policy. Additionally, include the **`--pending`** option to display policies that are pending eradication. To display a list that only includes policies that are pending eradication include the –**`-pending-only`** option.

SMB and NFS policy names can be changed. To rename an existing SMB or NFS policy, specify the current name of the policy followed by a new name with the **`purepolicy smb rename`** or

purepolicy nfs rename command, respectively. For example, the following command renames the **smb-simple** SMB policy to **smb-simple-1**.

```
$ purepolicy smb rename smb-simple smb-simple1
Name          Type  Enabled  Access Based Enumeration Enabled
smb-simple1   smb   True     False
```

Policies can be temporarily disabled and re-enabled with the sub-commands **disable** and **enable**, respectively, followed by the name of one or more policies.

SMB or NFS policies can be removed from target managed directories with the **purepolicy smb remove** or **purepolicy nfs remove** commands. This is similar to deleting an export with the **puredir export delete** command. Note that the policies are not deleted, only the export. I.e., the policies are removed from the specified target managed directory. To delete one or more SMB or NFS policies, use the **purepolicy smb delete** or **purepolicy nfs delete** command.

To remove one or more rules from an SMB or NFS policy, use the **purepolicy smb rule remove** or **purepolicy nfs rule remove** command.

NFS User Mapping

By disabling user mapping on NFS, the exports can be accessed without directory services and the array relies instead upon AUTH_SYS (previously known as AUTH_UNIX) RPC authentication. The array will trust the UID and GIDs from the host.

The recommendation is to not disable user mapping for exports when user mapping has been in use, since existing files and directories could become inaccessible. Do not remove, delete, or unjoin existing directory services; the existing files and directories have ACLs based on Active Directory or LDAP user IDs and memberships, and without these services, the files will become inaccessible. Similar accessibility issues might occur with SMB exports if applied on the same directory as NFS exports with user mapping disabled.

Disabling user mapping can be done when creating an NFS policy, by adding the **--disable-user-mapping** option. After creation, user mapping can be enabled or disabled with the **enable** and **disable** subcommands, respectively. When user mapping is disabled, the policy will display, when listed, User Mapping Enabled: False.

Auto Managed Policies

Given the correct permissions, connected clients create subdirectories on their mounted share through the SMB or NFS protocols. These non-managed directories cannot granularly be managed within Purity//FA. With the use of auto managed directory policies (autodir policies), subdirectories automatically become managed directories, one level below the first managed directory. Any subdirectories nested below this first level of managed subdirectories will become non-managed directories.

Auto managed directory policies are only allowed on managed directories that allow (that is, has room for) at least one more level of nested managed directories below itself.

The **`autodir`** subcommand creates and manages the auto managed directory policy type.

Use the **`purepolicy autodir`** command to manage the creation, deletion, renaming, and listing of auto managed directory policies. Use the **`disable`** or **`enable`** subcommands to disable or enable the policy.

To add or remove your managed directory to the autodir policy, use the **`add`** or **`remove`** subcommands. For example, the following command adds the directory `VMware-FS01:VMwareNFS-01` to the `NFS01` autodir policy:

```
purepolicy autodir add --dir VMware-FS01:VMwareNFS-01 NFS01
```

NFS Datastore

Managed directories are designed to give VMware administrators the ability to use FlashArray as an NFS datastore. Using vSphere 7.0 or later, NFS datastores can be created using NFS exports on FlashArray through the NFS protocol version 3 or 4.1.

Subdirectories are created automatically for each VM. The NFS datastore requires the no-root-squash option to be set. Add a rule to the NFS policy with **`--no-root-squash`** enabled. For granular control, add an auto managed directory policy to make subdirectories automatically created as managed directories. For fast copying of files, for example in relation to restoring VMs, refer to the **`purefile copy`** command.

The following steps summarize the process of creating the datastore:

1. Create a managed directory for the datastore.
2. Create the NFS policy for the export.
3. Add a rule to the NFS policy, select NFS version 3 or 4.1.

4 Optional: Create an autodir policy and add to it the managed directory from step 1.
5 Create an export by adding the managed directory to the NFS policy.
6 Finally, mount the NFS datastore through the vSphere client.

For getting started with NFS datastores on the FlashArray, refer to the VMware NFS Datastores on FlashArray Quick Start Guide on the Knowledge site at https://support.purestorage.com.

Directory Quota Policies

Directory quotas allow restriction of storage space, and notification, by using enforced and unenforced quota limits. Quotas are managed with directory quota policies. To create a quota policy, use the `purepolicy quota create` command, followed by one or more quota policy names to be created. Quota policies can be created in a pod that will be used for ActiveDR. You can copy quota policies from pod-to-pod by using the `purepolicy quota copy` command.

Define and add rules to one or more quota policies by using the `purepolicy quota rule add` command. Use the `--quota-limit` option to specify the size of the quota, `--enforced` for the one enforced quota per policy. Valid options include `--ignore-usage` to override directory usage scanning, and `--notifications` to specify notification recipients. Specify one or more quota policy names that will have the rule added.

Quota policies can be reused, creating quota limits for a number of managed directories, with the `purepolicy quota add` or the `puredir quota add` commands.

Quota limits are measured against logical file sizes, which include the actual data stored in the file as well as any empty space, such as with sparse files. Therefore, even if a large and mostly empty file occupies minimal physical space on the array, it still counts towards the set quota limit as if it was a fully populated file of the same size.

Directory quota is also unaware of data reduction and deduplication. Space used for snapshots or metadata (that is, data about files and directories, like ownership, permissions, timestamps, and more), is not counted towards the quota limit.

In the following example, a quota policy is created, named **quota_policy_main**:

```
purepolicy quota create quota_policy_main
```

Next, add a rule:

```
purepolicy quota rule add --quota-limit 1G --notifications user,group --enforced --
ignore-usage quota_policy_main
```

In this example, the limit is set to 1 GB, notifications enabled for both user and group (actual email addresses are looked up upon notification). The mode is enforced, meaning notifications are generated at 80% of the limit (informational), 90% (warning), and 100% (critical). At 100%, all future space increasing operations are being prevented with ENOSPC errors. The rule is added to the quota policy. “Ignore-usage” ensures that the rule can be added even if the quota limit is already exceeded.

Finally, add the quota policy to the managed directory **`FS1:Data`**:

```
purepolicy quota add --dir FS1:Data quota_policy_main
```

Once quota rules are defined and added to quota policies, they can be modified by using the `purepolicy quota rule setattr` command. Valid options include **`--quota-limit`**, **`--notifications`**, and **`--ignore-usage`**. Specify **`--rule-name`** followed by one or more rules to modify, separated by commas, and the quota policies where the rules are applied. To unenforce an existing quota limit, use **`purepolicy quota rule disable --enforced`**, followed by the rule names and quota policy names. Similarly, use `enable`, to make the quota limit enforced. When modifying or enforcing an existing quota limit, the **`--ignore-usage`** option can be used for overriding directory usage scanning, and to allow the changes.

To display a list of quota policies, use the `purepolicy quota list` command. Include the **`--pending`** option to display destroyed quota policies that are pending eradication. To display a list that only includes destroyed quota policies that are pending eradication include the **`--pending-only`** option. Similarly, for a list of rules, use the `purepolicy quota rule list` command.

To rename an existing quota policy, specify the current name of the policy followed by a new name with the `purepolicy quota rename` command.

Quota policies can be temporarily disabled and re-enabled with the sub-commands **`disable`** and **`enable`**, respectively, followed by the name of one or more policies.

Quota policies can be removed from target managed directories with the `purepolicy quota remove` command. Note that the policies are not deleted, only removed from the specified target managed directory. To delete one or more quota policies, use the `purepolicy quota delete` command.

To remove one or more rules from a quota policy, use the `purepolicy quota rule remove` command.

Snapshot Policies

The creation of a snapshot policy starts with the `purepolicy snapshot create` command, followed by one or more snapshot policy names. Snapshot policies can be created in a pod that will be used for ActiveDR. You can copy snapshot policies from pod-to-pod by using the `purepolicy snapshot copy` command.

Define and add rules for one or more snapshot policies by using the `purepolicy snapshot rule add` command.

Each snapshot policy can be reused, creating protection plans for a number of managed directories. To add a snapshot policy to one or more target directories, use the `purepolicy snapshot add` command.

To display a list of snapshot policies, use the `purepolicy snapshot list` command. Include the **`--pending`** option to display destroyed snapshot policies that are pending eradication. To display a list that only includes destroyed snapshot policies that are pending eradication include the **`--pending-only`** option. Similarly, for a list of rules, use the `purepolicy snapshot rule list` command.

To rename an existing snapshot policy, specify the current name of the policy followed by a new name with the `purepolicy snapshot rename` command. For example, the following command renames the **`snap-daily`** snapshot policy to **`snap-daily-1`**.

```
$ purepolicy snapshot rename snap-daily snap-daily-1
Name          Type      Enabled
snap-daily-1  snapshot  True
```

Policies can be temporarily disabled and re-enabled with the sub-commands **`disable`** and **`enable`**, respectively, followed by the name of one or more policies.

Snapshot policies can be removed from target directories with the `purepolicy snapshot remove` command. This is similar to removing the snapshot using the `puredir snapshot remove` command. The policies are not deleted, only removed from the specified target directory. To delete one or more snapshot policies, use the `purepolicy snapshot delete` command.

To remove one or more rules from a snapshot policy, use the `purepolicy snapshot rule remove` command.

Alert Policies

Alert policies determine which email address receive notifications about specific alerts coming from your FlashArray system. For example, an alert policy might contain a rule that prevents emails from Alert 40 to be sent from the array to the email address dcushing@purestorage.com. Or, an alert policy might specify which email addresses should receive emails about alerts on the array.

You can list all existing alert policies with `purepolicy alert-watcher list`. To create an alert policy, run `purepolicy alert-watcher create`. To delete an alert policy, run `purepolicy alert-watcher delete`. To disable an alert policy, run `purepolicy alert-watcher disable`. To re-enable a disabled policy, run `purepolicy alert-watcher enable`. To rename an alert policy, run `purepolicy alert-watcher rename`.

Within each alert policy, you can specify rules. Each rule in the alert policy has these properties: Email addresses for which the rule applies (specified with the `--address` option), minimum notification severity for the policy (specified with the `--minimum-notification-severity` option), a list of included alert codes (specified with the `--include-codes` option), and a list of excluded alert codes (specified with the `--exclude-codes` option).

You can list all existing rules within an alert policy with `purepolicy alert-watcher rule list`. To add a rule to an existing alert policy, run `purepolicy alert-watcher rule add`. To remove a rule from an existing alert policy, run `purepolicy alert-watcher rule remove`. To alter an existing alert policy rule, run `purepolicy alert-watcher rule setattr`.

Note: If one email address is involved in multiple alert policies with rules that conflict as to whether that email address should receive an alert (i.e. one rule says dcushing@purestorage.com should receive notifications about alert 40, but another rule under a different policy on the same array says that email address should not receive notifications about alert 40), then the email address will receive the alert. Having multiple rules that say an email address should receive an alert will not result in multiple notifications being sent to that email address about the same alert.

File Auditing

The File Auditing feature is designed to enhance security and compliance by providing detailed auditing capabilities for file access and operations. The feature includes support for SMB and

NFS protocols and allows for configuration of audit policies, log targets, and System Access Control Lists (SACLs) to specify the activities that are logged.

The audit policy, when linked to source managed directories, holds the relationship between those directories and audit log targets, ensuring that file operations within each directory are audited in line with the policy configuration and the target which are the destinations where audit logs are sent. The log target can be a local managed directory or a syslog server. When a local managed directory is used as a target, the directory must be empty at start and it is advisable to not place the target below the directory being audited. It is important to note that auditing is not guaranteed, which implies that under certain conditions or configurations, auditing may not capture all events or may not be supported.

Each audit policy can link to one or many source managed directories, and the same audit policy can also link to one or many log targets. To define the operations that get audited, System Access Control Lists (SACLs) are required. SACLs are configured over SMB or NFS 4.1 using third party tools external to the FlashArray.

File audit log targets are managed using the **`purelog file`** and **`purelog syslog`** commands.

After one or more log targets are created, file audit policies are managed using the **`purepolicy audit file`** command. To create one or more audit policies, use the **`create`** subcommand and specify one or more comma-separated log targets and one or more comma-separated policy names. The following example creates one policy named `audit_1` specifying one target named `target_1`:

```
purepolicy audit file create --targets target_1 audit_1
```

To attach an audit policy to one or more managed directories, use the **`add`** subcommand and specify one or more directories, followed by the name of the policy, for example:

```
purepolicy audit file add --dir FS1:Data1,FS1:Data2 audit01
```

To enable or disable a policy, use the **`enable`** or **`disable`** subcommands, for example:

```
purepolicy audit file enable audit01
```

Member directories can be removed from an audit policy by using the **`remove`** subcommand. Specify one or more directories to be removed, followed by the name of the policy.

An audit policy can be copied with the **`copy`** or renamed with the **`rename`** subcommands. Specify the existing and a new policy name. To delete one or more audit policies, use the **`delete`** subcommand. To set new targets for an audit policy, which replaces all existing targets, use the **`setattr`** subcommand.

To display a list of audit policies, use the `list` subcommand. Include the `--pending` option to display deleted audit policies that are pending eradication. To display a list that only includes deleted audit policies that are pending eradication, include the `--pending-only` option. Use the `--member` option to list audit policies attached to specific member directories.

Password Policies

Password policies can be optimized and configured to serve user needs within an organization. Use password policies to set attributes that are part of an array's global password policy called `management` and applies to all local users.

To enable the `management` policy use the `purepolicy password enable` command. Alternatively, use the `purepolicy password disable` command to disable the `management` policy.

When disabling a password use the `purepolicy password disable --username-check` command to skip verifying whether a password contains a username, if the username is 4 characters or more. Use the `--dictionary-check` option to skip the check for passwords with 6 or more characters against a list of comprised passwords.

Password policies, policy name, enabled or disabled status, and policy attributes can be listed using the `purepolicy password list` command.

Modify Password Policies

Password policies can be easily modified using the `purepolicy password setattr` command.

Use the `--min-password-length` option to adjust the minimum password length. The maximum login attempts can be set by using the `--max-login-attempts` option. Similarly, use the `--lockout-duration` option to set the lockout duration after a user has reached the maximum login attempts. The minimum number of characters per character group can be set using the `--min-characters-per-group`. There is a minimum number of required character groups to be present in a policy password. To adjust the number use the `--min-characters-groups` option. Track the number of past used passwords to prevent reuse by using the `--password-history` option. To prevent users constantly rotating passwords use the `purepolicy password setattr --min-password-age` command.

Management Access Policies

Management access policies replace the concept of roles, which in previous releases defined user permissions. In addition to roles, access policies also define both the scope of the permissions, meaning to which realm or array the permissions apply, and an aggregation strategy, which determines the effect of multiple access policies being attached to a user or group of users. The aggregation strategy can be set either to apply all permissions of the multiple access policies or to apply a permission only if it is granted by *every* access policy that is assigned to the user or group.

To use management access policies, first you define the access policies you require, then you attach those policies to users or groups as appropriate. An access policy can be attached to multiple users or groups as needed and detached as needed. The `pureadmin` and `pureds` CLI commands also attach a management access policy to a new user account and to a directory service group to role mapping.

Four access policies corresponding to the previous roles are built into Purity//FA, as shown in the following table. These policies are scoped to the local array.

Table 6. Built-in Management Access Policies and Their Roles

Built-in Management Access Policy	Current Role	Equivalent Role in Previous Releases
array_admin	admin	array_admin
storage_admin	storage	storage_admin
ops_admin	support	ops_admin
readonly	viewer	readonly

The roles used in previous releases are still available for backwards compatibility.

A management access policy defines the following three areas:

- **Role**. Identifies the set of permissions that the user is entitled to perform in specified scope. A capability is analogous to a role.
    - **Read**. Able to read all configuration but cannot create, modify, or delete any object. Specified by the `--role viewer` option.
    - **Support**. Able to open RemoteAssist sessions, plus all of read permissions. Specified by the `--role support` option.
    - **Storage**. Able to create, modify, or delete storage objects, like volumes or snapshots, plus all of read permissions. Specified by the `--role storage` option.

    - **All**. Able to read, create, modify, or delete all customer-facing objects. Specified by the `--role admin` option.
- **Scope**. The scope in which the permissions are granted. Resource specifies one of the following objects:
    - An array.
    - A realm.
- **Aggregation strategy**. The rule that merges this policy together with other policies, if they conflict:
    - **all-permissions**. Users are granted the union of the permissions of all applicable policies. This setting provides the most generous set of permissions. Default.
    - **least-common-permissions**. Users are granted the intersection of the permissions of all applicable policies. This setting is analogous to a least common denominator. A permission is granted only if it appears in *all* applicable policies.

Use the **`purepolicy management-access create`** command to create an access policy of the desired permissions and scope. The permission level is required and is set with the `--role` option to one of `viewer`, `support`, `storage`, or `admin`. The required scope is set with either `--array` or `--realm` and an array or realm name. Aggregation strategy is optional and defaults to `all-permissions`.

The following example creates an access policy for a realm administrator, with `admin` permissions, in the realm `realm1`.

```
$ purepolicy management-access create --realm realm1 --role admin r1_admin_policy
Name            Type              Enabled Role   Aggregation Strategy Resource Name

r1_admin_policy management-access True    admin  all-permissions      realm1
```

Use the **`purepolicy management-access add`** command to attach an access policy to existing users or groups. The following example attaches the `r1_admin_policy` access policy to an existing user named `admin1`:

```
$ purepolicy management-access add --admin admin1 r1_admin_policy
Name             Type                Member   Member Type

r1_admin_policy  management-access   admin1   admin
```

User `admin1` now has `admin` permissions for `realm1` in addition to any permissions `admin1` had previously.

When an access policy is created with the `least-common-permissions` aggregation strategy, the policy restricts the specified users or groups to this access. In this example, `admin2` has only `support` permission in `realm2`, and no array-wide permissions, regardless of the permissions `admin2` had before.

```
$ purepolicy management-access create --realm realm2 --role support --aggregation-strat
egy least-common-permissions r2_support_pol
Name            Type              Enabled Role     Aggregation Strategy       Resource
r2_support_pol  management-access True    support  least-common-permissions   realm2


$ purepolicy management-access add --admin admin2 r2_support_pol
Name             Type               Member   Member Type
r2_support_pol   management-access  admin2   admin
```

The **`purepolicy management-access list`** command lists the current access policies, including the built-in policies:

```
$ purepolicy management-access list
Name            Type              Enabled  Role     Aggregation Strategy       Resourc
array_admin     management-access True     admin    least-common-permissions   pure01
storage_admin   management-access True     storage  least-common-permissions   pure01
ops_admin       management-access True     support  least-common-permissions   pure01
readonly        management-access True     viewer   least-common-permissions   pure01
r2_support_pol  management-access True     viewer   all-permissions            realm2
r1_admin_pol    management-access True     admin    all-permissions            realm1
```

The **`pureadmin list`** command lists local users and the access policies attached to those users.

```
$ pureadmin list
Name       Type   Access Policy
pureuser   local  array_admin
admin1     local  r1_admin_pol
                  r2_support_pol
admin2     local  r1_admin_pol
```

Use the **`purepolicy management-access copy`** command to copy an existing access policy to one or more new policies. The new policy or policies must not already exist. Each new policy is created with the same permissions, scope, and aggregation strategy as the source policy. **purepolicy management-access list** shows the new policies (`newpolicy` and `newpolicy2`) with the same settings as the source policy, `r1_admin_pol`.

```
$ purepolicy management-access copy r1_admin_pol  newpolicy,newpolicy2
Name       Type                Enabled Role   Aggregation Strategy   Resource Name
newpolicy  management-access   True    admin  all-permissions        realm-1
newpolicy2 management-access   True    admin  all-permissions        realm-1
```

```
$ purepolicy management-access list
Name            Type               Enabled  Role     Aggregation Strategy        Resourc
array_admin     management-access  True     admin    least-common-permissions    pure01
storage_admin   management-access  True     storage  least-common-permissions    pure01
ops_admin       management-access  True     support  least-common-permissions    pure01
readonly        management-access  True     viewer   least-common-permissions    pure01
r2_support_pol  management-access  True     viewer   all-permissions             realm2
r1_admin_pol    management-access  True     admin    all-permissions             realm1
newpolicy       management-access  True     admin    all-permissions             realm1
newpolicy2      management-access  True     admin    all-permissions             realm1
```

The **purepolicy management-access delete** command to delete (completely erase) one or more access policies.

```
$ purepolicy management-access delete newpolicy
Name
newpolicy
```

```
$ purepolicy management-access list newpolicy
Error on newpolicy: Policy does not exist.
```

An access policy cannot be deleted if the policy is attached to any user or group, as shown in this example. First, use the **purepolicy management-access remove** command to detach the users and groups, or consider using `purepolicy management-access disable` instead.

```
$ purepolicy management-access delete testpolicy
```

```
Error on testpolicy: Policy is attached. Detach the policy from all members before
deletion.
```

Use the **purepolicy management-access disable** command to remove one or more access policies from use. The access policy or policies still exist but have no effect.

```
$ purepolicy management-access disable r2_support_pol
Name            Type               Enabled  Role     Aggregation Strategy   Resource
Name
r2_support_pol  management-access  False    viewer   all-permissions        realm2
```

Use the **purepolicy management-access enable** command to re-enable one or more disabled policies. Enabling an access policy is only necessary if the policy had been disabled.

```
$ purepolicy management-access enable r2_support_pol
Name            Type               Enabled  Role     Aggregation Strategy   Resource
Name
r2_support_pol  management-access  True     viewer   all-permissions        realm2
```

The **purepolicy management-access rename** command changes the name of an access policy. Other policy settings are not affected.

```
$ purepolicy management-access rename realm-2_admin_policy r2_admin_pol
Name            Type               Enabled  Role     Aggregation Strategy    Resource
Name
r2_admin_pol    management-access  True     admin    all-permissions         realm2
```

The **purepolicy management-access remove** command removes (detaches) one or more access policies from a local administrator or directory service role mapping. This example removes the access policy `r1_admin_policy` (and its permissions) from user `admin1`:

```
$ purepolicy management-access remove --admin admin1 r1_admin_policy
Name             Type               Member  Member Type
r1_admin_policy  management-access  admin1  admin
```

Use the **purepolicy management-access setattr** command to change the scope (array or realm), permissions, or aggregation policy of one or more access policies. This example changes the role to `storage` and the scope to `realm2`. The `purepolicy management-`

`access list` command shows the original policy settings. Note that changing a policy immediately affects users who have that policy attached.

```
$ purepolicy management-access list testpolicy
Name          Type                Enabled  Role     Aggregation Strategy   Resource Name
testpolicy    management-access    True     viewer   all-permissions        realm-1
```

```
$ purepolicy management-access setattr --realm realm-2 --role storage testpolicy
Name          Type                Enabled  Role     Aggregation Strategy   Resource Name
testpolicy    management-access   True     storage  all-permissions        realm-2
```

Examples

Example 1

Create a snapshot protection plan and attach it to a directory:

```
purepolicy snapshot create snap_policy_main
purepolicy snapshot rule add --every 1d --at 9pm --keep-for 3w
                             --client-name daily snap_policy_main
purepolicy snapshot add --dir FS1:Data snap_policy_main
```

1 Creates a snapshot policy named **`snap_policy_main`**.

2 Adds a rule to the snapshot policy notifying the scheduler to make a snapshot every one day, at 9 pm, and keep for three weeks. The customizable name visible to clients is **`daily`**.

3 Adds the snapshot policy to the managed directory **`FS1:Data`**.

Example 2

```
purepolicy snapshot disable snap_policy_main
purepolicy snapshot enable snap_policy_main
```

Disables and enables the snapshot policy. Enabling a snapshot policy is only necessary if first disabled.

Example 3

```
purepolicy snapshot list
```

Lists all snapshot policies.

Example 4

```
purepolicy snapshot list --member snap_policy_main
```

Lists all directories that are members of the snapshot policy named **`snap_policy_main`**.

Example 5

```
purepolicy list
```

List all policies of any type.

Example 6

```
purepolicy nfs create --disable-user-mapping sq
purepolicy nfs rule add --all-squash --anonuid=2708 --anongid=100 --version nfsv4 sq
```

Creates an NFS export policy named **`sq`**, disabling user mapping. Adds a rule to squash all users to uid 2708 and gid 100. NFS version number is set to **`nfsv4`** (NFS version 4.1).

Example 7

```
purepolicy smb create SMBanon
purepolicy smb rule add SMBanon
```

Creates an SMB export policy named **`SMBanon`**. Then add a rule with default settings: no clients are restricted, authentication is required, and with optional SMB encryption.

Example 8

```
purepolicy smb add ..dir FS1:Data ..export-name Share SMBanon
```

Creates an export named **`Share`**, by using the policy named **`SMBanon`** from the example above, adding it to the managed directory named **`Data`** on file system **`FS1`**.

Example 9

```
purepolicy smb create pod1::policy1

purepolicy smb rule add pod1::policy1
```

Creates an SMB export policy named **`policy1`** in **`pod1`**. Then add a rule with default settings: no clients are restricted, authentication is required, and with optional SMB encryption.

Example 10

```
purepolicy snapshot create pod1::policy1
purepolicy snapshot rule add pod1::policy1 --every 1d --at 9pm --keep-for 3w --client-
name daily
```

Creates a snapshot policy named **`policy1`**.

Adds a rule to the snapshot policy notifying the scheduler to make a snapshot every one day at 9pm, and keep for three weeks. The customizable name visible to clients is **`daily`**.

Example 11

```
purepolicy password setattr management --min-chracter-groups 3
```

Sets the global minimum character groups to a value of **`3`**.

Example 12

```
purepolicy alert-watcher create array-admin
```

Creates an alert policy named **`array-admin`**.

Example 13

```
purepolicy alert rule add array-admin --address admin-group@example.com --include-codes
25,60
```

Adds a rule to the alert policy **`array-admin`** which specifies that email address **`admin-group@example.com`** will receive notifications about alerts 25 or 60.

Example 14

```
purepolicy alert-watcher rule setattr array-admin --address admin-group@example.com --
minimum-notification-severity warning
```

Alters a rule in the alert policy **`array-admin`** for email address **`admin-group@example.com`** by making it so notifications will only be sent for alerts with the notification severity of **`warning`** or above.

Example 15

```
purepolicy management-access create --array pure01 --role support support_policy
```

Creates an access policy that grants `support` role permissions on the array `pure01`. `support` role permissions allow reading configuration and opening RemoteAssist sessions.

Example 16

```
purepolicy management-access create --array pure01 --role admin admin_policy
```

Creates an access policy that grants `admin` role permissions for the array `pure01`.

Example 17

```
purepolicy management-access create --realm realm1 --role admin r1_admin
```

Creates an access policy that grants `admin` role permissions for the `realm1` realm.

Example 18

```
purepolicy management-access list
```

Lists the current access policies, including both custom and built-in policies.

Example 19

```
purepolicy management-access add --admin admin1 r1_admin
```

Attach the access policy `r1_admin` to the local user `admin1`.

Example 20

```
purepolicy management-access add --ds-mapping dsmap1 r1_admin
```

Attach the access policy `r1_admin` to the directory service role to group mapping named `dsmap1`.

Example 21

```
purepolicy management-access remove --ds-mapping dsmap1 r1_admin
```

Removes (detaches) the access policy `r1_admin` from the directory service role to group mapping named `dsmap1`.

Example 22

```
purepolicy management-access copy admin_policy admin_policy_2
```

Copies an access policy.

Example 23

```
purepolicy management-access setattr --realm realm-1 admin_policy_2
```

Changes an attribute of the access policy `admin_policy_2`. Changes the scope of `admin_policy_2` to `realm-1`.

Example 24

```
purepolicy management-access disable admin_policy_2
```

Disables the access policy named `admin_policy_2`.

See Also

pureadmin, pureapiclient, puredir, pureds, purefile, purefs, purelog

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

pureport

pureport, pureport-list – manages an array's host connection ports

Synopsis

```
pureport list [--initiator] [ --csv | --nvp ] [--filter FILTER]
[--limit LIMIT] [--notitle] [--page] [--page-with-token] [--token
TOKEN] [--raw] [--sort SORT] [--context REMOTES]
```

Options

`-h | --help`
: Can be used with any command or subcommand to display a brief syntax description.

`--context REMOTES`
: Used to specify the remote array or arrays on which a command is processed. ***REMOTES*** is a comma-separated list of names of arrays within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--initiator`
: Displays host iSCSI Qualified Names (IQNs), NVMe Qualified Names (NQNs), and Fibre Channel World Wide Names (WWNs) - both those discovered by Purity//FA and those manually assigned by system administrators - along with the array ports (targets) on which they are eligible to communicate.

Options that control display format:

`--csv`
: Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--page-with-token`

Used to paginate output with a continuation token (**`--token TOKEN`**). Results are printed in `stdout` and the token is printed in `stderr`.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Description

The **`pureport list`** command displays the iSCSI Qualified Names (IQNs), NVMe Qualified Names (NQNs), and Fibre Channel World Wide Names (WWNs) assigned to an array's target ports. If the array port has failed over, the Failover column displays the name of the port to which this port has failed over. Include the **`--initiator`** option to display all host IQNs, NQNs, and WWNs known to the array, either through discovery or manually assigned, and the array ports on which they are eligible to communicate. IQNs, NQNs, and WWNs are manually assigned through the `purehost` command.

The **pureport list** output does not include information about network interfaces. For information about network interfaces, refer to purenetwork.

Examples

Example 1

```
pureport list --initiator --notitle
```

Displays initiator IQNs, NQNs, and WWNs known to the array, both discovered and manually assigned by system administrators. If an array port has failed over, also displays the name of the port to which the port has failed over. The output excludes column titles.

See Also

purearray, purehost, purenetwork

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purepreset

purepreset workload - creates and manages workload presets

Synopsis

purepreset `workload create –payload` ***PAYLOAD*** `[--context` ***FLEET***`]` ***NAME***

purepreset `workload upload [--context FLEET]` ***NAME***

```
purepreset workload download [--context FLEET] NAME
purepreset workload list [--context FLEET] [--cli | --csv | --nvp] [--
notitle] [--raw] [NAME ...]
purepreset workload update --payload PAYLOAD [--context FLEET] NAME
purepreset workload delete [--context FLEET] NAME
purepreset workload rename [--context FLEET] OLD-NAME NEW-NAME
```

Arguments

FLEET

The name of the fleet to create a fleet-scoped preset.

NAME

The preset name.

A preset name is 1 to 63 characters in length. Valid characters are letters (A-Z and a-z), digits (0-9), and the hyphen (-) character. The first and last characters of the name must be alphanumeric, and the name must contain at least one letter or '-'. Names are case insensitive with the exception of **`purepreset workload create`**.

NEW-NAME

The name of the new fleet.

OLD-NAME

The name of the old fleet.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--payload` ***PAYLOAD***

The JSON format that contains the preset information.

`--context` ***FLEET***

Used to specify the preset location of remote arrays. The **`FLEET`** name must be provided in the context to create a fleet-scoped preset. Depending on which **`purepreset workload`** command is used, `--context` will apply differently. In **`purepreset workload create`**, `--context` is used to create a preset in a fleet context.

Options that control display format:

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

Description

The **`purepreset workload`** command manages the creation, naming, deletion, and management of workload presets. A workload preset is a JSON document or template that contains all the necessary configurations for provisioning a workload, including volume and policy configurations. Fleet-scoped storage admins can standardize the way a fleet works by using a workload preset. This ensures that all configurations are consistently applied across different workloads. Create a preset by using the **`purepreset workload create`** command.

Fleet-scoped Objects

A fleet-scoped object is an object that can be created in a fleet and visible across all arrays in that fleet, including arrays that are added later to the fleet. Presets are a type of object that can be created in the fleet context. If any array leaves a fleet, it will then loose all access to fleet scoped objects.

Managing Workload Presets

Use the **`purepreset workload upload`** command to create or update a workload preset. Provide the current ***`FLEET`*** name to update an existing preset. Update the entire workload preset with the **`purepreset workload update`** command. The preset JSON document must be provided via the **`--payload`** option. View details on one or more presets with the **`purepreset workload list`** command. Similarly, use the **`purepreset workload download`** command to display the entire preset JSON of one preset.

Change the name of the workload preset with the **`purepreset workload rename`** command. Delete one or more workload presets on an array by using the **`purepreset workload delete`** command.

It is advisable to use the GUI for managing workload presets. Refer to the Presets section for more information.

Examples

Example 1

```
purepreset workload create --payload <PAYLOAD> --context myfleet
```

Creates a preset in the specified fleet context. **`PAYLOAD`** is the JSON schema of preset in string format.

Example 2

```
purepreset workload upload --context myfleet mypreset
```

Updates an existing preset **`mypreset`** with new configurations.

Example 3

```
purepreset workload download mypreset --context myfleet
```

Downloads the full preset in JSON format.

purerealm

purerealm, purerealm-create, purerealm-destroy, purerealm-eradicate, purerealm-eradication-config, purerealm-list, purerealm-monitor, purerealm-recover, purerealm-rename, purerealm-setattr – creates and manages realms

Synopsis

purerealm create [--bw-limit BANDWIDTH_LIMIT] [--iops-limit IOPS_LIMIT] [--quota-limit QUOTA_LIMIT] ***REALM...***

purerealm destroy [--destroy-contents] ***REALM...***

purerealm eradicate [--eradicate-contents] ***REALM...***

purerealm eradication-config {list} [--csv | --nvp] [--notitle] [--page] [--raw] [--pending | --pending-only] [***REALM...***]

purerealm list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--filter ***FILTER***] [--historical {1h,1y,24h,30d,3h,7d,90d}] [--limit ***LIMIT***] [--pending | --pending-only] [--qos] [--sort ***SORT***] [--space | --footprint] ***REALM...***

purerealm monitor [--csv] [--filter FILTER] [--historical {1h,1y,24h,30d,3h,7d,90d}] [--interval INTERVAL] [--latency] [--limit LIMIT] [--notitle] [--page] [--raw] [--repeat REPEAT] [--size] [--sort SORT] [--total] [--total-only] [***REALM...***]

purerealm recover ***REALM...***

purerealm rename ***OLD-NAME NEW-NAME***

purerealm setattr [--bw-limit BANDWIDTH_LIMIT] [--ignore-usage] [--iops-limit IOPS_LIMIT] [--quota-limit QUOTA_LIMIT] ***REALM...***

Arguments

NEW-NAME

Name by which the realm is to be known after the command executes.

OLD-NAME

Current name of the realm to be renamed.

REALM

The realm to be created, listed, etc.

A realm name is 1 to 63 characters in length. Valid characters are letters (A-Z and a-z), digits (0-9), and the hyphen (-) character. The first and last characters of the name must be alphanumeric, and the name must contain at least one letter or '-'. Names are case-insensitive on input. For example, `realm1`, `Realm1`, and `REALM1` all represent the same realm. Purity//FA displays names in the case in which they were specified when created or renamed.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--bw-limit` ***BANDWIDTH-LIMIT***

Sets the maximum QoS bandwidth limit for the realm. If set, the bandwidth limit must be between `1` MB/s and `512` GB/s. The bandwidth limit is specified as an integer, optionally followed by the suffix letter `M` (for megabytes) or `G` (gigabytes).
Use an empty string "" for unlimited bandwidth.
Whenever throughput exceeds the bandwidth limit, throttling occurs.

`--destroy-contents`

Allows the **`purerealm destroy`** command to destroy realm contents.
Destroys both the realm and its contents, including all pods, volumes, and host groups inside the realm.

`--eradicate-contents`

Allows the **`purerealm eradicate`** command to eradicate realm contents when the realm is not empty.

`--footprint`

Displays the maximum amount of physical space or effective used capacity the realm or realms would take up on any array, measured in bytes. Footprint represents the amount of physical space or effective used capacity the realm or realms would require if moved or copied to a different array. Footprint is an estimate, as differences in deduplication rates and in average compression rates could affect the actual physical space or effective used capacity requirement.

The `--footprint` option cannot be used in combination with the `--space` option.

`--ignore-usage`

Allows setting a bandwidth limit that is lower than the existing usage.

`--iops-limit` ***`IOPS-LIMIT`***

Sets the maximum IOPS limit for the realm. If set, the IOPS limit must be between 100 and 100`M`. The IOPS limit is specified as an integer, optionally followed by the suffix `K (10^3)` or `M(10^6).`
Use an empty string "" for unlimited IOPS.
Whenever the number of I/O operations per second exceeds the IOPS limit, throttling occurs.

`--latency`

Display latency statistics.

`--pending`

Includes destroyed realms that are in the eradication pending state. If not specified, realms that are pending eradication are not shown.

`--pending-only`

Only displays realms that are destroyed and in the eradication pending state.

`--quota-limit`

Sets an upper limit on the amount of data a realm can contain. When this option is set, new objects cannot be added if the realm would then take more space than the quota limit. It is by default not allowed to set a quota limit lower than the existing usage. To set a quota limit lower than the existing usage, use the **`--ignore-usage`** option. Destroyed objects are not charged against the quota limit. The quota limit can be removed by setting the value to empty quotes "".

`--space`

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--page`

Turns on interactive paging.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Object Names

Valid characters are letters (A-Z and a-z), digits (0-9), and the hyphen (-) character. The first and last characters of the name must be alphanumeric, and the name must contain at least one letter or '-'.

Most objects in Purity//FA that can be named, including host groups, hosts, volumes, protection groups, volume and protection group suffixes, SNMP managers, and subnets, can be 1-63 characters in length.

Array names can be 1-56 characters in length. The array name length is limited to 56 characters so that the names of the individual controllers, which are assigned by Purity//FA based on the array name, do not exceed the maximum allowed by DNS.

Names are case-insensitive on input. For example, **`vol1`**, **`Vol1`**, and **`VOL1`** all represent the same volume. Purity//FA displays names in the case in which they were specified when created or renamed.

All objects in a pod have a fully qualified name that include the pod name and object name. The fully qualified name of a volume in a pod is **`POD::VOLUME`**, with double colons (`::`) separating the pod name and volume name. The fully qualified name of a protection group in a pod is **`POD::PGROUP`**, with double colons (`::`) separating the pod name and protection group name. For example, the fully qualified name of a volume named **`vol01`** in a pod named **`pod01`** is **`pod01::vol01`**, and the fully qualified name of a protection group named **`pgroup01`** in a pod named **`pod01`** is **`pod01::pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target array, the fully qualified name of the protection group on the target array is **`POD:PGROUP`**, with single colons (`:`) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to target array **`array02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod01:pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target pod, the fully qualified name of the protection group on the target pod is **`POD:PGROUP`**, with single colons (`:`) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to a target pod **`pod02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod02::pod01:pgroup01`**.

All objects in a volume group have a fully qualified name that includes the volume group name and the object name, separated by a forward slash (`/`). For example, the fully qualified name of a volume named **`vol01`** in a volume group named **`vgroup01`** is **`vgroup01/vol01`**.

Realms, pods, and volume groups provide a namespace with unique naming conventions. All objects in a realm have a fully qualified name that include the realm name and object name or realm, pod, and object names. The fully qualified name of a pod in a realm is **`REALM::POD`**, with double colons (`::`) separating the realm name and pod name. The fully qualified name of a volume in a pod in a realm is **`REALM::POD::VOLUME`**, with double colons (`::`) separating the realm, pod, and volume names. The fully qualified name of a protection group in a pod in a realm is **`REALM::POD::PGROUP`**, with double colons (`::`) separating the realm, pod, and protection group names. The fully qualified name of an object in a realm but not in a pod is **`REALM::HOST`**, using host as an example.

Description

A realm is a unit of management and accounting for data, a feature that is used mostly by storage providers and others in multitenancy environments. Realms offer a way of creating storage areas that are fully separate and independent of each other, even on the same array.

Realms add the following support:

- **Delegated administration**. Administrators can be limited to accessing and modifying objects within one or more realms.
- **Consumption controls**. Array administrators can set provisioning quota limits as well as IOPS and bandwidth QoS limits on each realm.
- **Capacity footprint metrics**. Footprint is the consumed capacity for data within the realm as if that were the only data on the array, taking into account any applicable data reduction.

The **`purerealm`** command creates and manages realms.

Administrators and Permissions

A realm administrator by default controls their realm and does not have visibility to other realms or storage objects, if any, on the array. Access to other realms and to array objects must be explicitly assigned with a management access policy (through the `purepolicy management-access` command).

To a realm administrator, their own realm and its objects appear to be the entire array.

The following **`purerealm`** commands are available to realm administrators:

- **`purerealm eradication-config`**
- **`purerealm eradication-config list`**

- **`purerealm list`**
- **`purerealm monitor`**

The following commands are require array administrator permissions:

- **`purerealm create`**
- **`purerealm destroy`**
- **`purerealm eradicate`**
- **`purerealm recover`**
- **`purerealm rename`**
- **`purerealm setattr`**

See also "CLI Commands for a Realm Administrator" on page 481.

Naming Considerations

Realms, pods, and arrays share a namespace. A realm cannot have the same name as the local array, a connected array, other realms in either the local array or a connected array, or a pod.

Each realm is a separate namespace for the pods, volumes, hosts, and host groups the realm contains. For example, pods can have identical names in different realms.

Creating a Realm

The **`purerealm create`** command creates a realm on the local array. This example creates a realm named `realm-1`:

```
$ purerealm create realm-1
Name
realm-1
```

Include the **`--quota-limit`** option to set an upper limit on the data a realm can contain.

Include the **`--bw-limit`** option to set the absolute bandwidth limit of the realm.

Include the **`--iops-limit`** option to enforce maximum I/O operations processed per second for the realm.

Limits can be removed by setting the value to empty quotes " ".

After a realm has been created, add pods, volumes, hosts, and host groups to the realm.

Pods are created inside a realm through the **`purepod create`** command, as in the following example:

```
$ purepod create realm-1::pod2
Name           Source Array  Status Frozen At Member   Promotion Status Link Count Quota
realm-1::pod2 -       pure01 online -          realm-1  promoted  0       -
```

A pod can be moved inside a realm through the **`purepod move`** command, as in the following example:

```
$ purepod move --from pure01 --to realm-1 pod1
Name           Source Array  Status Frozen At Member   Promotion Status Link Count Quota
realm-1::pod1 -       pure01 online  -        realm-1  promoted  0       -
```

Volumes are created inside a realm through the **`purevol create`** command. Volumes can be moved into or out of a realm and protection groups through the **`purevol move`** command. Hosts are created inside a realm through the **`purehost create`** command. Host groups are created inside a realm through the **`purehgroup create`** command. It is not supported for hosts and host groups created inside a realm to have connections outside the realm.

Destroying a Realm

When a realm is no longer needed, it can be destroyed. Run the **`purerealm destroy`** command to destroy a realm.

When run with no options, the **`purerealm destroy`** command can only destroy a realm if the realm is empty, so before destroying a realm, ensure all pods, volumes, and host groups inside the realm have been either moved out of the realm or destroyed. Or add the **`--destroy-contents`** option to destroy both the realm and its contents, including all pods, volumes, snapshots, and host groups inside the realm.

The destruction of realms and their contents is not immediate. Instead, the realm is placed in an eradication pending period. The realm can be manually recovered or permanently eradicated within the eradication pending period. After the eradication pending period has elapsed, the realm is completely eradicated and not recoverable.

The length of eradication pending period is set by the enabled delay value if the protection group is not protected by SafeMode, or by the disabled delay value if the protection group is protected by SafeMode. (See "Eradication Delays" on page 88 in the `purearray` chapter for information on the disabled delay and enabled delay.)

Notes about a Destroyed Realm

- Storage objects in a destroyed realm are not available to GUI or CLI displays or operations.

  Pods that are members of a destroyed realm are also in a destroyed state and can be recovered only by recovering the realm.
- Pods and other storage objects within a destroyed realm can be eradicated separately, without eradicating the realm.
- Network interfaces and subnets are not owned by a realm. Destroying a realm does not remove the reference from a subnet to the realm. The subnet is still present if the realm is recovered. You can add or remove a subnet to or from a destroyed realm.

Recovering a Realm

A destroyed realm can be recovered at any time during the eradication pending period. Run the **`purerealm recover`** command to recover a realm.

Once a realm has been recovered, its destroyed volumes and protection groups can also be recovered to bring the realm and its contents back to their previous state.

Eradicating a Realm

A destroyed realm can be permanently eradicated at any time during the eradication pending period (unless the SafeMode manual eradication prevention feature is enabled). Run the **`purepod eradicate`** command to eradicate a realm.

The **`purerealm eradicate`** command only eradicates a realm if its destroyed pods and volumes are already eradicated. Therefore, before eradicating a realm, first eradicate all its destroyed pods and volumes. Or add the **`--eradicate-contents`** option to also eradicate the realm's destroyed pods and volumes.

Once a realm has been eradicated, it and its member objects can no longer be recovered.

Renaming a Realm

Run the **purerealm rename** subcommand to change the current name (**OLD-NAME**) of the realm to the new name (**NEW-NAME**). The name change is effective immediately and the old name is no longer recognized in CLI or GUI interactions. All pods, volumes, pod snapshots, and volume snapshots with the realm namespace are also renamed. (In the Purity//FA GUI, the new name appears upon page refresh.)

Additional purerealm Commands

The **purerealm eradication-config list** command displays the eradication configuration for the current realm or for the specified realm or realms.

```
$ purerealm eradication-config list
Name     Manual Eradication
realm-1  all-enabled
realm-2  all-enabled
```

The **purerealm list** command displays information about the current realm or about one or more specified realms. With no options or arguments, the **purerealm list** command displays the name of the current realm.

```
$ purerealm list
Name     Quota Limit
realm-1  -
```

The **purerealm monitor** command displays I/O performance information for the current realm or for the specified realm or realms.

```
$ purerealm monitor
Name     Quota Limit Bandwidth Limit (B/s)  IOPS Limit
realm-1  -                                  -
```

Use the **purerealm setattr** command to change the bandwidth, IOPS, or data limits for the current realm or for the specified realm or realms. Use the **--bw-limit** option to set the absolute bandwidth limit of the realm. Use the **--iops-limit** option to enforce maximum I/O operations processed per second for the realm. Use the **--quota-limit** option to set an upper limit on the data that the realm can contain. Add the **--ignore-usage** option with **--quota-**

`limit` to set a quota limit that is lower than the existing usage. Limits can be removed by setting the value to empty quotes "".

The following examples first set the bandwidth limit to 1G and then remove the bandwidth limit.

```
$ purerealm setattr --bw-limit 1G realm-1
Name     Bandwidth Limit (B/s) IOPS Limit
realm-1  1G                    -
```

```
$ purerealm setattr --bw-limit "" realm-1
Name     Bandwidth Limit (B/s) IOPS Limit
realm-1  -                     -
```

Objects within a Realm

The following objects are supported within a realm: pods, hosts, and host groups.

The following objects are supported within a pod in a realm: volumes, volume snapshots, protection groups, protection group snapshots, and volume groups.

The following are examples of creating realm objects.

Create a pod within the realm:

```
$ purepod create realm-1::pod1
```

Create a host within the realm:

```
$ purehost create realm-1::host1
```

Create a host group within the realm:

```
$ purehgroup create realm-1::hgroup1
```

Create volume within a pod in the realm:

```
$ purevol create --size 10G realm-1::pod1::vol1
```

Create a volume group within a realm pod:

```
$ purevgroup create realm-1::pod1::vgroup1
```

Create a protection group within a realm pod:

```
$ purepgroup create realm-1::pod1::pgroup1
```

CLI Commands for a Realm Administrator

Users who have both a realm admin management access policy and an array-level admin access policy have access to all CLI commands. This section discusses users who have only one or more realm management access policies attached but do not have an array-level access policy.

Realm administrators have access to the following CLI commands. However, realm administrators often do not have access to all subcommands.

- pureadmin
- purearray
- purehelp
- purehgroup
- purehost
- pureman
- purepgroup
- purepod
- purepolicy
- pureport
- purerealm
- purevgroup
- purevol

The command line syntax help typically lists which subcommands and options a realm administrator can run. Here, **`purepolicy -h`** lists **`password`** as the subcommand that is available to a realm admin.

(In this release, not all syntax help listings display only realm administrator subcommands.)

```
$ purepolicy -h
usage: purepolicy [-h] {password} ...
positional arguments:
   password    manage password policies
```

The following are examples of CLI commands for a realm admin.

List limited information about the array.

```
$ purearray list
Name    ID                     OS          Version
pure01  00abcdab-abcd-0123   Purity//FA 6.6.11
```

Create a host within the realm:

```
$ purehost create realm-1::host1
```

Create a host group within the realm:

```
$ purehgroup create realm-1::hgroup1
```

Create volume within a pod in the realm:

```
$ purevol create --size 10G realm-1::pod1::vol1
```

Create a volume group within a realm pod:

```
$ purevgroup create realm-1::pod1::vgroup1
```

Create a protection group within a realm pod:

```
$ purepgroup create realm-1::pod1::pgroup1
```

List eradication settings.

```
$ purearray eradication-config list
Enabled  Delay  Disabled  Delay  Manual  Eradication
1d        8d     -
```

List the current realm name and quota settings, if any.

```
$ purerealm list
Name      Quota Limit
realm-2  -
```

List the current realm QoS settings and space usage.

```
$ purerealm list --qos --space
Name      Provisioned Size   Virtual  Thin Provisioning   Data Reduction   Total
realm-2  0                   0.00     0%                  1.0 to 1       1.0 to 1
```

List the space footprint for the current realm.

```
$ purerealm list --footprint
```

```
Name      Footprint
realm-2   0.00
```

List the current password policy configuration.

```
$ purepolicy password list
Name        Enabled  Attribute                     Value
management  True     Minimum Password Length       1
                     Maximum Login Attempts        -
                     Lockout Duration              -
                     Minimum Characters Per Group  1
                     Minimum Character Groups      1
                     Password History              -
                     Minimum Password Age          -
                     Enforce Dictionary Check      False
                     Enforce Username Check        False
```

List port names and information.

```
$ pureport list
Name      WWN                          Portal IQN NQN Failover
CT0.FC0   50:01:50:01:50:01:50:00      -      -   -   -
CT0.FC1   50:01:50:01:50:01:50:01      -      -   -   -
CT0.FC2   50:0 1:50:01:50:01:50:04     -      -   -   -
```

List and monitor several storage objects.

```
$ purehgroup list
$ purehost list
$ purehost monitor
$ purepod list
$ purepod monitor
$ purepgroup list
$ purevgroup list
$ purevol list
```

Open pureman help for limited CLI commands.

```
$ pureman purehost
```

List supported CLI commands. (The `purehelp`, `help`, and `man` commands all list available CLI commands.)

```
$ purehelp
...
$ help
...
$ man
...
```

Errors

The following are examples of permissions errors that a realm administrator could encounter.

The subcommand `create` involves an operation that is not permitted for a realm administrator (who does not also have an array-level policy attached).

```
$ purehost create host1
Error on host1: Operation not permitted.
```

An operation on a different realm (for which this realm administrator does not have a management access policy attached) is not allowed. This realm administrator has an access policy for `realm-A`.

```
$ purerealm list --qos --space realm-A realm-B
Error on realm-B: Operation not permitted.
```

Array-level commands are not supported for a realm administrator (who does not also have an array-level policy attached).

```
$ puread account list -h
Error: Unknown command - puread
```

Array-level Commands

Most array-level operations are not available to realm administrators. Realm administrators do not have access to the following CLI commands, unless explicitly granted by an array-level management access policy.

- puread
- purealert
- pureapiclient
- pureapp
- pureaudit
- purecert
- pureconfig
- puredir
- puredns
- puredrive
- pureds
- purefile
- purefs
- purehw
- purekmip
- purelog
- puremaintenance
- puremessage
- puremultifactor
- purenetwork
- pureoffload
- pureplugin
- puresession
- puresmis
- puresmtp
- puresnmp

- puresso
- puresubnet
- puresupport
- puresw
- purevchost

Attempts to run one of those CLI commands by a user who does not have array admin permissions results in the following error: "Error: Operation not permitted".

Examples

Example 1

```
purerealm create realm1
```

Creates a realm named `realm1` on the current array.

Example 2

```
purerealm create --quota-limit 10G --bw-limit 10G realm2
```

Creates a realm with quota and bandwidth limits.

Example 3

```
purerealm setattr --bw-limit 2G realm2
```

Adds or changes the bandwidth limit for the existing realm `realm2`. The output includes both bandwidth and IOPS QoS limits for `realm2`.

Example 4

```
purerealm list
```

Lists the realms on the current array. The listing includes the quota limit values for each realm.

Example 5

```
purerealm list --qos
```

Lists the realms with their QoS bandwidth and IOPS limits (but not quota limits).

Example 6

```
purerealm list realm2 realm3
```

Lists the realms and quota limits for `realm2` and `realm3`. Returns an error if any specified realm does not exist.

Example 7

```
purerealm list --space
```

Lists space metrics for all realms on the array.

Example 8

```
purerealm destroy realm2
```

Destroys `realm2`, which must be empty.

Example 9

```
purerealm destroy --destroy-contents realm3
```

Destroys `realm3`, including its contents, if any.

Example 10

```
purerealm recover realm3
```

Recovers `realm3`, including any realm contents that has not been manually eradicated.

Example 11

```
purerealm eradicate realm2
```

Eradicates `realm2`, which must be empty, be in the destroyed state, and still be during its eradication pending period.

Example 12

```
purerealm eradicate --eradicate-contents realm3
```

Eradicates `realm3`, including its contents, if any. The realm must be in the destroyed state and still be during its eradication pending period.

See Also

pureadmin, purearray, purefleet, purepod, purepolicy

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puresession

puresession, puresession-list – displays user session logs

Synopsis

```
puresession list  [--csv | --nvp] [--notitle] [--raw] [--filter
FILTER] [--page] [--limit LIMIT] [--sort SORT]
```

Options

`-h | --help`
: Can be used with any command or subcommand to display a brief syntax description.

Options that control display format:

`--csv`
: Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`
: Lists information without column titles.

`--nvp`
: Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`
: Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Description

Purity//FA generates *user session logs*, which represent user login and authentication events performed in the Purity//FA GUI, Purity//FA CLI, and REST API. For example, logging in to the Purity//FA CLI with an invalid password, or opening a Pure Storage REST API session generates a user session log entry.

The **`puresession list`** command displays a list of user login and authentication events. For example,

```
puresession list
ID  Start Time           End Time                    Interface  User
54  2020-08-14 11:45:00  PDT 2020-08-14 12:00:00 PDT  REST       pureuser
57  2020-08-14 11:45:00  PDT 2020-08-14 12:00:00 PDT  REST       -
59  2020-08-14 11:45:47  PDT 2020-08-14 11:45:47 PDT  REST       pureuser
63  2020-08-14 11:45:47  PDT 2020-08-14 11:45:47 PDT  REST       root
66  2020-08-14 11:45:00  PDT 2020-08-14 12:00:00 PDT  REST       pureuser
69  2020-08-14 14:15:11  PDT 2020-08-14 15:59:30 PDT  GUI        pureuser
71  2020-08-25 17:25:15  PDT 2020-08-25 19:22:30 PDT  CLI        pureadmin
75  2020-08-25 19:40:37  PDT 2020-08-25 19:40:37 PDT  CLI        pureuser

Location          Event                    Method      Repeats
```

```
10.124.190.231   API token obtained      password    1
10.124.190.231   failed authentication   API token   6
10.124.190.231   user session            API token   -
10.124.190.231   user session            API token   -
10.124.190.231   request without session  session    2
10.123.14.220    user session             password   -
10.123.13.62     user session             public key -
127.0.0.1        user session             password   -
```

The **puresession list** command displays the following user session event information:

`ID`

Unique number assigned by the array to the alert, audit, or user session event. ID numbers are assigned to messages in chronological ascending order.

`Start Time`

For user login events, date and time the user logged in to the Purity//FA interface. For authentication events, start time of the time period in which the authentication action occurred.

`End Time`

For user login events, date and time the user logged out of the Purity//FA interface. For authentication events, end time of the time period in which the authentication action occurred.

`Interface`

Purity//FA GUI, Purity//FA CLI, or REST API through which the user session event was performed.

`User`

Username of the Purity//FA user who triggered the user session event.

`Location`

IP address of the user client connecting to the array.

`Event`

Event that was triggered. Login events include `login` and `user session`. Authentication events include `failed authentication`, `API token obtained`, and '`request without session`.

`Method`

Method by which the user attempted to log in, log out, or authenticate. Methods include `API token`, `password`, `public key`, and `saml2_sso`. `saml2_sso` indicates a session authenticated by an third-party identity provider through SAML2 SSO.

`Repeats`

Number of attempts in addition to an initial attempt that a user has performed an authentication action within the start and end time period.

Examples

Example 1

```
puresession list
```

Displays a list of user login/logout and authentication events.

See Also

purealert, pureaudit,

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puresmis

puresmis, puresmis-enable, puresmis-disable, puresmis-list – manages the Pure Storage Storage Management Initiative Specification (SMI-S) provider

Synopsis

```
puresmis disable [--slp] [--wbem-https]
puresmis enable [--slp] [--wbem-https]
puresmis list [--cli | --csv | --nvp] [--notitle] [--page] [--raw]
```

Options

`-h | --help`
: Can be used with any command or subcommand to display a brief syntax description.

`--slp`
: Enables or disables the Service Location Protocol (SLP) and its ports, TCP 427 and UDP 427.

  SLP is optional.

`--wbem-https`
: Enables or disables the SMI-S provider and its port, TCP 5989.

Options that control display format:

`--cli`
: Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Description

The **`puresmis`** command manages the Pure Storage Storage Management Initiative Specification (SMI-S) provider.

Enable the SMI-S provider to administer the array through an SMI-S client. The SMI-S provider is optional and must be enabled before its first use.

For more information about the SMI-S provider, including command usage and examples, refer to the Pure Storage SMI-S Provider Guide on the Knowledge site at `https://support.purestorage.com`.

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puresmtp

puresmtp, puresmtp-list, puresmtp-setattr – manages connections to Simple Mail Transfer Protocol (smtp)

Synopsis

puresmtp list [--cli | --csv | --nvp] [--notitle]

puresmtp setattr [--body-prefix ***BODY_PREIFX***] [--encryption-mode ***ENCRYPTION_MODE***] [--password] [--relay-host ***RELAY_HOST***] [--sender-domain ***SENDER_DOMAIN***] [--sender-username ***SENDER_USERNAME***] [--subject-prefix ***SUBJECT_PREFIX***] [--username ***USERNAME***]

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--body-prefix BODY_PREFIX`

Customizable string that gets inserted at the beginning of the body of every email that is sent from the array.

`--encryption-mode ENCRYPTION_MODE`

Sets the method of encryption for emails sent from the appliance. You can set it to TLS (i.e. "starttls") or the default by providing a blank string (i.e. "")

`--password`

Displays or sets the SMTP password used to authenticate into the relay host SMTP server. For SMTP servers that require authentication, specify the user name and password.

When used with **puresmtp list**, displays the password in masked form. A dash (–) in the `Password` column means that an SMTP password has not been set.

When used with **`puresmtp setattr`**, sets the SMTP password through interactive prompt. The SMTP user name and password must be set together. To clear the password, press **`Enter`** at the command prompts. Clearing the password implicitly clears the SMTP user name.

`--relay-host RELAY_HOST`

Displays or sets the hostname or IP address of the email relay server currently being used as a forwarding point for email alerts generated by the array. If specifying an IP address, enter the IPv4 or IPv6 address. If a relay host is not configured, Purity//FA sends alert email messages directly to recipient addresses rather than route them via the relay (mail forwarding) server.

For IPv4, specify the IP address in the form **`ddd.ddd.ddd.ddd`**, where **`ddd`** is a number ranging from `0` to `255` representing a group of 8 bits. If a port number is also specified, append it to the end of the address in the format **`ddd.ddd.ddd.ddd:PORT`**, where **`PORT`** represents the port number.

For IPv6, specify the IP address in the form **`xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx`**, where **`xxxx`** is a hexadecimal number representing a group of 16 bits. Consecutive fields of zeros can be shortened by replacing the zeros with a double colon (**`::`**). If a port number is also specified, enclose the entire address in square brackets (**`[]`**) and append the port number to the end of the address. For example, **`[xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx]:PORT`**, where **`PORT`** represents the port number.

`--sender-domain SENDER_DOMAIN`

Displays or sets the sender domain name. The domain name determines how Purity//FA logs are parsed and treated by Pure Storage Technical Services and Escalations. The domain name is also appended to alert email messages. The default domain name is `please-configure.me`. If this is not a Pure Storage test array, set the sender domain to the actual customer domain name.

`--sender-username SENDER_USERNAME`

The local part of the email address when the array sends emails.

`--subject-prefix SUBJECT_PREFIX`

Customizable string that gets inserted at the beginning of the subject of every email that is sent from the array.

`--username USERNAME`

Displays or sets the SMTP account name used to authenticate into the relay host SMTP server. For SMTP servers that require authentication, specify the user name and password. The SMTP user name and password must be set together.

To clear the user name, set to an empty string (**`""`**). Clearing the user name implicitly clears the SMTP password. A dash (**`-`**) in the User Name column means that an SMTP user name has not been set.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

Description

The relay host represents the hostname or IP address of the email relay server currently being used as a forwarding point for alert email notifications generated by the array. The **`puresmtp setattr --relay-host`** command sets the relay host. For SMTP servers that require authentication, also include the **`--username`** and **`--password`** options to specify the username and password, respectively. If a relay host is not configured, Purity//FA sends alert email messages directly to recipient addresses rather than route them via the relay (mail forwarding) server.

The **`puresmtp setattr --sender-domain`** command sets the sender domain name to determine how Purity//FA logs are parsed and treated by Pure Storage Technical Services and Escalations. It is crucial that you set the sender domain to the correct domain name. If the array is not a Pure Storage test array, set the sender domain to the actual customer domain name. For example, `mycompany.com`. If the sender domain is set to the default `please-configure.me` domain name, the array will be treated as an internal Pure Storage test array, potentially blocking alerts and other important warnings. Purity//FA also includes the sender domain name in the email addresses of outgoing alert messages. For example, `pure-ct0@mycompany.com`.

The **`puresmtp setattr --sender-username`** command enables you to set the local part of the email address from which emails like alert notifications are sent from an appliance. For example, if the default email address is `nonspecificarray@mycompany.com`, you can use this command to set it instead to `specificarray@mycompany.com`.

The **`puresmtp setattr --subject-prefix`** command inserts user-created text into the email subject line to always appear before the normal array email subject text is generated. For example, if the array would send an alert email with the following email subject line: "Warning - vm-pureuser23: Insecure default pureuser password [194]", and you used the command to set the prefix, "DC:45 RM:23 RACK:12" so you would be able to quickly know where the array is that is sending you the email, the resulting email would have the subject line: "DC:45 RM:23 RACK:12 Warning - vm-pureuser23: Insecure default pureuser password [194]".

The **`puresmtp setattr --body-prefix`** command performs a similar function, inserting user-created text into the email body before the email body text.

The **`puresmtp list`** command displays the SMTP relay host settings, including relay host name, SMTP user name and password, and sender domain.

The **`puresmtp setattr --encryption-mode`** command enables you to select the method of encryption for emails sent from the appliance. The two available methods include enforcing TLS encryption via starttls or the default method of opportunistic TLS. To use the TLS method, type **`puresmtp setattr --encryption mode "starttls"`**. Emails will not be accepted without TLS encryption.To set it to the default method, type **`puresmtp setattr --encryption mode ""`**.

Examples

Example 1

```
puresmtp list --csv
```

Displays a CSV output of the SMTP relay host settings, including relay host name, SMTP user name and password, and sender domain.

Example 2

```
puresmtp setattr --relay-host relayhostname --username smtpusername --password
Enter password for relayhost:
Retype password for relayhost:
```

Sets the SMTP relay host to `relayhostname`. Also sets the SMTP user name and password, which are used to authenticate into the SMTP server. The password is entered interactively.

Example 3

```
puresmtp setattr --username ""
```

Clears the SMTP user name, which also clears the SMTP password.

Example 4

```
puresmtp setattr --password
Enter password for relayhost: <Press Enter>
Retype password for relayhost: <Press Enter>
```

Clears the SMTP password, which also clears the SMTP user name.

Example 5

```
puresmtp setattr --sender-domain mycompany.com
```

Sets the sender domain name to `mycompany.com`.

Example 6

```
puresmtp setattr --sender-username "ArrayR4D4"
```

Sets the local part of the email address from which an appliance sends alerts to `R4D4`.

Example 7

```
purestmp setattr --subject-prefix "DC3, RM2, Rack21"
```

Inserts `DC3, RM2, Rack21` as a prefix to the subject line of emails sent from the appliance.

Example 8

```
purestmp setattr --body-prefix "Maintenance Scheduled for 2/3/24"
```

Inserts `Maintenance Scheduled for 2/3/24` as a prefix to the body of every email sent from the appliance.

Example 9

```
purestmp setattr --encryption-mode "starttls"
```

Sets the encryption mode for the appliance to `TLS`.

Example 10

```
puresmtp setattr --encryption-mode ""
```

Sets the encryption mode for the appliance back to the default.

Example 11

```
puresmtp setattr --subject-prefix 'DC:00 RM:23 RACK:08' --body-prefix 'Classification:
UNCLASSIFIED'
```

Sets the values to the user-created `DC:00 RM:23 RACK:08` prefix and changes the email body text before general email body text to `Classification: UNCLASSIFIED` for SMTP based email alerts.

See Also

purearray

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puresnmp

puresnmp, puresnmp-create, puresnmp-delete, puresnmp-list, puresnmp-rename, puresnmp-setattr, puresnmp-test – manages connections to Simple Network Management Protocol (SNMP) managers

Synopsis

```
puresnmp create [--auth-passphrase] [--auth-protocol AUTH-PROTOCOL
{MD5 | SHA}] [--community] --host HOST [--notification {trap |
inform}] [--privacy-passphrase] [--privacy-protocol PRIVACY-PROTOCOL
{AES | DES}] [--user USER] [--version {v2c | v3}] MANAGER
```

```
puresnmp delete MANAGER...
```

```
puresnmp list [--cli | --csv | --nvp ] [--engine-id] [--notitle]
[--page] [--raw] [MANAGER...]
```

```
puresnmp rename OLD-NAME NEW-NAME
```

```
puresnmp setattr [--auth-passphrase] [--auth-protocol AUTH-PROTOCOL
{MD5 | SHA}] [--community] --host HOST [--notification {trap |
inform}] [--privacy-passphrase] [--privacy-protocol PRIVACY-PROTOCOL
{AES | DES}] [--user USER] [--version {v2c | v3}] MANAGER...
```

```
puresnmp test MANAGER
```

Arguments

MANAGER

Name used by Purity to identify an SNMP Network Management System ("Manager").

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--auth-passphrase`

SNMPv3 only. Passphrase used by Purity to authenticate the array with the specified managers. Purity prompts for the passphrase. The passphrase must be at least 8 characters in length. Allowed characters are [A-Z], [a-z], [0-9], _ (underscore), and - (hyphen). To clear the passphrase, press **`Enter`** at the prompts. Note that the authentication protocol must be configured in order to set the authentication passphrase.

`--auth-protocol` ***`AUTH-PROTOCOL`***

SNMPv3 only. Hash algorithm used to validate the authentication passphrase. Valid values are `MD5`, `SHA`, or null (set to **`""`**). Values are case sensitive.

`--community`

SNMPv2c only. Manager community string under which Purity is to communicate with the specified managers. Purity prompts twice for the community string. Allowed characters are [A-Z], [a-z], [0-9], _ (underscore), - (hyphen), and the following special characters: `!"#$%'()=^~|@`[{;+:*]},<.>/?`

To remove the array from the community, press **`Enter`** at the prompts.

`--engine-id`

SNMPv3 only. Displays the SNMPv3 engine ID generated by Purity for the array. (Some managers require the engine ID in their configuration.)

`--host HOST`

DNS hostname or IP address of a computer that hosts an SNMP manager to which Purity is to send messages when it generates alerts. If specifying an IP address, enter the IPv4 or IPv6 address.

For IPv4, specify the IP address in the form **`ddd.ddd.ddd.ddd`**, where **`ddd`** is a number ranging from `0` to `255` representing a group of 8 bits. If a port number is also specified, append it to the end of the address in the format **`ddd.ddd.ddd.ddd:PORT`**, where **`PORT`** represents the port number.

For IPv6, specify the IP address in the form **`xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx`**, where **`xxxx`** is a hexadecimal number representing a group of 16 bits. Consecutive fields of zeros can be shortened by replacing the zeros with a double colon (**`::`**). If a port number is also specified, enclose the entire address in square brackets (**`[]`**) and append the port number to the end of the address. For example, **`[xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx]:PORT`**, where **`PORT`** represents the port number.

Note: This host option refers to SNMP managers and is not related to host options in other CLI commands.

`--notification`

Notification type that determines whether the recipient (remote host) acknowledges receipt of SNMP messages. Valid options are **`trap`** (default) and **`inform`**. An SNMP trap is an unacknowledged (asynchronous) SNMP message, meaning the recipient does not acknowledge receipt of the message. An SNMP inform request is an acknowledged trap. Values are case sensitive.

`--privacy-passphrase`

SNMPv3 only. Passphrase used to encrypt SNMP messages. Purity prompts for the passphrase, which must be at least 8 non-space ASCII characters. To clear the passphrase, press **`Enter`** at the prompts. Note that the privacy protocol must be configured in order to set the privacy passphrase.

`--privacy-protocol` ***`PRIVACY_PROTOCOL`***

SNMPv3 only. Encryption protocol for SNMP messages. Valid values are `AES`, `DES`, or null (set to `""`). Values are case sensitive. Note that the authentication settings must be configured in order to set the privacy options.

`--user`

Required for SNMPv3. User ID that Purity uses in communications with SNMP managers. Allowed characters are [A-Z], [a-z], [0-9], _ (underscore), and - (hyphen).

`--version`

Version of the SNMP protocol to be used by Purity in communications with the specified manager(s). Valid values are `v2c` (default) and `v3`. Values are case sensitive.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed

both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Description

The Simple Network Management Protocol (SNMP) is used by SNMP agents and SNMP managers to send and retrieve information. FlashArray supports SNMP versions v2c and v3.

In the FlashArray, the built-in SNMP agent has local knowledge of the array. The agent collects and organizes this array information and translates it via SNMP to or from the SNMP managers running in hosts. The agent, named `localhost`, cannot be deleted or renamed. The managers are defined by creating SNMP manager objects on the array. The managers communicate with the agent via the standard TCP port 161, and they receive notifications on port 162.

In the FlashArray, the `localhost` SNMP agent has two functions, namely, responding to GET-type SNMP requests and transmitting alert messages.

The agent responds to GET-type SNMP requests made by the SNMP managers, returning values for an information block, such as purePerformance, or individual variables within the block, depending on the type of request issued. The variables supported are:

`pureArrayReadBandwidth`	**`Current array-to-host data transfer rate`**
`pureArrayWriteBandwidth`	**`Current host-to-array data transfer rate`**
`pureArrayReadIOPS`	**`Current read request execution rate`**
`pureArrayWriteIOPS`	**`Current write request execution rate`**
`pureArrayReadLatency`	**`Current average read request latency`**
`pureArrayWriteLatency`	**`Current average write request latency`**

The FlashArray Management Information Base (MIB) describes the purePerformance variables. Download the MIB through the **Settings > System > SNMP** panel of the GUI.

Note that SNMP does not support MIB-2 object identifiers (OID). The MIB is entirely custom and only contains the OIDs that are officially supported.

The SNMP agent generates and transmits messages to the SNMP manager as traps or inform requests (informs), depending on the notification type that is configured on the manager. An SNMP trap is an unacknowledged SNMP message, meaning the SNMP manager does not acknowledge receipt of the message. An SNMP inform is an acknowledged trap.

To configure the notification type, include the **`--notification`** option when creating or modifying the attributes of the SNMP manager. If the SNMP manager notification type is set to `trap`, the agent sends the SNMP message (trap) without expecting a response. If the SNMP manager is set to `inform`, the agent sends the SNMP message (inform) and waits for a reply from the manager confirming message retrieval. If the agent does not receive a response within a certain timeframe, it will retry until the inform has passed through successfully. If the notification type is not set, the manager defaults to `trap`.

The **`puresnmp create`** command creates a Purity SNMP manager object. Include the required **`--host`** and **`--version`** options when creating an SNMP manager object. Include the `--host` option to specify the DNS hostname or IP address of the computer that hosts the SNMP manager. Include the **`--version`** option to specify the version of the SNMP protocol. Valid versions are `v2c` and `v3`.

SNMPv2 uses a type of password called a community string to authenticate the messages that are passed between the agent and manager. The community string is sent in clear text, which is considered an unsecured form of communication. SNMPv3 supports secure communication between the agent and manager through the use of authentication and privacy encryption methods. As such, SNMPv2c and SNMPv3 have different security attributes.

To configure the SNMPv2c agent and managers, include the **`--community`** option to set the community string under which the agent is to communicate with the managers. The agent and manager must belong to the same community; otherwise, the agent will not accept requests from the manager. When setting the community, Purity prompts twice for the community string. To remove the agent or manager from the community, press **`Enter`** at the community prompts.

To configure the SNMPv3 agent and managers, include the **`--user`** option to specify the user ID that Purity uses to communicate with the SNMP manager. Also set the authentication and privacy encryption security levels for the agent and managers. SNMPv3 supports the following security levels:

- **noAuthNoPriv.** Authentication and privacy encryption is not set. Similar to SNMPv2c, communication between the SNMP agent and managers is not authenticated and not encrypted. noAuthNPriv security requires no configuration.

- **authNoPriv.** Authentication is set, but privacy encryption is not set. Communication between the SNMP agent and managers is authenticated but not encrypted. Password authentication is based on `MD5` or `SHA` hash authentication. To configure authNoPriv security, set the **`--auth-protocol`** and **`--auth-passphrase`** attributes.
- **authPriv.** Communication between the SNMP agent and managers is authenticated and encrypted. Password authentication is based on `MD5` or `SHA` hash authentication. Traffic between the FlashArray and SNMP manager is encrypted using encryption protocol `AES` or `DES`. To configure authPriv security, set the **`--auth-protocol`**, **`--auth-passphrase`**, **`--privacy-protocol`**, and **`--privacy-passphrase`** attributes.

Note that privacy cannot be configured without authentication.

Once an SNMP manager is created, the FlashArray immediately starts transmitting SNMP messages and alerts to the manager.

The **`puresnmp list`** command displays the communication and security attributes of the SNMP agent and managers. Include the **`--engine-id`** option to display the array's engine ID, which may be required to configure some of the SNMP managers.

The **`puresnmp rename`** command changes the name of the specified SNMP manager object. SNMP manager names are used in Purity administrative commands, and have no external significance. The name change is effective immediately. The SNMP `localhost` agent cannot be renamed.

The **`puresnmp setattr`** command changes the hostname, IP address, notification, or SNMP version, corresponding protocol and security attributes, and notification type of the specified SNMP manager.

Changing the SNMP version clears all of the previous version settings.

The **`puresnmp test`** command sends a test SNMP message (trap or inform) to the specified manager. For SNMP traps, successful transmission is verified at the destination. For SNMP informs, the recipient sends a response, acknowledging receipt of the SNMP message.

The **`puresnmp delete`** command stops communication with the specified SNMP manager and deletes the SNMP manager object from Purity.

Examples

Example 1

```
puresnmp create --host mysnmp.example.com --community MySNMPManager
Enter community:
Retype community:
```

Creates an SNMP manager object named `MySNMPManager` for the SNMP manager running in host `mysnmp.example.com`. Purity prompts for the new community string. Purity uses SNMPv2c to communicate in the new SNMP community.

Example 2

```
puresnmp delete MySNMPManager
```

Stops transmission of any future messages to `MySNMPManager` and deletes the SNMP manager. If `MySNMPManager` is recreated at a later time, its attributes must be re-defined.

Example 3

```
puresnmp list MySNMPManager
```

Displays the protocol and security attributes for `MySNMPManager`.

Example 4

```
puresnmp rename MySNMPManager MySNMPManager1
```

Changes the name of `MySNMPManager` to `MySNMPManager1`. None of the manager object's protocol or security attributes are changed.

Example 5

```
puresnmp setattr --host 172.169.0.12 MySNMPManager
```

Changes the IPv4 address associated with `MySNMPManager` to `172.169.0.12`.

Example 6

```
puresnmp setattr --host [2001:db8:85a3::8a2e:370:7334] MySNMPManager
```

Changes the IPv6 address associated with `MySNMPManager` to `2001:db8:85a3::8a2e:370:7334`.

Example 7

```
puresnmp setattr --auth-passphrase MySNMPManager
Enter auth passphrase:
Retype auth passphrase:
Name    Host              Version Community User  Auth Protocol Auth Pass ...
MyV3Mgr mysnmp.example.com v3      -         User1 SHA           ****      ...
```

For an existing SNMPv3 manager named `MySNMPManager`, sets the authentication passphrase to a new value. Purity prompts for the new authentication passphrase value.

Example 8

```
puresnmp setattr --auth-protocol MD5 MySNMPManager
Name          Host               Version Community User  Auth Protocol Auth Pass ...
MySNMPManager mysnmp.example.com v3      -         User1 MD5           ****      ...
```

For an existing SNMPv3 manager named `MySNMPManager`, sets the authentication protocol to `MD5`.

Example 9

```
puresnmp setattr --auth-protocol SHA --auth-passphrase MySNMPManager
Enter auth passphrase:
Retype auth passphrase:
Name          Host               Version Community User Auth Protocol Auth Pass ...
MySNMPManager mysnmp.example.com v3      - User1          SHA           ****      ...
```

For an existing SNMPv3 manager named `MySNMPManager`, sets the authentication protocol attribute to `SHA` and sets the authentication passphrase to a new value. Purity prompts for the new authentication passphrase value.

Example 10

```
puresnmp setattr --community MySNMPManager
Enter community:
Retype community:
```

```
Name          Host                 Version Community User Auth Protocol Auth Pass ...
MySNMPManager mysnmp.example.com v2c     ****      -    -             -         ...
```

For an existing SNMPv2c manager named `MySNMPManager`, sets the community string to a new value. Purity prompts for the new community string.

Example 11

```
puresnmp setattr --version v3 --user User1 localhost
Name      Host      Version Community User  Auth Protocol Auth Pass ...
localhost localhost v3      -         User1 -             -         ...
```

Sets the existing `localhost` agent to SNMPv3, with a user named `User1`. SNMP Managers can now communicate with the built-in SNMP agent using the SNMPv3 protocol.

Example 12

```
puresnmp setattr --notification inform MySNMPManager
Name          Host                 Version Community User Auth Prot ... Notification
MySNMPManager mysnmp.example.com v2c     ****      -    -         ... inform
```

Sets the notification type for SNMP manager `MySNMPManager` to `inform`. SNMP messages sent will be acknowledged.

See Also

purearray, puremessage

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puresso

puresso-saml2, puresso-saml2-create, puresso-saml2-delete, puresso-saml2-disable, puresso-saml2-enable, puresso-saml2-list, puresso-saml2-setattr, puresso-saml2-test – configure, display, manage, and test SAML2 SSO settings

Synopsis

puresso `saml2 create [--array-url` ***`ARRAY_URL`***`] [--signing-credential` ***`SIGNING_CREDENTIAL`***`] [--decryption-credential` ***`DECRYPTION_CREDENTIAL`***`] [--idp-entity-id` ***`IDP_ENTITY_ID`***`] [--idp-url` ***`IDP_URL`***`] [--idp-metatadata-url` ***`IDP_METADATA_URL`*** `[--verification-certificate] [--sign-request] [--encrypt-assertion]` ***`IDP-NAME`***

puresso `saml2 delete [`***`IDP-NAME`***`]`

puresso `saml2 disable [--sign-request] [--encrypt-assertion] [`***`IDP-NAME`***`]`

puresso `saml2 enable [--sign-request] [--encrypt-assertion] [`***`IDP-NAME`***`]`

puresso `saml2 list [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--filter` ***`FILTER`***`] [--limit` ***`LIMIT`***`] [--sort` ***`SORT`***`] [--verification-certificate] [`***`IDP-NAME`***`]`

puresso `saml2 setattr [--array-url` ***`ARRAY_URL`***`] [--signing-credential` ***`SIGNING_CREDENTIAL`***`] [--decryption-credential` ***`DECRYPTION_CREDENTIAL`***`] [--idp-entity-id` ***`IDP_ENTITY_ID`***`] [--idp-url` ***`IDP_URL`***`] [--idp-metatadata-url` ***`IDP_METADATA_URL`***`] [--verification-certificate] [`***`IDP-NAME`***`]`

puresso `saml2 test [`***`IDP-NAME`***`]`

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--array-url` ***`ARRAY_URL`***

Specifies the URL for the FlashArray.
Example: `https://myarray.mycompany.com`.

`--decryption-credential` ***`DECRYPTION_CREDENTIAL`***

Specifies the name of the credential that the service provider uses to decrypt the encrypted SAML authentication assertions from the identity provider. The credential is managed by the `purecert` CLI command.

`--encrypt-assertion`

With the **`puresso saml2 create`** command, sets the SSO configuration to enable assertion encryption by the identity provider.

With the **`puresso saml2 enable`** command, enables assertion encryption by the identity provider.

With the **`puresso saml2 disable`** command, disables assertion encryption by the identity provider.

Note: Assertion encryption must also be enabled in the identity provider.

`--idp-entity-id` ***`IDP_ENTITY_ID`***

Specifies the globally unique name for the identity provider.

Example: `https://myidp.company.com/adfs/services/trust`.

`--idp-metatadata-url` ***`IDP_METADATA_URL`***

Specifies the URL of the identity provider metadata XML file. When this option is provided, the IdP URL, IdP entity ID, and verification certificate are taken from the metadata file if those options are not present on the command line.

Example: `https://myidp.company.com/federationmetadata/2007-06/federationmetadata.xml`.

`--idp-url` ***`IDP_URL`***

Specifies the URL of the identity provider.

Example: `https://myidp.company.com/adfs/ls`.

`--sign-request`

With the **puresso saml2 create** command, sets the SSO configuration to enable signed authentication requests by the service provider.

With the **puresso saml2 enable** command, enables signed authentication requests by the service provider.

With the **puresso saml2 disable** command, disables signed authentication requests by the service provider.

Note: For authentication to work with signed authentication requests by the service provider, the signature verification certificate for these requests must also be configured on the identity provider.

`--signing-credential` ***SIGNING_CREDENTIAL***

Specifies the name of the credential that the service provider uses to sign the SAML authentication request. The credential is managed by the `purecert` CLI command.

`--verification-certificate`

Prompts for the X.509 PEM format verification certificate (including the "`-----BEGIN CERTIFICATE-----`" and "`-----END CERTIFICATE-----`" lines.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Arguments

`IDP-NAME`

The local display name for the SAML2 SSO configuration on the array.

Description

Purity supports single sign-on (SSO) integration with the Microsoft® Active Directory Federation Services (AD FS) authentication identity management system. When SSO is configured and enabled, SAML users are redirected to AD FS for authentication during login attempts to the Purity//FA management GUI. AD FS integration can optionally be configured to enable multi-factor authentication (MFA).

The **`puresso`** command configures SSO attributes in both the service provider (SP), which is included in Purity//FA, and in the third-party identity provider.

Completing the SSO configuration requires additional steps not available with the **`puresso`** CLI command. Those changes are in the AD FS identity provider (IdP) configuration and a recommended end-to-end configuration test that is available only in the Purity//FA GUI. See "Summary of Configuration Steps" on page 517 and "Typical Configuration Run" on page 518 for details.

Run the **puresso saml2 create** command to create a SAML2 SSO configuration for a FlashArray.

With the **--idp-metadata-url** option, if any of the IdP provider, IdP entity ID, and verification certificate options is not included in the command, **puresso** reads that information from the specified metadata XML file.

```
puresso saml2 create --array-url https://myarray.mycompany.com
  --idp-metadata-url ./metadata.xml new-idp
```

If the **--verification-certificate** option is included in the command, **puresso** prompts for the verification certificate. Options are shown here on separate lines for readability but are entered as a single line:

```
puresso saml2 create --array-url https://myarray.mycompany.com
   --idp-entity-id https://myidp.mycompany.com/adfs/services/trust
   --idp-url https://myidp.mycompany.com/adfs/ls
   --verification-certificate new-idp
   Please enter certificate data followed by Enter and then Ctrl-D:
   -----BEGIN CERTIFICATE-----
   MIIC/jCCAeagAwIBAgIQRn+gw3HqIoVGtWHEhX9DVzANBgkqhkiG9w0BAQsFADA7
   ... B8W7SRLXavvimLsHW5f37G6PIWsC/j7TvhsZ8wlOhWk9fC7t2J/c4WlPuPPe
   -----END CERTIFICATE-----
```

Any attributes not specified with the **create** command can be configured later with the **puresso saml2 setattr** command.

Note: The newly created configuration is not usable until corresponding changes are made in the IdP and the configuration has been tested and enabled.

Run the **puresso saml2 setattr** command to change configuration attributes. Combinations of options are supported. The following command changes the FlashArray URL attribute:

```
puresso saml2 setattr --array-url https://pure01.mycompany.com new-idp
```

After creating the SAML2 configuration, use the **puresso saml2 test** command to verify the SP configuration and the verification certificate. If the test fails, use the **puresso saml2 setattr** command to adjust the configuration and then test again.

The following is example output of a successful **puresso saml2 test** run:

```
# puresso saml2 test
Name          Type         Description          Destination   Success Details
example-idp   correctness   array URL            https://pure01 True    OK
example-idp   correctness   verification cert    -              True    OK
management    correctness   directory service    ldaps://fa-ds  True    OK
```

The `puresso saml2 test` command is strongly recommended at the initial configuration and also after any configuration changes. The CLI `puresso saml2 test` command verifies the array URL, directory service, metadata URL (if provided), and verification certificate. In contrast, the GUI SAML2 SSO test (Settings > Access tab > SAML2 SSO pane > Test) runs a complete end-to-end test that includes the IdP settings.

Run the `puresso saml2 enable` command to enable SAML2 SSO on the array, but only after passing a CLI `puresso saml2 test` run and optionally also the Purity//FA GUI SAML2 SSO test.

Later use `puresso saml2 enable` with the `--encrypt-assertion` option to enable SAML assertion encryption by the identity provider, and the `--sign-request` option to enable signed SAML requests by the service provider. Both are optional and both require additional configuration steps in the identity provider.

To display the SAML2 SSO configuration attributes, use the `puresso saml2 list` command. To display the verification certificate, add the `--verification-certificate` option.

Run the `puresso saml2 disable` command with no options to disable SSO on the array. This command returns the array to its previous authentication mechanism. Use the `--encrypt-assertion` option to disable assertion encryption. Use the `--sign-request` option to disable signed SAML requests by the service provider.

To delete an existing SAML2 SSO configuration associated with an identity provider, use the `puresso saml2 delete` command.

Prerequisites

- The directory service used with the array must be the same directory service instance as used by the identity provider in the relying party trust configuration for this array.
- Access to the AD FS Management Tool or coordination with an AD FS administrator. Configuration steps in Purity//FA require the following information about AD FS:
  - The IdP entity ID, which specifies the globally unique name for the identity provider .

- The IdP URL, which specifies the URL of the identity provider.
- The IdP verification certificate

The IdP metadata file contains this information and having the IdP metadata file URL is sufficient.

- The AD FS server must be configured to support TLS 1.2 or 1.3 with strong authentication, if AD FS monitoring is to be configured for the array relying party trust.

Important: Before configuring SSO, create a strong password for the `pureuser` and other array administrator accounts and save those passwords according to your organization's security policies. For example, generate a random password and store it in a password vault. In case your IdP system becomes unavailable, these administrator accounts can still log into the array with password authentication.

Notes about SAML2 SSO

- In this release, only one SAML2 SSO configuration can be created at a time on an array.
- Certificates must be X.509 certificates in PEM format.
- In this release, SSO authentication applies only to GUI logins. SSH logins continue to use their existing password authentication or other authentication mechanism, including LDAP authentication or multi-factor authentication with RSA SecurID® software.

Notes about User Accounts and Sessions

- When SAML2 SSO is enabled, all SAML users logging into the Purity//FA management GUI must be authenticated through SSO and the identity provider.
- If a user is deactivated in AD FS, that user's current login session is not affected. The user is denied access at the next login attempt.
- By default, an SSO session times out after eight hours. A different time out length can be configured in the identity provider.

Note about the Array URL Configuration and its Implications

Part of the SSO configuration is an Array URL field for the FlashArray URL, which is based either on a hostname (such as `pure01`) or on an FQDN (such as `pure01.mycompany.com`). Consider which form you want to use for the FlashArray URL before opening a browser to perform the SAML2 SSO configuration.

There are two implications from your choice of Array URL.

- The browser URL displayed for Purity user sessions.

  The Array URL field determines the displayed browser URL for Purity//FA GUI user sessions authenticated through SAML2 SSO, no matter which form a user specifies when opening the browser.

  For example, if the Array URL in the SSO configuration is the FQDN, `https://pure01.mycompany.com`, then all user sessions authenticated through SAML2 SSO display the FQDN in the browser URL (`https://pure01.mycompany.com/dashboard`, `https://pure01.mycompany.com/storage/array`, etc.), even after a user enters `https://pure01` to log into Purity//FA.
- Potential failure to find test results.

  Whichever form you decide on, also use that form to open the Purity GUI when you start to configure SAML2 SSO.

  If there is a mismatch (hostname vs FQDN) between the form in the browser where the end-to-end test is run and the form in the configuration Array URL, then after the end-to-end configuration test, the browser will encounter an error and will report a failure to find results for the end-to-end test.

Summary of Configuration Steps

This list summarizes the steps to configure and enable SAML2 SSO on an array. Configurations handled by the **`puresso`** command are in bold.

1. Configure directory service groups and group-to-role mappings with management access policies.
2. **Configure the service provider**.

3 **Run the basic configuration test**.
4 Configure the Active Directory Federation Services IdP.
5 Perform the end-to-end test in the Purity//FA GUI.
6 **Enable SAML2 SSO authentication.**

The **`puresso`** command configures the service provider, runs the basic test, and enables SAML2 SSO. The **`pureds`** command configures directory service groups and group-to-role mappings. The IdP is configured with third-party tools such as the AD FS Management Tool. The end-to-end test is run from the Purity//FA GUI (Settings > Access > SAML2 SSO).

Typical Configuration Run

These sections describe the initial basic configuration of SAML2 SSO components.

Configure Directory Service Groups and Group-to-role Mappings

This step is not required if the directory service already meets the following:

- The directory service configured with the array must be the same directory service instance used in the IdP relying party trust configuration for this array.
- The directory service configuration must include groups and group-to-role mappings for SAML users.

This is an example CLI command to configure the directory service:

```
pureds setattr --uri ldaps://ad1.mycompany.com
               --base-dn DC=mycompany,DC=com
               --bind-user ldapreader
```

Alternatively, the directory service can be configured in the Purity//FA GUI (Settings > Access tab > Directory Service pane).

Configure the Service Provider

1 Create the SAML2 SSO configuration with either one of the following commands:

```
puresso saml2 create example-idp --array-url https://pure01
        --idp-url https://fa-saml2/adfs/ls
        --idp-entity-id https://fa-saml2/adfs/services/trust
        --verification-certificate
```

or:

```
puresso saml2 create example-idp --array-url https://pure01
        --idp-metadata-url ./metadata_file
```

When the `--idp-metadata-url` version is used, the IdP URL, IdP entity ID, and verification certificate are taken from the metadata file if those attributes are not provided in the command.

2 Run the CLI configuration test:

```
puresso saml2 test
Name        Type        Description       Destination    Success Details
example-idp correctness array URL         https://pure01    True    OK
example-idp correctness verification cert -                 True    OK
management  correctness directory service ldaps://fa-saml2 True    OK
```

Configure the Active Directory Federation Services IdP

1 Optionally run the following command to list the service provider information required for the AD FS configuration, to use for copying into the IdP configuration:

```
puresso saml2 list --nvp
```

2 On the machine running AD FS, open **Control Panel > System and Security > Administrative Tools** and select **AD FS Management.**

3 In the left panel, select **Relying Party Trusts**. In the right panel, select **Add Relying Party Trust....**

4 The Add Relying Party Trust Wizard opens.

Table 7. **Actions Required in the Add Relying Party Trust Wizard**

Wizard Page	Action
Welcome	Ensure **Claims aware** is selected
Select Data Source	Select **Enter data about the relying party manually**
Specify Display Name	Enter a display name for the new relying party (for convenience, this name could match the SP configuration name, as set by the `puresso saml2 create` command)
Configure Certificate	No action
Configure URL	Select **Enable support for the SAML 2.0 Web SSO protocol** In the **Relying party SAML 2.0 SSO service URL** field, enter the Assertion Consumer URL from the Purity SAML2 SSO pane

Wizard Page	Action
Configure Identifiers	In the **Relying party trust identifier** field, enter the SP entity ID
Choose Access Control Policy	Select **Permit everyone**
Ready to Add Trust	No action
Finish	Select **Configure claims insurance policy for this application**

Note: Access Control Policy **Permit everyone** corresponds to password authentication. Multi-factor authentication can be configured later if required. Consider waiting until after the SSO configuration passes with password authentication.

5 The new relying party now appears in the Relying Party Trusts table. Select the relying party and in the right panel, select **Edit Claims Insurance Policy...**.

6 The Edit Claims Insurance Policy page opens. **Click Add Rule...** in the bottom right. These rules specify the content the IdP returns in an assertion to the SP. A rule for two attributes is required: one for a unique identity for the user, the second for group information.

7 The Add Transform Claim Rule Wizard opens. In the Select Rule Template page, in the **Claim rule template** field, select **Send LDAP Attributes as Claims** and then click **Next**.

8 Complete the Choose Rule Type page, according to the following table.

Table 8. **Actions in the Add Transform Claim Rule Wizard for Users**

Field	Action
Claim rule name	Enter a descriptive name for the rule, such as **map name id**
Attribute store	Select **Active Directory**
Mapping table LDAP Attribute column	Select **SAM-Account-Name**
Outgoing Claim Type column	Select **Name ID** Click **Finish**

The new rule for name mapping appears in the Edit Claims Insurance Policy page Insurance Transform Rules table.

a Again click **Add Rule...**, this time for the group information rule.

b The Add Transform Claim Rule Wizard opens. In the Select Rule Template page, in the **Claim rule template** field, select **Send LDAP Attributes as Claims**, and then click **Next**.

c Complete the Choose Rule Type page, according to the following table.

Table 9. Actions in the Add Transform Claim Rule Wizard for Groups

Field	Action
Claim rule name	Enter a descriptive name for the rule, such as **group pass info**
Attribute store	Select **Active Directory**
Mapping table LDAP Attribute column	Select **Is-Member-Of-DL**
Outgoing Claim Type column	Select **Group** Click **Finish**

The new rule for group mapping appears in the Edit Claims Insurance Policy page Insurance Transform Rules table.

Perform the End-to-end Test in the Purity//FA GUI

The GUI test is optional but strongly recommended, as it performs an end-to-end test of the authentication process. The test does not affect current user sessions or attempts to login.

1. Open the Purity//FA GUI for the FlashArray.
2. Open the Settings > Access tab and scroll to the SAML2 SSO pane.
3. Click **Test**.

   If the test reports any error, check that the service provider and AD FS configurations match, make any correction if necessary, and rerun the test.

Optionally Enable Multi-factor Authentication

The types of multi-factor authentication available depend on the IdP.

If the GUI end-to-end test is successful with password authentication, optionally enable MFA in the IdP server. The types of MFA supported depend on the IdP server.

1. In the AD FS Management tool Relying Party Trust page, select the relying party you created. In the right panel, select **Edit Access Control Policy...**.
2. On the "Choose an access control policy" page, select **Permit everyone and require MFA** or other MFA option. Click **Apply**.
3. In the AD FS > Service > Authentication Methods > Authentication Methods pane, click **Next** next to Multi-factor Authentication Methods.
4. In the Edit Authentication Methods, select the MFA method. Click **Apply**.
5. Next steps, such as configuring a certificate, depend on the type of MFA selected.
6. After completing the MFA configuration in the IdP, again run the end-to-end test to verify the configuration. See "Perform the End-to-end Test in the Purity//FA GUI" above.

Important: To avoid interrupting login access, run the end-to-end test after any configuration change.

Enable SAML2 SSO Authentication

Warning: Only enable SAML2 SSO authentication if the end-to-end test passes! Otherwise ***all array users could be locked out*** of the Purity//FA GUI. Then the only GUI access to the array is the login page Local Access link.

Run this command to enable SAML2 SSO authentication. Replace `example-idp` with the name of your configuration.

```
puresso saml2 enable example-idp
```

After the SAML2 SSO configuration is enabled, all new login attempts to the Purity//FA management GUI by SAML users are authenticated through the identity provider. Existing user sessions are not affected.

SSH logins continue to use their existing password authentication or other authentication mechanism without SSO.

Examples

Note: The examples show some **`puresso`** commands on multiple lines only because of space imitations. Enter each command on a single line.

Example 1

```
puresso saml2 create --array-url https://myarray.mycompany.com
    --idp-entity-id https://myidp.mycompany.com/adfs/services/trust
    --idp-url https://myidp.mycompany.com/adfs/ls
    --verification-certificate MyIdP
```

Creates a SAML2 SSO configuration named MyIdP for the array `https://myarray.mycompany.com` and the IdP provider `https://myidp.mycompany.com/adfs/ls` with the IdP entity ID `https://myidp.mycompany.com/adfs/services/trust`. The command prompts for the verification certificate.

Example 2

```
puresso saml2 create --array-url https://myarray.mycompany.com
 --idp-metadata-urlchttps://myidp.mycompany.com/federationmetadata/
  2007-06/federationmetadata.xml MyIdP2
```

Creates a SAML2 SSO configuration named MyIdP2, using the Array URL and IdP metadata URL options. IdP provider, IdP entity ID, and verification certificate information is read from the specified metadata file.

Example 3

```
puresso saml2 test
```

Verifies the current service provider configuration and the verification certificate. Does not perform an end-to-end test of all SSO components.

Example 4

```
puresso saml2 setattr --verification-certificate MyIdP
```

Change the verification certificate used in the current configuration. The command prompts for the new certificate.

Any configuration change should be followed by another run of `puresso saml2 test` and optionally the GUI SAML2 SSO end-to-end test (recommended).

Example 5

```
puresso saml2 enable MyIdP
```

Enable the MyIdP SAML2 SSO configuration.

Example 6

```
puresso saml2 enable --encrypt-assertion MyIdP
```

Enable assertion encryption by the identity provider. Assertion encryption also requires a configuration step in the identity provider.

See Also

pureds

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puresubnet

puresubnet, puresubnet-create, puresubnet-delete, puresubnet-disable, puresubnet-enable, puresubnet-list, puresubnet-rename, puresubnet-setattr – manages the subnets and VLANs used to organize the network interfaces

Synopsis

```
puresubnet create [--gateway GATEWAY] [--mtu MTU] --prefix PREFIX [--
vlan VLAN] SUBNET
```

```
puresubnet delete SUBNET
```

```
puresubnet disable SUBNET...
```

```
puresubnet enable SUBNET...
```

```
puresubnet list [--cli | --csv | --nvp] [--notitle] [--page]
[--raw] [--vlan VLAN] [SUBNET...]
```

```
puresubnet rename OLD-NAME NEW-NAME
```

```
puresubnet setattr [--gateway GATEWAY] [--mtu MTU] [--prefix PREFIX]
[--vlan VLAN] SUBNET
```

Arguments

SUBNET
: Subnet name.

NEW-NAME
: Name by which the subnet is to be known after the command executes.

OLD-NAME
: Current name of the subnet to be renamed.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--gateway GATEWAY`

IP address of the gateway through which the specified interface is to communicate with the network. For IPv4, specify the gateway IP address in the form **`ddd.ddd.ddd.ddd`**. For IPv6, specify the gateway IP address in the form **`xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx`**. When specifying an IPv6 address, consecutive fields of zeros can be shortened by replacing the zeros with a double colon (`::`).

To remove the gateway specification, set to a null value (`""`). The interfaces in the subnet inherit the gateway address.

`--mtu`

Specifies the maximum message transfer unit (packet) size for the subnet in bytes. Valid values are integers between 568 and 9000 (inclusive). The default value is 1500. The interfaces in the subnet inherit the MTU value.

`--prefix PREFIX`

IP address of the subnet prefix and prefix length. For IPv4, specify the subnet prefix in the form **`ddd.ddd.ddd.ddd/dd`**. For IPv6, specify the subnet prefix in the form **`xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx:xxxx/xxx`**. When specifying an IPv6 address, consecutive fields of zeros can be shortened by replacing the zeros with a double colon (`::`).

`--vlan VLAN`

VLAN ID number. Specify the VLAN ID if the subnet is being created for VLAN tagging purposes. The VLAN interfaces in the subnet inherit the VLAN ID. To remove the VLAN ID, set `--vlan` to 0.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Description

In Purity//FA, interfaces with common attributes can be organized into subnetworks, (or subnets) to enhance the efficiency of data (iSCSI, NVMe-RoCE, or file), management, and replication traffic if applicable.

If the subnet is assigned a valid IP address, once it is created, all of its enabled interfaces are immediately available for connection. The subnet inherits the services from all of its interfaces. Likewise, the interfaces contained in the subnet inherit the netmask, gateway, MTU, and VLAN ID (if applicable) attributes from the subnet.

Physical, virtual, and bond interfaces can belong to the same subnet. VLAN interfaces can only belong to subnets with other VLAN interfaces.

Physical, virtual, and bond interfaces in a subnet share common address, netmask, and MTU attributes. The subnet can contain a mix of physical, virtual, and bond interfaces, and the interface services can be of any type, such as iSCSI, management, NVMe-RoCE, or replication services. To add physical, virtual, or bond interfaces to a subnet, create the subnet and then run **`purenetwork eth setattr --subnet`** to add the interfaces to the subnet.

A *VLAN interface* is a dedicated virtual network interface that is designed to be used with an organization's virtual local area network (VLAN). Through VLAN interfaces, Purity//FA employs VLAN tags to ensure the data passing between the array and VLANs is securely isolated and routed properly.

VLAN Tagging

VLAN tagging allows customers to isolate traffic through multiple virtual local area networks (VLANs), ensuring data routes to and from the appropriate networks. The array performs the work of tagging and untagging the data that passes between the VLAN and array.

VLAN tagging is supported for the following service types: iSCSI, NVMe-RoCE, and file. Before creating a VLAN interface, verify that one or more of these are configured on the physical interface.

Each port can support multiple VLANs. Note that each array supports up to 32 VLAN IDs, so if multiple hosts connect through multiple VLANs, it is possible to quickly reach the maximum number of paths that a host would support.

The **`puresubnet create`** command creates subnets.

To create and attach VLAN interfaces to a subnet, first, create a subnet and assign it with a VLAN ID number, and second, create a VLAN interface for each of the corresponding physical network interfaces you want to associate with the VLAN. Run **`puresubnet create --prefix --vlan`** to create a subnet for each of the VLANs in the organization, assigning each subnet with the appropriate VLAN ID. For each of the physical network interfaces you want to associate with a VLAN, run **`purenetwork eth create vif --subnet`** to create a VLAN interface (`vif`) and attach it to the subnet. Include the **`--address`** option to create a reachable VLAN interface that is ready to use. All of the VLAN interfaces within a subnet must be in the same VLAN.

In Purity//FA, VLAN interfaces have the naming structure CT***x***.ETH***y***.***z***, where ***x*** denotes the controller (0 or 1), ***y*** denotes the interface (0 or 1), and ***z*** denotes the VLAN ID number. For example, `ct0.eth1.500`.

In the following example, subnet `192.168.100.0/24`, named `ESXHost001` and assigned to VLAN `50`, is being created. The physical interfaces `ct0.eth4` and `ct0.eth5` are being attached to subnet `192.168.100.0/24` (named `ESXHost001`) as VLAN interfaces, with VLAN ID number `50` appended to the interface name to match the VLAN ID number of the subnet.

```
puresubnet create --gateway 192.168.1.1 --mtu 9000 --prefix 192.168.100.0/24
                  --vlan 50 ESXHost001

purenetwork eth create vif --address 192.168.1.10 --subnet ESXHost001 ct0.eth4.50

purenetwork eth create vif --address 192.168.1.11 --subnet ESXHost001 ct0.eth5.50
```

If a physical interface has multiple VLANs, create a subnet for each VLAN, and then create the VLAN interfaces mapping the physical interface to the given VLANs.

When creating the VLAN interface, the VLAN ID number in the VLAN interface name must match the VLAN ID of the subnet. In the following example, the second command will not work because the VLAN ID number of the `ESXHost002` subnet (`600`) does not match the VLAN ID number in the VLAN interface name (`500`).

```
puresubnet create --prefix 2001:db8:85a3::/64 --gateway 2001:0db8:85a3::1
                  --vlan 600 ESXHost002
purenetwork eth create vif --address 2001:0db8:85a3::ae26:8a2e:0370:7334
                       --subnet ESXHost002 ct0.eth1.500
```

If **`--mtu`** is not specified during subnet creation, the value defaults to `1500`. Note that the MTU of a VLAN interface cannot exceed the MTU of the corresponding physical interface. Since an interface inherits the MTU value of its subnet, verify the MTU of the new subnet is valid.

Once a subnet is created, the interfaces within the subnet that are enabled are automatically available and ready to connect.

The **`puresubnet delete`** command deletes subnets that are no longer needed. A subnet can only be deleted if it is empty, so before you delete a subnet, remove all of its interfaces by running **`purenetwork eth delete`**.

The **`puresubnet enable`** and **`puresubnet disable`** commands respectively enable and disable a subnet.

When a subnet is enabled, the interfaces within the subnet that are enabled are automatically available and ready to connect. Interfaces within the subnet that are in disable status remain disabled and cannot be reached. Newly created subnets are automatically enabled.

Disabling a subnet disables all of its interfaces - including the ones that are enabled at the interface level.

Take caution when disabling a subnet. If you disable a subnet that contains interfaces through which administrative sessions are being conducted, the interfaces will lose SSH connection.

The **`puresubnet list`** command displays the attributes of the subnets, including its physical, virtual, bond, and VLAN interfaces. Include the **`--vlan`** option to list only the subnets that are configured with the specified VLAN ID.

The **`puresubnet rename`** command changes the current (OLD-NAME) name of a subnet to the new name (NEW-NAME). The name change is effective immediately and the old name is no longer recognized in CLI, GUI, or REST interactions. Renaming a subnet does not affect any of the interface connections attached to the subnet.

The **`puresubnet setattr`** subcommand modifies the subnet attributes, including subnet gateway, MTU size, prefix, and VLAN ID.

The `--gateway` and `--mtu` options can be set at any time. Note that the MTU of a VLAN interface cannot exceed the MTU of the corresponding physical interface. Since an interface inherits the MTU value of its subnet, verify the MTU is valid.

The `--prefix` and `--vlan` options can only be set if the subnet does not contain interfaces. Only specify the VLAN ID number if the subnet is being created for VLAN tagging purposes. To remove the VLAN ID from a subnet, set `--vlan` to `0`.

Examples

Example 1

```
puresubnet create --prefix 192.168.1.0/24 --vlan 100 ESXHost001
```

Creates subnet `ESXHost001` with prefix `192.168.1.0/24` and VLAN ID `100`.

Example 2

```
$ puresubnet create --prefix 2001:db8:85a3::/64 --vlan 200 ESXHost002
```

Creates subnet `ESXHost002` with prefix `2001:db8:85a3::/64` and VLAN ID `200`.

Example 3

```
puresubnet delete ESXHost001
```

Deletes subnet `ESXHost001`. Subnets can only be deleted if they do not contain network interfaces.

Example 4

```
puresubnet setattr --vlan 50 ESXHost001
```

Sets subnet `ESXHost001` to VLAN ID `50`.

Example 5

```
puresubnet list --vlan 50
```

Displays only subnets set to VLAN ID `50`.

See Also

purearray

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puresupport

puresupport, puresupport-connect, puresupport-diagnostics, puresupport-disable, puresupport-disconnect, puresupport-enable, puresupport-list, puresupport-remoteassist, puresupport-setattr, puresupport-test – manage the phone home facility, remote assistance sessions, proxy settings, Pure1 Edge Service, and Purity optimizations from Pure1 Manage

Synopsis

```
puresupport connect[--duration DURATION]
puresupport diagnostics list
puresupport diagnostics update
puresupport disable {edge-management | edge-agent-update | mitigation
| phonehome}
puresupport disconnect
puresupport enable {edge-management | edge-agent-update | mitigation |
phonehome}
puresupport list [--connect] [--path] [--cli | --csv | --nvp] [--
notitle] [--raw]
puresupport remoteassist connect --access-level-override ACCESS_LEVEL
puresupport remoteassist list
puresupport remoteassist global list
puresupport remoteassist global setattr --default-access-level ACCESS_
LEVEL
puresupport setattr [--proxy PROXY]
puresupport test [--phonehome] [--remoteassist]
```

Arugments

ACCESS_LEVEL
: Either "restricted" or "elevated".

Options

`-h | --help`
: Can be used with any command or subcommand to display a brief syntax description.

`--access-level-override` ***ACCESS_LEVEL***
: Used with **`puresupport remoteassist connect`** to either restrict or elevate the access level of Pure Storage support engineers during a remote assist session, temporarily overriding the default access level.

`--connect`
: When used with **`puresupport list`**, displays remote assistance connection details.

`--default-access-level` ***ACCESS_LEVEL***
: Used with **`puresupport global setattr`** to set the default access level of Pure Storage support engineers during remote assist sessions.

`--duration` ***DURATION***
: Adjust the duration of the remote assistance session by setting a time limit between 4 hours and 48 hours (e.g., 12h). The default is 24 hours.

`--edge-management`
: Enables or disables the opt-in Pure1 Edge Service (PES) agent on the array.

`--edge-agent-update`
: Enables or disables automatic updates to the PES agent on the array.

`--mitigation`
: Enables or disables opt-in self-service optimizations, tuning, and mitigations initiated from Pure1 Manage.

`--phonehome`
: Used by Pure Storage Technical Services to troubleshoot the phone home facility status and the network connectivity state.

`--proxy` ***PROXY***

Sets the server to be used as the HTTP or HTTPS proxy. Specifies the server name, scheme and proxy port number. The format for the option value in the **puresupport setattr** subcommand is: **HTTP(S)://HOSTNAME:PORT**, where **HOSTNAME** is the name of the proxy host, and **PORT** is the TCP/IP port number used by the proxy host. **HTTP(S)://USERNAME@HOSTNAME:PORT**, where **USERNAME** is the required username, can also be used. The CLI prompts for a password interactively.

`--purity-default`

When used with **puresuppport diagnostics update**, this option sets diagnostics to the default version.

`--remoteassist`

Enables Pure Storage Technical Services to test a remote assistance status.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **--cli** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **--csv** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **ITEMNAME=VALUE**. Argument names and information items are displayed flush left. The **--nvp** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **purearray monitor** output, the unformatted version of column title **us/op (read)** is **usec_per_read_op**. The **--raw** output is used to sort and filter list results.

Description

The `puresupport` command manages the phone home facility, remote assistance (RA) sessions, Pure1 Edge Service (PES), and proxy settings.

Phone Home Facility

The phone home facility provides a secure direct link between the array and the Pure Storage Technical Services team. The link is used to transmit log contents and diagnostic information to the Pure Storage Technical Services team. Optionally configure the proxy host for HTTPS communication.

The `puresupport enable phonehome` and `puresupport disable phonehome` commands enable and disable, respectively, automatic hourly transmission of array logs to Pure Storage Technical Services.

The `puresupport list` command displays the phone status (enabled or disabled).

Pure1 Edge Service

The Pure1 Edge Service (PES) provides an opt-in secure communication between management applications in the Pure1 Cloud and companion agents on the FlashArray. PES makes possible opt-in applications, such as self-service upgrade, Pure Fusion, and others, that increase customer autonomy and simplify array management.

The `puresupport enable edge-management` and `puresupport disable edge-management` commands enable and disable, respectively, the PES agent on the array.

The `puresupport enable edge-agent-update` and `puresupport disable edge-agent-update` commands enable and disable, respectively, automatic updates to the PES agent on the array.

The `puresupport list` command displays the status of PES and of the PES agent automatic update facility (`True` for enabled, `False` for disabled).

Both the PES agent and the PES agent automatic update facility require that the phone home facility be enabled and operational.

Purity//FA Optimizations

The `puresupport enable mitigation` command enables the Purity//FA optimization agent, which allows you to initiate self-service Purity//FA optimizations, tuning, and mitigations through Pure1 Manage. In the context of this facility, the term “optimizations” refers to tuning and mitigations, as well as optimizations. Optimizations that you initiate through Pure1 Manage replace the involvement of Pure Storage Technical Servicest to apply these changes through remote assist sessions. In Pure1 Manage, you can view the available optimizations and choose to which to apply (or not apply) on an array-by-array, optimization-by-optimization basis.

The Purity//FA optimization agent requires that the phone home facility be enabled and operational.

The `puresupport disable mitigation` command disallows initiating these self-service changes from Pure1 Manage. When the mitigation facility is disabled, contact Pure Storage Technical Services to apply these changes to Purity//FA through remote assist sessions.

Remote Assistance (RA) Sessions

In some cases, the most efficient way for Pure Storage Technical Services to service a FlashArray array or diagnose problems is through direct access to the array. A remote assistance (RA) session grants Pure Storage Technical Services direct and secure access to the array through a reverse tunnel which you, the administrator, open. This is a two-way communication.

RA sessions can be opened and closed at any time.

The `puresupport connect` command opens an RA session, giving Pure Storage Technical Services the ability to log into the array and effectively establishing an administrative session. Once the RA session is successfully established, the array returns connection details, including the date and time when the session was opened, the date and time when the session expires, and the proxy status (true, if configured). To view the details of an open RA session, run the **`puresupport list --connect`** command.

After the Pure Storage Technical Services team has performed all of the necessary diagnostic or maintenance functions, run the **`puresupport disconnect`** command to terminate the RA session. A remote assist active status of false confirms that the remote assist status has been disconnected and the session has successfully closed. Alternatively, a user can set the duration of an RA session to be between 4h and 48h using the **`puresupport connect --duration`** command. The default is 24h. To reset the duration of an active RA session, re-run the **`puresupport connection --duration`** command at anytime.

An open RA session automatically terminates (disconnects) after two days have elapsed.

To view the connection status of the RA session, run the `puresupport list` command.

Support Shell

In some situations, you may prefer that Pure Storage support engineers have restricted access to your array during RA sessions. There are two levels of access, "Elevated" and "Restricted". By default, support engineers have elevated access during RA sessions.

To list the default access level for your array, use the `puresupport remoteassist global list` command.

```
puresupport remoteassist global list
Default Access Level
elevated
```

You can change the default access level by using the `puresupport remoteassist global setattr --default-access-level` command.

```
puresupport remoteassist global setattr --default-access-level restricted
Default Access Level
restricted
```

To list the access level for your current RA session, as well as the start and end times for that session, use the `puresupport remoteassist list` command. If RA is not live, the output will indicate the Status as disconnected.

```
puresupport remoteassist list
Status       Opened  Expires  Access Level
disconnected  - - -
```

If the RA session is live, the output will indicate the Status as connected, and provide the times the RA session opened, when it will close, and the access level for the session.

```
puresupport remoteassist list
Status     Opened Expires Access Level
connected  2025-02-27 14:46:49 PST   2025-02-28 14:46:49 PST restricted
```

In some cases, it might be necessary to grant Pure Storage support elevated access to your array during an RA session. To change access level during the current session without changing the default value, use the `puresupport remoteassist connect --access-level-override` command.

```
puresupport remoteassist connect --access-level-override elevated
```

```
Active Status      Opened                          Expires Access Level
True     connected   2025-02-27 14:48:49 PST    2025-02-28 14:48:49 PST elevated
```

Only users with admin RBAC privileges can change the access level.

Proxy Hostnames

The `puresupport setattr --proxy` command configures the proxy hostname. The format for the proxy host name is `http(s)://hostname:port`, where `hostname` is the name of the proxy host, and `port` is the TCP/IP port number used by the proxy host. Use the `puresupport setattr --proxy https://localhost:8080` command to configure a proxy without a username and password. When a username and password are required, run the `puresupport setattr --proxy https://username@localhost:8080` command. Only the username should be specified. The CLI prompts for a password interactively

To view the current proxy settings, run the `puresupport list` command.

Displaying Support Connection Details

The `puresupport list` command displays the following support connection details:

Remote assistance session status as open (RA Active is True) or closed (RA Active is False). Include the `--connect` option to view the details of an open RA session.

- Proxy host setting
- Phonehome status as enabled or disabled

You can also display the following information about the remote assistance connection path by issuing the `puresupport list --connect --path` command:

- The name of the fabric module
- Session status for each fabric module

Diagnostics for Dark Site Arrays

The Diagnostics bundle contains logic for monitoring, alerting and mitigating issues. The bundle enables detection of known issues for dark site arrays on previously released versions of Purity. The bundle also contains the latest logic for each alert, allowing dark site arrays to have access to updated alert logic without requiring a Purity upgrade. To download the upgrade bundle, run **puresw bundle download**.

The **puresupport diagnostics list** command lists all installed diagnostics. The pure-support diagnostics update command can be used to update your installed diagnostics.The **puresupport diagnostics update** command can be used with the **--purity-default** option to revert your diagnostics to the default version.

Troubleshooting

The **puresupport test** command displays information to help Pure Storage Technical Services troubleshoot connection issues. To display the current phone home status, run the **puresupport test --phonehome** command. To display the current remote assist status, run the **puresupport test --remote assist** command.

If you are experiencing connection issues, contact a member of the Pure Storage account team or email Pure Storage Technical Services at support@purestorage.com.

Examples

Example 1

```
puresupport list
```

Displays the remote assistance (RA), proxy, phone home, and Pure1 Edge Service settings for the FlashArray.

Example 2

```
puresupport list --connect
```

Displays the connection details and status of a remote assistance (RA) session.

Example 3

```
puresupport list --connect --path
```

Displays the connection status of a remote assistance (RA) session per fabric module.

Example 4

```
puresupport connect
```

Opens a remote assistance (RA) session between the current array and Pure Storage Technical Services, giving Pure Storage Technical Services the ability to establish an administrative session.

Example 5

```
puresupport disconnect
```

Closes a remote assistance (RA) session between the current array and Pure Storage Technical Services.

Example 6

```
puresupport enable phonehome
```

Enables the automatic transmission of array logs and diagnostic information to Pure Storage Technical Services.

Example 7

```
puresupport setattr --proxy http://proxy.example.com:8080
```

Sets the proxy host for HTTPS communication between the array and Pure Storage Technical Services.

Example 8

```
puresupport test --phonehome
```

Displays the current phone home status for each fabric module.

Example 9

```
puresupport test --remoteassist
```

Displays the current remote assist status for each fabric module.

Example 10

```
puresupport enable edge-management
puresupport enable edge-agent-update
```

Enables the Pure1 Edge Service agent on the array and enables automatic updates to the Pure1 Edge Service agent.

See Also

purealert, purearray

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

puresw

puresw, puresw-app, puresw-bundle, puresw-check, puresw-download, puresw-global, puresw-install, puresw-list, puresw-patch-install, puresw-patch-list, puresw-remove, puresw-upgrade – manages software updates, Purity optimizations, and app installations and upgrades

Synopsis

```
puresw app install NAME
puresw app uninstall NAME ...
puresw app list [--cli | --csv | --nvp] [--notitle] [--raw] [--filter
FILTER] [--page] [--limit LIMIT] [--sort SORT] [--catalog] [NAME ...]
puresw bundle download URL
puresw bundle list [--csv | --nvp ]
puresw check start --version VERSION
puresw check list [--name NAME] [--version VERSION] [--csv] [--nvp]
puresw check show-detail [--name NAME] [--version VERSION]
puresw check delete --version VERSION
puresw download --version VERSION
puresw global enable [--auto-download]
puresw global disable [--auto-download]
puresw global list [--auto-download] [--cli | --csv | --nvp] [--
notitle] [--raw] [--page]
puresw install --version VERSION
puresw list [--cli | --csv | --nvp] [--catalog]
puresw patch install [--allow-ha-reduction] NAME
puresw patch list [--csv] [--nvp] [--catalog]
puresw remove --version VERSION
puresw upgrade abort STEP-ID
puresw upgrade continue [--override CHECK] [--persistent-override
CHECK] STEP-ID
puresw upgrade list [--name NAME] [--history] [--csv | --nvp]
puresw upgrade retry [--override [CHECK | CHECK,ARG...]] [--
persistent-override [CHECK | CHECK,ARG...]] STEP-ID
puresw upgrade show-plan --version VERSION
puresw upgrade show-step [--name NAME] [--csv] [--nvp]
puresw upgrade show-step-detail [--name NAME | --step-id STEP-ID][--
csv] [--nvp]
puresw upgrade start --mode {check-only, one-click, interactive |
semi-interactive} --version VERSION [--override check_name,args,..][--
persistent-override check_name,args,..][--upgrade-parameter name,
value] NAME
puresw upgrade show-upgrade-parameters [--csv] [--nvp]
```

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--allow-ha-reduction`

Allows installing a Purity optimization that requires a reduction in high availability, such as a controller failover. Used with the **`puresw patch install`** command.

`--auto-download`

When enabled, automatically downloads the latest software installation files to the array in preparation for a software update. Must be used with the **`enable`** or **`disable`** option.

Displays a list of apps that are either installed or available to be installed on the array. Newer versions of currently installed apps are also listed in the app catalog. It also displays a list of software versions that are available to be installed on the array.

`--catalog`

Lists available upgrade versions, apps, or Purity optimizations.

`--history`

Displays the array's installation and upgrade history.

`--mode`

Specifies the type of upgrade run:

`interactive`

Initiates an interactive software upgrade.

`one-click`

Initiates a one-click non-interactive software update.

Note: The argument **`one-click`** previously was **`one_click`**.

`semi-interactive`

Initiates a semi-interactive software upgrade.

`--persistent-override`

Overrides the specified upgrade check for the entire upgrade. Any errors from that upgrade check are ignored.

Can be entered as **`-p`**.

`--override`

Overrides the specified upgrade check for one upgrade step. Any errors from that upgrade check are ignored in the current or next upgrade step.

Can be entered as **`-o`**.

`--version` ***`VERSION`***

Updates the software to the specified version.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed

both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

`--upgrade-parameters`

Customizes parameters in a name and value pair format (**`name`**,**`value`**) during a Purity upgrade.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Arguments

ARG

Argument to an override option. Optional.

Only the **`HostIOCheck`** check takes an argument, which is one or more host names.

CHECK

Name of the upgrade check being overridden.

The following upgrade checks can be overridden:

- **`AppHealth`**:

  Verifies that enabled, healthy apps remain healthy through the upgrade.
- **`HostIOCheck`**:

Verifies that each host has balanced I/O to both controllers.

- **`HostPathCheck`**:

  Verifies that each host is connected to both controllers.

NAME

The name of the software package to be upgraded. For **`puresw app install`** and **`puresw patch install`**, the name of the app or Purity optimization to be installed.

STEP-ID

A unique ID for each step of an interactive **`puresw upgrade`** run.

URL

URL of the upgrade bundle.

Description

Software updates are necessary because they add or enhance Purity features and functionality. Performing periodic software updates guarantees you will get the most out of your Purity system.

We recommend you pursue software updates through the Pure1 Manage UI. The Pure1 Manage UI has all the array-level software update functionality that the **`puresw`** command has, as well as additional upgrade functionality at a higher level. For more information on Self-Service Upgrades through Pure1 Manage, please see the Pure1 Manage Software Lifecycle (Upgrades and Optimizations) article.

For environments where Pure1 access is unavailable, you can use the **`puresw`** command to manage Purity software updates and app installations from the Purity console one array at a time.

You cannot switch between **`puresw`** and self-service upgrades to upgrade an array. Whatever method you use to start an array upgrade, you should use to complete that upgrade.

Software Updates

A software update can be performed in an interactive mode, or semi-interactive mode that each pause on error to allow you to address upgrade failures before continuing the upgrade process. A one-click non-interactive mode is also available. In the one-click mode, any failure will abort the entire upgrade immediately. We strongly recommend you use the interactive or semi-interactive modes instead of the one-click mode. The one-click mode is designed for single array upgrades only. It is not a recommended option for fleet upgrades. It is designed to support high-availability environments where a fast failover is required during upgrades.

Interactive

An interactive upgrade goes through several steps and pauses at three points; after pre-upgrade checks, after mid-upgrade checks, and after post-upgrade checks. An explicit command is required to resume the upgrade, whether or not the step ran successfully. In the event of failure, the type of failure is given and further details are also available.

You can address the failure causes while the upgrade is paused and then resume the upgrade from the failed step. The upgrade restarts from the last successful step, not from the beginning.

An interactive upgrade is recommended in environments that require manual intervention after a controller reboot.

Semi-interactive

The semi-interactive mode combines the convenience of the one-click mode with the flexibility of the interactive mode. If no failures occur, the upgrade proceeds without pausing at each step. However, if a failure does occur, the upgrade pauses for you to correct the issue and then continue the upgrade.

One-click

One-click runs the update as a single step for each Purity version ("hop") required in the update. Each hop either succeeds completely or fails with no change. If a failure occurs, the upgrade is aborted and the array continues to run the last successfully installed version. The one-click update process is the same as offered on the GUI **Settings > Software** page **Updates** pane.

In the event of a failure with a one-click update, the failing hop is rolled back completely. Any successful hops are left updated. Otherwise, Purity is left unchanged.

Note: During the Purity//FA software update/upgrade process, you are logged out of the software when the controller is rebooted.

Note: Both modes support multi-hop updates/upgrades when required.

Note: OneClick is for single array upgrades, and not recommended for fleet management.

Caveats for ActiveCluster Arrays

You should NOT upgrade both arrays in an Active Cluster pair at the same time. Instead, the arrays should be upgraded to matching versions one at a time. If the upgrade requires one or more intermediate version upgrades (hops) before the final version is reached, you must upgrade BOTH arrays one at a time to the intermediate versions before continuing the upgrade on either.

To do this, follow these steps:

1. Upgrade one array until it pauses after completing the first hop.
2. Stop, go to the other array and start the identical upgrade.
3. Once it reaches the matching version of the first array, stop. Go back to the other array and continue to the next version.
4. Repeat this back-and-forth upgrade procedure until both arrays are successfully completed to the final version.

Failure to follow this procedure will result in an array outage.

Self-Service Upgrades

To begin self-service upgrades in the Purity console, use the `puresw list --catalog` command to list the upgrade versions available on the array. Choose the desired upgrade version and enter `puresw check start` command to begin pre-check. The pre-check will check for any potential health issues for the specified upgrade version. For example,

```
puresw check start Purity//FA --version 6.2.0
Name        Upgrade Path          Check Start Time     Check End Time  Status  Detail
Purity//FA 5.1.16->5.3.10->6.2.o 2021-01-29 11:34:00  -               queued  Checks
queued to run
```

Enter the `puresw check list` command to view pre-check health status once the pre-check has finished.

```
puresw check list
Name        Upgrade Path           Check Start Time     Check End Time
Purity//FA  5.1.16->5.3.10->6.2.0   2021-01-29 11:34:00  2021-01-29 11:38:00
Status    Detail
```

```
passed    All checks finished and passed
```

Enter the `puresw check show-detail` command to display the current details of the latest run.

```
puresw check show-detail --name Purity//FA --version 6.2.0
Name         Upgrade Path              Check Name         Status    Detail
Purity//FA   5.1.16->5.3.10->6.0.3     HostIOCheck        passed    details...
Purity//FA   5.1.16->5.3.10->6.0.3     InterfaceHealth    failed    why interface check
failed and how to fix it
Purity//FA   5.1.16->5.3.10->6.0.3     InternalCheck      passed    all internal checks
look good
```

If pre-check is healthy, the upgrade may continue. Use the `puresw download` command to begin software download. Note that if there is already an available upgrade created for the array, the `puresw download` command will fail. The existing upgrade in the array must be deleted first for the desired upgrade to continue. If installation is already in progress enter the `puresw upgrade abort` command to remove an existing upgrade while installation is paused. Enter the `puresw list` command to view the status.

```
puresw list
Name         Version  Status        Progress
Purity//FA   6.2.0   downloading  0.1%
```

Self-service upgrades are intended to allow ease of use for the user to upgrade their own arrays without support. However, if problems occur that cannot be resolved during the upgrade process, contact Pure Storage Technical Services.

Users can also remove downloading upgrades or a downloaded upgrade by using the `puresw remove` command. The software upgrade can be removed if the installation is downloading, downloaded, aborted, or finished. If the software package is currently downloading, the download will be canceled and any downloaded files will be removed.

Interactive Software Upgrades

The `puresw upgrade` commands manage interactive upgrades. Use the `puresw list` command to display the list of available software versions.

Enter the `puresw upgrade start` command to initiate an interactive upgrade. Use the `--mode check-only` option to perform pre-upgrade checks without actually performing the upgrade. This option downloads the required software installation files before running the checks.

Use the `--mode one-click` option to run a one-click update (this option is equivalent to the `puresw install` command and to the update process offered on the GUI **Settings > Software** page **Updates** pane). Use the `--mode interactive` option to run an interactive upgrade.

Use the `--mode semi-interactive` option to run a semi-interactive upgrade. A semi-interactive upgrade pauses only on error, allowing you to correct the reason for the failure. If there are no errors, the upgrade runs to completion without pausing, as a one-click upgrade does.

Customize upgrade parameters by using the `--upgrade-parameter` option. Specify the parameters in a name and value pair format, with a character limit of 1024 characters. Available parameters will be closely associated with specific Purity versions and features. A specific Knowledge Base article will be provided whenever new parameters are introduced. Do not set any parameters without consulting the article.

Enter the `puresw upgrade list` command to show the status of upgrades in progress. If no in-progress upgrades are found, this command shows information about the most recent interactive upgrade.

For upgrades currently in progress, the information displayed includes the step ID, status, and percentage of steps completed. If the upgrade is paused, the output includes details about the last step and instructions on your next action for the upgrade.

An interactive upgrade has one of the following statuses, as reported by `puresw upgrade list`:

- **downloading**: The upgrade is downloading the required software files.
- **installing**: The software is being upgraded.
- **paused**: The upgrade has completed a step and is waiting for user action.
- **finished**: The upgrade has completed successfully.
- **aborted**: The upgrade was canceled.

The progress reported by `puresw upgrade list` is based on the number of completed steps compared to the overall number of steps.

Use the `puresw upgrade show-upgrade-parameters` command to display the relevant upgrade parameters associated with the most recent active or installed upgrade.

The `puresw upgrade show-plan` command lists the steps that are typically performed to upgrade to the specified software and version but does not perform the upgrade. The Version column shows the progression from the current software version through the final upgrade version.

The `puresw upgrade show-step` command lists any running upgrades with their current step. If no upgrades are running currently, shows the steps of the previous upgrade. The progress reported by `puresw upgrade show-step` is based on the number of completed checks compared to the overall number of checks.

The `puresw upgrade show-step-detail` command provides more detail on the previous step or on the specified step. At any pause involving a failed check, the `puresw upgrade show-step-detail` command provides information on issues that need to be resolved before resuming the upgrade.

The `puresw upgrade continue` command resumes a paused upgrade after a successful step. The `puresw upgrade retry` command retries a failed step.

Enter the `puresw upgrade abort` command to cancel an interactive upgrade. Canceling an interactive upgrade has this effect:

- For a single hop upgrade, canceling leaves the array unchanged.
- For a multi-hop upgrade in which the first hop fails, canceling leaves the array unchanged.
- For a multi-hop upgrade in which at least one hop has completed successfully, canceling leaves the array at the Purity version of the last successful hop.

Because an array with a partial upgrade might not successfully complete a failover, an interactive upgrade should not be left in a paused state any longer than is required. Instead, the upgrade should be canceled.

The continue, retry, and abort subcommands require the step ID reported by the previous step of the upgrade. An incorrect step ID causes the command to fail. The current step ID is also reported by the `puresw upgrade show-step-detail` and `puresw upgrade list` commands.

Upgrades may issue a "Broken pipe" notice when a controller is rebooted. "Broken pipe" is not an error message and can safely be ignored.

Interactive upgrades are intended to provide the information required for a successful upgrade. However, if you encounter problems that you cannot resolve during the upgrade process, contact Pure Storage Technical Services.Interactive Software Upgrades

The `puresw upgrade` commands manage interactive upgrades. As with a one-click upgrade, use the `puresw list` command to display the list of available software versions.

Enter the `puresw upgrade start` command to initiate an interactive upgrade. Use the **`--mode check-only`** option to perform pre-upgrade checks without actually performing the upgrade. This option downloads the required software installation files before running the checks.

Use the `--mode one-click` option to run a one-click update (this option is equivalent to the `puresw install` command and to the update process offered on the GUI **Settings > Software** page **Updates** pane). Use the `--mode interactive` option to run an interactive upgrade.

Use the `--mode semi-interactive` option to run a semi-interactive upgrade. A semi-interactive upgrade pauses only on error, allowing you to correct the reason for the failure. If there are no errors, the upgrade runs to completion without pausing, as a one-click upgrade does.

Enter the `puresw upgrade list` command to show the status of upgrades in progress. If no in-progress upgrades are found, this command shows information about the most recent interactive upgrade.

For upgrades currently in progress, the information displayed includes the upgrade's step ID, status, and percentage of steps completed. If the upgrade is paused, the output includes details about the last step and instructions on your next action for the upgrade.

An interactive upgrade has one of the following statuses, as reported by `puresw upgrade list`:

- **downloading**: The upgrade is downloading the required software files.
- **installing**: The software is being upgraded.
- **paused**: The upgrade has completed a step and is waiting for user action.
- **finished**: The upgrade has completed successfully.
- **aborted**: The upgrade was canceled.

The progress reported by `puresw upgrade list` is based on the number of completed steps compared to the overall number of steps.

Use the `puresw upgrade show-upgrade-parameters` command to display the relevant upgrade parameters associated with the most recent active or installed upgrade.

The `puresw upgrade show-plan` command lists the steps that are typically performed to upgrade to the specified software and version but does not perform the upgrade. The Version column shows the progression from the current software version through the final upgrade version.

The `puresw upgrade show-step` command lists any running upgrades with their current step. If no upgrades are running currently, it shows the steps of the previous upgrade. The progress reported by `puresw upgrade show-step` is based on the number of completed checks compared to the overall number of checks.

The `puresw upgrade show-step-detail` command provides more detail on the previous step or on the specified step. At any pause involving a failed check, the **`puresw upgrade show-step-detail`** command provides information on issues that need to be resolved before resuming the upgrade.

The `puresw upgrade continue` command resumes a paused upgrade after a successful step. The `puresw upgrade retry` command retries a failed step.

Enter the `puresw upgrade abort` command to cancel an interactive upgrade. Canceling an interactive upgrade has this effect:

- For a single hop upgrade, canceling leaves the array unchanged.
- For a multi-hop upgrade in which the first hop fails, canceling leaves the array unchanged.
- For a multi-hop upgrade in which at least one hop has completed successfully, canceling leaves the array at the Purity version of the last successful hop.

Because an array with a partial upgrade might not successfully complete a failover, an interactive upgrade should not be left in a paused state any longer than is required. Instead, the upgrade should be canceled.

The continue, retry, and abort subcommands require the step ID reported by the previous step of the upgrade. An incorrect step ID causes the command to fail. The current step ID is also reported by the `puresw upgrade show-step-detail` and `puresw upgrade list` commands.

Upgrades may issue a "Broken pipe" notice when a controller is rebooted. "Broken pipe" is not an error message and can safely be ignored.

Interactive upgrades are intended to provide the information required for a successful upgrade. However, if you encounter problems that you cannot resolve during the upgrade process, contact Pure Storage Technical Services.

One-click Software Updates

We strongly recommend you use the interactive or semi-interactive modes instead of the one-click mode.

The `puresw global` command manages the configuration setting across all one-click software updates. The `puresw global enable --auto-download` command enables Auto Download. When Auto Download is enabled, any software installation files that Pure Storage Technical Services send to the array are automatically downloaded and ready to install. The `puresw global disable --auto-download` command disables Auto Download. When

Auto Download is disabled, the software installation files that Pure Storage Technical Services send to the array are only downloaded during the software update process. You cannot change the Auto Download status while a software update is in progress. Note that the Auto Download feature impacts software updates only. The Auto Download feature does not impact Purity apps. Auto Download is disabled by default.

The `puresw global list` command displays the current global configuration that affects all software updates. Include the `--auto-download` option to display the global configuration for the Auto Download feature.

In the following example, the `puresw global list --auto-download` command displays the global Auto Download status for software updates. Auto Download is set to enabled, which means that any software installation files that Pure Storage Technical Services send to the array will be automatically downloaded and ready to install. Purity apps are not affected by global settings, so as Purity apps become available, those installation files are not automatically downloaded to the array; instead, they are downloaded to the array during the app installation process.

```
$ puresw global list
Auto Download

enabled
```

Enter the `puresw list` command to display a list of software versions available for update and the installation status of each software version. A software version that is available for update will have one of the following statuses:

- **available**: A software update for this version is available, but the installation files have not been downloaded to the array. Instead, the files will be downloaded to the array during the installation process. When scheduling the software update, make sure to factor enough time for the download process.
- **downloaded**: The installation files for this software version have been successfully downloaded to the array.

Enter the `puresw install` command to run the software update. Include the required `--version` option to specify the software version to be updated. As the software update process progresses, the following statuses will appear:

- **downloading**: Purity is downloading the installation files to the array for this software version. If Auto Download is enabled, any software installation files that Pure Storage Technical Services send to the array will be automatically downloaded and ready to install. If Auto Download is disabled, the software installation files that Pure Storage Technical Services send to the array will only be downloaded during the software

update process.
- **installing**: Purity is updating the software. Be prepared to be logged out of the software during the update process.

Note: The `puresw install` command will be deprecated in a future release. Use the `puresw upgrade start --mode one-click` command instead.

During the update process, you will be logged out of the software. After you have been logged out, log back in to continue monitoring the process. The software update process is complete when the software update no longer appears in the `puresw list` output.

If the software update fails to fully complete, the system reverts to the previous software version, ready to resume operations. If the update path spans multiple versions, the system stops the installation process at the most current successfully installed, highly-available software version. If you encounter any problems during the update process, contact Pure Storage Technical Services.

App Installations

The Purity Run platform extends array functionality by integrating add-on services into the Purity//FA operating system. Each service that runs on the platform is provided by an app.

The `puresw app` command manages the app installation process. Apps require CPU, memory, network, and storage resources. For this reason, apps are not installed by default.

Enter the `puresw app list` command to display a list of apps that are currently installed on the array, along with the following attributes for each app:

- **Name**: App name. The app name is pre-assigned and cannot be changed.
- **Version**: App version that is currently installed on the array.
- **Status**: Status of the app installation. Possible app statuses include:
  - **Available**: App (new or updated version of an existing one) is available to be installed.
  - **Downloading**: App installation files are being downloaded in preparation for an installation.
  - **Downloaded**: App installation files have been successfully downloaded. Installation will begin shortly.
  - **Installing**: App is currently being installed.
  - **Installed**: App has been successfully installed.
  - **Uninstalling**: App is currently being uninstalled.

  - **Aborted**: App installation process has encountered issues. The installation is rolling back. If the app is no longer available, it will not reappear in the list; otherwise, it will eventually return to its previous "Available" status. If the app is available, try the installation again. If you continue to encounter issues, contact Pure Storage Technical Services.
- **Progress**: Download progress during the installation process.
- **Description**: Description of the app.

The `puresw app list --catalog` command displays a list of apps, both available to and installed on the array. If a newer version of a currently installed app is available, it will also be listed in the app catalog.

The `puresw app install` command installs a new app or updates a currently installed app. An app can be installed at any time with no disruption to array activity, though the installation of larger, more intensive apps may impact performance.

The app installation process is three-fold: first, the latest installation files are downloaded to the array in preparation for the app installation, second, the app is installed, and third, the app is enabled. The app installation process is complete when the app is enabled.

Once the app installation process is complete, enter the pureapp list command to verify its status. The app is ready to service requests when it is in a healthy status. If the app is in an unhealthy status, refer to pureapp to perform additional steps to bring the app to a healthy status.

After an app has been installed, enter the `pureapp` command to manage the app. For more information about apps, refer to pureapp.

When an app is no longer needed, enter the `puresw app uninstall` command to uninstall the app. An app must be disabled before it can be uninstalled. Enter the `pureapp disable` command to disable an app. After an app has been uninstalled, the volumes connected to the app host and the data volume that was created during app installation remain on the array, but they can no longer reach the app service. The boot volume that was created during app installation is deleted.

Purity Optimizations

The `puresw patch` command lists, downloads, and installs Purity optimizations on the array without assistance from Pure Storage Technical Services. Purity optimizations, which include remediation packages and security patches, are provided to resolve issues that occasionally arise and affect your systems. Self-service management of Purity optimizations requires explicit opt-in and also requires setup steps, including access to Pure1 Manage, installation of the Purity//FA Optimization Agent on the array, and other steps described in the Knowledge Base article Pure1 Manage - Software Lifecycle. Available Purity optimizations are listed by the **puresw patch list** command (described below) and are also displayed in the Pure1 Manage UI, in the Dashboard > Software Lifecycle > Purity Optimizations tab.

The rest of this section describes using the **puresw patch** command for self-service installation of Purity optimizations.

Run **puresw patch list --catalog** to see any available Purity optimizations to install. The following example shows optimization `fa-22-01-01`, which requires a controller failover or other reduction in high availability. Optimizations that do not require a reduction in high availability are listed with '`False`' in the HA Reduction Required column. Alert codes are described in the Knowledge Base article Self-Help Alerts. (Optimization names and descriptions are artificially shortened in these examples.)

```
puresw patch list --catalog
Patch Name    Alert Code   Description          HA Reduction Required
fa-22-01-01   -            Cumulative patch     True
```

Run **puresw patch install *NAME*** to download and install an available Purity optimization. Add the **--allow-ha-reduction** option for an optimization that requires a reduction in high availability, such as a controller failover, as shown in the next example. An optimization that requires reduction in high availability cannot be installed if the **--allow-ha-reduction** option is not present.

```
puresw patch install --allow-ha-reduction fa-22-01-01
Patch Name  Description  Status       Progress HA Reduction Req'd  Details
fa-22-01-01 Cumulative   downloading  0.00%    True                Downloading security
```

Run **puresw patch list** to list any installed or installing Purity optimizations.

```
puresw patch list
Patch Name  Description  Status      Progress HA Reduction Req'd  Details
fa-22-01-01 Cumulative   Installing  80.00%   True                Installing security
```

Those Purity optimizations that are security patches are cumulative. New security patches include the content of older security patches.

Exceptions

Running the **`puresw app list --catalog`** command always returns null on Cloud Block Store.

Examples

Example 1

```
puresw list
```

Displays a list of software versions that are either running on the array or available for update.

Example 2

```
puresw global disable --auto-download
```

Disables Auto Download. Any software installation files that Pure Storage Technical Services send to the array will only be downloaded during the software update process.

Example 3

```
puresw check --version 6.2.0 Purity
```

Performs a series of checks to ensure the array is healthy and prepared for the software update to Purity 6.2.0.

Example 4

```
puresw upgrade start --mode one-click --version 6.2.0 Purity//FA
```

Updates the Purity software to version 6.2.0.

Example 5

```
puresw list
Name Version Status Progress
Purity//DFS 1.1.2 available -


puresw install Purity//DFS --version 1.1.2
Name Status
```

```
Purity//DFS installing


puresw list
Name Version Status Progress
Purity//DFS 1.1.2 downloading 10.96%
```

Shows DirectFlash Shelf (DFS) software available for update, its update command, and the DFS update in progress.

Example 6

```
puresw app list
```

Displays a list of installed apps and the installation status of each app.

Example 7

```
puresw app list --catalog
```

Displays a list of apps, both available to and installed on the array.

Example 8

```
puresw app install offload
```

Installs the offload app.

Example 9

```
puresw app list --catalog
puresw app install linux
puresw app list
```

Displays a list of apps that are available to be downloaded and installed. Downloads, installs, and enables the linux app. Displays the progress of the installation process.

Example 10

```
pureapp disable linux
puresw app uninstall linux
pureapp list
```

Disables the linux app in preparation for the uninstallation. Uninstalls the linux app. Displays a list of installed apps to verify that the linux app has been successfully uninstalled and no longer appears in the list.

Example 11

The next several examples show an example interactive upgrade run. In the output, times and step IDs have been shortened for ease of reading. Also, long output lines are displayed in two parts.

Begin by checking for an available software release. This output shows that the array can be upgraded to Purity//FA 6.2.0.

```
puresw list
Name Version Status Progress
Purity//FA 6.2.0 available -
```

Confirm that an upgrade is not already in progress. An empty response indicates an upgrade is not in progress and has never been run.

```
puresw upgrade list
```

Example 12

Optionally enter the **puresw upgrade show-plan** command to preview the list of steps involved in the upgrade.

```
puresw upgrade show-plan Purity//FA --version 6.2.0
Planned Step Name                  Version   Description
pre-upgrade-check                  6.0.3     Run the pre-upgrade checks which will
verify if the array is ready to upgrade.
upgrade secondary controller       6.0.3     Upgrade the secondary controller to the
new version. Will perform a reboot on the secondary controller.
mid-upgrade-check                  6.0.3     Run the mid-upgrade checks which will
verify if the current primary can upgrade to the new version.
upgrade new secondary controller   6.2.0     Upgrade the new secondary controller to
the new version. Will perform a reboot on the new secondary controller.
post-upgrade-check                 6.2.0     Run the post-upgrade checks which will
verify if the array performed the upgrade successfully.
```

Example 13

Start the upgrade, using the version information from **`puresw list`**.

```
puresw upgrade start Purity//FA --version 6.1.0 --mode interactive
Software Name  Plan  Mode           Start Time   End Time     Current Step ID
Purity//FA     6.1.0 interactive    10:21:06 PST -            -
Status
downloading
```

Example 14

```
puresw upgrade list
Software Name  Plan   Mode         Start Time    End Time   Current Step ID
Purity//FA     6.1.0  interactive 10:21:06 PST  -          9024d7c9-41d2
Status Progress  Details
paused 4.00%     Upgrade has been paused since 10:30:16. Finished pre-upgrade check,
                 41 checks in total, all passed.
```

Checks the status of an interactive upgrade. In this case, an interactive upgrade passed the previous step and is paused waiting for user input.

Example 15

```
puresw upgrade continue --step-id 3f2c9f02-190o
Software Name  Plan   Mode         Start Time    End Time    Current Step ID
Purity//FA     6.2.0  interactive 10:21:06 PST   -           3f2c9f02-41d2
Status        Progress     Details
installing    4.00%        Continued installation
```

Resumes an interactive upgrade after a successful step. The step ID of the paused step is required.

Example 16

```
puresw upgrade list
Software Name Plan   Mode         Start Time   End Time   Current Step ID
Purity//FA    6.2.0  interactive 10:21:06 PST -          9024d7c9-41d2
```

```
Status        Progress     Details
paused        4.00%        Upgrade has been paused since 10:30:16. Finished pre-upgrade
                            check, 41 checks in total, 1 failed (Timezones).
Run command "puresw upgrade show-step-detail" for more information.
To proceed with the upgrade, either:
1) Retry checks - Run "puresw upgrade retry 9024d7c9-41d2"
2) Abort the upgrade - Run "puresw upgrade abort 9024d7c9-41d2"
The secondary controller will be upgraded to 6.1.0 after continuing the upgrade.
```

Shows an interactive upgrade that did not successfully pass all checks.

Example 17

```
puresw upgrade show-step-detail
Software Name  Step Name          Step ID       Check Name  Check Status
Purity//FA     pre-upgrade check  9024d7c9-41d2 Timezones   failed
Check Details
Controller timezones don't match
                   | CT0              | CT1
------------------------------------------------------
cat /etc/timezone | America/Chicago | America/Denver
date +%z+%z       | -0700+MST       | -0700+MST
Recommended Action: Make the timezones match on both controllers
```

Displays more details about the failure of an interactive upgrade step, including a recommendation to resolve the error.

Example 18

```
puresw upgrade retry --step-id 9024d7c9-41d2
Software Name  Plan   Mode         Start Time   End Time   Current Step ID
Purity//FA     6.2.0  interactive  10:21:06 PST -         9024d7c9-41d2
Status         Progress    Details
installing     4.00%       Retrying last step
```

Resumes an interactive upgrade after a failed step, using the step ID of that step. To be used after resolving the issues causing the check failure.

Example 19

```
puresw upgrade show-step
```

Lists the current step of interactive upgrades in progress.

Example 20

```
puresw upgrade abort --step-id 775hlhb2-172b
Software Name  Plan   Mode          Start Time   End Time    Current Step ID
Purity//FA     6.1.0  interactive   10:21:06 PST -           775hlhb2-172b
Status   Progress    Details
aborted  27.00%      Aborting installation
```

Cancels an interactive upgrade. Requires the step ID of the current upgrade step.

Example 21

```
puresw upgrade retry --step-id 775hlhb2-172b --override HostIOCheck,host-1,host-2
```

Resumes an interactive upgrade after a failed step and specifies that any I/O balance errors related to host host-1 or host-2 during the current step are ignored.

Example 22

```
puresw upgrade start Purity//FA --version 6.1.0 --mode interactive \
       --persistent-override HostIOCheck
```

Starts an interactive upgrade and specifies that all I/O balance errors are ignored during the entire upgrade.

Example 23

```
puresw upgrade start Purity//FA --version 6.1.0 --mode semi-interactive
```

Starts an upgrade in semi-interactive mode.

Example 24

```
puresw bundle download https://local/upgrade_bundle.sh
```

Downloads the upgrade bundle from the local server https://local to the array.

Example 25

```
puresw bundle list
```

Displays the progress of the bundle upgrade download to the FlashArray.

Example 26

```
puresw remove Purity//FA --version 6.1.0
```

Removes an in-progress download or a downloaded software upgrade.

Example 27

```
puresw patch list --catalog
```

Lists Purity optimizations, if any, that are available to be installed on the array.

Example 28

```
puresw patch install fa-2022-01-02
```

Installs the specified Purity optimization.

Example 29

```
puresw patch install --allow-ha-reduction fa-2022-01-01
```

Installs the specified Purity optimization, for an optimization that requires a controller failover or other reduction in high availability.

Example 30

```
puresw patch list
```

Lists any installed or installing Purity optimizations.

Example 31

```
puresw upgrade start Purity//FA --version 6.8.0 --mode interactive
      --upgrade-parameter test-parameter1,test-value1
      --upgrade-parameter test-parameter2,test-value2
```

Specifies custom parameters **`test-parameter1`** and **`test-parameter2`** to be set when upgrading to Purity version 6.8.0.

See Also

pureapp

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purevchost

purevchost, purevchost-connection, purevchost-create, purevchost-list, purevchost-delete, purevchost-certificate, purevchost-endpoint – manages virtual hosts for communication between VMWare vCenters and VMware APIs for Storage Awareness (VASA) providers.

Synopsis

purevchost certificate list [--remote]

purevchost certificate add ***CERTIFICATE*** [--vchost ***VCHOST***] [--endpoint ***ENDPOINT***]

purevchost certificate remove ***CERTIFICATE*** [--vchost ***VCHOST***]

purevchost connection create [--allow-stretched-multi-vchost] --protocol-endpoint ***PROTOCOL_ENDPOINT VCHOST***

purevchost connection list [***VCHOST***]

purevchost connection delete --protocol-endpoint ***PROTOCOL_ENDPOINT VCHOST***

purevchost create [--vcuuid ***VCENTER-UUID***] ***VCHOST***

purevchost delete ***VCHOST***

purevchost endpoint add ***ENDPOINT*** [--certificate ***CERTIFICATE***] [--vchost ***VCHOST***]

purevchost endpoint list [--remote]

purevchost endpoint remove ***ENDPOINT***[--vchost ***VCHOST***]

purevchost endpoint setattr [--certificate ***CERTIFICATE***] [--vchost ***VCHOST***] ***ENDPOINT***

purevchost list

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--allow-stretched-multi-vchost`

This option makes it possible to add a connection to a stretched container that already has a private connection. In principle, a stretched container can only see one vCenter at a time. This option can be useful if you need to add a second connection before you can remove the first one.

`--certificate`

Specifies the certificate that will be used on an endpoint for a vchost

`--endpoint`

Specifies the IP address endpoint on which a certificate should be used.

`--remote`

Includes vchost certificates and endpoints from remote arrays.

`--vchost`

Specifies the vCenter virtual host for which a certificate should be used.

`--vcuuid`

Specifies the vCenter server unique identifier (UUID).

Arguments

VCENTER-UUID

The uuid of the vCenter server. The uuid is used in the SNI (Server Name Identification) header in requests that are made by the vCenter.

VCHOST

The name of the virtual host that facilitates communication between your VMware vCenter and vSphere API for Storage Awareness (VASA) provider.

CERTIFICATE

The certificate that authenticates clients in VASA.

ENDPOINT

The IP address of the controller where a specific certificate should be sent (e.g. 10.20.53.108)

PROTOCOL-ENDPOINT

The name of the protocol endpoint (e.g. pod1::pe1).

Description

Using the **`purevchost`** command, as an array admin, you can manage virtual hosts for communication between vCenters and VASA providers. You can pre-create a vCenter virtual host (vchost) with the certificates that you intend to use for communication between vCenter and a VASA provider. You can see which certificates are used on which endpoints and for which vCenter. You can assign new certificates to vCenters and endpoints (controllers).

Each vCenter server has an ID (uuid). The uuid is used in the SNI header (Server Name Identification) in requests that are made by the vCenter. The SNI is needed by Purity to serve the corresponding certificate to the vCenter client when requests are made. Each connection between a vCenter and a VASA provider is mediated by a vCenter virtual host (vchost). A vchost can either be pre-created by a storage admin or it gets created automatically during the VASA provider registration workflow. Certificates can be associated to a virtual hosts prior to registration if the usage of a custom certificate is desired. Otherwise the certificate will be generated automatically during the registration workflow.

Automatically Created vchosts

For vchosts that were automatically created through the VASA provider registration workflow, you can list the vchosts and the associated vcenter user IDs (uuids) by running **`purevchost list`**. You can list the certificates, vchosts, and endpoints by running **`purevchost certificate list`**. If you register a second controller of the same FlashArray as a VASA provider on the same vCenter, **`purevchost certificate list`** will show an additional certificate at another endpoint. The two listed certificates will also appear in the **`purecert`** list.

You can run the **`purevchost certificate list --remote`** command to include in the output certificates on another connected array connected by a stretched pod.

Pre-created vchosts

If the usage of a custom certificate is desired, a vchost should be created prior to registration. Run `purevchost create --vcuuid` to create the vchost. Initially, vchosts are not associated with any certificates. If you want to use your own custom certificate for communication between vCenter and a VASA provider, you must create that certificate using `purecert` and then link that certificate to the relevant vchost. Link the certificate to the host with `purevchost certificate add` or the `purevchost endpoint add`. With the pre-created vchost and certificate linkages, a new certificate will not be generated during the VASA provider registration workflow. Instead the pre-created certificates will be exposed on the respective controller.

Certificate Rotation

To use a specific certificate on a specific controller with a specific endpoint, use `purevchost endpoint setattr`. If you associate a new certificate with an endpoint that already has an old certificate associated to it, the old certificate will continue to show up when you run `purevchost certificate list`, but it won't any longer be associated with an endpoint.

vchosts can only be deleted when there are no certificate linkages assigned to them. To remove certificate linkages, run `purevchost certificate remove`.

Connecting vchosts and Protocol Endpoints

Before vSphere can access your pod, you must create a connection between the vchost and the protocol endpoint in the pod. You can do this using the `purevchost connection create` command. For example, the command below creates a connection between a protocol endpoint (pe1, located in pod1) and a vchost (vchost1).

```
purevchost connection create --protocol-endpoint pod1::pe1 vchost1
Protocol-endpoint Vchost
pod1::pe1 vchost1
```

You can create a public connection by using * in place of the vchost name.

```
purevchost connection create --protocol-endpoint pod2::pe2 *
Protocol-endpoint Vchost
pod2::pe2 *
```

Multiple vchosts can connect to the same protocol endpoint, and multiple protocol endpoints can connect to the same vchost. However, if a pod is stretched, the default behavior is that it can only have one connection unless otherwise specified using the **`--allow-stretched-multi-vchost`** option. This option may be useful in situations where a stretched pod has a private connection, and another connection must be made, perhaps before deleting the private connection. This option only works for a protocol endpoint in a stretched pod. Non-stretched pods can already have multiple connections by default.

```
purevchost connection create --allow-stretched-multi-vchost --protocol-endpoint
pod3::stretchedPE3 vchost2
Protocol-endpoint Vchost
pod3::stretchedPE3 vchost2
```

You can list existing connections with the **`purevchost connection list`** command. You can specify a vchost to filter the list that displays to only connections to that vchost.

You can delete existing connections with the **`purevchost connection delete`** command.

Each of these **`purevchost connection`** commands has a corresponding **`purevol vc-connection`** command.

Examples

Example 1

```
purevchost list
```

Shows the vchost and vCenter uuid on the array.

Example 2

```
purevchost certificate list
```

Shows the name of the certificates as well as the vchosts and endpoints associated with each certificate.

Example 3

```
purevchost create --vcuuid vCenterID vchostname
```

Creates a vchost called **`vchostname`** that links to a vCenter with the **`vCenterID`** uuid.

Example 4

```
purevchost certificate add certID --vchost vchostname --endpoint endpointIP
```

Creates and adds certificate **`certID`** to the vchost **`vchostname`** at the endpoint **`endpointIP`**.

Example 5

```
purevchost endpoint setattr --certificate certID --vchost vchostname endpointIP
```

Moves the certificate **`certID`** to the vchost **`vchostname`** at the endpoint **`endpointIP`**.

Example 6

```
purevchost delete vchostname
```

Deletes vchost **`vchostname`**.

Example 7

```
purevchost certificate remove certID --vchost vchostname
```

Removes certificate **`certID`** from vchost **`vchostname`**.

Example 8

```
purevchost endpoint remove endpointIP --vchost vchostname
```

Removes the endpoint **`endpointIP`** from the vchost **`vchostname`**.

Example 9

```
purevchost certificate list --remote
```

Lists certificates associated with vchosts for both the local array, and the remote array to which a pod has been stretched.

Example 10

```
purevchost connection create --protocol-endpoint pod1::pe1 vchost1
```

Creates a connection between vchost **`vchost1`** and protocol endpoint **`pe1`**, which is located in **`pod1`**.

Example 11

```
purevchost connection create --allow-stretched-multi-vchost --protocol-endpoint
pod2::pe2 vchost2
```

Creates a connection between vchost **`vchost2`** and protocol endpoint **`pe2`**, which is located in **`pod2`**. This endpoint is created even though a private connection already exists.

Example 12

```
purevchost connection list vchost1
```

Lists all protocol endpoint connections by vchost1.

Example 13

```
purevchost connection delete --protocol-endpoint pod1::pe1 vchost1
```

Deletes the connection between vchost **`vchost1`** and protocol endpoint **`pe1`**, which is located in **`pod1`**.

purevgroup

purevgroup, purevgroup-create, purevgroup-destroy, purevgroup-eradicate, purevgroup-list, purevgroup-listobj, purevgroup-monitor, purevgroup-recover, purevgroup-rename, purevgroup-setattr – manages the creation, naming, and destruction of Purity//FA volume groups.

Synopsis

purevgroup create [--bw-limit ***BANDWIDTH-LIMIT***] [--iops-limit ***IOPS-LIMIT***] [--priority-adjustment ***ADJUSTMENT***] [--context ***REMOTE***] ***VGROUP...***

purevgroup destroy [--destroy-contents] [--context ***REMOTE***] ***VGROUP...***

purevgroup eradicate [--eradicate-contents] [--context ***REMOTE***] ***VGROUP...***

purevgroup list [--historical {1h | 3h | 24h | 7d | 30d | 90d | 1y}] [--pending | --pending-only] [--qos] [--space] [--total] [--cli | --csv | --nvp] [--notitle] [--page] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [--workload ***WORKLOAD***] [--context ***REMOTES***] [***VGROUP...***]

purevgroup listobj [--pending | --pending-only] [--type {vgroup | vol}] [--csv] [***VGROUP...***]

purevgroup monitor [--interval ***SECONDS***] [--latency] [--mirrored] [--qos] [--repeat ***REPEAT-COUNT***] [--size] [--total] [--historical {1h | 3h | 24h | 7d | 30d | 90d | 1y}] [--csv] [--notitle] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--page] [--sort ***SORT***] [***VGROUP...***]

purevgroup recover [--context ***REMOTE***] ***VGROUP...***

purevgroup rename [--context ***REMOTE***] ***OLD-NAME*** ***NEW-NAME***

purevgroup setattr [--bw-limit ***BANDWIDTH-LIMIT***] [--iops-limit ***IOPS-LIMIT***] [--priority-adjustment ***ADJUSTMENT***] [--context ***REMOTE***] ***VGROUP...***

Arguments

`NEW-NAME`
: Name by which the volume or volume snapshot is to be known after the command executes.

`OLD-NAME`
: Current name of the volume or volume snapshot to be renamed.

`VGROUP`
: Volume group to be created, renamed, destroyed, eradicated, recovered, viewed, or monitored.

Object Names

Valid characters are letters (A-Z and a-z), digits (0-9), and the hyphen (-) character. The first and last characters of the name must be alphanumeric, and the name must contain at least one letter or '-'.

Most objects in Purity//FA that can be named, including host groups, hosts, volumes, protection groups, volume and protection group suffixes, SNMP managers, and subnets, can be 1-63 characters in length.

Array names can be 1-56 characters in length. The array name length is limited to 56 characters so that the names of the individual controllers, which are assigned by Purity//FA based on the array name, do not exceed the maximum allowed by DNS.

Names are case-insensitive on input. For example, **`vol1`**, **`Vol1`**, and **`VOL1`** all represent the same volume. Purity//FA displays names in the case in which they were specified when created or renamed.

All objects in a pod have a fully qualified name that include the pod name and object name. The fully qualified name of a volume in a pod is **`POD::VOLUME`**, with double colons (**`::`**) separating the pod name and volume name. The fully qualified name of a protection group in a pod is **`POD::PGROUP`**, with double colons (**`::`**) separating the pod name and protection group name. For example, the fully qualified name of a volume named **`vol01`** in a pod named **`pod01`** is **`pod01::vol01`**, and the fully qualified name of a protection group named **`pgroup01`** in a pod named **`pod01`** is **`pod01::pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target array, the fully qualified name of the protection group on the target array is **POD:PGROUP**, with single colons (**:**) separating the pod name and protection group name. For example, if protection group **pod01::pgroup01** on source array **array01** asynchronously replicates data to target array **array02**, the fully qualified name of the protection group on target array **array02** is **pod01:pgroup01**.

If a protection group in a pod is configured to asynchronously replicate data to a target pod, the fully qualified name of the protection group on the target pod is **POD:PGROUP**, with single colons (**:**) separating the pod name and protection group name. For example, if protection group **pod01::pgroup01** on source array **array01** asynchronously replicates data to a target pod **pod02**, the fully qualified name of the protection group on target array **array02** is **pod02::pod01:pgroup01**.

All objects in a volume group have a fully qualified name that includes the volume group name and the object name, separated by a forward slash (**/**). For example, the fully qualified name of a volume named **vol01** in a volume group named **vgroup01** is **vgroup01/vol01**.

Realms, pods, and volume groups provide a namespace with unique naming conventions. All objects in a realm have a fully qualified name that include the realm name and object name or realm, pod, and object names. The fully qualified name of a pod in a realm is **REALM::POD**, with double colons (**::**) separating the realm name and pod name. The fully qualified name of a volume in a pod in a realm is **REALM::POD::VOLUME**, with double colons (**::**) separating the realm, pod, and volume names. The fully qualified name of a protection group in a pod in a realm is **REALM::POD::PGROUP**, with double colons (**::**) separating the realm, pod, and protection group names. The fully qualified name of an object in a realm but not in a pod is **REALM::HOST**, using host as an example.

Volume Sizes

Volume sizes are specified as an integer, followed by one of the suffix letters **K**, **M**, **G**, **T**, **P**, representing KiB, MiB, GiB, TiB, and PiB, respectively, where "Ki" denotes 2^10, "Mi" denotes 2^20, and so on.

Volumes must be between one megabyte and four petabytes in size. If a volume size of less than one megabyte is specified, Purity//FA adjusts the volume size to one megabyte. If a volume size of more than four petabytes is specified, the Purity//FA command fails.

Volume sizes cannot contain digit separators. For example, **1000g** is valid, but **1,000g** is not.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--historical` ***TIME***

When used with **`purevgroup list --space`**, displays historical size and space consumption information over the specified range of time. Valid time ranges include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year.

When used with **`purevgroup monitor`**, displays historical performance data over the specified range of time. Valid time ranges include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year.

`--bw-limit` ***BANDWIDTH-LIMIT***

Sets the maximum QoS bandwidth limit for the volume groups. Whenever throughput exceeds the bandwidth limit, throttling occurs. If set, the bandwidth limit must be between `1` MB/s and `512` GB/s. The bandwidth limit is specified as an integer, optionally followed by the suffix letter `M` (for megabytes) or `G` (gigabytes).

`--context` ***REMOTE***

Used to specify the remote array on which a command is processed. ***REMOTE*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

If used with **`purevgroup list`**, `--context` can specify multiple arrays and ***REMOTES*** is a comma-separated list of array names.

`--destroy-contents`

Allows the **`purevgroup destroy`** command to destroy volume group contents.

`--eradicate-contents`

Allows the **`purevgroup eradicate`** command to eradicate volume group contents.

`--interval` ***SECONDS***

Sets the number of seconds between displays of real-time performance data. At each interval, the system displays a point-in-time snapshot of the performance data. If omitted, the interval defaults to every 5 seconds.

`--iops-limit` ***IOPS-LIMIT***

Sets the maximum IOPS limit for the volume groups. Whenever the number of I/O operations per second exceeds the IOPS limit, throttling occurs. If set, the IOPS limit must be between 100 and 100`M`. The IOPS limit is specified as an integer, optionally followed by the suffix `K(10^3)` or `M(10^6)`.

`--latency`

Displays real-time and historical I/O latency information.

`--mirrored`

In an ActiveCluster replication configuration, includes performance data for mirrored writes (i.e., writes to volumes in stretched pods). If one array of a stretched pod is offline, the writes processed on the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again.

`--pending`

Includes destroyed volumes or snapshots that are in the eradication pending state. If not specified, volumes or snapshots that are pending eradication are not shown.

`--pending-only`

Only displays destroyed volumes or snapshots that are in the eradication pending state.

`--priority-adjustment` ***ADJUSTMENT***

Sets or changes the performance priority of members of the specified volume group or groups. ***ADJUSTMENT*** can be `+10` or `-10` for higher or lower priority, respectively, or `[+0]` to set the priority to `0` (default). The volume group priority adjustment affects the priority adjustment of volumes belonging to the group (with exceptions). See "Priority Adjustments" on page 587.

`--qos`

Displays the QoS information for each volume group, including the bandwidth limit, the IOPS limit, and priority adjustment. If the bandwidth limit is not set, the value appears as a dash (-), representing unlimited throughput. If the IOPS limit is not set, the value appears as a dash (-), representing unlimited IOPS.

`--repeat` ***REPEAT-COUNT***

Sets the number of times to display real-time performance data. If omitted, the repeat count defaults to `1`.

`--size`

For **`purevgroup create`**, sets the virtual capacity of the volume group as perceived by hosts.

For **`purevgroup monitor`**, displays the average I/O sizes per read and write operation.

`--space`

Displays the following information about provisioned (virtual) size and physical storage consumption for each specified volume:

Provisioned Size

Total provisioned size of all volumes. Represents storage capacity reported to hosts.

Thin Provisioning

Percentage of volume sectors that do not contain host-written data because the hosts have not written data to them or the sectors have been explicitly trimmed.

Data Reduction

Ratio of mapped sectors within a volume versus the amount of physical space the data occupies after data compression and deduplication. The data reduction ratio does not include thin provisioning savings.

For example, a data reduction ratio of 5:1 means that for every 5 MB the host writes to the array, 1 MB is stored on the array's flash modules.

Total Reduction

Ratio of provisioned sectors within a volume versus the amount of physical space the data occupies after reduction via data compression and deduplication *and* with thin provisioning savings. Total reduction is data reduction with thin provisioning savings.

For example, a total reduction ratio of 10:1 means that for every 10 MB of provisioned space, 1 MB is stored on the array's flash modules.

Unique

Physical space that is occupied by data of both volumes and file systems after data reduction and deduplication, but excluding metadata and snapshots.

Snapshots

Physical space occupied by data unique to one or more snapshots.

Total

Total physical space occupied by system, shared space, volume, and snapshot data.

`--total`

Follows output lines with a single line containing column totals in columns where they are meaningful.

`--type`

Specifies the type of information (connected hosts, snapshots, or echoed volume names) about specified volumes to be produced in whitespace-separated list format suitable for scripting.

`--workload` ***`WORKLOAD`***

The name of the workload. Include the **`--workload`** option to display the workload the volume group is in. Include the ***`WORKLOAD`*** name to display the vgroups that are part of the specified workload.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--notitle`

Lists information without column titles.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--page`

Turns on interactive paging.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Description

Volume groups organize FlashArray volumes into logical groupings.

If virtual volumes are configured, each volume group on the FlashArray array represents its associated virtual machine, and inside each of those volumes groups are the FlashArray volumes that are assigned to the virtual machine. Volume groups that are associated with virtual machines have names that begin with "`vvol-`" and end with the virtual machine name. For more information about virtual volumes, including configuration steps, refer to the Pure Storage vSphere Web Client Plugin for vSphere User Guide on the Knowledge site (https://support.purestorage.com).

Volume groups can also be created through the **`purevgroup create`** command. Once a volume group has been created, create new volumes directly in the volume group or move existing ones into the volume group.

Creating Volume Groups

The **`purevgroup create`** command creates a volume group on the array. The volume group itself does not contain any meaningful content; instead, it acts as a container that is used to organize volumes.

Once a volume group has been created, volumes can be created inside the volume group or moved into and out of the volume group.

Creating Volumes in Volume Groups

The **`purevol create --size`** command creates one or more Purity//FA virtual storage volumes of the specified size. To create a volume in a volume group, include the volume group namespace in the volume name.

For example, run the following command to create a volume named `vol03` in volume group `vgroup01`:

```
$ purevol create vol03 vgroup01 --size 1T
Name Size Source Created Serial
vgroup01/vol03 1T - 2019-02-02 16:55:15 PST 23364F86CC654C2900011013
```

A volume can only be created inside an existing volume group; you cannot create a volume group while creating its volume.

Moving Volumes into and out of Volume Groups

The `purevol move` command moves one or more volumes between storage containers. Volumes can be moved into, out of, and between pods and volume groups. For more information about moving volumes, refer to purevol.

Renaming Volume Groups

The `purevgroup rename` command changes the current (OLD-NAME) name of a volume group to the new name (NEW-NAME). The name change is effective immediately and the old name is no longer recognized in CLI, GUI, or REST interactions. In the Purity//FA GUI, the new name appears upon page refresh.

Destroying Volume Groups

When a volume group is no longer needed, it can be destroyed. Run the **`purevgroup destroy`** command to destroy a volume group.

When run with no options, the **`purevgroup destroy`** command can only destroy an empty volume group. Before destroying a volume group, ensure all volumes inside the volume group have been either moved out of the volume group or destroyed. Or add the **`--destroy-contents`** option to destroy both the volume group and all volumes inside the volume group.

The destruction of volume groups is not immediate. Instead, the volume group enters an eradication pending period. The eradication pending period is controlled by the applicable eradication delay. See the "Eradication Delays" on page 88 section in the `purearray` chapter. The volume group can be manually recovered or permanently eradicated within the eradication pending period. After the eradication pending period has elapsed, the volume group is completely eradicated and not recoverable.

Recovering Volume Groups

A destroyed volume can be recovered at any time during the eradication pending period. Run the **`purevgroup recover`** command to recover a volume group.

Once a volume group has been recovered, its destroyed volumes can also be recovered to bring the volume group and its contents back to their previous state.

Eradicating Volume Groups

A destroyed volume group can be permanently eradicated at any time during the eradication pending period. Run the **`purevgroup eradicate`** command to eradicate a volume group. Add the **`--eradicate-contents`** option to eradicate the volume group contents as well as the volume group.

Once a volume group has been eradicated, it can no longer be recovered.

Listing Volume Group Details

The **`purevgroup list`** command displays a list of all volume groups on the array and their volumes. Include the **`--pending`** option to display a list of all volume groups, including ones that have been destroyed and are in the eradication pending state. Include the **`--pending-only`** option to only display a list of volume groups that have been destroyed and are in the eradication pending state. Include the **`--space`** option to display the space consumption or effective used capacity of individual volume groups in detail. To display the total space consumption or effective used capacity of all volume groups, specify both the **`--space`** and the **`--total`** options.

The **`purevgroup listobj`** command displays a list of volume group attributes, either in whitespace or comma-separated form, suitable for scripting.

Include the **`--pending`** option to display a list of all volume groups, including ones that have been destroyed and are in the eradication pending state. Include the **`--pending-only`** option to only display a list of volume groups that have been destroyed and are in the eradication pending state.

Include the **`--qos`** option to display the bandwidth limit, the IOPS limit, and priority adjustment for each volume group. If the bandwidth limit for a volume group is not set, the value appears as a dash (-), representing unlimited throughput. If the IOPS limit for a volume group is not set, the value appears as a dash (-), representing unlimited IOPS.

Include the **`--type`** option to list FlashArray objects that are associated with one or more volume groups.

The **`--type`** option accepts the following arguments:

`--type vgroup` (default if `--type` option not specified)
: Display a list of volume groups that have been created on the array. If no volume groups are specified, the list contains the names of all volume groups.

`--type vol`
: Displays a list of volumes that reside in volume groups. If no volume groups are specified, the list contains the names of all volumes that reside in all volume groups.

Include the **`--workload`** option to display the volume groups that are part of a workload.

Monitoring Volume Group I/O Performance

The **`purevgroup monitor`** command displays real-time and historical I/O performance information for all of the specified volume groups. The output includes the following bandwidth, IOPS, and latency data:

- **Name**: Object name.
- **Time**: Current time.
- **B/s**: Bandwidth. Number of bytes read/written.
- **op/s**: IOPS. Number of read, write, or mirrored write requests processed per second.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve a read, write, or mirrored write I/O request. Latency does not include SAN time or QoS rate limit time.
- **B/op**: IOPS. Average I/O size per read, write, mirrored write, and both read and write (all) operations. Must include the **`--size`** option to see the B/op columns.
- **Bandwidth Limit**: Maximum QoS bandwidth limit. Must include the **`--qos`** option to see the Bandwidth Limit column.
- **IOPS Limit**: Maximum QoS IOPS limit. Must include the **`--qos`** option to see the IOPS Limit column.

Include the **`--mirrored`** option to display performance data for mirrored writes (i.e., writes to volumes in stretched pods). If one array of a stretched pod is offline, the writes processed on the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again.

Monitoring Volume Group Latency

The `purevgroup monitor --latency` command displays real-time and historical I/O latency information for volumes in volume groups. The **`purevgroup monitor --latency`** output includes the following data:

- **SAN us/op**: SAN time. Average time, measured in microseconds, required to transfer data between the initiator and the array. Slow data transfers will result in higher SAN times.
- **Queue us/op**: Queue time. Average time, measured in microseconds, that an I/O request spends in the array waiting to be served. The time is averaged across all I/Os of the selected types.
- **QoS Rate Limit us/op**: QoS rate limit time. Average time, measured in microseconds, that reads, writes, or mirrored writes spend in queue as a result of bandwidth limits reached on one or more volumes.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve a read, write, or mirrored write I/O request. Latency does not include SAN time or QoS rate limit time.

Include the **`--mirrored`** option to display latency data for mirrored writes (i.e., writes to volumes in stretched pods).

Quality of Service Limits and Priority Adjustments

Quality of Service (QoS) limits define the maximum level of throughput and the maximum number of I/O operations per second for a volume group. You can set the following QoS limits:

- QoS bandwidth limit
- QoS IOPS limit

QoS limits are not enforced on volume groups that do not have the bandwidth limit or the IOPS limit set.

You can also set a relative priority on a volume group, to ensure that more important workloads receive performance priority (when supported by the FlashArray hardware).

QoS Bandwidth Limit

QoS bandwidth limit can be set on a volume group to enforce maximum allowable throughput. Whenever throughput exceeds the bandwidth limit, throttling occurs. This limit is the aggregate bandwidth for all the volumes in the volume group.

To set the absolute bandwidth limit of a volume group, include the **`--bw-limit`** option when creating or configuring the volume group. If set, the bandwidth limit must be between `1` MB/s and `512` GB/s. The bandwidth limit is specified as an integer, optionally followed by the suffix letter `M` (for megabytes) or `G` (gigabytes).

To clear the bandwidth limit, giving the volume group unlimited throughput, specify an empty string (`""`). To change the existing bandwidth limit of a volume group, use the **`purevgroup setattr --bw-limit`** command.

By default, the QoS bandwidth limit for a volume group is unlimited. When you move a volume to a volume group that has a specified bandwidth limit, the volume inherits the bandwidth limit of the volume group regardless of whether the volume has a previously set bandwidth limit. When you move the volume out of the volume group, the bandwidth limit of the volume defaults to unlimited bandwidth.

QoS IOPS Limit

QoS IOPS limit can be set on a volume group to enforce maximum I/O operations processed per second. Whenever the number of I/O operations per second exceeds the IOPS limit, throttling occurs. This limit is the aggregate IOPS of all the volumes in the volume group.

To set the absolute IOPS limit of a volume group, include the **`--iops-limit`** option when creating or configuring the volume group. If set, the IOPS limit must be between 100 and 100`M`. The IOPS limit is specified as an integer, optionally followed by the suffix letter `K(10^3` or `M(10^6)`. To clear the IOPS limit, specify an empty string (`""`), giving the volume group unlimited IOPS. To change the existing IOPS limit of a volume group, use the **`purevgroup setattr --iops-limit`** command.

By default, the QoS IOPS limit of a volume group is unlimited. When you move a volume to a volume group that has a specified IOPS limit, the volume inherits the IOPS limit of the volume group regardless of whether the volume has a previously set IOPS limit. When you move the volume out of the volume group, the IOPS limit of the volume defaults to unlimited IOPS.

For example, run the following command to create a volume group named `VGROUP02` with the QoS bandwidth limit of 1 gigabyte and the IOPS limit of `5K`.

```
$ purevgroup create VGROUP02 --bw-limit 1G --iops-limit 5K
Name Bandwidth Limit (B/s) IOPS Limit
VGROUP02 1G 5K
```

In the following example, the `purevgroup setattr` command changes the bandwidth limit and the IOPS limit of `VGROUP02` to 2 gigabytes and `10K`, respectively.

```
$purevgroup setattr VGROUP02 --bw-limit 2G --iops-limit 10K
Name Bandwidth Limit (B/s) IOPS Limit
VGROUP02 2G 10K
```

Priority Adjustments

Use a priority adjustment to increase or decrease the performance priority of volumes in the specified volume group or volume groups. Priority values can be `+10`, `0`, and `-10`. Whenever some volumes on the array have a higher priority than others, those volumes at the highest priority level receive performance priority, when supported by the array. If all volumes have the same priority level, no volumes receive performance priority. If all three priority levels are in use, volumes at `+10` receive performance priority and there is no difference in the priority given to volumes at `0` and volumes at `-10` (volumes at `0` do not receive priority over volumes at `-10`).

Use the `purevgroup create --priority-adjustment` command with `+10` and `-10` set the priority adjustment to `+10` and `-10`, respectively. Use the `purevgroup setattr --priority-adjustment` command to change the priority adjustment of volumes in the specified volume group or volume groups. The effect of adjustments with the **`purevgroup setattr`** command depends in part on the previous adjustment value, as follows:

- `+10` raises the priority adjustment from `0` to `+10` or from `-10` to `0`, and has no effect if the priority adjustment was already at `+10`.
- `-10` lowers the priority adjustment from `+10` to `0` or from `0` to `-10`, and has no effect if the priority adjustment was already at `-10`.
- `+0` sets the priority adjustment to `0`, regardless of the previous adjustment value.

Once a priority adjustment is configured for a volume group, volumes that later become members of that group inherit the priority adjustment of the volume group. The exceptions are any volumes that had their priority adjustment set with the **`purevol create`** or **`purevol setattr`** commands and an equals sign ('**=**') with the **`--priority-adjustment`** option. When a volume has its priority adjustment set with the equals sign, the volume retains that priority adjustment and is unaffected by any volume group membership or volume group priority adjustment changes.

(See also the Priority Adjustments section in the `purevol` chapter.)

Examples

Example 1

```
purevgroup create VGROUP10 VGROUP11
```

Creates two volume groups named `VGROUP10` and `VGROUP11`.

Example 2

```
purevgroup create VGROUP01
purevol create --size 1T VGROUP01/VOL01 VGROUP01/VOL02
purevol move VOL10 VGROUP01
```

Creates volume group `VGROUP01`, creates two volumes in volume group `VGROUP01` named `VGROUP01/VOL01` and `VGROUP01/VOL02`, and moves volume `VOL10` from the root of the array to volume group `VGROUP01`.

Example 3

```
purevol move VGROUP01/VOL01 VGROUP02
purevol move VGROUP01/VOL02 ""
purevol move VOL02 POD01
```

Moves volume `VGROUP01/VOL01` from volume group `VGROUP01` to volume group `VGROUP02`, and then moves volume `VGROUP01/VOL02` from volume group `VGROUP01` to pod `POD01` by first moving the volume to the root of the array and then moving the volume from the root of the array to the pod.

Example 4

```
purevol move vgroup10/vol06 vgroup10/vol07 ""
purevgroup list
purevgroup destroy VGROUP10
```

Moves volumes `VGROUP10/VOL06` and `VGROUP10/VOL07` from volume group `VGROUP10` to the root of the array, lists the volume group details to verify that volume group `VGROUP10` is empty, and destroys volume group `VGROUP10`.

Example 5

```
purevgroup monitor --qos
```

Displays the QoS bandwidth limit and QoS IOPS limit for each volume group.

Example 6

```
purevgroup setattr VGROUP02 --bw-limit 2G --iops-limit 10K
```

Sets the bandwidth limit and the IOPS limit on volume group `VGROUP02` to 2 gigabytes and `10K`, respectively.

Example 7

```
purevgroup setattr --iops-limit "" VGROUP02
```

Clears the IOPS limit set on volume group `VGROUP02`.

Example 8

```
purevgroup list --space --total
```

Displays the total space consumption or effective used capacity of all volume groups in detail, including size, thin provisioning, data reduction, total reduction, unique physical space, and snapshots.

Example 9

```
purevgroup setattr --priority-adjustment +10 VGROUP03
```

Increases the priority value for volumes in `VGROUP03`.

Exceptions:

- Volumes that already had priority value `+10` are not changed.
- Volumes whose priority values were previously set with `=10`, `=0`, or `=-10` (by a `purevol` command) are not changed.

See Also

purepod, purevol

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purevol

purevol, purevol-copy, purevol-create, purevol-destroy, purevol-eradicate, purevol-recover, purevol-rename, purevol-snap, purevol-tag, purevol-vc-connection – manage the creation, naming, QoS limits, and destruction of FlashArray virtual storage volumes and snapshots of their contents, as well as the reclamation of physical storage occupied by the data in them

purevol-add, purevol-remove – manage adding and removing of FlashArray virtual storage volumes to and from protection groups, respectively

purevol-monitor – monitors volume I/O performance

purevol-get, purevol-send – manage sending volume snapshots to offload targets or remote arrays and the retrieval of those volume snapshots

purevol-move – manages adding and removing FlashArray volumes to and from storage containers

purevol-tag – manages adding and removing tags to and from objects

purevol-vc-connection - manages connections between vchosts and protocol endpoints.

Synopsis

purevol add --pgroup ***PGROUP VOL...***

purevol copy [--overwrite] [--add-to-protection-group-names ***PGROUP***] [--without-default-protection] [--allow-throttle] [--dry-run] [--context ***REMOTE***] ***SOURCE TARGET***

purevol create --size ***SIZE*** [--bw-limit ***BANDWIDTH-LIMIT***] [--iops-limit ***IOPSLIMIT***] [--priority-adjustment ***ADJUSTMENT***] [--add-to-protection-group-names ***PGROUP***] [--without-default-protection] [--workload ***WORKLOAD***] [--workload-configuration ***WORKLOAD_CONFIGURATION***] [--context ***REMOTE***] ***VOL...***

purevol create --protocol-endpoint [--container-version ***VERSION***] [--context ***REMOTE***] ***PROTOCOL-ENDPOINT***

purevol destroy [--replication-snapshot] [--on ***REMOTE***] [--context ***REMOTE***] ***VOL...***

purevol eradicate [--replication-snapshot] [--on ***REMOTE***] [--context ***REMOTE***] ***VOL...***

purevol get --on ***REMOTE*** [--suffix ***SUFFIX***] ***VOL...***

purevol monitor [--array] [--historical { 1h | 3h | 24h | 7d | 30d | 90d | 1y }] [--interval ***SECONDS***] [--latency] [--mirrored] [--qos] [--repeat ***REPEAT-COUNT***] [--size] [--total] [--csv] [--notitle] [--page] [--raw] [--filter ***FILTER***] [--limit ***LIMIT***] [--sort ***SORT***] [***VOL...***]

purevol move [--add-to-protection-group-names ***PGROUP***] [--remove-from-protection-group-names ***PGROUP***] ***VOL... CONTAINER***

purevol recover [--on ***REMOTE***] [--context ***REMOTE***] ***VOL...***

purevol remove --pgroup ***PGROUP VOL...***

purevol rename [--context ***REMOTE***] ***OLD-NAME NEW-NAME***

purevol send [--suffix ***SUFFIX***] [--to ***REMOTE***] ***VOL...***

purevol snap [--suffix ***SUFFIX***] [--allow-throttle] [--dry-run] ***VOL...***

purevol tag --key ***KEY*** --value VALUE [--namespace ***NAMESPACE***] [--non-copyable] ***VOL...***

purevol truncate --size ***SIZE*** [--context ***REMOTE***] ***VOL...***

purevol untag --key ***KEY*** [--namespace ***NAMESPACE***] ***VOL...***

purevol vc-connection create [--allow-stretched-multi-vchost] --vchost ***VCHOST PROTOCOL_ENDPOINT***

purevol vc-connection delete --vchost ***VCHOST PROTOCOL_ENDPOINT***

purevol vc-connection list [***PROTOCOL_ENDPOINT***]

For **purevol connect** and **purevol disconnect**, see purehost connect.

For **purevol list**, see purevol-list.

For **purevol setattr**, see purevol-setattr.

Arguments

CONTAINER

Storage container to where the volume is to be moved. A storage container can be the root of the array, a pod, or a volume group.

KEY

Part of a key/value pair that is used to identify a tag.

NAMESPACE

Mechanism that is used to prevent key conflicts by differentiating tags that have identical keys but different definitions. A namespace can consist of up to 256 bytes and uses DNS namespace rules. If no **NAMESPACE** is specified, the **default** namespace is assigned.

NEW-NAME

Name by which the volume or volume snapshot is to be known after the command executes.

OLD-NAME

Current name of the volume or volume snapshot to be renamed.

PROTOCOL-ENDPOINT

Name of the protocol endpoint created with **purevol create**.

SOURCE

Volume or snapshot from where data is copied. The data is copied to the **TARGET** volume.

TARGET

Volume to which data is copied. The data is copied from the **SOURCE** volume or snapshot.

VALUE

Part of a key/value pair that is used to identify a tag.

VCHOST

The name of the virtual host that facilitates communication between your VMware vCenter and vSphere API for Storage Awareness (VASA) provider.

VERSION

The container version corresponding to compatibility with several vSphere versions. Version 1 is the default, and is compatible with vSphere versions 7.0.1 and above. Version 2 is compatible with vSphere versions 8.0.0 and above. Version 3 is compatible with vSphere versions 8.0.1 and above. Enter it as just the number (e.g. "1").

VOL

Volume or volume snapshot to be created, destroyed, eradicated, or recovered. For **`purevol snap`**, volume to be snapped. For **`purevol move`**, volume to be moved. For **`purevol send`** and **`get`**, the volume snapshot to be sent or restored.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--add-to-protection-group-names` ***PGROUP***

Adds the volume or volumes to the protection group or groups in ***PGROUP***, a comma-separated list of protection groups. Has no effect on volumes already associated with the protection group.

`--allow-stretched-multi-vchost`

By default, stretched pods can only have one connection between a vchost and a protocol endpoint. This option makes it possible to add multiple connections to a stretched pod.

`--allow-throttle`

Used with **`purevol copy`** and **`purevol snap`** to check the internal health of the array to determine if the volume copy or snapshot would incur too much overhead on the array. If conditions are not optimal at the current time, the volume copy or snapshot is disallowed (throttled).

`--array`

In an ActiveCluster replication configuration, breaks down performance data by the array to which the I/O is directed.

`--bw-limit` ***BANDWIDTH-LIMIT***

Sets the maximum QoS bandwidth limit for the volume. Whenever throughput exceeds the bandwidth limit, throttling occurs. If set, the bandwidth limit must be between 1 MB/s and 512 GB/s. Like volume sizes, the bandwidth limit is specified as an integer, optionally followed by the suffix letter `M` (for megabytes) or `G` (gigabytes).

`--container-verison`

Used with **`purevol create --protocol-endpoint`** and **`purevol setattr`**. Specifies the version of the container when creating a protocol endpoint using the purevol command. This version is intended to ensure compatibility with specific vSphere versions or higher. The option allows users to choose from predefined version numbers, each corresponding to compatibility with different vSphere versions. Version 1 is compatible with vSphere version 7.0.1, version 2 with 8.0.0, version 3 with 8.0.1. The default version is 1 if none is specified.

`--context` ***REMOTE***

Used to specify the remote array on which a command is processed. REMOTE is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--dry-run`

Used with **`purevol copy`** and **`purevol snap`**, simulates the operation without actually executing it. The dry-run checks all conditions the operation would require, and returns a result indicating whether the volume copy or snapshot operation would succeed or not.

`--historical` ***TIME***

Display historical performance data over the specified range of time. Valid time ranges include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year.

`--interval` ***SECONDS***

Sets the number of seconds between displays of real-time performance data. At each interval, the system displays a point-in-time snapshot of the performance data. If omitted, the interval defaults to every 5 seconds.

`--iops-limit` ***IOPS-LIMIT***

Sets the maximum IOPS limit for the volume. Whenever the number of I/O operations per second exceeds the IOPS limit, throttling occurs. If set, the IOPS limit must be between 100 and 100`M`. The IOPS limit is specified as an integer, optionally followed by the suffix letter `K` (10^3) or `M` (10^6).

`--key` ***KEY***

A unique identifier for an item of data. Keys are attached to an object as part of a key/-value pair. Keys are used for filtering and search and may also be used to organize or select subsets of objects. Keys may consist of 1-64 Unicode characters and are case-sensitive. A key may also contain white-space and the following characters: **_**, **.**, **/**, **+**, and **-**.

Note: In order to support filtering syntax, the **=**, *****, and **:** characters are not allowed in keys.

`--latency`

Displays real-time and historical I/O latency information for all or specified volumes.

`--mirrored`

In an ActiveCluster replication configuration, includes performance data for mirrored writes (i.e., writes to volumes in stretched pods). If one array of a stretched pod is offline, the writes processed on the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again.

`--namespace` ***NAMESPACE***

A mechanism that is used to prevent key conflicts by differentiating tags that have identical keys, but different definitions. Namespaces may consist of up to 256 ASCII characters and are case-sensitive. If no **`namespace`** is specified for a tag, the **`default`** namespace is assigned. After a tag is assigned to a **`namespace`**, users must include the value of the namespace when attempting to list tags that are assigned to it. For example, **`purevol list --tag --namespace FOO`**.

`--non-copyable`

Prevents tags from being copied when a volume copy is made.

`--on` ***REMOTE***

For **`purevol destroy`**, destroys snapshots on a connected offload target.

For **`purevol send`**, sends volume snapshots to a connected remote target or remote array.

For **`purevol get`**, restores volume snapshots from a connected remote target or remote array.

For **`purevol eradicate`**, terminates a snapshot on a connected offload target.

For **`purevol recover`**, recover the destroyed snapshot on a connected offload target.

`--overwrite`

Allows **`purevol copy`** to overwrite an existing volume. Without this option, if the target volume already exists, the command fails.

`--pgroup` ***PGROUP***

Comma-separated list of protection groups to which the specified volumes are added or from which the specified volumes are removed. Has no effect on volumes already associated with the protection group.

`--priority-adjustment` ***ADJUSTMENT***

With the **`purevol create`** command, sets the performance priority of the specified volume or volumes. ***ADJUSTMENT*** can be `+10` or `-10` for higher or lower priority, respectively, and those priorities can later be changed by inheriting volume group priorities.

ADJUSTMENT can also be `=10` (high), `=0` (default), or `=-10` (low), to set the priority values that cannot be changed by volume group priority. (See also "Priority Adjustments" on page 615.)

`--protocol-endpoint`

Creates a protocol endpoint, which is used to form connections between FlashArray virtual volumes and VMware ESXi hosts and host groups. For more information about virtual volumes, including configuration steps, refer to the Pure Storage vSphere Web Client

Plugin for vSphere User Guide on the Knowledge site at `https://support.purestorage.com`.

`--remove-from-protection-group-names` ***`PGROUP`***

Removes the volume or volumes from the protection group or groups in ***`PGROUP`***, a comma-separated list of protection groups.

`--replication-snapshot`

Destroy sending and base line snapshots.

`--qos`

Displays the QoS information for each volume, including the bandwidth limit, IOPS limit, and DMM priority adjustment. If the bandwidth limit is not set, the value appears as a dash (-), representing unlimited throughput. If the IOPS limit is not set, the value appears as a dash (-), representing unlimited IOPS.

`--repeat` ***`REPEAT-COUNT`***

Sets the number of times to display real-time performance data. If omitted, the repeat count defaults to 1.

`--size`

For **`purevol create`**, sets the virtual capacity of the volume as perceived by hosts.

For **`purevol monitor`**, displays the average I/O sizes per read and write operation.

`--suffix` ***`SUFFIX`***

If specified with the **`purevol get`**, **`purevol send`**, and **`purevol snap`** commands, the suffix is appended to the names of snapped or recovered volumes to create snapshot names. If not supplied, Purity//FA constructs snapshot names of the form **`VOL.NNN`**, where **`NNN`** is a unique number assigned by Purity//FA.

`--to` ***`REMOTE`***

Used with **`purevol send`** to send one or more volume snapshots to a connected remote target or remote array.

`--total`

Follows output lines with a single line containing column totals in columns where they are meaningful.

`--value` ***`VALUE`***

The data that is identified by a key or a pointer to the location of the data that is represented by a key. Attached to an object as part of a key/value pair. Paired with a key, values are used for filtering and search and may also be used to organize or select subsets of objects. Values can consist of up to 256 Unicode characters and are case-sensitive. Values can be empty but cannot be null.

`--without-default-protection`

Creates or copies the volume without automatic default protection. Ignores the default protection group list, if any.

`--workload`***WORKLOAD***

The name of the workload. Configures the volume according to the settings associated with workload configuration.

`--workload-configuration` ***WORKLOAD_CONFIGURATION***

Used with **`purevol create`** to configure the workload volume. Configures the volume according to the settings associated with workload configuration.

Options that control display format:

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--page`

Turns on interactive paging.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

Object Names

Valid characters are letters (A-Z and a-z), digits (0-9), and the hyphen (-) character. The first and last characters of the name must be alphanumeric, and the name must contain at least one letter or '-'.

Most objects in Purity//FA that can be named, including host groups, hosts, volumes, protection groups, volume and protection group suffixes, SNMP managers, and subnets, can be 1-63 characters in length.

Array names can be 1-56 characters in length. The array name length is limited to 56 characters so that the names of the individual controllers, which are assigned by Purity//FA based on the array name, do not exceed the maximum allowed by DNS.

Names are case-insensitive on input. For example, **`vol1`**, **`Vol1`**, and **`VOL1`** all represent the same volume. Purity//FA displays names in the case in which they were specified when created or renamed.

All objects in a pod have a fully qualified name that include the pod name and object name. The fully qualified name of a volume in a pod is **`POD::VOLUME`**, with double colons (`::`) separating the pod name and volume name. The fully qualified name of a protection group in a pod is **`POD::PGROUP`**, with double colons (`::`) separating the pod name and protection group name. For example, the fully qualified name of a volume named **`vol01`** in a pod named **`pod01`** is **`pod01::vol01`**, and the fully qualified name of a protection group named **`pgroup01`** in a pod named **`pod01`** is **`pod01::pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target array, the fully qualified name of the protection group on the target array is **`POD:PGROUP`**, with single colons (`:`) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to target array **`array02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod01:pgroup01`**.

If a protection group in a pod is configured to asynchronously replicate data to a target pod, the fully qualified name of the protection group on the target pod is **`POD:PGROUP`**, with single colons (`:`) separating the pod name and protection group name. For example, if protection group **`pod01::pgroup01`** on source array **`array01`** asynchronously replicates data to a target pod **`pod02`**, the fully qualified name of the protection group on target array **`array02`** is **`pod02::pod01:pgroup01`**.

All objects in a volume group have a fully qualified name that includes the volume group name and the object name, separated by a forward slash (`/`). For example, the fully qualified name of a volume named **`vol01`** in a volume group named **`vgroup01`** is **`vgroup01/vol01`**.

Realms, pods, and volume groups provide a namespace with unique naming conventions. All objects in a realm have a fully qualified name that include the realm name and object name or realm, pod, and object names. The fully qualified name of a pod in a realm is **`REALM::POD`**, with double colons (`::`) separating the realm name and pod name. The fully qualified name of a volume in a pod in a realm is **`REALM::POD::VOLUME`**, with double colons (`::`) separating the realm, pod, and volume names. The fully qualified name of a protection group in a pod in a realm is **REALM::POD::PGROUP**, with double colons (`::`) separating the realm, pod, and protection group names. The fully qualified name of an object in a realm but not in a pod is **REALM::HOST**, using host as an example.

Volume Sizes

Volume sizes are specified as an integer, followed by one of the suffix letters **K**, **M**, **G**, **T**, **P**, representing KiB, MiB, GiB, TiB, and PiB, respectively, where "Ki" denotes 2^10, "Mi" denotes 2^20, and so on.

Volumes must be between one megabyte and four petabytes in size. If a volume size of less than one megabyte is specified, Purity//FA adjusts the volume size to one megabyte. If a volume size of more than four petabytes is specified, the Purity//FA command fails.

Volume sizes cannot contain digit separators. For example, **1000g** is valid, but **1,000g** is not.

Description

This page describes the management of volumes and volume snapshots.

Volumes

A volume can reside in one of the following types of storage containers: root of the array (`""`), pod, or volume group. The most simple of array configurations is one that contains volumes at the root of the array. Each pod and volume group is a separate namespace for the volumes it contains.

Pods are created and configured to store volumes and protection groups that need to be fully synchronized with other arrays.

Each volume in a pod consists of the pod namespace identifier and the volume name, separated by a double colon (`::`). The naming convention for a volume inside a pod is **POD::VOL**, where:

- **POD** is the name of the container pod.
- **VOL** is the name of the volume inside the pod.

For example, the fully qualified name of a volume named **`vol01`** inside a pod named pod01 is **`pod01::vol01`**.

For more information about pods, refer to "Pods" on page 1.

Volume groups organize volumes into logical groupings. If virtual volumes are configured, a volume group is automatically created for each virtual machine that is created.

Each volume in a volume group consists of the volume group namespace identifier and the volume name, separated by a forward slash (`/`). The naming convention for a volume inside a volume group is **`VGROUP/VOL`**, where:

- **`VGROUP`** is the name of the container volume group.
- **`VOL`** is the name of the volume in the volume group.

For example, the fully qualified name of a volume named vol01 inside a volume group named vgroup01 is **`vgroup01/vol01`**.

For more information about volume groups, refer to "Volume Groups" on page 1.

Volumes that reside in one storage container are independent of the volumes that reside in other containers. In the following example, an array has four volumes with names that contain "**`vol01`**". Volume **`vol01`** represents a volume named **`vol01`** that resides on the root of the array, volume **`vgroup01/vol01`** represents a volume named **`vol01`** that resides in volume group **`vgroup01`**, volume **`vgroup02/vol01`** represents a volume named **`vol01`** that resides in volume group **`vgroup02`**, and volume **`pod01::vol01`** represents a volume named **`vol01`** that resides in pod **`pod01`**. Though all four volumes have "**`vol01`**" in their names, they are completely independent of one another.

```
$ purevol list *vol01
Name           Size Source Created                 Serial
pod::vol01     1T   -      2018-02-05 14:27:23 PST 23364F86CC654C2900011015
vgroup01/vol01 1T   -      2018-02-02 16:49:18 PST 23364F86CC654C2900011011
vgroup02/vol01 5G   -      2018-02-02 16:28:18 PST 23364F86CC654C2900011009
vol01          1G   -      2018-02-02 10:39:32 PST 23364F86CC654C2900011010
```

When administering volumes, always include the namespace identifier in the volume name.

Creating Volumes

The **`purevol create`** command creates one or more FlashArray virtual storage volumes of the host-visible size specified by the **`--size`** option.

Volumes do not consume physical storage until data is actually written to them, so volume creation has no immediate effect on an array's physical storage consumption.

Volumes created by the **purevol create** subcommand have a null value for the source attribute.

With the **--add-to-protection-group-names** option, the newly created volume or volumes are added to the protection group or groups in a comma-separated list of protection groups. The protection group or groups must already exist. Any volumes already in those groups are not affected.

With the **--without-default-protections** option, the default protection group list is ignored and the newly created volume or volumes do not receive automatic default protection. Refer to "Automatic SafeMode™ Default Protection for Volumes" on page 86, in the `purearray` section, for information about default protection group lists.

With the **--workload** option, the new volume is associated with an existing workload. For more information about workloads, refer to "pureworkload" on page 647.

Connecting Volumes to Hosts or Host Groups

Volumes must be *connected* to hosts in order for the hosts to read and write data on them. The **purevol connect** command establishes either *private* connections between volumes and individual hosts or *shared* connections between volumes and host groups. The same connections can be established through the **purehost connect** and **purehgroup connect** commands. For more information about private connections, refer to "purehost-connect" on page 231. For more information about shared connections, refer to "purehgroup-connect" on page 204.

The **purevol connect** command also connects volumes to apps. For more information about Pure Apps and the **pureapp** command.

Moving Volumes

The **purevol move** command moves one or more volumes between storage containers. Volumes can be moved into, out of, and between pods and volume groups. Run the **purevol move** command to move volumes from the root of the array to a volume group, or from a volume group to the root of the array (**""**).

For example, run the following command to move two volumes named **vol03** and **vol04** from the root of the array to volume group **vgroup01**:

```
$ purevol move vol03 vol04 vgroup01
Name           Size Source Created                 Serial
vgroup01/vol03 1T   -      2018-02-02 16:55:15 PST 23364F86CC654C2900011013
vgroup01/vol04 1T   -      2018-02-02 16:55:15 PST 23364F86CC654C2900011013
```

As another example, run the following command to move volume **`vgroup01/vol03`** from volume group **`vgroup01`** to the root of the array:

```
$ purevol move vgroup01/vol03 ""
Name  Size Source Created                  Serial
vol03 1T   -      2018-02-02 16:55:15 PST  23364F86CC654C2900011013
```

You can move a volume out of a protection group, but only if you specify it in the command.

```
$ purevol move --remove-from-protection-group-names pg, pg2 vol pod
Name Size Source Created Serial Promotion Status
pod::vol 2G - 2021-07-20 16:45:33 PDT 7E3A13D5E45F4C1E000479D6 promoted
```

You can also simultaneously move a volume into a different protection group using the **`--add-to-protection-group-names`** option.

Moving Volumes in Stretched Pods

Use the **`purevol move`** command to move volumes between stretched pods, if these conditions are met:

- The source pod and the destination pod are stretched between the same pair of arrays and are in sync.
- Each pod is online on its respective array.

Notes about moving volumes between stretched pods:

- Following any interruption or failure when moving volumes between stretched pods, the two pods would be recovered through resync. Before running the volume move operation, ensure there is a symmetric configuration from their initiators to these two pods' volumes.
- For the case of xcopy between two different volumes within the same pod, in order to move those volumes, use a single **`purevol move`** request to move both volumes. Moving those volumes one at a time would cause interruptions in I/O. This example shows a single command to move both volumes to another pod (`pod-1` and `pod-2` must be stretched between the same pair of arrays):

```
      purevol move pod-1::vol-1,pod-1::vol-2 pod-2
```

Moving Volumes with SafeMode Enabled

A volume in a SafeMode-enabled protection group can only be moved to a SafeMode-enabled protection group with equal or better SafeMode protections, as determined by the following configuration characteristics:

- Snapshot schedule frequency and retention.
- Replication schedule frequency and retention.
- Target retention number of days and number retained per day.
- Blackout window (if the current protection group has a blackout window).

If the target protection group does not match or exceed the current protection group on all of these configurations, the volume cannot be moved.

Notes on volume move with SafeMode:

- For volumes in protection groups based on host or hostgroup membership, Purity does not ensure that the target protection group has equal or better SafeMode protections.
- Contact Pure Technical Support to move a volume that is currently a member of more than one protection group.

Renaming Volumes

Run the **`purevol rename`** subcommand to change the current (OLD-NAME) name of the volume or volume snapshot to the new name (NEW-NAME). The name change is effective immediately and the old name is no longer recognized in CLI, GUI, or REST interactions. In the Purity//FA GUI, the new name appears upon page refresh.

Copying Volumes

The **`purevol copy`** command can be used to create a volume from another volume, or to replace the contents of an existing volume with those of another volume. Volumes created in this way have a **`Source`** attribute whose value is the name of the originating volume. An *undo snapshot* is automatically taken (providing an eradication pending period during which the previous contents can be retrieved). A volume cannot be copied across pods if the source and target volumes are both stretched but on different pairs.

With the **`--add-to-protection-group-names`** option, the copied volume or volumes are added to the protection group or groups in a comma-separated list of protection groups. The protection group or groups must already exist. Any volumes already in those groups are not affected.

With the **`--without-default-protections`** option, the default protection group list is ignored and the copied volume or volumes do not receive automatic default protection. Refer to "Automatic SafeMode™ Default Protection for Volumes" on page 86, in the **`purearray`** section, for information about default protection for volumes.

Throttling a Volume Copy

An array may become overloaded when taking volume copies. Use the **`purevol copy --allow-throttle`** command to check the internal health of the array to determine if a copy is recommended at that point in time. This check allows for throttling of volume copies to lessen the impact on the array performance.

```
$ purevol copy --allow-throttle vol1 vol2
Name  Size Source Created                  Serial        Protection
vol3  1T   vol1   2024-02-23 10:21:43 PST  EF9CF729113EB  pgroup-auto
```

Simulating a Volume Copy

The **`purevol copy --dry-run`** option checks all the conditions the volume copy would require. Use this command to determine if the volume copy operation would return an error or return successfully (without creating the volume copy). This example returns an error:

```
$ purevol copy --dry-run vol1 vol2
Success  Details
False    Error on vol2: Volume already exists.
```

This example returns success:

```
$ purevol copy --dry-run vol1 vol3
Success  Details
True     Volume(s) can be created from source volume.
```

Destroying Volumes

When a volume or snapshot is no longer required, it can be *destroyed* to reclaim the physical storage occupied by its data (**`purevol destroy`** command). Destroying a volume implicitly destroys all of its snapshots.

Destroying an individual snapshot only reclaims space that is uniquely charged to the snapshot. Space shared with older snapshots or with the originating volume remains allocated. Before a volume can be destroyed, it must be disconnected (**`purevol disconnect`** command).

Destruction of volumes and snapshots is not immediate. The **`purevol destroy`** command places volumes and snapshots in the *eradication pending* state, making them immediately inaccessible, while leaving their data intact for an *eradication pending* period. The eradication pending period is controlled by the applicable eradication delay. See the "Eradication Delays" on page 88 section in the `purearray` chapter. At any time during the eradication pending period, an administrator can use the `purevol recover` command to recover the contents of a destroyed volume or destroyed snapshot.

When a destroyed volume's or snapshot's eradication pending period has elapsed (or if an administrator eradicates the volume or snapshot during the eradication pending period), Purity//FA reclaims the physical storage occupied by its data.

If the physical storage occupied by a destroyed volume's or destroyed snapshot's data is required immediately (for example, if a large amount of new data is to be written), an administrator can use the `purevol eradicate` command to terminate the eradication pending period and initiate immediate storage reclamation. The `purevol eradicate` command only acts on previously destroyed volumes and snapshots. Manual eradication is not supported if the SafeMode manual eradication prevention feature is enabled.

Once reclamation starts, whether because an eradication pending period has elapsed or because a `purevol eradicate` command was executed, a destroyed volume's or destroyed snapshot's data can no longer be recovered.

Adding Volumes to Protection Groups

The `purevol add` command adds existing volumes to existing protection groups. Multiple volumes can be added to multiple protection groups. Enter multiple protection groups in comma-separated format.

If a protection group already includes other volumes, those volumes are unaffected.

Only members of the same object type can belong to a protection group. For example, a volume cannot be added to a protection group that already contains hosts or host groups.

The `purevol add` command only accepts volumes that do not already belong to any of the specified protection groups. If any of the volumes already belong to any of the protection groups, the entire command fails.

Run the `purevol list --protect` command to see a list of all protected volumes and their associated protection groups.

The `purevol remove` command removes one or more volumes from one or more protection groups. All of the specified volumes must belong to all of the specified protection groups in the command; otherwise, the command fails.

Volumes can also be added to protection groups through default protection group lists. See "Automatic SafeMode™ Default Protection for Volumes" on page 86, in the `purearray` section, for information about default protection group lists.

Volumes can also be added to and removed from protection groups through the `purepgroup setattr` command.

Monitoring Volume I/O Performance

The `purevol monitor` command displays real-time and historical I/O performance information for all or specified volumes. The `purevol monitor` output includes the following data about bandwidth, IOPS, and latency:

- **Name**: Object name.
- **Time**: Current time.
- **B/s**: Bandwidth. Number of bytes read/written.
- **op/s**: IOPS. Number of read, write, or mirrored write requests processed per second.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve a read, write, or mirrored write I/O request. Latency does not include SAN time or QoS rate limit time.
- **B/op**: IOPS. Average I/O size per read, write, mirrored write, and both read and write (all) operations. Must include the **`--size`** option to see the B/op columns.
- **Bandwidth Limit**: Maximum QoS bandwidth limit. Must include the **`--qos`** option to see the Bandwidth Limit column.
- **IOPS Limit**: Maximum QoS IOPS limit. Must include the **`--qos`** option to see the IOPS Limit column.

By default, the `purevol monitor` command displays real-time performance data. Include the **`--interval`** option to specify the number of seconds between each real-time update. Include the **`--repeat`** option to specify the number of times to repeat the real-time update. The **`--interval`** and **`--repeat`** options can be combined.

Include the **`--historical`** option to display historical performance data over any of the following ranges of time: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, or 1 year. Include the **`--qos`** option to display the bandwidth limit and the IOPS limit for each volume. If the bandwidth limit for a volume is not set, the value appears as a dash (-), representing unlimited throughput. If the IOPS limit is not set, the value appears as a dash (-), representing unlimited IOPS.

In an ActiveCluster replication configuration, include the **`--array`** option to break down performance data by the array to which the I/O is directed. Include the **`--mirrored`** option to display performance data for mirrored writes (i.e., writes to volumes in stretched pods). If one array of a stretched pod is offline, the writes processed on the other array are reported as local writes rather than mirrored ones. Once the pod is back online, in sync, and processing each write on both sides, the writes to its volumes are reported as mirrored writes again.

Monitoring Latency

The `purevol monitor --latency` command displays real-time and historical I/O latency information for all volumes. The `purevol monitor --latency` output includes the following data:

- **SAN us/op**: SAN time. Average time, measured in microseconds, required to transfer data between the initiator and the array. Slow data transfers will result in higher SAN times.
- **Queue us/op**: Queue time. Average time, measured in microseconds, that an I/O request spends in the array waiting to be served. The time is averaged across all I/Os of the selected types.
- **QoS Rate Limit us/op**: QoS rate limit time. Average time, measured in microseconds, that reads, writes, or mirrored writes spend in queue as a result of bandwidth limits reached on one or more volumes.
- **us/op**: Latency. Average time, measured in microseconds, it takes the array to serve a read, write, or mirrored write I/O request. Latency does not include SAN time or QoS rate limit time.

Include the **`--mirrored`** option to display latency data for mirrored writes (i.e., writes to volumes in stretched pods).

Remote Commands

The `--context` ***`REMOTE`*** option is used to execute a command on a different array within the same fleet. For more information on how remote commands are used, see "purefleet" on page 177.

Volume Snapshots

A volume snapshot is a point-in-time image of the contents of a volume.

The volume snapshot naming convention is **`VOL.NNN`**, where:

`VOL` is the name of the volume.

`NNN` is a unique monotonically increasing number or a manually-assigned volume snapshot suffix name.

Creating Volume Snapshots

The `purevol snap` command creates point-in-time *snapshots* of the contents of one or more volumes. FlashArray snapshots make use of Purity//FA deduplication technology; they consume space only when data is written to the volumes on which they are based.

All snapshots taken as a result of a single `purevol snap` command are atomic. They represent the specified volumes' contents as of a single instant in time.

Snapshots are *immutable*; they cannot be connected to hosts or host groups, and therefore the data they contain cannot be read or written.

Snapshot Throttling

An array may become overloaded when taking snapshots. Use the **`purevol snap --allow-throttle`** command to check the internal health of the array to determine if a snapshot is recommended at that point in time. This allows for throttling of snapshots to lessen the impact on the array performance.

Simulating Snapshots

The **`purevol snap --dry-run`** option checks all the conditions the snapshot would require. Use this command to determine if the command would return an error or return successfully (without taking a snapshot).

Renaming Volume Snapshots

Run the `purevol rename` subcommand to change the current snapshot suffix name (OLD-NAME) of the volume to the new snapshot suffix name (NEW-NAME). The name change is effective immediately and the old name is no longer recognized in CLI, GUI, or REST interactions. In the Purity//FA GUI, the new name appears upon page refresh.

Destroying Volume Snapshots

When a snapshot is no longer required, it can be *destroyed* to reclaim the physical storage occupied by its data (`purevol destroy` command).

Destroying an individual snapshot only reclaims space that is uniquely charged to the snapshot. Space shared with older snapshots or with the originating volume remains allocated.

Destruction of snapshots is not immediate. The `purevol destroy` command places snapshots in the *eradication pending* state, making them immediately inaccessible, while leaving their data intact for an eradication pending period. At any time during the eradication pending period, an administrator can use the `purevol recover` command to recover the contents of a destroyed snapshot.

When a destroyed snapshot's eradication pending period has elapsed (or if an administrator eradicates the snapshot during the eradication pending period), Purity//FA reclaims the physical storage occupied by its data.

The length of the eradication pending period is controlled by the applicable eradication delay setting. See "Eradication Delays" on page 88 in the `purearray` chapter.

If the physical storage occupied by a destroyed snapshot's data is required immediately (for example, if a large amount of new data is to be written), an administrator can use the **`purevol eradicate`** command to terminate the eradication pending period and initiate immediate storage reclamation. The **`purevol eradicate`** command only acts on previously destroyed snapshots.

Once reclamation starts, whether because an eradication pending period has elapsed or because a **`purevol eradicate`** command was executed, a destroyed snapshot's data can no longer be recovered.

Restoring Volumes from Volume Snapshots

The **`purevol copy`** command can be used to restore a volume by creating the volume from a snapshot, or to replace the contents of the existing volume with those of the snapshot. Restoring a volume brings the volume back to the state it was when the snapshot was taken. When a volume is restored from a snapshot, Purity//FA overwrites the entire volume with the snapshot contents. After you restore a volume, the created date of the overwritten volume is set to the snapshot created date. Purity//FA automatically takes an undo snapshot of the overwritten volume. The undo snapshot enters an *eradication pending* period, after which time the snapshot is destroyed. During the pending period, the undo snapshot can be viewed, recovered, or permanently eradicated through the Destroyed Volumes folder.

Volumes created in this way have a **`Source`** attribute whose value is the name of the originating volume.

A snapshot's *size* attribute cannot be changed. When a snapshot is restored to create new or replace an existing volume, the volume's size becomes that associated with the snapshot.

Restoring Volume Snapshots from an Offload Target or Remote Array

The offload target feature enables system administrators to replicate point-in-time volume snapshots from the array to an external storage system, such as an S3 bucket. The **`purevol get --on`** command restores volume snapshots from an offload target or remote array. Include the **`--suffix`** option to replace the default snapshot suffix with a custom string. After a volume snapshot has been restored from an offload target or remote array, run the **`purevol copy`** command to copy the restored volume snapshot to create a new volume. For more information about offload targets, refer to "pureoffload" on page 316.

Sending Volume Snapshots

Run the **purevol send** command to send a volume snapshot to an offload target or to a remote array. This command provides an alternative to scheduled offloads through protection group replication. If the target is another array, the volume snapshot can be sent from a protection group, from outside a pod, or from a pod, stretched or unstretched. For offload targets, the snapshot can only be sent from outside a pod or from a protection group. Include the **--suffix** option to replace the default snapshot suffix with a custom string. In this example, **testSnapshot_vol1.1** has been created on the source array, **array1**, and being sent to **array2** with the suffix **abc**.

```
pureuser@array1:~# purevol send testSnapshot_vol1.1 --to array2 --suffix abc
Name               Source           Remote Created
testSnapshot_vol1.1 testSnapshot_vol1 array2 2023-06-12 10:57:29 PDT
```

On the target array, the snapshot appears with the suffix **abc**.

```
pureuser @array2:~# purevol list --snap
Name                         Size Source                   Created Serial
array1:testSnapshot_vol1.abc 2M  array1:testSnapshot_vol1 2023-06-11 23:19:48 PDT
B1DF344572554520000017729
```

Tagging Volumes

Tags are used to organize and filter resources in a customer-defined way. Tags are defined with **key**/**value** pairs and may be assigned to a **namespace** in order to prevent conflicts from occurring when tags have identical keys but different definitions. Tags may also be designated as **non-copyable** to prevent them from being copied along with volumes during a **purevol copy** operation.

Virtual Volumes

VMware Virtual Volumes (vVols) storage architecture is designed to give VMware administrators the ability to perform volume operations and apply protection group snapshot and replication policies to FlashArray volumes directly through vSphere.

On the FlashArray side, virtual volumes are created and then connected to VMware ESXi hosts or host groups via a protocol endpoint (also known as a conglomerate volume). The protocol endpoint itself does not serve I/Os; instead, its job is to form connections between FlashArray volumes and ESXi hosts and host groups.

Each protocol endpoint can connect multiple virtual volumes to a single host or host group, and each host or host group can have multiple protocol-endpoint connections.

LUN IDs are automatically assigned to each protocol endpoint connection and each virtual volume connection. Specifically, each protocol endpoint connection to a host or host group creates a LUN (PE LUN), while each virtual volume connection to a host or host group creates a sub-LUN. The sub-LUN is in the format **`x:y`**, where **`x`** represents the LUN of the protocol endpoint through which the virtual volume is connected to the host or host group, and **`y`** represents the sub-LUN assigned to the virtual volume.

In the following example, one virtual volume named **`pure_vVol`** and one protocol endpoint named **`pure_PE`** are connected to host **`host01`**. The virtual volume is identified by the sub-LUN (**`7:1`**).

```
$ purevol list --connect --sort lun
Name           Size LUN Host Group Host
pod01::vol006  1T   1   -          host01
pod100::vol007 1T   2   -          host01
pod01::vol008  1T   3   -          host01
vol009         1T   4   -          host01
vol043         2T   5   -          host01
pure_PE        -    7   -          host01
pure_vVol      1T   7:1 -          host01
```

Note: Virtual volumes are primarily configured through the vSphere Web Client plugin.

For more information about virtual volumes, including steps on how to configure them, refer to the Pure Storage vSphere Web Client Plugin for vSphere User Guide on the Knowledge site at https://support.purestorage.com.

Multi-tenancy for Virtual Volumes

Pod protocol endpoints provide a mechanism to map pods as VASA storage containers to vSphere datastores. When its first VASA provider is registered, Purity//FA automatically creates a protocol endpoint at the root of the array. The `purevol create --protocol-endpoint` command supports creating additional protocol endpoints in pods. A pod with a protocol endpoint is considered a VASA storage container by vSphere and maps to a specific vVol datastore in vSphere. The new storage container inherits the name of the pod and appears in vSphere in the datastore "Backing Storage Container" list. Access to, visibility of, and control over objects to each storage container can be restricted through the vSphere datastore to provide support for multi-tenancy for virtual volumes.

The `purevol create --protocol-endpoint` command creates a protocol endpoint in the specified pod:

```
purevol create --protocol-endpoint pod1::pe1
Name Source Created Serial Container Version
pod1::pe1 - 2022-12-11 19:32:37 PST F0105F9E82944C9900011010 1
```

Connect the pod protocol endpoint to an ESXi host group connected to vSphere with `purevol connect` command:

```
purevol connect --hgroup hg-esxi1 pod1::pe1
Name Host Group Host LUN
pod1::pe1 hg-esxi1 - 254
```

Note: Pod protocol endpoints currently are not supported with ActiveDR replica links.

Protocol Endpoints and Container Versions

When creating a protocol endpoint, specifying the `--container-version` option with one of the version numbers ensures that the pod that vSphere is accessing (i.e. the container) will be compatible with the associated vSphere version or higher. (Version 1, the default version, is compatible with vSphere version 7.0.1 and above. Version 2 is compatible with vSphere version 8.0.0 and above. Version 3 is compatible with vSphere version 8.0.1 and above.) Implementing a container version happens at the protocol endpoint level, but affects the pod to which it belongs. If a protocol endpoint is created without specifying the version number, the pod will get version 1 by default. The last protocol endpoint that is explicitly given a version number will determine the version for the pod. If a pod is running a higher version and a new PE is created without specifying the version, the pod will not change versions back to version 1.

You can use the `--protocol-endpoint` and `--container-version` options with the `purevol create` command to specify the container version for the storage container in which the protocol endpoint will be located.

```
purevol create --protocol-endpoint --container-version 1 PE1
```

```
Name Source Created Serial Container Version
PE1 - 2023-09-01 00:00:01 PDT 1B 1
```

You can also modify the container version using the `purevol setattr --container-version` command.

```
purevol setattr --container-version 1 pod1::PE1
Name Source Created Serial Container Version
pod1::PE1 - 2023-09-01 00:00:01 PDT 1B 1
```

Connecting vchosts and Protocol Endpoints

Before vSphere can access your pod, you you must create a connection between the vchost and the protocol endpoint in the pod. You can do this using the `purevol vc-connection create` command. For example, the command below creates a connection between a protocol endpoint (pe1, located in pod1) and a vchost (vchost1).

```
purevol vc-connection create --vchost vchost1 pod1::pe1
Protocol-endpoint Vchost
pod1::pe1 vchost1
```

You can create a public connection by using * in place of the vchost name.

```
purevol vc-connection create --vchost * pod2::pe2
Protocol-endpoint Vchost
pod2::pe2 *
```

Multiple vchosts can connect to the same protocol endpoint, and multiple protocol endpoints can connect to the same vchost. However, if a pod is stretched, the default behavior is that it can only have one connection unless otherwise specified using the `--allow-stretched-multi-vchost` option. This option may be useful in situations where a stretched pod has a private connection, and another connection must be made, perhaps before deleting the private connection. This option only works for a protocol endpoint in a stretched pod. Non-stretched pods can already have multiple connections by default.

```
purevol vc-connection create --allow-stretched-multi-vchost --vchost vchost2
pod3::stretchedPE3
Protocol-endpoint Vchost
pod3::stretchedPE3 vchost2
```

You can list existing connections with the `purevol vc-connection list` command. You can specify a protocol endpoint to filter the list that displays to only connections to that protocol endpoint.

You can delete existing connections with the `purevol vc-connection delete` command.

Each of these `purevol vc-connection` commands has a corresponding `purevchost connection` command.

Quality of Service Limits and DMM Priority Adjustments

Quality of service (QoS) limits define the maximum level of throughput and the maximum number of I/O operations per second for a volume. You can set the following QoS limits:

- QoS bandwidth limit
- QoS IOPS limit

QoS limits are not enforced on volumes that do not have the bandwidth limit or the IOPS limit set.

You can also set a relative priority on a volume, volumes, or volume group, to ensure that more important workloads receive performance priority (when supported by the FlashArray hardware).

QoS Bandwidth Limit

QoS bandwidth limit can be set on a volume to enforce maximum allowable throughput. Whenever throughput exceeds the bandwidth limit, throttling occurs. For example, performance caps for mission-critical workloads can be set to higher bandwidth limits relative to other volumes.

To set the absolute bandwidth limit of a volume, include the **`--bw-limit`** option when creating or configuring the volume. If set, the bandwidth limit must be between 1 MB/s and 512 GB/s. Like volume sizes, the bandwidth limit is specified as an integer, optionally followed by the suffix letter `M` (for megabytes) or `G` (gigabytes).

By default, the QoS bandwidth limit of a volume is unlimited. When you move a volume to a volume group that has a specified bandwidth limit, the volume inherits the bandwidth limit of the volume group regardless of whether the volume has a previously set bandwidth limit. When you move the volume out of the volume group, the bandwidth limit of the volume defaults to unlimited bandwidth.

For example, run the following command to create a volume named **vol05** of 10 gigabytes with the bandwidth limit of 1 gigabyte.

```
$ purevol create vol05 --size 10G --bw-limit 1G
Name  Size Source Created                 Serial                   Bandwidth Limit (B/s)
vol05 10G  -      2019-05-20 16:07:40 PDT 3FBF2D29EE18492000011012 1G
```

QoS IOPS Limit

QoS IOPS limit can be set on a volume to enforce maximum I/O operations processed per second. Whenever the number of I/O operations per second exceeds the IOPS limit, throttling occurs.

To set the absolute IOPS limit of a volume, include the **--iops-limit** option when creating or configuring the volume. If set, the IOPS limit must be between 100 and 100**M**. The IOPS limit is specified as an integer, optionally followed by the suffix letter **K** (10^3) or **M** (10^6).

By default, the QoS IOPS limit of a volume is unlimited. When you move a volume to a volume group that has a specified IOPS limit, the volume inherits the IOPS limit of the volume group regardless of whether the volume has a previously set IOPS limit. When you move the volume out of the volume group, the IOPS limit of the volume defaults to unlimited IOPS.

For example, run the following command to create a volume named **vol06** of 10 gigabytes with the IOPS limit of **5K**.

```
$ purevol create vol06 --size 10G --iops-limit 5K
Name  Size Source Created                 Serial                   IOPS Limit
vol06 10G  -      2019-05-20 16:06:53 PDT 3FBF2D29EE18492000011011 5K
```

Priority Adjustments

DMM Priority adjustments can be set on volumes to reflect the relative importance of their workloads. For example, a priority adjustment can configure a higher performance priority for volumes that run critical, latency-sensitive workloads and configure a lower priority for volumes that run workloads with less latency sensitivity. Supported priority adjustments are `+10`, `0`, and `-10`.

By default, all volumes have the same priority adjustment of `0`. If you increase the priority adjustments for some volumes, then those volumes receive performance priority relative to all other volumes.

If all volumes are set to the same priority, even the higher priority (`+10`), then all volumes have the same relative priority and no volume receives performance priority. If all three priority levels are in use, volumes at `+10`receive performance priority and there is no difference in the priority given to volumes at `0` and volumes at `-10` (volumes at `0` do not receive priority over volumes at `-10`).

To set the priority adjustment of a volume, include the **`--priority-adjustment`** option when creating the volume with the **`purevol create`** command, with an adjustment value of **`+10`** (higher priority), **`+0`** (default priority), or **`-10`** (lower priority). These values are affected by the volume group priority adjustment when the volume becomes a member of a volume group.

In general, the priority value of a volume in a volume group is affected by the priority adjustment of that volume group. The volume group priority adjustment is added to the volume priority adjustment to determine to volume priority value. Use an adjustment value of **`=10`** (high), **`=0`** (default), or **`=-10`** (low), to set a priority adjustment that cannot be changed by volume group membership or volume group priority adjustment changes.

The following command to create a volume named **`vol07`** of 10 gigabytes with a higher performance priority.

```
$ purevol create vol07 --size 10G --priority-adjustment +10
Name  Size Source  Created         Serial                   Priority Adjustment  Priority
vol07 10G  - 2021-10-05 PDT  31E95A77440B4D8300011014  + 10                10
```

See also Priority Adjustments in the purevgroup chapter.

Volume and Snapshot Space Accounting

FlashArray snapshots are inherently *space saving* in that a snapshot only consumes physical storage when data on its source volume is overwritten. Similarly, a volume created from a snapshot only consumes space when its contents change.

Output of the **`purevol list --snap`** command displays the amount of physical storage charged to specified volumes and their snapshots.

For example, if two snapshots of a volume, **`s1`** and **`s2`**, are taken at times *t1* and *t2* (*t1* < *t2*), space consumption is accounted as follows:

Volume data written before *t1*

If no snapshots exist, the volume is charged for space occupied by host-written data.

Volume data written after t1 but before *t2*

Snapshot **`s1`** is charged for space occupied by the pre-update contents of the updated volume data.

Volume data written after *t2*

Snapshot **`s2`** is charged for space occupied by the pre-update contents of the updated volume data (**`s1`** shares the pre-update contents with **`s2`**).

If **`s1`** is destroyed but **`s2`** remains, storage charged to **`s1`** is reclaimed because there is no longer a need to preserve pre-update content for updates made prior to *t2*. If **`s2`** is destroyed, however, storage charged to it must be preserved (it represents pre-update content for updates made after *t1*, which must be preserved as long as **`s1`** exists).

To generalize, if a volume has two or more snapshots taken at different times:

Destroying the oldest snapshot

Space charged to the destroyed snapshot is reclaimed (after the eradication pending period has elapsed or after a **`purevol eradicate`** command is executed).

Destroying a snapshot other than the oldest

Space charged to the destroyed snapshot is charged to the next older snapshot unless it is already reflected in that snapshot (because the same volume block address was written both after the next older snapshot and after the snapshot being destroyed), in which case it is reclaimed.

Exceptions

Destroyed volumes cannot be truncated directly. They must be recovered to their original sizes first (provided that eradication has not begun) and then truncated. Truncation is immediate and irrevocable.

Volumes cannot be eradicated unless (a) they have previously been destroyed, (b) they are within their eradication pending periods, and (c) the SafeMode manual eradication prevention feature has not been enabled.

The names of destroyed volumes cannot be reused to create other volumes until eradication has begun.

When a volume is destroyed, all of its existing snapshots are destroyed as well. Similarly, when a volume is recovered, snapshots that were implicitly destroyed as a consequence of the volume's destruction are recovered as well. Snapshots that are destroyed explicitly are not recovered by a **`purevol recover`** command specifying the volumes from which they were created.

Examples

Example 1

```
purevol create --size 100G VOL5
purevol create --size 1T VGROUP01/VOL5
purevol create --size 1T VGROUP02/VOL5
purevol create --size 50G POD01::VOL5
```

Creates four volumes. One volume (**`VOL5`**) is on the root of the array, two volumes (**`VGROUP01/VOL5`** and **`VGROUP02/VOL5`**) are in volume groups, and one volume (**`POD01::VOL5`**) is in a pod. All four volumes reside in storage containers that are completely independent of one another.

Example 2

```
purevol create --size 100G --bw-limit 10M VOL1 VOL2 VOL3
```

Creates three volumes called **`VOL1`**, **`VOL2`**, and **`VOL3`**, each with a host-visible capacity of 100 gigabytes (10^2 * 2^30 bytes), and sets the maximum QoS bandwidth limit of each volume to 10 MB/s.

Example 3

```
purevol create --size 100G VOL5 VOL6
```

... *time passes*...

```
purevol destroy VOL5
```

...*eradication pending period*...

```
purevol eradicate VOL5
```

Creates 100-gigabyte volumes **`VOL5`** and **`VOL6`**. Some time later, destroys **`VOL5`**, making it inaccessible to hosts and starting the eradication pending period after which Purity//FA reclaims the physical storage occupied by its data. The **`purevol eradicate`** command begins reclamation of physical storage occupied by **`VOL5`**'s data immediately; **`VOL5`** can no longer be recovered after this point.

The **`purevol eradicate`** command is not supported when the SafeMode manual eradication prevention feature is enabled.

Example 4

```
purevol destroy VOL4.415
```

... *eradication pending period*...

```
purevol recover VOL4.415
```

Destroys volume snapshot **`VOL4.415`** and starts the eradication pending period after which Purity//FA would begin to reclaim the physical storage occupied by the volume snapshot's data. During the eradication pending period, the volume snapshot is restored to active status with its data intact.

Example 5

```
purevol get --on nfs-target array01:pgroup01.25.vol01
purevol copy array01:vol01.restore-of.array01-pgroup01-25-vol01 restore1
```

Restores volume snapshot **`array01:pgroup01.25.vol01`** from offload target **`nfs-target`**, resulting in a volume snapshot with the name **`array01:vol01.restore-of.array01-pgroup01-25-vol01`**, which is then copied to create a new volume named **`restore1`**.

Example 6

```
purevol monitor
```

Displays real-time performance data for all volumes.

Example 7

```
purevol monitor --repeat 60 --size
```

Displays real-time performance data for all volumes. The output includes average I/O sizes. Sixty (60) point-in-time updates are displayed, with each update taken at the default interval of every five (5) seconds.

Example 8

```
purevol monitor --historical 24h --csv VOL1 VOL2
```

Displays a CSV output of historical performance data for volumes **`VOL1`** and **`VOL2`** over the past 24 hours.

Example 9

```
purevol monitor --qos
```

Displays the QoS bandwidth limit and the QoS IOPS limit for each volume.

Example 10

```
purevol move VGROUP01/VOL01 ""
purevol move VOL01 POD01
```

Moves volume **VOL1** from volume group **VGROUP01** to the root of the array, and then moves the same volume from the root of the array to pod **POD01**.

Example 11

```
purevol rename VOL1 VOL100
purevol rename VGROUP02/VOL5 VGROUP02/VOL500
purevol rename POD01::VOL8 POD01::VOL800
```

Changes the names of volume **VOL1** to **VOL100**, volume **VGROUP02/VOL5** to **VGROUP02/VOL500**, and volume **POD01::VOL8** to **POD01::VOL500.**

Example 12

```
purevol rename VOL1.SUFFIX1 VOL1.SUFFIX2
```

Changes the suffix of volume snapshot **VOL1.SUFFIX1** from **SUFFIX1** to **SUFFIX2**.

Example 13

```
purevol snap VOL1
purevol snap --suffix MONDAY1 VOL1
purevol snap --suffix MONDAY2 VOL1 VOL2
```

1. Create snapshot **VOL1.1**. (Purity//FA chooses the unique number.)
2. Create snapshot **VOL1.MONDAY1**, representing **VOL1** at time *t1*.
3. Create two snapshots:
   - **VOL1.MONDAY2**, representing **VOL1** at time *t2*.
   - **VOL2.MONDAY2**, representing **VOL2** at time *t2*.

Note: The new snapshot automatically inherits the tags that are originally on VOL1.

Example 14 (Using a snapshot to recover from data corruption)

```
purevol snap --suffix SNAPSHOT VOL1
```

... *applications using* **`VOL1`** *encounter data corruption...*

```
purevol copy VOL1.SNAPSHOT GOODVOL1
```

... *hosts connect to* **`GOODVOL1`**, *validate data, and resume...*

```
purevol destroy VOL1.SNAPSHOT
purevol eradicate VOL1.SNAPSHOT
```

1 Create snapshot **`VOL1.SNAPSHOT`** at a time when data is known to be correct. Some time later, applications using **`VOL1`** encounter data corruption.
2 Create volume **`GOODVOL1`** from snapshot **`VOL1.SNAPSHOT`**. The volume contains a pre-corruption data image of **`VOL1`**. Hosts recover lost information, for example by replaying logs, and continue processing data on **`GOODVOL1`**.
3 Destroy snapshot **`VOL1.SNAPSHOT`** when it is no longer required.
4 Start the process of reclaiming physical storage charged to **`VOL1.SNAPSHOT`**.

Example 15 (Cloning multiple volumes from a snapshot)

```
purevol snap --suffix TUESDAY VOL1 VOL2
```

... *main application continues using* **`VOL1`** *and* **`VOL2`**...

```
purevol copy Vol1.TUESDAY Vol1Backup
purevol copy Vol2.TUESDAY Vol2Backup
purevol connect --host HostBackup Vol1Backup Vol2Backup
```

... *backup starts* ...

```
purevol copy Vol1.TUESDAY Vol1Analytics
purevol connect --host HostAnalytics Vol1Analytics
```

... *analytics starts, using* `VOL1` *image only* ...

... *backup completes* ...

```
purevol disconnect --host HostBackup Vol1Backup Vol2Backup
purevol destroy Vol1Backup Vol2Backup
```

... *analytics completes* ...

```
purevol disconnect --host HostAnalytics Vol1Analytics
purevol destroy Vol1Analytics
purevol eradicate Vol1Analytics Vol1Backup Vol2Backup
```

1 Create snapshots **VOL1.TUESDAY** and **VOL2.TUESDAY**, representing the data on the respective volumes at a single instant in time. The main application continues to operate using **VOL1** and **VOL2**.
2 Create accessible volumes **Vol1Backup** and **Vol2Backup**.
3 Connect the two backup volumes to **HostBackup**, making it possible to back up the data images they contain.
4 (Assuming that the analytics application only requires data from **VOL1**.) Create volume **Vol1Analytics** from the same snapshot used to create **Vol1Backup**.

5 Connect **Vol1Analytics** to **HostAnalytics**, making it possible to analyze the data image from **VOL1**.
6 When backup application completes, disconnect the two backup volumes from **HostBackup** and destroy them.
7 When analytics application completes, disconnect **Vol1Analytics** from **HostAnalytics** and destroy the volume.
8 Eradicate the three volumes used by the ancillary backup and analytics applications, starting the process of reclaiming the space charged to them.

Example 16 (Destroying and Recovering Snapshots and Volumes)

```
purevol snap --suffix FirstSnap VOL1
purevol snap --suffix SecondSnap VOL1
purevol snap --suffix ThirdSnap VOL1
purevol snap --suffix FourthSnap VOL1
purevol destroy VOL1.FirstSnap
purevol eradicate VOL1.FirstSnap
purevol destroy VOL1.SecondSnap
purevol destroy VOL1
purevol recover VOL1
purevol recover VOL1.SecondSnap
```

1 Create snapshots of **VOL1** at four different points in time.
2 Destroy and eradicate **VOL1.FirstSnap**. The snapshot is no longer recoverable.
3 Destroy, but do not eradicate, **VOL1.SecondSnap**. After the command executes, the eradication pending period for the destroyed snapshot begins. During the eradication pending period, **VOL1.SecondSnap** can be recovered.
4 Destroy, but do not eradicate, **VOL1**, and implicitly, its two remaining snapshots, **VOL1.ThirdSnap** and **VOL1.FourthSnap**.
5 (Executes within the eradication pending period of **VOL1.ThirdSnap** and **VOL1.FourthSnap**.) Recover **VOL1** and the snapshots that were implicitly destroyed along with it. Recover **VOL1.ThirdSnap** and **VOL1.FourthSnap**, but not **VOL1.SecondSnap** (because it was destroyed separately), or **VOL1.FirstSnap** (because it was destroyed and

Example 17 (Refreshing Volume Contents from Snapshots and Volumes)

```
purevol snap --suffix SNAPSHOT VOL1
```

... applications continue to access VOL1...

```
purevol copy VOL1.SNAPSHOT VOL1Image
```

... update data format on VOL1Image...

```
purevol disconnect --host HOST1 VOL1
purevol copy --overwrite VOL1Image VOL1
purevol connect --host HOST1 VOL1
```

... restart applications using updated data format...

... detect problem with updated format...

```
purevol disconnect --host HOST1 VOL1
purevol copy --overwrite VOL1.SNAPSHOT VOL1
purevol connect --host HOST1 VOL1
```

... resume applications using older data format...

1 Create snapshot **`VOL1.SNAPSHOT`** of volume **`VOL1`**.
2 Create volume **`VOL1Image`**. Use it to update data format for applications that use **`VOL1`**.
3 Momentarily disconnect **`HOST1`** from its data, and overwrite said data with data in the updated format from volume **`VOL1Image`**. Applications detect problems with updated data format.
4 Momentarily disconnect **`HOST1`** from its volume, revert data on **`VOL1`** to previous format by copying from snapshot **`VOL1.SNAPSHOT`**, and reconnect **`VOL1`** to **`HOST1`** so that applications can resume processing using older format data.

Example 18

```
purevol tag --key owner --value asmith vol01
```

Creates a tag, specifying a **`key`**/**`value`** pair for the designated volume. Also used to modify the tag.

Example 19

```
purevol tag --key owner --value asmith vol01 --namespace Engineering
purevol tag --key owner --value asmith vol01 --namespace HR
```

Creates two tags on vol01, both of whom have a person with the ID, **`asmith`**. The two are differentiated by adding each of them to a separate namespace.

Example 20

```
purevol tag --key owner --value egivens --namespace Engineering --non-copyable vol05
```

Creates a tag and designates it as **`non-copyable`**, which prevents the tag from being copied during a **`purevol copy`** operation.

Example 21

```
$ purevol create vol01 --size 10G --priority-adjustment -10
```

Creates a volume with a lower performance priority.

Example 22

```
$ purevol create vol01 vol02 --size 10G --add-to-protection-group-names pg1,pg2
```

Creates volumes **`vol01`** and **`vol02`** and adds those volumes to the protection groups **`pg1`** and **`pg2`** in addition to each of the protection groups in the default protection group list (if any) for the root of the array. The protection groups **`pg1`** and **`pg2`** must already exist. Any volumes that are already members of those protection groups are not affected.

Example 23

```
$ purevol move --remove-from-protection-group-names pg, pg2 VOL01 POD01
```

Moves protected volume **`VOL01`** to **`POD01`** after removing protection groups pg1 and pg2.

Example 24

```
$ purevol move --add-to-protection-group-names pod::ratcheted-pg --remove-from-pro-
tection-group-names ratcheted-pg,unlocked-pg VOL01 POD01
```

Moves protected volume **`VOL01`** to **`POD01`** after removing the ratcheted protection group ratcheted-pg and the protection group unlocked-pg.

Example 25

```
$ purevol move VOL1 VOL2 pod1 --remove-from-protection-group-names pg1, pg2
```

Removes both **`VOL01`** and **`VOL02`** from whatever protection groups each is in from among the list pg1, pg2.

Example 26

```
purevol snap --allow-throttle --dry-run vol1
```

Assesses the internal health of the array and simulates taking a snapshot of the contents of **vol1** without generating a snapshot.

Example 27

```
purevol copy --allow-throttle VOL1.SNAPSHOT VOL1Image
```

Assesses the internal health of the array and proceeds only if array conditions support the volume copy operation. If conditions are correct, copies **VOL1.SNAPSHOT** to **VOL1Image**.

Example 28

```
purevol create --protocol-endpoint --container-version 1 PE1
```

Creates a protocol endpoint called **PE1** with a specified container version of 1.

Example 29

```
purevol vc-connection create --vchost vchost1 pod1::pe1
```

Creates a connection between vchost **vchost1** and protocol endpoint **pe1**, which is located in **pod1**.

Example 30

```
purevol vc-connection create --allow-stretched-multi-vchost --vchost vchost2 pod2::pe2
```

Creates a connection between vchost **vchost2** and protocol endpoint **pe2**, which is located in **pod2**. This endpoint is created even though a private connection already exists.

Example 31

```
purevol vc-connection list pod2:pe2
```

Lists all connections from vchosts to protocol endpoint **pe2**, which is located in **pod2**.

Example 32

```
purevol vc-connection delete --vchost vchost1 pod1::pe1
```

Deletes the connection between vchost **vchost1** and protocol endpoint **pe1**, which is located in **pod1**.

See Also

purevol-list, purevol-setattr
pureapp, purehgroup, purehgroup-connect, purehost, purehost-connect, purepod

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purevol-list

purevol-list, purevol-listobj – displays attributes of volumes and information about the virtual and physical storage they consume

Synopsis

```
purevol list [--all | --connect | --protect | --space] [--historical
{1h | 3h | 24h | 7d | 30d | 90d | 1y}] [--host-encryption-key] [--
namespace NAMESPACE] [--on REMOTE] [--pending | --pending-only] [--
pgrouplist PGROUPLIST] [--private | --shared] [--protocol-endpoint] [-
-qos] [--remote] [--snap] [--tags] [--transfer] [--total] [--cli | --
csv | --nvp] [--filter FILTER] [--limit LIMIT] [--notitle] [--page] [-
-page-with-token] [--token TOKEN] [--raw] [--sort SORT] [--workload
WORKLOAD] [--context REMOTES] [VOL...]
```

```
purevol listobj [--csv] [--pending | --pending-only] [--pgrouplist
PGROUPLIST] [--type {host | snap | vol}] [VOL...]
```

Arguments

NAMESPACE

Used to identify tags of a specified namespace when listing tags.

VOL

Volume for which the information specified by options is to be displayed

Options

Options that control information displayed:

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--all`

Displays names, virtual sizes, connected hosts and host groups, and host and array IQNs, NQNs, or WWNs for the specified volumes.

`--connect`

Displays hosts and host groups associated with the specified volumes, and the LUNs used by each to address them.

`--context` ***REMOTES***

Used to specify the remote array or arrays on which a command is processed. ***REMOTES*** is a comma-separated list of names of arrays within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--historical` ***TIME***

Used with **`purevol list --space`** to display historical size and space consumption or effective used capacity information over the specified range of time. Valid time ranges include: 1 hour, 3 hours, 24 hours, 7 days, 30 days, 90 days, and one year.

`--host-encryption-key`

Displays the host encryption key status for each volume or specified volumes.

`--namespace` ***NAMESPACE***

Used with **`purevol list --tags`** to display tags in the specified namespace.

`--on` ***REMOTE***

Used with **`purevol list --snap`** to display a list of volume snapshots on the specified offload target.

`--pending`

Includes destroyed volumes or snapshots that are in the eradication pending state. If not specified, volumes or snapshots that are pending eradication are not shown.

`--pending-only`

Only displays destroyed volumes or snapshots that are in the eradication pending state.

`--pgrouplist`

Used with `purevol list --snap` to only display volume snapshots that have been created as part of the specified protection group or protection group snapshot. Enter multiple protection group or protection group snapshot arguments in comma-separated format.

`--private`

Used with `purevol list --connect` to restrict the display to specified volumes' privately connected hosts. Invalid when combined with other options.

`--protect`

Displays protected volumes and their associated protection groups, except volumes protected through SafeMode volume protection.

`--protocol-endpoint`

Displays the virtual volumes used to form connections between FlashArray volumes and VMware ESXi hosts and host groups. For more information about virtual volumes, including configuration steps, refer to the Pure Storage vSphere Web Client Plugin for vSphere User Guide on the Knowledge site (https://support.purestorage.com) .

`--qos`

Displays the QoS information for each volume, including the bandwidth limit, IOPS limit, last priority adjustment, and current priority setting. If the bandwidth limit is not set, the value appears as a dash (-), representing unlimited throughput. If the IOPS limit is not set, the value appears as a dash (-), representing unlimited IOPS.

`--remote`

Used with `purevol list --connect` to include a list of remote volumes.

`--shared`

Used with `purevol list --connect` to restrict the display to specified volumes' shared connections. Invalid when combined with other options.

`--snap`

Displays information for snapshots of the specified volumes rather than for the volumes themselves.

`--space` (for purchased arrays)

Displays the following information about provisioned (virtual) size and physical storage consumption for each specified volume:

Provisioned Size

Total provisioned size of all volumes. Represents storage capacity reported to hosts.

Thin Provisioning

Percentage of volume sectors that do not contain host-written data because the hosts have not written data to them or the sectors have been explicitly trimmed.

Data Reduction

Ratio of mapped sectors within a volume versus the amount of physical space the data occupies after data compression and deduplication. The data reduction ratio does not include thin provisioning savings.

For example, a data reduction ratio of 5:1 means that for every 5 MB the host writes to the array, 1 MB is stored on the array's flash modules.

Total Reduction

Ratio of provisioned sectors within a volume versus the amount of physical space the data occupies after reduction via data compression and deduplication *and* with thin provisioning savings. Total reduction is data reduction with thin provisioning savings.

For example, a total reduction ratio of 10:1 means that for every 10 MB of provisioned space, 1 MB is stored on the array's flash modules.

Unique

Physical space that is occupied by data of both volumes and file systems after data reduction and deduplication, but excluding metadata and snapshots.

Snapshots

Physical space occupied by data unique to one or more snapshots.

Total

Total physical space occupied by system, shared space, volume, and snapshot data.

`--space` (on Evergreen//One™ subscription storage)

Displays capacity metrics reflecting storage consumption based on effective used capacity (EUC).

Name

Name of the volume.

Size

Effective used capacity available in the volume from a host’s perspective, including both consumed and unused storage.

Virtual

The amount of data that the host has written to the volume as perceived by the array, before any data deduplication or compression.

Unique

Effective used capacity data of volume after removing clones, but excluding metadata and snapshots.

Snapshots

Effective used capacity consumed by data unique to one or more snapshots.

Total

Total effective used capacity containing user data, including Shared, Snapshots, and Unique storage.

`--tags`

Displays all tags that have been attached to specified objects along with key/value pairs, namespace, and copyable status. If no namespace is specified, the **`purevol list`** command only displays tags with the `default` namespace. If one or more namespaces are specified, the **`purevol list`** displays all tags with the specified namespace(s).

`--total`

Follows output lines with a single line containing column totals in columns where they are meaningful.

`--transfer`

Used with the **`--snap`** option to display data transfer statistics, including data transfer start time, data transfer end time, data transfer progress, as well as the amount of logical and physical data transferred.

`--type`

Specifies the type of information (connected hosts, snapshots, or echoed volume names) about specified volumes to be produced in whitespace-separated list format suitable for scripting.

`--workload` ***`WORKLOAD`***

The name of the workload. Include the **`--workload`** option to display the workload the protection group is in. Include the **`WORKLOAD`** name to display the protection groups that are part of the specified workload.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--page-with-token`

Used to paginate output with a continuation token (**`--token TOKEN`**). Results are printed in `stdout` and the token is printed in `stderr`.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Options that manage display results:

`--filter`

Displays only the rows that meet the filter criteria specified.

`--limit`

Limits the size of the list output to the specified maximum number of rows.

`--sort`

Sorts the list output in ascending or descending order by the column specified.

See "Filtering" on page 14 and "Sorting" on page 13 for usage details and examples.

Description

The **`purevol list`** command displays a list of Purity//FA virtual storage volumes on the array. The list includes volume attributes such as volume (virtual) size, volume creation date, and volume serial number (generated by Purity//FA when the volume is created).

The **`purevol list`** command includes the mutually exclusive **`--all`**, **`--connect`**, **`--protect`**, **`--qos`**, and **`--space`** options, which are used to modify the display output of **`purevol list`**. The **`--all`** option displays volume connection details, including volume size, LUN number, and "paths" to connected hosts (host and corresponding array port IQNs, NQNs, or WWNs). The **`--connect`** option displays the hosts and host groups to which the specified volumes are connected, and the LUNs used by each to address them. The **`--protect`** option displays all protected volumes and their associated protection groups.

The **`--qos`** option displays the bandwidth limit, IOPS limit, last priority adjustment, and current priority setting. If the bandwidth limit is not set, the value appears as a dash (-), representing unlimited throughput. If the IOPS limit is not set, the value appears as a dash (-), representing unlimited IOPS.

The **`--space`** option displays information about provisioned (virtual) size and storage consumption or effective used capacity for each volume. The **`--remote`** option, which must be used with the **`--connect`** option, includes a list of remote volumes with host or host group connections.

The **`purevol list --historical`** command, which must be used with the **`--space`** option, displays historical size and storage consumption or effective used capacity details for each volume.

The **`purevol list --pending`** command displays a list of all volumes, including ones that have been destroyed and are in the *eradication pending* state. The **`purevol list --pending-only`** command only displays a list of volumes that have been destroyed and are in the *eradication pending* state.

The **`purevol list --total`** command displays the aggregate host-visible size of the volumes. The **`--total`** option cannot be run with the **`--all`** or **`--connect`** option because these options can conceivably display more than one output line per volume.

The **`purevol listobj`** command displays a list volume attributes, either in whitespace or comma-separated form, suitable for scripting. The **`--type`** option accepts the following arguments:

`--type host`
: Display a list of hosts to which the specified volumes are connected. If no volumes are specified, the list contains the names of all hosts connected to any volume.

`--type snap`
: Displays a list of volume snapshots, generated through volume or protection group snapshots, for the specified volumes. If no volumes are specified, the list contains the names of all volume snapshots.

`--type vol` (default if **`--type`** option not specified)
: Lists the specified volume names. If no volume names are specified, the list contains the names of all volumes. Snapshots are not included in the list.

Lists are whitespace-separated by default. Include the **`--csv`** option to produce a comma-separated list.

Use the `purevol list --workload` command to displays volumes that are part of a workload.

Displaying and Listing Snapshots

The `purevol list --snap` command displays a list of volume snapshots rather than volumes. Both volume and snapshot arguments can be included in the same command. The **`--snap`** option can be specified with the **`--pending`**, **`--pending-only`**, **`--space`**, and **`--transfer`** options.

The `purevol list --snap --pgrouplist` command only displays volume snapshots that are created as part of the specified protection group or protection group snapshot. Enter multiple protection group or protection group snapshot arguments in comma-separated format.

The **`--snap`** option is not valid with the **`--all`**, **`--connect`**, **`--private`**, or **`--shared`** options.

The `purevol listobj --type snap` command displays the names of volume snapshots generated through volume or protection group snapshots.

Displaying and Listing Snapshots on Offload Targets

The offload target feature enables system administrators to replicate point-in-time volume snapshots from the array to an external storage system, such as an S3 bucket. The `purevol list --snap --on` command displays a list of volume snapshots on an offload target. The `purevol get --on` command restores volume snapshots from an offload target. For more information about offload targets, refer to pureoffload.

Displaying Host Encryption Key Status

To enable deduplication on encrypted data, you need to configure the FlashArray to retrieve encryption keys from the Key Management Interoperability Protocol (KMIP) server. To show whether a volume has a host encryption key, use the **`purevol list --host-encryption-key`** command to display the host encryption status.

A volume can have one of the following encryption statuses:

- `none`: A volume is not encrypted.
- `detected`: A valid header (or key) is detected, but no fetched encryption key from the KMIP server.
- `fetched`: A valid header (or key) and a fetched encryption key from the KMIP server are detected.

The following example displays the host encryption key status of volumes `vol1`, `vol2`, and `vol3`.

```
$ purevol list --host-encryption-key
Name Status
vol1 none
vol2 detected
vol3 fetched
```

App Volumes

The Purity Run platform extends array functionality by integrating add-on services into the Purity//FA operating system. Each service that runs on the platform is provided by an app.

For each app that is installed, a boot volume is created. For some apps, a data volume is also created. Boot and data volumes are known as app volumes. Run the **`purevol list`** command to see a list of app volumes. Boot and data app volume names begin with a distinctive `@` symbol.

The naming convention for app volumes is `@APP_boot` for boot volumes and `@APP_data` for data volumes, where `APP` denotes the app name.

The following example displays the boot and data volumes for the `linux` app.

```
$ purevol list @linux*
Name Size Source Created Serial
  ...
@linux_boot 15G - 2016-11-09 15:13:37 MST AC97A330F2544A3C00011010
@linux_data 16T - 2016-11-09 15:14:17 MST AC97A330F2544A3C00011012
  ...
```

The boot volume represents a copy of the boot drive of the app. Do not modify or save data to the boot volume. When an app is upgraded, the boot volume is overwritten, completely destroying its contents including any other data that is saved to it. The data volume is used by the app to store data.

For more information about Apps, refer to pureapp.

Exceptions

None.

Examples

Example 1

```
purevol list
```

Displays names, sizes, and serial numbers of all volumes. Both directly connected hosts and hosts connected by virtue of their associations with host groups are displayed.

Example 2

```
$ purevol list @*
```

Displays a list of all app volumes on the array.

Example 3

```
purevol list --space --csv --notitle VOL1 VOL2 VOL3
```

On purchased arrays, displays names, sizes, data reduction ratios, and physical storage space occupied by volume data and RAID-3D check data for volumes `VOL1`, `VOL2`, and `VOL3`, in comma-separated value format.

On subscription storage, displays effective used capacity.

The **`--notitle`** option suppresses the line of column titles that would ordinarily precede the output.

Example 4

```
purevol list --space --historical 3h --csv --notitle VOL1 VOL2 VOL3
```

On purchased arrays, displays names, sizes, data reduction ratios, and physical storage space occupied by volume data and RAID-3D check data for volumes `VOL1`, `VOL2`, and `VOL3`, for the past 3 hours, in comma-separated value format.

On subscription storage, displays effective used capacity.

The **`--notitle`** option suppresses the line of column titles that would ordinarily precede the output.

Example 5

```
purevol list --connect --shared --nvp VOL4 VOL5
```

For volumes `VOL4` and `VOL5`, displays names, sizes, connected host groups and their associated hosts, and LUNs used to address the volumes, all in *name-value pair* (***`ATTRIBUTE-NAME=ATTRIBUTE-VALUE`***) format. Only shared connections are included.

Example 6

```
purevol list --connect --shared VOL1
```

Displays host groups to which `VOL1` has shared connections and the LUNs used by hosts in the groups to address it.

Example 7

```
purevol list --connect --remote
```

Displays a list of all volumes on both the local and remote arrays, their associated hosts or host groups, and the LUNs used by each volume to address the host or host group.

Example 8

```
purevol list --qos
```

Displays the QoS bandwidth limit, QoS IOPS limit, last priority adjustment, and current priority setting for each volume.

Example 9

```
purevol destroy VOL2
purevol destroy VOL3
purevol list --space --pending-only
```

Places `VOL2` and `VOL3` in the *eradication pending* state. Produces a space report of volumes that are pending eradication (including `VOL2` and `VOL3`). `VOL2` and `VOL3` continue to occupy space until the eradication pending period has elapsed or until the volumes have been manually eradicated by the **`purevol eradicate`** command.

Example 10

```
purevol list --snap --pgrouplist PG1.1
```

Produces a list of volume snapshots created for protection group snapshot `PG1.1`.

Example 11

```
purevol list --snap --pgrouplist PG1,PG2 V1
```

Displays a list of volume snapshots of `V1` that were created for all snapshots of protection groups `PG1` and `PG2`.

Example 12

```
purevol list --host-encryption-key vol1
```

Displays the host encryption key status of volume `vol1`.

Example 13

```
purevol list --all --filter "host = 'ESXi-IT-Cluster01-H0001'"
```

Displays a list of volumes that are directly connected to host `ESXi-IT-Cluster01-H0001`.

Example 14

```
purevol list --sort created
```

Displays a list of volumes that are sorted by date created in ascending order.

Example 15

```
purevol list --sort created-
```

Displays a list of volumes that are sorted by date created in descending order.

Example 16

```
purevol list --tags
```

Displays all tags in the `default` namespace. The `default` namespace is assigned to tags when no namespace is specified.

Example 17

```
purevol list --tags --namespace FOO
```

Displays all tags assigned to the `FOO` namespace.

Note: Make a note of any namespaces that you configure. There is currently no command that displays existing namespaces.

Example 18

```
purevol list --protocol-endpoint
```

Displays the virtual volumes used to form connections with VMware ESXi hosts and host groups. Includes the default protocol endpoint at the root of the array and also pod protocol endpoints, if any.

See Also

purevol and purevol-setattr

pureapp, purehgroup, purehgroup-connect, purehost, purehost-connect

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

purevol-setattr

purevol-setattr – manage QoS limits and performance priorities of volumes and configure volume sizes

purevol-truncate – increase and decrease volume sizes

Synopsis

```
purevol setattr [--bw-limit BANDWIDTH-LIMIT] [--container-version
VERSION][--iops-limit IOPS-LIMIT] [--priority-adjustment ADJUSTMENT]
[--size NEW-SIZE] [--workload WORKLOAD] [--context REMOTE] VOL...
```

```
purevol truncate --size NEW-SIZE [--context REMOTE] VOL...
```

Arguments

VOL

Name of the volume object whose size is to be changed.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--bw-limit` ***BANDWIDTH-LIMIT***

Sets a QoS bandwidth limit for the specified volume. Whenever throughput exceeds the bandwidth limit, throttling occurs. If set, the bandwidth limit must be between 1 MB/s and 512 GB/s. Like volume sizes, the bandwidth limit is specified as an integer, optionally followed by the suffix letter `M` (for megabytes) or `G` (gigabytes). To clear the bandwidth limit, giving the volume unlimited throughput, set to an empty string (`""`).

`--container-verison` ***VERSION***

Used with **`purevol create`** and **`purevol setattr`**. Specifies the version of the container with a protocol endpoint using the **`purevol`** command. This version is intended to ensure compatibility with specific vSphere versions or higher. The option allows users to choose from predefined version numbers, each corresponding to compatibility with different vSphere versions. Version 1 is compatible with vSphere version 7.0.1, version 2 with 8.0.0, version 3 with 8.0.1. Enter the version as just the number (e.g. "1").

`--context` ***REMOTE***

Used to specify the remote array on which a command is processed. ***REMOTE*** is an array within the same fleet as the current array. Note that this option only applies to arrays within a fleet. For more details on fleets, see "purefleet" on page 177.

`--iops-limit` ***IOPS-LIMIT***

Sets a QoS IOPS limit for the specified volume. Whenever the number of I/O operations exceeds the IOPS limit, throttling occurs. If set, the IOPS limit must be between 100 and 100`M`. The IOPS limit is specified as an integer, optionally followed by the suffix letter `K` (10^3) or `M` (10^6). To clear the IOPS limit, set to an empty string (`""`), giving the volume unlimited IOPS.

`--priority-adjustment` ***ADJUSTMENT***

Increases or decreases the performance priority of the specified volume or volumes. ***ADJUSTMENT*** can be `+10` or `-10`, to increase or decrease the priority, respectively, or `=10` (high), `=0` (default), or `=-10` (low), to set the priority values that cannot be changed by volume group priority.

See the "Setting Priority Adjustments" on page 643 section.

`--size`

Sets the virtual size for the specified volume.

`--workload` ***WORKLOAD***

The name of the workload. To disassociate a volume from a workload, the only valid value is an empty string (`""`).

Volume Sizes

Volume sizes are specified as an integer, followed by one of the suffix letters **`K`**, **`M`**, **`G`**, **`T`**, **`P`**, representing KiB, MiB, GiB, TiB, and PiB, respectively, where "Ki" denotes 2^10, "Mi" denotes 2^20, and so on.

Volumes must be between one megabyte and four petabytes in size. If a volume size of less than one megabyte is specified, Purity//FA adjusts the volume size to one megabyte. If a volume size of more than four petabytes is specified, the Purity//FA command fails.

Volume sizes cannot contain digit separators. For example, **`1000g`** is valid, but **`1,000g`** is not.

Description

The **`purevol setattr`** command changes the attribute values of existing volumes.

Configuring Volume Sizes

The **`purevol setattr --size`** command increases the provisioned size of a volume (virtual size as perceived by hosts). The **`purevol truncate`** command decreases the provisioned volume size. Changes in volume size are immediate and always affect the highest-numbered sector addresses.

Changes in volume size are visible to connected hosts immediately. When a volume is truncated, Purity//FA automatically takes an undo snapshot, providing a 24-hour window during which the previous contents can be retrieved. Note that the data in truncated sectors cannot be retrieved by increasing the size of a truncated volume. When a volume's size is increased, reading data in the new virtual sector addresses before they are written returns zeros.

Setting QoS Limits

Quality of service (QoS) bandwidth limits can be set on a per-volume basis to enforce maximum allowable throughput. The **`purevol setattr --bw-limit`** command sets the absolute bandwidth limit of a volume. Whenever throughput exceeds the bandwidth limit, throttling occurs. If set, the bandwidth limit must be between `1` MB/s and `512` GB/s. Like volume sizes, the bandwidth limit is specified as an integer, optionally followed by the suffix letter `M` (for megabytes) or `G` (gigabytes). To clear the bandwidth limit, giving the volume unlimited throughput, set to an empty string (`""`). Note that you cannot set a QoS bandwidth limit on an individual volume in a volume group.

Quality of service (QoS) IOPS limits can be set on a per-volume basis to enforce maximum allowable I/O operations per second. The **`purevol setattr --iops-limit`** command sets the absolute IOPS limit of a volume. Whenever the number of I/O operations per second exceeds the IOPS limit, throttling occurs. If set, the IOPS limit must be between 100 and 100`M`. The IOPS limit is specified as an integer, optionally followed by the suffix letter `K(10^3)` or `M (10^6)`. To clear the IOPS limit, specify an empty string (`""`), giving the volume unlimited IOPS.

In the following example, the **`purevol create`** command creates a volume named `vol01` of 10 gigabytes with the QoS bandwidth limit of 1 gigabyte and the IOPS limit of `5K`; the **`purevol setattr`** command changes the bandwidth limit to `2` gigabytes and the IOPS limit to unlimited IOPS.

```
$ purevol create vol01 --size 10G --bw-limit 1G --iops-limit 5K
Name Size Source Created Serial Bandwidth Limit(B/s) IOPS Limit
vol01 10G - 2019-05-20 16:24:58 PDT 3FBF2D29EE18492000011013 1G 5K
$purevol setattr vol01 --bw-limit 2G --iops-limit ""
Name Size Bandwidth Limit (B/s) IOPS Limit
vol01 10G 2G -
```

Setting Priority Adjustments

Use a priority adjustment to increase, decrease, or set the performance priority of the specified volume or volumes. The **`purevol setattr --priority-adjustment`** ***`ADJUSTMENT`*** command configures the priority adjustment of the specified volume or volumes.

The ***`ADJUSTMENT`*** value can be `+10` or `-10`, to increase or decrease the priority, respectively, or `+0` to set the priority adjustment to `0` (the default), , and those priorities can later be changed by inheriting volume group priorities.

`ADJUSTMENT` can also be `=10` (high), `=0` (default), or `=-10` (low), to set the priority values that cannot later be changed by volume group priority.

When used with `purevol setattr`:

- `+10` raises the priority from `0` to `+10` or from `-10` to `0`, and has no effect if the priority was already at `+10`.
- `-10` lowers the priority from `+10` to `0` or from `0` to `-10`, and has no effect if the priority was already at `-10`.
- `+0` sets the priority adjustment to `0`.

The following example increases the priority for the existing volumes `vol02` and `vol03`.

```
$ purevol setattr --priority-adjustment +10 vol02 vol03
Name Size Priority Adjustment Priority
vol02 6G   + 10                10
vol03 6G   + 10                10
```

See also the Priority Adjustment section in the "purevol" on page 590 chapter.

Adjusting Workloads

The **`purevol create`** command associates a new volume with a workload. To undo this, use the **`purevol setattr --workload`** command.

Exceptions

The following exceptions apply to the **`purevol setattr`** command:

The **`purevol setattr`** command cannot reduce the size of a volume, nor can **`purevol truncate`** increase the size of a volume.

The size of a volume cannot be reduced to less than one megabyte or increased to more than 4 petabytes. If a **`--size`** smaller than one megabyte is specified, Purity//FA changes the volume's size to one megabyte. If a **`--size`** greater than 4 petabytes is specified, the command fails.

Examples

Example 1

```
purevol setattr --size 1T VOL1 VOL2
```

Increases the sizes of `VOL1` and `VOL2` to 1 terabyte each. If either volume is larger than 1 terabyte, an error is logged, and that volume's size does not change.

Example 2

```
purevol setattr --size 1T VOL2
```

Increases the size of `VOL2` to 1 terabyte, assuming that the volume's current size is less than that amount.

Example 3

```
purevol setattr --bw-limit 50G VOL2
```

Sets the maximum QoS bandwidth limit of `VOL2` to 50 gigabytes.

Example 4

```
purevol setattr --iops-limit 5000 VOL2
```

Sets the maximum QoS IOPS limit of `VOL2` to 5000 IOPS.

Example 5

```
purevol setattr --iops-limit "" VOL2
```

Clears the maximum QoS IOPS limit set on `VOL2`.

Example 6

```
purevol setattr --priority-adjustment +10 VOL3
```

Raises the priority for `VOL3`.

If `VOL3` previously was at the default `0` priority, its priority is now set to `+10`.

If `VOL3` previously was at low `-10` priority, its priority is now set to `0`.

If `VOL3` previously was at high `+10` priority, its priority remains at `+10`.

Example 7

```
purevol setattr --priority-adjustment -10 VOL4
```

Lowers the priority for `VOL4`.

If `VOL4` previously was at high `+10` priority, its priority is now set to `0`.

If `VOL4` previously was at the default `0` priority, its priority is now set to `-10`.

If `VOL4` previously was at low `-10` priority, its priority remains at `-10`.

Example 8

```
purevol setattr --priority-adjustment =0 VOL5
```

Sets the priority for `VOL5` to `0`.

If any volumes on the array have the priority `+10`, `VOL5` does not receive performance priority.

If all other volumes on the array have the priority `-10`, then `VOL5` does receive performance priority.

Example 9

```
purevol setattr --priority-adjustment =10 VOL5
```

Sets the priority for `VOL5` to `10`.

`VOL5` receives performance priority unless all other volumes on the array also have the priority `+10`.

Example 10

```
purevol setattr --priority-adjustment =-10 VOL5
```

Sets the priority for `VOL5` to `-10`.

`VOL5` does not receive performance priority.

Example 11

```
purevol setattr --priority-adjustment =10  VOL6
purevol setattr --priority-adjustment =0   VOL7
purevol setattr --priority-adjustment =-10 VOL8
```

Sets the priority for `VOL6` to `10`.

Sets the priority for `VOL7` to `0`.

Sets the priority for `VOL8` to `-10`.

`VOL6` receives performance priority.

`VOL7` and `VOL8` do not receive priority and there is no difference in their priority treatment.

See Also

purevol and purevol-list,

purehgroup, purehgroup-connect, purehost, purehost-connect

Author

Pure Storage Inc. <documentfeedback@purestorage.com>

pureworkload

pureworkload, pureworkload-create, pureworkload-destroy, pureworkload-eradicate, pureworkload-list, pureworkload-recommendation, pureworkload-recover, pureworkload-rename, pureworkload-tag, pureworkload-untag - create and manage workloads

pureworkload-recommendation - provide placement recommendations for new workloads

Synopsis

pureworkload recommendation create --preset ***PRESET_NAME*** [--parameter ***PARAMETERS***] [--targets ***TARGETS***] [--projection-months ***PROJECTION_MONTHS***] [--results-limit ***RESULTS_LIMIT***] [--context ***REMOTE***]

pureworkload recommendation list [--results ***RECOMMENDATION***] [--projection-type {average, blended-max}] [--show-days-until-full] [--context ***REMOTES***] [--csv | --nvp] [--notitle] [--raw]

pureworkload create [--preset ***PRESET_NAME***] [--parameter ***NAME,VALUE***] [--context ***REMOTE***] ***WORKLOAD_NAME***

pureworkload list [--details] [--pending] [--pending-only] [--tags] [--namespace ***NAMESPACE***] [--csv | --nvp] [--notitle] [--page] [--raw] [--page-with-token] [--token ***TOKEN***] [--context ***REMOTES***] ***WORKLOAD_NAME***

pureworkload destroy [--context ***REMOTE***] ***WORKLOAD_NAME***

pureworkload recover [--context ***REMOTE***] ***WORKLOAD_NAME***

pureworkload rename [--context ***REMOTE***] ***OLD_NAME NEW_NAME***

pureworkload eradicate [--context ***REMOTE***] ***WORKLOAD_NAME***

pureworkload tag --key ***KEY*** --value ***VALUE*** [--namespace ***NAMESPACE***] [--non-copyable] [--context ***REMOTE***] ***WORKLOAD_NAME***

```
pureworkload untag --key KEY [--namespace NAMESPACE] [--context
REMOTE] WORKLOAD_NAME
```

Arguments

KEY
: Part of a key/value pair that is used to identify a tag.

NEW_NAME
: New workload name.

OLD_NAME
: Old workload name.

PRESET_NAME
: The workload preset name.

VALUE
: Part of a key/value pair that is used to identify a tag.

WORKLOAD_NAME
: The workload name.

Options

`-h | --help`

Can be used with any command or subcommand to display a brief syntax description.

`--context` ***REMOTE***

Used to perform actions on a remote array.

Used with the **`pureworkload recommendation create`** command to create a placement recommendation on a remote array.

Used with the **`pureworkload recommendation list`** command to display all placement recommendation requests on a remote array.

Used with the **`pureworkload create`** command to create a workload on a remote array.

Used with the **`pureworkload list`** command to display workload details of remote arrays.

For **`pureworkload destroy`**, destroys one or more workloads on a remote array.

For **`pureworkload recover`**, recovers one or more workloads on a remote array.

For **`pureworkload rename`**, renames a workload on a remote array.

For **`pureworkload eradicate`**, eradicates one or more workloads on a remote array.

For **`pureworkload tag`**, updates workload tags on a remote array.

For **`pureworkload untag`**, deletes one or more workload tags on a remote array.

`--details`

Used to show additional details of a workload.

`--key` ***`KEY`***

Tag key.

`--namespace` ***`NAMESPACE`***

Specify a namespace, or when used with **`pureworkload list`**, list tags in a specific namespace.

Used with the **`pureworkload list`** command, to display tags.

Used with the **`pureworkload tag`** command, to create a tag for a workload preset.

Used with the **`pureworkload untag`** command, to remove a tag from the workload preset.

`--non-copyable`

Specify if a tag is non-copyable.

`--parameter` ***`PARAMETERS`***

Parameter value.

Used with the **`pureworkload recommendation create`** and the **`pureworkload create`** command.

`--pending`

Include destroyed workloads pending eradication.

`--pending-only`

Only include destroyed workloads pending eradication.

`--preset` ***`PRESET_NAME`***

Specify the preset name.

Used with **`pureworkload recommendation create`** to specify

Used with **`pureworkload create`**.

`--projection-months` ***`PROJECTION_MONTHS`***

The projection months.

Used with **`pureworkload recommendation create`**, to specify the months.

`--projection-type`

Displays the results of a recommendation. Use the `average` or `blended-max` parameters when used with **`pureworkload recommendation list`**.

`--results` ***`RECOMMENDATION`***

Shows results of a recommendation.

Used with **`pureworkload recommendation list`**.

`--results-limit` ***`RESULTS_LIMIT`***

Maximal count of items in the recommendation result.

Used with **`pureworkload recommendation create`**, to specify the maximum count of items in the recommendation result.

`--tags`

List metadata tags.

`--targets` ***`TARGETS`***

Targets for workload placement.

Used with **`pureworkload recommendation create`**, to specify possible targets for workload placement.

`--value` ***`VALUE`***

Tag value.

Options that control display format:

`--cli`

Displays output in the form of CLI commands that can be issued to reproduce the current configuration. The **`--cli`** output is not meaningful when combined with immutable attributes.

`--csv`

Lists information in comma-separated value (CSV) format. The **`--csv`** output can be used for scripting purposes and imported into spreadsheet programs.

`--notitle`

Lists information without column titles.

`--nvp`

Lists information in name-value pair (NVP) format, in the form **`ITEMNAME=VALUE`**. Argument names and information items are displayed flush left. The **`--nvp`** output is designed both for convenient viewing of what might otherwise be wide listings, and for parsing individual items for scripting purposes.

`--page`

Turns on interactive paging.

`--page-with-token`

Used to paginate output with a continuation token (**`--token TOKEN`**). Results are printed in `stdout` and the token is printed in `stderr`.

`--raw`

Displays the unformatted version of column titles and data. For example, in the **`purearray monitor`** output, the unformatted version of column title **`us/op (read)`** is **`usec_per_read_op`**. The **`--raw`** output is used to sort and filter list results.

Description

Workloads are a set of storage objects created for a common purpose, to support a specific use case or application. The **`pureworkload`** command is designed to manage workloads. It is used to create, update, and delete workloads.

Managing a Workload

Use the **`pureworkload create`** command to create a new workload from a workload preset. Specify various parameters to customize the workload.

```
pureworkload create --preset preset-name --parameter "key1=val1,key2=val2"
```

Use the **`pureworkload list`** command to view details of one or more workloads. Include the --details option to display more comprehensive details.

The **`pureworkload destroy`** command destroys a workload, without immediately removing it from the system. The destroyed workload remains pending until it is eradicated using the **`pureworkload eradicate`** command or when the eradication timer expires. Once a workload is eradicated the action cannot be undone. Only a destroyed workload can be recovered using the **`pureworkload recover`** command.

Create or update tags for a workload with the **`pureworkload tag`** command. Use the **`pureworkload untag`** command to delete tags from a workload. Tags are key-value pairs that can be used to add metadata to workloads for efficient management.

Recommendations

Recommendations are supplementary objects designed to identify potential targets within the fleet for the purpose of generating new workloads based on a specified preset. To create recommendations for a new workload, use the **`pureworkload recommendation create`** command and specify parameters such as preset, targets, projection months, and results limit to tailor the recommendations.

To review the placement suggestions that have been generated, use the **`pureworkload recommendation list`** command, which provides options to show results, projection type, days until full, and output format. For example:

```
$ pureworkload recommendation list
Name         Created            Expires           Preset    Targets  Computation mode

Status

oracle_db.1 2024.09.24 13:54  2024.09.25 13:54 Oracle DB  fleet    local            com-

pleted

Projection months   Results limit

3                   10

mongodb.10 2024.09.24 13:54  2024.09.25 13:54 MongoDB   arrayA  cloud assisted  com-

pleted

12                  10

                                                        arrayB

                                                        arrayC

oracle_db.2 2024.09.24 13:54 2024.09.25 13:54 Oracle DB AZ1    best available in pro-

gress(70%)

6                  10
```

To return details on a single recommendation, use the **`--results`** option. The results of a single recommendation is sorted with the most recommended array at the top of the list. For example:

```
$ pureworkload recommendation list --results oracle_db.1
Rank Score     Target      Model     Usable capacity Baseline used capacity Projected used capacity
Baseline load Projected load (average) Warnings
60% -
1. Optimal     First Array  FA-X20R2 10TiB 35%                    50%                      50%
```

```
2. Acceptable SecondArray FA-X20R2 5TiB          40%                        60%
70%
80%
```

Examples

Example 1

```
pureworkload create --preset myfleet:mypreset --parameter coutn,2 --parameer billing_
id,examplebillingid workload1
```

Creates a new workload **`workload1`** from a workload preset.

Example 2

```
pureworkload destroy workload1
```

Destroys **`workload1`**, but is not immediately removed from the system.

Example 3

```
pureworkload list --tags workload1
```

Deprecated Commands

This chapter lists each of the Purity//FA CLI commands that are no longer supported in Purity. For a detailed description of each command, please reference a Purity CLI user guide from a version before that which is listed in the "Deprecated Version" column.

Command	Subcommands	Deprecated Version	Reason for Deprecation
pureplugin	--install	6.4.3	The pureplugin command was used specifically for the VMWare vSphere local plugin. In vSphere versions 8.0 and above, local plugins are no longer supported. Instead, only remote plugins are supported. For more detailed reasons about the switch from the local plugin to the remote plugin, see this article: https://support.purestorage.com/Solutions/VMware_Platform_Guide/Release_Notes_for_VMware_Solutions/Why_Pure_Storage_is_Deprecating_the_Local_vSphere_Plugin_in_Favor_of_the_Remote_vSphere_Plugin. For step-by-step instructions on installing the remote plugin, see this article: https://support.purestorage.com/Solutions/VMware_Platform_Guide/User_Guides_for_VMware_Solutions/Using_the_Pure_Storage_Plugin_for_the_vSphere_Client/vSphere_Plugin_User_Guide%3A_Installing_the_Remote_vSphere_Plugin_with_the_Pure_Storage_VMware_Appliance.
puremessage	<every subcommmand>	6.8.0	The functionality that used to be handled by `puremessage` is now handled instead by the `purealert`, `pureaudit`, and `puresession` commands.

CLI Getting Started

FlashArrays present disk-like volumes to hosts. From the Purity//FA standpoint, volumes and hosts are abstract objects which must be created by an administrator and connected to each other before actual host computers can write and read data on volumes.

Creating Volumes

Once a FlashArray is installed and initially configured, an administrator's typical first task is creation of volumes that hosts will subsequently format and use to store and access data. " Creating Volumes" below illustrates three examples of volume creation using the CLI.

Figure 55-1. Creating Volumes

```
$ purevol create vol1 1
usage: purevol create [-h] --size SIZE VOL ...
purevol create: error: argument --size is required
$ purevol create vol1 --size 100g 2
Name Size Serial
vol1 100G 7CBFCE0B2660CC3000010001
$ purevol create vol2 vol3 vol4 --size 200g 3
Name Size Serial
vol2 200G 7CBFCE0B2660CC3000010002
vol3 200G 7CBFCE0B2660CC3000010003
vol4 200G 7CBFCE0B2660CC3000010004
```

1 Command fails, because no virtual size is specified for the volume.

2 Creates a volume that hosts will perceive as containing 100 gigabytes of storage, although it occupies almost no physical space until hosts write data to it.

3 Creates three volumes in a single command. There is no practical limit to the number of volumes that can be created in a single command, but only one `--size` option can be specified in the command, so all volumes initially have the same virtual size.

The output of the `purevol create` command lists the name, size, and Purity//FA-generated serial number of each volume created.

Creating Hosts

Internally, Purity//FA represents host computers that use FlashArray storage services as host objects (usually just called "hosts"). A Purity//FA host object has a name and one or more iSCSI Qualified Names (IQNs), NVMe Qualified Names (NQNs), or Fibre Channel World Wide Names (WWNs). IQNs, NQNs, and WWNs correspond to initiators, typically host computers or host computer ports with which a FlashArray exchanges commands, data, and messages. Purity//FA represents a host's IQNs, NQNs, and WWNs as a multi-valued attribute, specified via either the **`--iqnlist`** (alias **`--iqn`**), **`--nqnlist`** (alias **`--nqn`**), or **`--wwnlist`** (alias **`--wwn`**) option when hosts are created.

iSCSI IQNs conform to Internet fully-qualified domain name (FQDN) standards. NVMe NQNs conform to the naming standards set by NVM Express. Fibre Channel WWNs consist of exactly 16 hexadecimal digits, and may be specified either as continuous strings or as eight digit pairs separated by colons. "Creating Hosts" below illustrates examples of host creation.

Figure 55-2. **Creating Hosts**

```
$ purehost create host1 1
Name   WWN
host1 -
$ purehost create --wwnlist 0123456789abcde2 host2 2
Name   WWN
host2 01:23:45:67:89:AB:CD:E2
$ purehost create --wwnlist 0123456789abcde3,01:23:45:67:89:ab:cd:e4 host3 3
Name   WWN
host3 01:23:45:67:89:AB:CD:E3
    01:23:45:67:89:AB:CD:E4
$ purehost create --wwn 01:23:45:67:89:ab:cd:e4 host4 4
Error on host4: The specified WWN is already in use.
```

1 Creates host object `host1`. At the time of creation, `host1` cannot connect to volumes because no IQNs, NQNs, or WWNs are associated with it.

2 Creates `host2` and assigns WWN `0123456789abcde2` to it. Volumes can be connected to `host2` immediately.

3 Creates `host3` and associates two WWNs with it (typically because the host is equipped with two Fibre Channel ports). This example also illustrates the two forms in which world wide names can be entered--continuous digit string and colon-separated two-digit groups.

4 Attempts to create `host4` and fails, because the WWN specified is already associated with another host. Because they identify host ports on the storage network, WWNs must be unique within an array.

Connecting Hosts and Volumes

For a host to read and write data on a volume, a connection between the two must be established. A connection is effectively permission for an array to respond to I/O commands from a given set of storage network addresses (host IQNs, NQNs, or WWNs). In SCSI terminology, a connection is an *Initiator-Target-LUN* (I_T_L) nexus, a completely specified path for message and data exchange. By default, Purity//FA chooses a LUN based on availability at both the initiator (host) and target (array) when a connection is established. Alternatively, an administrator can specify an unused LUN when establishing a connection.

The **`purehost connect`** command establishes private connections between a single volume and one or more hosts. The **`purevol connect`** command establishes connections between a single host and one or more volumes. The two commands are functionally identical. Both commands are provided for administrative convenience. "Establishing Host-Volume Connections" below illustrates the use of both commands.

Figure 55-3. Establishing Host-Volume Connections

```
$ purehost connect --vol vol1 host1 host2 host3 1
Name   Vol   LUN
host1 vol1 1
host2 vol1 1
host3 vol1 1
```

```
$ purevol connect --host host1 vol2 vol3 vol4 2
Name Host Group Host  LUN
vol2 -          host1 2
vol3 -          host1 3
vol4 -          host1 4
$ purevol connect --lun 249 --host host1 vol5 3
Name Host Group Host  LUN
vol5 -          host1 249
$ purehost disconnect --vol vol1 host2 4
Name  Vol
host2 vol1
$ purevol disconnect --host host3 vol1 5
Name Hgroup Host
vol1        host3
```

1 Connects `vol1` to hosts `host1`, `host2`, and `host3`.

2 Connects `host1` to volumes `vol2`, `vol3`, and `vol4`.

3 Connects `host1` to `vol5`, manually associating LUN `249` with the connection. Any available LUNs in the range [`1...4095`] can be used for private or shared connections.

4, 5 **`The purehost disconnect`** or **`purevol disconnect`** commands in the example break the private connections between `host2` and `vol1`, and between `host3` and `vol1` respectively. Either can be used to break a private connection between a host and a volume, regardless of how the connection was established. As with the connect subcommands, **`purehost disconnect`** can disconnect multiple hosts from a single volume, and **`purevol disconnect`** can disconnect multiple volumes from a single host. The two subcommands are provided for administrative convenience.

Whichever way a connection is made, the host administrator can immediately rescan the host's flash modules (if necessary) to discover the logical units represented by newly-connected volumes, create file systems, and mount them for storing and retrieving data.

Resizing Volumes

FlashArray administrators also perform routine management tasks on objects within an array. Typically the most frequent of these are operations on volumes. You can change the size (host-visible storage capacity) of a volume, disconnect it from hosts, rename it, destroy it, eradicate all or part of the data it contained, and under certain circumstances, restore the contents of a destroyed volume or truncated sector range.

To minimize the possibility for inadvertent errors, the CLI uses different subcommands for increasing and decreasing the host-perceived sizes of volumes. The `purevol setattr` command increases the sizes of one or more volumes to a given level; the `purevol truncate` command decreases volumes' sizes. Both commands can operate on multiple volumes, but `purevol setattr` can only increase volume sizes, and `purevol truncate` can only decrease them.

"Resizing Volumes" below illustrates the use of these commands to change volumes' sizes.

Figure 55-4. **Resizing Volumes**

```
$ purevol list vol1 vol2 vol3 vol4
Name Size Serial
vol1 100G 7CBFCE0B2660CC3000010001
vol2 200G 7CBFCE0B2660CC3000010002
vol3 200G 7CBFCE0B2660CC3000010003
vol4 200G 7CBFCE0B2660CC3000010004
$ purevol setattr --size 300g vol1 vol2 1
Name Size Serial
vol1 300G 7CBFCE0B2660CC3000010001
vol2 300G 7CBFCE0B2660CC3000010002
$ purevol truncate --size 100g vol3 vol4 2
Name Size Serial
vol3 100G 7CBFCE0B2660CC3000010003
vol4 100G 7CBFCE0B2660CC3000010004
$ purevol setattr --size 250g vol2 vol3 3
Name Size Serial
vol3 250G 7CBFCE0B2660CC3000010003
```

```
Error on vol2: Implicit truncation not permitted.
$ purevol list vol2 vol3
Name Size Serial
vol2 300G 7CBFCE0B2660CC3000010002
vol3 250G 7CBFCE0B2660CC3000010003
$ purevol truncate --size 275g vol2 vol3 4
Name Size Serial
vol2 275G 7CBFCE0B2660CC3000010002
Error on vol3: Target size exceeds volume size.
$ purevol list vol2 vol3
Name Size Serial
vol2 275G 7CBFCE0B2660CC3000010002
vol3 250G 7CBFCE0B2660CC3000010003
```

1 Increases the sizes of `vol1` and `vol2` from 100 gigabytes and 200 gigabytes respectively to 300 gigabytes.

2 Reduces the sizes of `vol3` and `vol4` from 200 gigabytes to 100 gigabytes.

3 Attempts to change the sizes of `vol2` and `vol3` to 250 gigabytes. The command succeeds for `vol3`, but fails for `vol2` because it is already larger than 250 gigabytes.

4 Attempts to change the sizes of `vol2` and `vol3` to 275 gigabytes. The command succeeds for `vol2`, but fails for `vol3` because it is already smaller than 275 gigabytes (the “target size”).

Increasing a volume's host-perceived size does not increase its consumption of physical storage. Volumes only consume storage when hosts write data to previously unused volume block addresses or overwrite blocks with less-compressible data.

When a volume is truncated, Purity//FA automatically takes an *undo snapshot* (providing a 24-hour window during which the previous contents can be retrieved). If the physical storage occupied by data in truncated sectors is urgently required, the eradication pending period can be terminated, and space reclamation initiated immediately by calling **`purevol eradicate`** on the undo snapshot.

Destroying Volumes

When the data stored in a volume is no longer of interest, the **purevol destroy** command obliterates the data it contains and frees the storage it occupies. Destroyed volumes become invisible to hosts and to most CLI and GUI displays immediately, but their data is not obliterated, nor is the storage it occupies reclaimed until the *eradication pending* period has elapsed. During this eradication pending period, a destroyed volume can be recovered with its data intact. The **purevol eradicate** command terminates the eradication pending period and begins storage reclamation immediately. Once eradication has begun, a volume can no longer be recovered.

The length of the eradication pending period is controlled by the applicable eradication delay setting. See the "Eradication Delays" on page 88 section in the `purearray` chapter.

"Destroying Volumes" below illustrates the use of the **purevol destroy** command.

Figure 55-5. **Destroying Volumes**

```
$ purevol list --connect vol1 1
Name Size LUN Hgroup Host
vol1 100G 1          host1
$ purevol destroy vol1 2
Error on vol1: Cannot destroy volume because it is currently connected.
$ purevol disconnect --host host1 vol1
Name Hgroup Host
vol1        host1
root@gt2-ct0:~# purevol destroy vol1 3
Name
vol1
$ purevol list 4
Name Size Serial
vol2 200G 27CA508DA38AF9F400010001
vol3 300G 27CA508DA38AF9F400010002
vol4 300G 27CA508DA38AF9F400010003
$ purevol list --pending 5
Name Size Time Remaining Source Created                Serial
```

```
vol1 100G 1 day, 0:00:00 -      2013-04-01 15:45 PDT 27CA508DA38AF9F400010000
vol2 100G -              -      2013-04-01 15:46 PDT 27CA508DA38AF9F400010001
vol3 100G -              -      2013-04-01 15:47 PDT 27CA508DA38AF9F400010002
vol4 100G -              -      2013-04-01 15:48 PDT 27CA508DA38AF9F400010003
```

1 As background, illustrates that `vol1` is connected to `host1`

2 Attempt to destroy `vol1` fails, because it has a connection.

3 After `vol1`'s connection to `host1` is broken, the **`purevol destroy`** command succeeds.

4, 5 Illustrates that a volume in the eradication pending state (`vol1`) is only displayed by the **`purevol list`** command, when the **`--pending`** option is specified.

Recovering and Eradicating Destroyed Volumes

To provide a margin of error for administrators, Purity//FA retains a volume's contents and structure for an eradication pending period after it is destroyed. The **`purevol recover`** command restores a destroyed volume to its size at the time of destruction with its contents intact, as "Recovering and Eradicating Volumes" below illustrates.

Figure 55-6. **Recovering and Eradicating Volumes**

```
$ purevol destroy vol1
Name
vol1
$ purevol list --pending vol1
Name Size Source Time Remaining  Created              Serial
vol1 100G -      1 day, 0:00:00  2013-04-12 15:45 PDT 27CA508DA38AF9F400010004
$ purevol recover vol1 1
Name
vol1
$ purevol list vol1
Name Size Source  Created              Serial
vol1 100G -       2013-04-12 15:45 PDT 27CA508DA38AF9F400010004
$ purevol destroy vol2
```

```
Name
vol2
$ purevol eradicate vol2 2
Name
vol2
$ purevol recover vol2 3
Error on vol2: Volume does not exist.
```

1 Restores destroyed volume `vol1` (provided that the command executes within the volume's eradication pending period).

2 Obliterates the data from destroyed volume `vol2`, and begins reclamation of the storage it had occupied.

3 Fails to recover `vol2`, illustrating that once it is eradicated, a volume can no longer be recovered.

Renaming Volumes

Volume names identify volumes in CLI and GUI commands and displays. Volume names are assigned at creation, and, as with other objects can be changed at any time by the `purevol rename` command, as "Renaming Volumes" below illustrates. Renaming a volume has no effect on its attributes or connections.

Figure 55-7. Renaming Volumes

```
$ purevol rename vol4 NewVol4
Name
NewVol4
$ purevol list vol4
Error on vol4: Volume does not exist.
$ purevol list NewVol4
Name    Size Source Created                 Serial
NewVol4 300G -      2013-04-12 15:48 PDT 27CA508DA38AF9F400010003
```

The name of `vol4` is changed to `NewVol4`. The **`purevol list`** commands that follow illustrate that after renaming, a volume is immediately visible under its new name, and is no longer visible under its former name. All attributes, including the volume serial number, which is visible to hosts, remain unchanged.

Ongoing Administration of Hosts

Purity//FA host administration includes the following tasks:

- Creation, deletion and renaming of host objects
- Connecting hosts to and disconnecting them from volumes
- Changing the IQNs, NQNs, or WWNs associated with a host
- Adding hosts to and removing them from host groups.

"Creating Hosts" on page 657 illustrates host creation.

"Establishing Host-Volume Connections" on page 658 illustrates both connection of multiple volumes to a single host (**`purevol connect`**) and multiple hosts to a single volume (**`purehost connect`**).

"Administrative Operations on Hosts" below illustrates the remaining host operations.

Figure 55-8. **Administrative Operations on Hosts**

```
$ purehgroup listobj --type host hgroup1 1
host1
host2
$ purehost list --connect host1 2
Name  LUN Vol Host Group
host1 1   vol1
$ purehost delete host1 3
Error on host1: Could not delete host.
$ purehgroup setattr --remhost host1 hgroup1
Name    Hosts
hgroup1 host1
$ purehost delete host1 4
Error on host1: Could not delete host.
```

```
$ purehost disconnect --vol vol1 host1 5
Name Vol
host1 vol1
$ purehost delete host1
Name
host1
```

1, 2 These commands illustrate that `host1` is associated with host group `hgroup1`, and has a private connection to `vol1`.

3 Attempt to delete `host1` fails because it is associated with a host group.

4 After `host1` is removed from `hgroup1`, the attempt to delete it still fails because it is still connected to volume `vol1`.

5 Deletion of `host1` succeeds because it is not associated with a host group and has no private connections to volumes.

Monitoring I/O Performance

The **`purevol monitor`** command produces periodic displays of I/O performance information for some or all of an array's volumes, as "Monitoring I/O Performance" below illustrates. Use **`purehost monitor`**, **`purehgroup monitor`**, and **`purearray monitor`** commands to display I/O performance information for host, host group, or the whole array.

Figure 55-9. **Monitoring I/O Performance**

```
$ purevol monitor --interval 1 --repeat 3 --total
Name Time     B/s(read) B/s(write) op/s(read) op/s(write) us/op(read) us/op(write)
vol1 13:23:19 3.16M     1.51M      808        386         119         155
vol2 13:23:19 1.53M     1.58M      392        405         93          156
total13:23:19 4.69M     3.09M      1K         791         110         156
Name Time     B/s(read) B/s(write) op/s(read) op/s(write) us/op(read) us/op(write)
vol1 13:23:20 3.23M     1.47M      828        376         117         154
vol2 13:23:20 1.57M     1.63M      401        417         92          153
total13:23:20 4.80M     3.10M      1K         793         109         154
Name Time     B/s(read) B/s(write) op/s(read) op/s(write) us/op(read) us/op(write)
Name  Time     B/s(read) B/s(write) op/s(read) op/s(write) us/op(read) us/op(write)
Name  Time     B/s(read) B/s(write) op/s(read) op/s(write) us/op(read) us/op(write)
vol1  13:23:21 3.05M     1.55M      780        398         119         155
vol2  13:23:21 1.57M     1.53M      401        391         91          157
total13:23:21 4.61M     3.08M      1K         789         109         156
```

The `--interval` option specifies the number of seconds between displays of data. The `--repeat` option specifies the number of displays that the command produces. The default interval is 5 seconds; the default repetition count is 1.

Displays read and write operations and data transfer performance for volumes `vol1` and `vol2` with totals. Three sets of results are produced (`--repeat 3`) at intervals of one second (`--interval 1`).

Getting Started with File Storage

Contact Pure Storage Technical Services to have file services activated on the FlashArray.

FlashArray file services present file systems to clients through file exports (that is, shares). From the Purity//FA standpoint, file systems, managed directories, and exports are abstract objects that must be created by an administrator and connected before actual client computers can write and read data.

Creating a Network Interface for File

Before clients can connect to and use the file storage, there must be a virtual interface in place for the `File` service. Create a virtual interface by using the `purenetwork` command or open the Purity//FA GUI and go to the **Settings > Network** page. Use the `purenetwork list` command for a list of available interfaces. "Creating a Network Interface For File" below illustrates an example of network interface creation for file services using the CLI.

Figure 55-10. **Creating a Network Interface For File**

```
$ purenetwork eth create vif --subinterfacelist ct0.eth4,ct0.eth5,ct1.eth4,ct1.eth5
filevip 1
$ purenetwork eth setattr --address 192.168.20.211 --netmask 255.255.255.0
                      --gateway 192.168.20.1 filevip 2
```

```
$ purenetwork eth enable filevip 3
```

1 Creates a floating virtual interface named filevip, using four interfaces on two controllers, automatically adding the file service to the interface.

2 Sets an IP address, netmask, and optionally a gateway for the interface.

3 Enables the virtual interface.

Joining Directory Services

Clients can use the file system once the storage is joined to a directory service such as Active Directory (AD) or LDAP. Another approach is to use local users for file services, a directory of users and groups locally stored on the array. After the file system is joined or connected to a directory service, or local users and groups are created, clients can mount and access shares according to the defined authorization.

For AD or LDAP, make sure that the following is in place before proceeding:

1. There is an AD or LDAP server prepared and accessible from the FlashArray.
2. The network has been set up so that FlashArray can connect with the server.

For configuration of AD or Kerberos, use the `puread account` command. For LDAP, use the `pureds` command, and for local users, the `pureds local` command. For AD and LDAP through the Purity//FA GUI, go to the **Settings > Access** page.

"Active Directory" below illustrates an example of joining the array file services to an Active Directory domain, while "LDAP" below illustrates an example of preparing the connection to an LDAP server.

Figure 55-11. **Active Directory**

```
$ puread account create --domain ad1.local file-fa-1
```

Joins the array file services to the domain named `ad1.local` under the name `file-fa-1`.

Figure 55-12. **LDAP**

The following example represents a typical command line using OpenLDAP:

```
$ pureds setattr data --base-dn dc=mydomain,dc=com --bind-password
--bind-user cn=admin,dc=mydomain,dc=com
--uri ldap://192.168.20.2
```

Prepares the connection of the FlashArray file services to an LDAP server, using the *data* service. The base-dn used here is an example. The bind user (here: admin) is the admin user that exists on the LDAP server. The **`--bind-password`** option must be included to prompt for the bind user password which is then set for the connection. The **`--uri`** option sets the URI of the directory server.

LDAP is not activated until the service is enabled as shown in the following example:

```
$ pureds enable data
```

Activates the use of LDAP directory service, allowing users in the LDAP directory to log in to the file services.

Creating a File System

Once the FlashArray file service is activated, the first task is to create at least one file system. Creating a file system also creates the root directory. The root directory, which is automatically created, is a managed directory named `root` that cannot be destroyed by itself.

To create a file system, you can either go to the **Storage > File Systems** page or use the **`purefs`** command.

"Creating a File System" below illustrates an example of file system creation using the CLI.

Figure 55-13. **Creating a File System**

```
$ purefs create FS1
```

Creates a file system named `FS1`.

Creating a Managed Directory

Create managed directories by using the `puredir` command or go to the **Storage > File Systems** page.

"Creating a Managed Directory" below illustrates an example of managed directory creation using the CLI.

Figure 55-14. **Creating a Managed Directory**

```
$ puredir create --path /data FS1:Data
```

Creates a managed directory `data`, named `Data`, on the file system `FS1`.

Creating an Export

Exports are entry points for clients to connect to the file system. When granted access, clients only see the file system from the point of the export, including its subdirectories.

An export is created by adding a policy to a managed directory. The policy, which has one or more rules added to it, can be reused for other managed directories. Export policies can be created for the SMB protocol or for the NFS protocol, and both types of policies can be used for the same directory if needed.

Note that predefined export policies may exist, which can be used to create SMB and NFS exports. These policies should be reviewed and updated according to actual requirements before use. See the GUI or run the `purepolicy smb rule list` or the `purepolicy nfs rule list` command, respectively, for a view of all available SMB and NFS policy rules.

To create a policy, go to the **Storage > Policies** page, or use the `purepolicy` command:

- **`purepolicy smb/nfs create`** creates the policy.
- **`purepolicy smb/nfs rule add`** adds an export rule to the policy.

To create an export, go to the **Storage > Policies** page, or use the `puredir` command:

- **`puredir export create`** adds the export policy to the directory.

Note that the two exports in the following two examples are allowed to share the same name, since SMB and NFS exports reside in different namespaces.

"SMB" below illustrates an example of SMB policy and export creation using the CLI, while "NFS" on the next page illustrates an example of NFS policy and export creation.

Figure 55-15. **SMB**

```
$ purepolicy smb create smb_all 1
$ purepolicy smb rule add --anonymous-access-allowed smb_all 2
$ puredir export create --dir FS1:Data --policy smb_all DataShare 3
```

1 Creates an SMB policy named `smb_all`.

2 Adds a rule to the policy, allowing anonymous access to all users.

3 Adds the policy to a managed directory `Data`, creating an SMB export named `DataShare`.

Figure 55-16. NFS

```
$ purepolicy nfs create nfs_all 1
$ purepolicy nfs rule add --root-squash nfs_all --anonymous-access smb_all 2
$ puredir export create --dir FS1:Data --policy nfs_all DataShare 3
```

1 Creates an NFS policy named `nfs_all`.

2 Adds a rule to the policy, allowing anonymous access for root users (the default) and allowing both read and write requests. NFS version 3 is the default. To allow version 4.1, add **`--version nfsv4`** before the policy name, or **`--version nfsv3,nfsv4`** to allow both version 3 and 4.1.

3 Adds the policy to a managed directory `Data`, creating an NFS export named `DataShare`.

Connecting a Client

The client should now be able to connect to an export on the file system. For SMB, connect to an export by mapping to a network drive similar to this: `\\server\share`

For NFS, mount the file system as shown in the following client side Linux command:

```
$ mount -t nfs -o nfsvers=3 server:/export_name mount_point
```

Replace `server` with the IP address or URL for the file service. Replace `export_name` with the export name, for example `DataShare`, and replace `mount_point` with the client side path. Use **`nfsvers`** to specify the NFS protocol version, `3` or `4` (4.1), especially in cases where both versions are enabled on the FlashArray.

Creating Scheduled Snapshots

Snapshots can be scheduled and created automatically through snapshot policies. First create a snapshot policy with one or more rules, and then attach the policy to one or more managed directories. The resulting snapshots will be located in a hidden `.snapshot` directory. Snapshots are also available for the user through SMB Previous Versions.

Multiple directories can be protected using one or many snapshot policies.

Note that predefined snapshot policies may exist and can be used to create scheduled snapshots. See the GUI or run the `purepolicy snapshot rule list` command for details. These policies should be reviewed and updated according to actual requirements before use.

To create a snapshot policy and attach it to directories, go to the **Storage > Policies** page, or use the `purepolicy` and `puredir` commands:

- **`purepolicy snapshot create`** creates the snapshot policy.
- **`purepolicy snapshot rule`** `add` adds one or more rules to the policy.
- **`puredir snapshot add`** adds the policy to the directory.

"Creating Scheduled Snapshots" below illustrates an example of scheduled snapshot creation using the CLI.

Figure 55-17. **Creating Scheduled Snapshots**

```
$ purepolicy snapshot create data_snap 1
$ purepolicy snapshot rule add --every 1d --at 9pm --keep-for 3w

                               --client-name daily data_snap 2

$ puredir snapshot add --policy data_snap FS1:Data 3
```

1 Creates a snapshot policy named `data_snap`.

2 Adds a rule to the snapshot policy notifying the scheduler to make a snapshot every day at 9 p.m. and keep each for three weeks. With the **`--client-name`** option, the client visible portion of the snapshot is being named `daily` and with a counter added to the name.

3 Adds the snapshot policy to the managed directory named `Data` on `FS1`.