Dell EMC BoostFS for Linux

Version 1.2

Configuration Guide

302-003-987
REV. 03

DELL EMC

Published July 2018

Dell EMC
Hopkinton, Massachusetts 01748-9103
1-508-435-1000 In North America 1-866-464-7381
www.DellEMC.com

CONTENTS

FIGURES

TABLES

CHAPTER 1

Introduction to BoostFS for Linux

Revision history

The following table presents the revision history of this document.

Table 1 Revision history of BoostFS for Linux Configuration Guide, version 1.2

Revision	Date	Description
03 (1.2)	July 2018	This revision contains updates to the Mount Options and Automounter sections.
02 (1.1)	January 2018	This revision contains an update to the Supported Environments section.
01 (1.1)	June 2017	This revision contains information about these new features and tools: • Adds information about the new shared lockbox feature, which allows you to create a common lockbox file for all BoostFS clients rather than a separate lockbox file for each unique BoostFS client. • Adds four new commands to manage lockbox access. • Adds information about the `_netdev` option you can use if BoostFS fails to mount after a reboot.

Introduction to BoostFS

Data Domain Boost Filesystem (BoostFS) 1.2 provides a general file-system interface to the DD Boost library, allowing standard backup applications to take advantage of DD Boost features.

Advantages of BoostFS

By leveraging the DD Boost technology, BoostFS helps reduce bandwidth, can improve backup-times, offers load-balancing, allows in-flight encryption, and supports the Data Domain multi-tenancy feature set.

As a file server system implementation, the BoostFS workflow is similar to NFS but also leverages the DD Boost protocol. In addition, BoostFS improves backup times compared to NFS and various copy-based solutions.

BoostFS supports single-node Data Domain systems, high-availability (HA) systems, Extended Retention systems, Data Domain Virtual Edition, and Extended Distance Protection.

Purpose

This document describes how to install and configure BoostFS on client systems.

Terminology

Term	Definition
FUSE	Filesystem in User Space (FUSE) is an open-source interface that enables non-privileged users to securely create and mount their own file-system implementations.
Puppet	An open-source software configuration management tool. For more information, see About Puppet on page 48.
Push	A process that involves using a centralized server to connect to specified clients and run commands remotely; for BoostFS, that means downloading and remotely installing the installation package on each client.

Supported environments

Environments that use BoostFS 1.2 must meet the following specifications.

BoostFS for Linux requires the following:

- Data Domain Operating System version 6.0 or later
- FUSE 2.8 or later

The following Linux distributions are supported:

- Red Hat Enterprise Linux versions 6 and 7
- CentOS 7
- SUSE Linux Enterprise Server versions 11 and 12
- Ubuntu 14.04 and 15
- Oracle Linux version 7

Supported applications

BoostFS for Linux supports the following applications:

- Commvault Simpana versions 10 and 11
- MySQL Community 5.6. and 5.7
- MySQL Enterprise Manager 5.6 and 5.7
- MongoDB Community 2.6, 3.0, and 3.2

Information about integrating BoostFS with other applications can be found in the following white paper on the Data Domain Community site: Boost Everywhere - Data Domain BoostFS Integration Guide: Application Validation and Best Practices for the DD Boost File System Plug-In.

Boost features supported by BoostFS

Transport Layer Security (TLS) anonymous authentication is supported to provide encryption.

Note

If you select TLS, be aware that there is no configuration option to enable TLS from the client. It must be enabled through the Data Domain System.

Boost features not supported by BoostFS

- Managed File Replication (MFR)
- DD Boost-over-Fibre Channel (DFC)
- Retention Lock

CHAPTER 2

Preparing the Data Domain system for BoostFS

Prepare the Data Domain system for BoostFS

Every Data Domain system that is enabled for Data Domain Boost deduplication must have a unique name. You can use the DNS name of the Data Domain system, which is always unique.

Procedure

1. On the Data Domain system, log in as an administrative user.
2. Verify that the file system is enabled and running by entering:

```
$ filesys status
The file system is enabled and running.
```

3. Verify DD Boost is already enabled:

```
$ ddboost status
DD Boost status: enabled
```

If the DD Boost status is reported as disabled, enable it by entering:

```
$ ddboost enable
DD Boost enabled
```

4. Verify distributed segment processing is enabled:

```
ddboost option show
```

You should see the following output:

```
Option                                Value
----------------------------------    --------
distributed-segment-processing        enabled
virtual-synthetics                    enabled
fc                                    disabled
global-authentication-mode            none
global-encryption-mode                medium
-----------------------------------   ---------
```

If distributed segment processing is shown as disabled, enable it by entering:

```
ddboost option set distributed-segment-processing enabled
```

Note

- If secure multi-tenancy (SMT) is used, the user role must be set as `none`.
- Users who run backup applications that connect to Data Domain systems must have their user names configured on the Data Domain system. For more information, refer to the *Data Domain Operating System Administration Guide*.
- Multiple applications can use DD Boost to access a Data Domain system, and multiple users can be configured for DD Boost access. The username, password, and role must have already been set up on the Data Domain system using the DD OS `user add` command:

```
user add <user> [password <password>]
[role {admin | limited-admin | security | user | backup-operator |
data-access}]
[min-days-between-change <days>] [max-days-between-change <days>]
[warn-days-before-expire <days>] [disable-days-after-expire <days>]
[disable-date <date>] [force-password-change {yes | no}]
```

For example, to add a user with a login name of **jsmith** and a password of **mP34$muk*E** with administrative privilege, enter:

```
$ user add jsmith password mP34$muk*E role admin
```

Once the user has been created on the Data Domain system, the user must be made a DD Boost user. To add **jsmith** to the DD Boost user list, enter:

```
$ ddboost user assign jsmith
```

BoostFS and existing Data Domain commands

You must create one or more storage units on each Data Domain system enabled for BoostFS. Data Domain administrators can use existing DD OS CLI commands to create and manage storage units used by BoostFS.

Assign multiple users to BoostFS

When, as a system administrator, you create the storage units that users employ with the backup applications, you associate a username with each storage unit. This associated username can be changed after creation of the storage unit.

Storage units are accessible only to applications with the username that owns the storage unit.

Each storage unit is owned by one username, and the same username can own multiple storage units. The application passes the username and password to BoostFS, and DD Boost passes them to the Data Domain system when attempting to connect to the Data Domain system. The Data Domain system then authenticates the username and password. The username and password can be shared by different applications.

When a storage unit is created with a valid Data Domain system local user but not assigned to DD Boost, the user is automatically added to the DD Boost users list in the same way that a user is added via the `ddboost user assign` command.

Assign one or more users to the DD Boost users list:

```
$ ddboost user assign user1 user2
User "user1" assigned to DD Boost.
User "user2" assigned to DD Boost.
```

To verify and display the users in the users list, enter:

```
$ ddboost user show

DD Boost user   Default tenant-unit   Using Token Access
-------------   -------------------   ------------------
user1           Unknown               Yes
user2           Unknown               -
user3           Unknown               Yes
user4           Unknown               -
user5           Unknown               -
user6           Unknown               -
user7           Unknown               Yes
user8           Unknown               -
-------------   -------------------   ------------
```

To unassign the user from the users list, enter:

```
$ ddboost user unassign user1
User "user1" unassigned from DD Boost.
```

Create storage units

You need to create one or more storage units on each Data Domain system enabled for BoostFS.

Procedure

1. Create a storage unit on the Data Domain system:

   ```
   $ ddboost storage-unit create NEW_STU1 user user1
   Created storage-unit "NEW_STU1" for "user1".
   ```

   A storage unit name must be unique on any given Data Domain system. However, the same storage unit name can be used on different Data Domain systems.

   The username owns the storage unit and ensures that only connections with this username's credentials are able to access this storage unit. See the section on ddboost storage-unit commands in the *Data Domain Operating System Command Reference Guide* for details on command options.

2. Repeat the previous step for each storage-unit needed on the Data Domain system.
3. If you want to modify a storage unit on the Data Domain system, enter:

   ```
   $ ddboost storage-unit modify NEW_STU1 user user2
   Storage-unit "NEW_STU1" modified for user "user2".
   ```

   The `ddboost storage-unit modify` command allows the backup application to change the username ownership of the storage unit. Changing the username does not require that attributes of every file on the storage unit be changed.

4. Display the users list for the storage units:

   ```
   $ ddboost storage-unit show
   ```

   After entering the command, the output you see should be similar to the following:

Figure 1 Sample output of `ddboost storage-unit show`

```
# ddboost storage-unit show
Name                Pre-Comp (GiB)   Status   User       Report Physical
                                                           Size (MiB)
-----------------   --------------   ------   --------   ---------------
backup                         3.0   RW       sysadmin                 -
DDBOOST_STRESS_SU             60.0   RW       sysadmin                 -
task2                          0.0   RW       sysadmin                 -
tasking1                       0.0   RW       sysadmin                 -
DD1                            0.0   RW       sysadmin                 -
D6                             5.0   RW       sysadmin                 -
TEST_DEST                      0.0   D        sysadmin                 -
STU-NEW                        0.0   D        ddul                     -
getevent                       0.0   RW       ddul                     -
DDP-5-7                      120.0   RW       sysadmin                 -
TESTME                       150.0   RW       sysadmin                 -
DDP-5-7-F                    100.0   RW       sysadmin                 -
testSU                         0.0   RW       sysadmin               200
-----------------   --------------   ------   --------   ---------------
D    : Deleted
Q    : Quota Defined
RO   : Read Only
RW   : Read Write
RD   : Replication Destination
```

Logical stream limits for storage units (optional)

BoostFS is restricted to the same stream limit and storage quota features as DD Boost. See the *DD Boost for Partner Integration Administration Guide* for more information.

Client Groups and BoostFS

The Client Group feature identifies specific client loads when clients are associated with groups.

The `client group` command set is supported only for clients that use DD Boost or NFS protocols. For more information about Client Groups, see the *Data Domain Operating System Command Reference Guide*.

Distributed segment processing option

BoostFS supports distributed segment processing as supported by DD Boost. For more information, refer to the *Data Domain Operating System Administration Guide*.

Note

Enabling or disabling the distributed segment processing option does not require a restart of the Data Domain file system.

CHAPTER 3

Installing BoostFS for Linux

Installation overview

There is a single RPM installation package for BoostFS for Linux that both enterprise and small-scale users can download. It is available in both RPM and .deb formats. The RPM package includes the boostfs executable.

Check the following before beginning the process:

- The FUSE version on the client must be 2.8 or higher.

While the BoostFS process is running:

- BoostFS mount points must be deactivated.
- You cannot upgrade BoostFS.
- You cannot uninstall BoostFS.

Components of the BoostFS for Linux client

The BoostFS for Linux client is composed of the following:

- A daemon process that supports various commands
- Two shared libraries: `libDDBoost.so` and `libDDBoostFS.so`
- `.rsalib`: A hidden directory that contains redistributable RSA libraries
- A configuration file
- A manual page

`libDDBoost.so`, a FUSE-agnostic library built on the DD Boost library, provides such services as connection management, a retry mechanism, and client logging.

The packaging defaults to the Red Hat Package Manager (RPM) format, but the native packaging for other operating systems is also supported.

The following packages are available:

- Ubuntu: `DDBoostFS_1.1.0.1_565134_amd64.deb`
- Red Hat: `DDBoostFS-1.1.0.1-565134.rhel.x86_64.rpm`
- SUSE: `DDBoostFS-1.1.0.1-565134.sles.x86_64.rpm`

Note

Verify that you are using the appropriate package for your client OS.

BoostFS on Linux systems

Employing the Linux Filesystem Hierarchy Standard 3.0, the BoostFS for Linux client is installed in `/opt/emc/boostfs` and contains the following subdirectories:

- `bin`: boostfs command(s) are installed here.
- `lib`: Contains these libraries.
  - libDDBoost.so
  - libDDBoostFS.so
- `.rsalib`: Contains redistributable RSA libs.

- `etc`: Contains configuration files (sample and production).
- `man`: Contains standard man pages.

The role of FUSE in BoostFS for Linux

BoostFS for Linux uses FUSE, an open-source software interface that enables non-privileged users to securely create and mount their own file-system implementations.

FUSE allows you to export a virtual file system to the Linux kernel. Write operations through BoostFS and FUSE benefit from Data Domain's distributed segment processing.

Using FUSE and the DD Boost plug-in, BoostFS exports a storage unit on a Data Domain system to a mountpoint on a client. On the client, file system operations conducted on the mountpoint are captured by the kernel before being passed through FUSE to BoostFS.

BoostFS runs as a daemon on a client. As a software module, BoostFS serves as a layer between FUSE and DD Boost.

BoostFS in this release is only supported on some Linux systems in the initial release. For a list of supported environments, see Supported applications on page 11

FUSE consists of three parts:

- A kernel module: `fuse.ko`
- A user space library: `libfuse`
- A mount utility: `fusermount`

Note

BoostFS requires the "user_allow_other" option for FUSE; it will add the option to the `/etc/fuse.conf` file if it is not already present. Be aware that this may change the behavior of other FUSE-based applications you are using.

Upgrade the BoostFS client

Upgrade BoostFS for Linux using the BoostFS RPM package. Before performing the upgrade, you must stop all BoostFS processes.

The shared lockbox feature is introduced in BoostFS 1.1. When you upgrade from BoostFS 1.0 to BoostFS 1.1 or later, you must create a new lockbox and add current user credentials.

Note

If you are upgrading from BoostFS 1.1 or later, this procedure is not required.

If you use the BoostFS lockbox for user authentication, you must perform the following steps to upgrade:

Procedure

1. Upgrade BoostFS to 1.1 or later.
2. Remove all previous lockbox files:

```
# rm /opt/emc/boostfs/lockbox/*
```

3. Create the new lockbox by entering user credentials with the `boostfs lockbox set` command:

```
# /opt/emc/boostfs/bin/boostfs lockbox set <parameters>
```

4. Enter the remaining user credential pairs as needed.

Results

BoostFS is upgraded with the new lockbox ready for authentication use. See Shared lockbox files on page 35 for more information about configuring a common lockbox file for all BoostFS clients.

CHAPTER 4

Configuring BoostFS for Linux

The BoostFS for Linux configuration file

The Boost Filesystem has two configuration options.

- Command-line interface (CLI)
- The configuration file: `boostfs.conf`

This file is located in `/opt/emc/boostfs/etc`, and can be edited by the "root" user or someone with sudo privileges.

Parameters can be specified either in the config file or on the command line, or both.

The configuration file has a global section and a mount-point specific section. Configuration parameters configured using the command line take the highest priority and override any values in the config file. Mount-specific parameter values override global parameter values.

The following is a sample configuration file:

```
##############################################################################
# BoostFS 1.2 example input file
#
#
# The configuration file is divided into sections, delineated by brackets [].
# Options that are to apply to all mount points are in the [global] section.
# More details on the various configuration options can be found in the
# BoostFS manual. Command line options override what is in this file.
#
# Format:
# # - Identifies a comment line, and must be at the start. Configuration
# parameters can be disabled by adding a “#” to the start of the line.
#
# Values which contains spaces should use double quotations around the
# entire value.
#
# No whitespace is allowed between the option and the value, i.e.
# log-dir = /path is not allowed.
#
# Comments are not allowed after the option value pair.
#
##############################################################################

[global]
# Data Domain Hostname or IP address
# data-domain-system=dd2500-1.yourdomain.com

# Storage Unit
# storage-unit=su-name

# Security option used for authentication (default: lockbox)
# security=<krb5|lockbox>

# Storage Unit Username (should only be used in conjunction with Kerberos authentication)
# storage-unit-username=sysadmin

# Subdirectory within the storage-unit to mount to
# directory-name=path/to/subdir

# Lockbox path (default: /opt/emc/boostfs/lockbox/boostfs.lockbox)
# lockbox-path=path/to/lockbox

# Enable logging (default: true)
# log-enabled=<true|false>

# Log level (default: info)
```

```
# log-level=<debug|info|warning|error>

# Directory for log files (default: /opt/emc/boostfs/log)
# log-dir=/path/to/log

# Log file name (default: ddboostfs_<uid>_<gid>.log)
# log-file=output.log

# Maximum log size in MB (default: 100MB)
# log-maxsize=100

# Number of log files to save (default: 8)
# log-rotate-num=10

# Text string that describes the application using boostfs with additional information such
as the version.
# app-info="text_string"


# Allow users other than the owner of the mount to access the mount
# allow-others=<true|false>

#
# Mount point sections are delineated by [mountpoint]
#

# [/path/to/mount]
# Data Domain Hostname or IP address
# data-domain-system=dd2500-1.yourdomain.com

# Storage Unit
# storage-unit=su-name

# Security option used for authentication (default: lockbox)
# security=<krb5|lockbox>

# Storage Unit Username (should only be used in conjunction with Kerberos authentication)
# storage-unit-username=sysadmin

# Subdirectory within the storage-unit to mount to
# directory-name=path/to/subdir

# Enable Boost multithreading (default: true)
# mtboost-enabled=<true|false>

# Number of threads to use in multithreaded Boost mode for writing each file (default: 2)
# This does not have any significance if mtboost-enabled=false
# Min value is 0 (this means mtboost-threads will be intelligently calculated by boostfs by
querying CPU information)
# Max value is 16
#
# mtboost-threads=16

# Maximum number of connections that can be used at the same time (default: 128).
# Min value is 64. Max value is 256.
# max-connections=128
```

Configuring a BoostFS client and Data Domain system for Kerberos

You follow a specific sequence of steps to configure a BoostFS client and a Data Domain system to use Kerberos authentication.

Prepare the Data Domain system for Kerberos

If you choose Kerberos as the authentication method, you should ensure that all your systems can access the Key Distribution Center (KDC).

If the systems cannot reach the KDC, check the domain name system (DNS) settings at `/etc/resolv.conf`.

Check the following to ensure your system is ready for Kerberos:

- Verify the KDC can ping both the Data Domain system and the client.
- Verify the client can ping the Data Domain system and the KDC.
- Verify the Data Domain system can ping the Client and the KDC.
- Check the NTP settings in the configuration.
  All systems must have the same time stamp (within a margin of 5 minutes).

Set the host name and domain name on the Data Domain system

Set the host name and the domain name on the Data Domain system using the **net set** CLI command.

Procedure

1. On the Data Domain system, type the following:

   ```
   # net set hostname [host]
   # net set {domain name [local-domain-name]}
   ```

   For more information on **net** commands, see the *Data Domain Operating System Command Reference Guide*.

Configuring BoostFS for a Windows Active Directory environment

In this environment, the Windows server that hosts the Microsoft Active Directory service also acts as the Key Distribution Center (KDC) and also a domain name system (DNS).

You should verify that the client and the Data Domain system can reach the Active Directory domain using the DNS provided by the Windows Active Directory client.

Add users to the Active Directory KDC server

Procedure

1. Log in the Windows Active Directory system.
2. In Active Directory Users and Computers, add the user-owner of the storage-unit:

   ```
   <su_username> Type: "User" Password: <user_password>
   ```

   The password must match the user's password on the Data Domain system.

Configure a Data Domain system for an Active Directory configuration

Procedure

1. Enter the following command on the Data Domain system:

   ```
   # authentication kerberos set realm <realm> kdc-type windows
   ```

You should see the following text and be prompted to enter additional information:

```
- Enter domain user: <enter KDC Domain username>
- Enter domain password: <enter KDC Domain password>
```

2. Optionally, configure the client access list for DD Boost on the Data Domain system to use Kerberos authentication:

   # ddboost client add <*client-name*> authentication-mode kerberos

   The configuration process is now complete; the required keytab has been generated and applied to the Data Domain system.

Configure the BoostFS client in a Windows Active Directory environment

Procedure

1. Verify you are pointing to the correct Key Distribution Center (KDC) by checking the `/etc/krb5.conf` file.
2. Enter the `kerberos set` command to provide the Kerberos Ticket Granting Ticket (TGT) for the storage-unit user:

   ```
   # boostfs kerberos set -u <su-username> -s <su-name>
   ```

3. Verify the Kerberos TGT has been granted for the storage-unit user:

   ```
   # boostfs kerberos query -u <su-username> -s <su-name>
   ```

   The client configuration is complete.

Mount a storage unit using BoostFS

Procedure

1. Enter the `boostfs mount` command to mount the storage unit:

   ```
   # boostfs mount -d <dd-system> -s <storage-unit> -o
   security=krb5 -o -u <storage-unit-username> -m <kerberos-
   username> <mount-point>
   ```

   Note

   BoostFS does not support files being executed on the mount point.

Configuring BoostFS for a UNIX KDC environment

In this type of environment, a UNIX server hosts the Key Distribution Center (KDC) service.

The Kerberos file contains a "shared secret" (a password, pass phrase, or other unique identifier) between the KDC server and the Data Domain system. When using a UNIX KDC, the DNS server does not have to be the KDC server; it can be a separate server.

For Kerberos, you must transfer a keytab file from the UNIX KDC server, where it is generated, to the Data Domain system. If you are using more than one Data Domain system, you need a separate keytab file for each system. That means additional steps are required to configure multiple Data Domain systems on a UNIX KDC.

Add Data Domain principals to the UNIX KDC

Procedure

1. Log in to the Key Distribution Center (KDC).
2. Enter KDC admin mode using the following command: **kadmin**

   The commands in the subsequent steps apply to the KDC after entering **kadmin** mode.

3. Add Data Domain principals to the Key Distribution Center (KDC) using the Kerberos `addprinc` command: **# addprinc boostfs/<*ddsystem-hostname*>@<*realm*>**
4. Confirm the client principals have been added by entering the following Kerberos command: **listprincs**
5. Import host and BoostFS credentials to a temporary keytab file on the KDC by entering the Kerberos `ktadd -k` command:**# ktadd -k /tmp/<*keytab-file-name-for-ddsystem*> boostfs/<*ddsystem-hostname*>@<*realm*>**

   The keytab file for the Data Domain system is generated and needs to be imported to the Data Domain system.
6. Rename the file to `krb5.keytab` and copy it to `/ddr/var folder`.
7. Copy the keytab file generated in Step 3 from the KDC to the Data Domain system directory **/**`ddr/var/releases`.

Add client principals to the KDC

Procedure

1. Add the host and BoostFS service principals to the KDC using the Kerberos `addprinc` command:

   ```
   # addprinc boostfs/<client-hostname>@<realm>
   # addprinc host/<client-hostname>@<realm>
   ```

2. Confirm the client principals have been added using the following Kerberos command: **listprincs**
3. Import the host and BoostFS credentials to a temporary keytab file on the KDC by entering the Kerberos `ktadd -k` command:

   ```
   # ktadd -k /tmp/<keytab-file-name-for-client> boostfs/<client-hostname>@<realm>
   # ktadd -k /tmp/<keytab-file-name-for-client> host/<client-hostname>@<realm>
   ```

4. Copy the keytab file generated in Step 3 from the KDC to the client as `/etc/krb5.keytab` file.

Add users to the UNIX KDC server

The following command requires to access the `kadmin` interface on the Key Distribution Center (KDC) server.

Procedure

1. Add the storage-unit user from the Data Domain system to the KDC using the Kerberos `addprinc` command:

```
# addprinc <su-username>@<realm>
```

Configure Data Domain systems for the UNIX KDC

Procedure

1. Rename the *keytab_file_for_ddsystem* file located on the `/ddvar/releases` directory to `krb5.keytab`.

   See Add client principals to the KDC on page 28 for information on creating the keytab file for the Data Domain system.

2. On the Data Domain system, import the keytab file moved in Step 1 to `/ddr/etc` using the following command:

   # authentication kerberos keytab import

3. Confirm the configuration using the `authentication` command:

   # authentication kerberos show config

4. Set the realm on the Data Domain system using the authentication command:

   # authentication kerberos set realm <*realm*> kdc-type unix kdcs <*KDC-hostname*>

5. Optionally, configure the client access list for DD Boost to use Kerberos authentication:

   # ddboost client add <*client-name*> authentication-mode kerberos

   Note

   If you perform this optional step, note that a BoostFS client configured to use Kerberos must use Kerberos for the connection to succeed. If that BoostFS client uses RSA Lockbox, the connection will fail.

Configure the BoostFS client

Procedure

1. Check the `/etc/krb5.conf` file to make sure you point to the correct KDC server.
2. Enter the `boostfs kerberos set` command to get the Kerberos Ticket Granting Ticket (TGT) for the storage unit user:

   # boostfs kerberos set -u <*su-username*> -s <*su-name*>

3. Verify the Kerberos TGT is granted for the storage-unit user by using the following `boostfs kerberos query` command:

   # boostfs kerberos query -u <*su-username*> -s <*su-name*>

   The client configuration is complete.

Mount a storage unit using BoostFS

Procedure

1. Enter the `boostfs mount` command to mount the storage unit:

```
# boostfs mount -d <dd-system> -s <storage-unit> -o
security=krb5 -o -u <storage-unit-username> -m <kerberos-
username> <mount-point>
```

Note

BoostFS does not support files being executed on the mount point.

CHAPTER 5

Using BoostFS for Linux

BoostFS command overview

You use the `boostfs` command to establish the FUSE mount, create the lockbox (if desired), and set up Kerberos credentials if you choose Kerberos as the authentication method.

boostfs kerberos

The `boostfs kerberos` command allows you to add, verify, and remove Kerberos credentials.

`boostfs kerberos set [-u | --storage-unit-username] <storage-unit-username>`
`[-s |--storage-unit] <storage-unit-name>`
`[[-r |--kerberos-realm] <kerberos-realm>]`
`[[-m | --kerberos-username] <kerberos-username>]`
Allows you to add Kerberos credentials. Role required: admin.

`boostfs kerberos query [[-u |--storage-unit-username] <storage-unit-username>`
`[-s |--storage-unit] <storage-unit-name>]`
`[[-m |--kerberos-username] <kerberos-username>]`
Checks for Kerberos credentials. Role required: admin.

`boostfs kerberos remove [-u | --storage-unit-username] <storage-unit-name>`
Removes Kerberos credentials. Role required: admin.

boostfs lockbox

The boostfs lockbox command allows you to set the RSA lockbox values.

`boostfs lockbox add-hosts hostname [hostname]`
Adds clients that can access the shared lockbox. When you are adding and removing access to the shared lockbox, you must do so from the machine where the lockbox was initially created. Role required: admin.

`boostfs lockbox delete-hosts all`
Deletes all client access to the shared lockbox. Role required: admin.

`boostfs lockbox delete-hosts hostname [hostname]`
Deletes specific client access to the shared lockbox. Role required: admin.

`boostfs lockbox {remove | query} [-d | --data-domain-system] data-domain-system`
`[-s | --storage-unit] storage-unit-name`
If the credentials have been stored in an RSA lockbox, this command returns the username after the query is submitted with the specified Data Domain hostname and storage-unit. Role required: admin.

`boostfs lockbox set [-d |--data-domain-system] data-domain-system`
`[-u | --storage-unit-username] storage-unit-username`
`[-s | --storage-unit] storage-unit-name`
To store credentials in an RSA lockbox, the user specifies the Data Domain hostname, the storage-unit name, and the storage-unit user. After providing that information, the user is prompted for the password. Role required: admin.

Note

The command `boostfs lockbox set` fails if there is an existing Lockbox file in the same location. This includes Lockbox files generated with older versions of BoostFS. For example, the existence of a BoostFS 1.1 Lockbox causes the creation of a Lockbox with BoostFS 1.2 to fail.

`boostfs lockbox remove [-d | --data-domain-system]` *`data-domain-system`*
`[-s | --storage-unit]` *`storage-unit-name`*
Removes the stored RSA lockbox credentials in the specified Data Domain system and storage-unit. Role required: admin.

`boostfs lockbox show-hosts`
Shows all clients that can access the shared lockbox. Role required: admin.

boostfs mount

The `boostfs mount` command allows you to establish the BoostFS FUSE mount.

`boostfs mount [-d|--data-domain-system]` *`<data-domain-system>`*
`[-s|--storage-unit]` *`<storage-unit>`*
`[[-o|--option` *`<param>`*`=`*`<value>`*`] ...]` *`<mount-point>`*
Mount the BoostFS file system. Role required: none.

`boostfs umount` *`<mount-point>`*
Unmount the BoostFS file system. Role required: none.

Argument Definitions

mount-point
The mount-point for the BoostFS system.

storage-unit
The target storage-unit on the Data Domain system.

Command options

The following options are valid for the boostfs mount command:

Option	Description
`-o directory-name=path/to/subdir`	Subdirectory within the storage-unit you select for mounting (default: root of the storage unit). You must create the subdirectory after mounting at the root path, unmounting, and adding the parameter to the subsequent mount command or config file.
`-o security=<krb5 | lockbox>`	Security option used for authentication (default: lockbox)
`-o allow-others=<true | false>`	Allow users other than the owner of the mount to access the mount
`-o log-enabled=<true | false>`	Enable logging (default: true)
`-o log-level=<debug | info| warning| error>`	Log level (default: info)

Option	Description
`-o log-dir=/path/to/log`	Directory for log files (default: `/opt/emc/boostfs/log`)
`-o log-file=output.log`	Log file name (default: `/opt/emc/boostfs/log/ddboostfs_<uid>_<gid>.log`)
`-o log-maxsize=100`	Maximum log size in MB (default: 100 MB)
`-o log-rotate-num=8`	Number of log files to save (default: 8)
`-o app-info="text string"`	Text string describing the application using boostfs (version, etc) (default: FUSE version)

BoostFS and high availability

If you are configuring a Data Domain high availability (HA) system, you should make sure the IP address (or hostname) that you specify for the system is one of the floating IP addresses. Only the floating IP addresses in an HA system are accessible after a failover.

If you incorrectly specify one of the fixed HA addresses, you will not be able to connect to the Data Domain system in the event of a recoverable failure.

Authentication methods

BoostFS has two authentication options:

- RSA Lockbox (default)
- Kerberos

RSA Lockbox-based authentication

RSA Lockbox is the default password manager for BoostFS for Linux.

To use RSA Lockbox, you need to set the lockbox using the `boostfs lockbox set` command. Beginning with BoostFS 1.1, you can also set up a shared BoostFS lockbox file.

Kerberos-based authentication

BoostFS for Linux supports Kerberos-based authentication, which offers some important advantages.

Note

Before using Kerberos for BoostFS, you should verify that the Kerberos client libraries for your Linux distribution are installed on your machine.

These advantages include the following:

- Single sign-on capability
- Passwords never exchanged over the wire between client and server
- Automatic renewal of Kerberos tickets with strong encryption and configuration support

Note

There are tools such as `kinit -R` that administrators need to run as part of a Cron job to ensure the Kerberos credentials are renewed. This is part of a standard Kerberos configuration.

There are three main entities involved with Kerberos Authentication:

- BoostFS client
- Kerberos Key Distribution Center (KDC), which can be either one of the following:
  - An active directory on a domain controller in a Windows environment
  - A UNIX-based KDC
- Data Domain Restorer (DDR)

Firewall configurations
If you are creating a firewall configuration, you should use the BoostFS TCP Port 2049. Kerberos networking ports must also be added to the firewall.

Supported topologies
The following topologies support Kerberos:

- Data Domain systems running DD OS version 6.0 and later
- All Linux clients that run the BoostFS executable
- KDC/Active Directory (AD) on a Windows 2012 R2 server
- KDC on a POSIX-based operating system with optional NIS lookups

The BoostFS client and Kerberos authentication
The BoostFS client uses the General Security Services (GSS) API to negotiate Kerberos authentication with the Data Domain system. GSS-API provides a generic interface which can be layered atop different security mechanisms, meaning that communicating peers that have GSS-API credentials for the same security mechanism can establish a security context.

To communicate using Kerberos authentication, the BoostFS client must meet the following conditions:

- Be configured as a Kerberos client for a specified domain or realm
- Be capable of acquiring Kerberos credentials for a specified node and access to the desired services

Kerberos tickets
You need two types of tickets to access different services through Kerberos:

- A Kerberos Ticket Granting Ticket (TGT)
- A Kerberos ticket for various services (service tickets) that the client will use (BoostFS, DNS, CIFS, NFS, etc.)

Shared lockbox files

Beginning with BoostFS 1.1, you can create a common lockbox file for all BoostFS clients. This feature allows you to avoid having to create a separate lockbox file for each unique BoostFS client.

Sharing a common lockbox file enables you to create a single management point for BoostFS clients to access BoostFS mount points on Data Domain systems.

Lockbox files created with BoostFS 1.0 are incompatible with BoostFS 1.1. To solve this problem, you must erase a BoostFS 1.0 installation if you have one, and then install BoostFS 1.1.

Due to a change in the Lockbox format, you must recreate your Lockbox when upgrading from BoostFS 1.0 to BoostFS 1.1. To do this, remove the files located under `/opt/emc/boostfs/lockbox/`. Then after upgrading, re-enter any credentials using the `boostfs lockbox set` command.

Note

A BoostFS 1.0 client can use a lockbox created with a 1.1 or later client, but a 1.1 or later client cannot use a lockbox created with a 1.0 client. A BoostFS 1.1 or later client can use a lockbox created with a 1.1 or later client, even if they are not the same version.

Methods of sharing a BoostFS lockbox file

If you want to share a BoostFS lockbox file or the directory, you can choose different methods of doing so.

The following are three ways to share a lockbox file or directory:

1. Sharing the lockbox directory
2. Installing copies of the master lockbox file
3. Creating a symlink to a common NFS share

Share the lockbox directory

On the server where the master lockbox resides, you can edit the `/etc/exports` file to add the lockbox directory to that file.

You must add the following options to give clients access to the shared master lockbox:

```
</path/to/lockbox/dir>
*(rw,insecure,sync,no_root_squash,no_subtree_check)
```

Note

You must have BoostFS installed on the client that needs access to the shared lockbox.

The following example shows the path to the shared lockbox with options needed to access it:

```
/opt/emc/boostfs/lockbox
*( rw,insecure,sync,no_root_squash,no_subtree_check)
```

When access is gained, the client issues a standard Linux NFS mount command:

```
mount FQDN:/opt/emc/boostfs/lockbox /opt/emc/boostfs/lockbox
```

Install copies of the master lockbox file
You can use a secure copy program (`scp`) to copy the master lockbox file to all the clients that require access to it.

```
# scp /opt/emc/boostfs/lockbox/boostfs.lockbox \
root@<ip address | dns name>:/opt/emc/boostfs/lockbox
```

Create a symlink to a common NFS share
If the clients already have access to an NFS shared filesystem through an automounter program or a common mount point, you can copy the `boostfs.lockbox` file to the shared filesystem and create a symbolic link on the client that points to that filesystem.

In the following example, `/auto/tools/etc` is a mountable filesystem configured and controlled by an automounter program. In this case, you can create a symbolic link by entering the following:

```
# ln -s /auto/tools/etc/boostfs.lockbox /opt/emc/boostfs/lockbox/
boostfs.lockbox
```

In this case, the shared filesystem `/mnt/fs` is mounted from an NFS server to multiple clients. If the `boostfs.lockbox` file is copied to that filesystem, a symlink on the clients would resemble the following:

```
# ln -s /mnt/fs/path/to/boostfs.lockbox /opt/emc/boostfs/lockbox/
boostfs.lockbox
```

Note

The file `boostfs.lockbox` must have rw-rw-rw- (666) permissions to ensure all clients can access it.

Mount options

BoostFS allows you to mount a BoostFS file system using the Linux/UNIX `mount` command.

Mounting a BoostFS file system with the `mount` command works the same way mounting NFS or any other file system works. Because the standard `mount` command is supported, other standard facilities that use the `mount` command also work.

Example 1 Basic use of `mount`

The most basic use of the `mount` command is as follows:

Example 1 Basic use of `mount` (continued)

```
mount -t boostfs myddr:/mystu /mnt
```

In the example, **`myddr`** is the hostname of the DDR, **`mystu`** is the name of the DD Boost storage unit, and **`/mnt`** is the mount point where the file system is to be mounted.

Use of the file systems table

During system start and some other times, the `mount` command consults the file systems table (fstab) that is stored in the `/etc/fstab` file for direction on what file systems should be mounted. For example, if the `mount -a` command is executed, `mount` tries to mount all of the file systems that are documented in the `/etc/fstab` file according to the fstab rules.

Example 2 Use of `mount` with fstab

In this example, the fstab entry shown mounts the storage unit **`mystu`** from the DDR **`myddr`** onto `/mnt` as a BoostFS filesystem.

```
myddr:/mystu /mnt boostfs defaults,_netdev 0 0
```

When using this command, set the mount point to the location where you want the file system to be mounted, such as `/mnt`.

Example 3 Allowing multiple users

In this example, the fstab entry includes the BoostFS option `allow-others`, which allows access to users other than the user that mounted the file system. Because the file system is mounted during system start, the user that mounted the file system is the root user.

```
myddr:/mystu /mnt boostfs defaults,_netdev,bfsopt(allow-
others=true) 0 0
```

Table 2 `mount` command options

Command Option	Description
`[-o]`	Precedes an option.
`[username]=<valid user name>`	The specified username is used when root is the mounter and the administrator wants the mount to be performed on behalf of the specified user.
`[uid]=###`	If the username option is not specified, the uid option is used. The specified uid is used when

Table 2 `mount` command options (continued)

Command Option	Description
	root is the mounter and the administrator wants the mount to be performed on behalf of the specified uid user. If the username option is specified, the uid option is ignored.
[gid]=###	If the uid option is used and the gid option is also specified, the specified gid is used as the effective gid of the mount. If the username option is specified, the gid option is ignored.
[umask]=###	If the uid or the username option is used and the umask option is specified, the specified umask is used as the effective umask of the mount point at the time of the mount. This option is necessary to change the permissions of transient mountpoints as is the case with automounter and other automatic mount mechanisms. It is recommended that the umask be set to "0000" ("umask=0000").
[bfsopt](<valid boostfs option>,...)	The bfsopt is used to introduce boostfs options on the mount command line. These options are passed directly to boostfs at the time of the mount. Any valid boostfs option may be specified.

Compressed restoration

When the mount option `ddboost-read-compression` is set to `true`, data is compressed on the server before being sent to the client. When the client receives the data, it must decompress the data. Sending and receiving compressed data uses less network bandwidth, but compressing and decompressing the data requires a significant amount of CPU power. By default, this option is set to `false`.

This option can be used in one of the following two ways:

- As a command-line option:
  ```
  boostfs mount -o ddboost-read-compression=true /mnt/bfs-mount
  ```
- As an option configured in the `boostfs.conf` file:
  ```
  ddboost-read-compression=true
  ```

Automounter

To mount file systems dynamically, use the Linux automounter with the `autofs` command. Mounts created with the `automount` command are automatically unmounted when not in use.

To enable the automounter, edit the `/etc/auto.master` file. If a program refers to a file within an automount-defined file system, the system mounts the file system to honor the request. The mounting process is transparent to the user and application.

The `auto.master` file introduces the file system to be mounted to `automount` and refers the `automount` facility to a script that controls the mount. This file is read when `automount` is started, usually by an `init.d` or `systemd` script.

In recent versions of Linux, the `systemctl` command is used to perform a service operation such as `systemctl [start | stop | restart] autofs`, where the `start`, `stop`, or `restart` option is specified.

For more information about the `automount` facility, refer to the Linux man pages for `mount`, `automount`, `auto.master`, and `autofs`.

Example 4 Using the automounter with BoostFS

In this example, the script to which the `auto.master` file refers the `automount` facility is `auto.boost`. The `auto.boost` script receives the directory to be mounted as a parameter. The script returns the mount options that are used.

Sample line in `/etc/auto.master` that enables `/etc/auto.boost` to mount to `/boost`:

```
/boost program,sun:/etc/auto.boost --timeout=10
```

A sample `/etc/auto.boost` script file:

```
#!/bin/bash
opts="-fstype=boostfs,rw,noauto,exec,bfsopt(allow-others=true)"
opts2="-
fstype=boostfs,rw,umask=0000,username=auser,exec,bfsopt(allow-
others=false)"
case "$1" in
    userdir)
    echo "$opts2 myddr:/mystu"
    ;;
    backup)
    echo "$opts myddr:/mystu"
    ;;
    *)
    ;;
esac
```

In this example, the directory `/boost` is automatically created when `automount` is started. When a program or shell command refers to `/boost/userdir`, `automount` creates the directory `/boost/userdir` and mounts the BoostFS file system to that

mount point. When the mount operation completes successfully, the user process executes with the files at that mount point. If the mount point remains dormant for more than 10 seconds, it is automatically unmounted.

This example shows an additional mount point, `/boost/backup`, with different options. When using the automounter, you must specify the user for whom the file system is mounted or use the `boostfs` option **`allow-others`**. The options for the mount point `/boost/backup` show the **`allow-others`** option.

Because the file `/etc/auto.boost` is an executable script, you must give it `+x` permissions. To test the script, run it with a specified parameter and check the printed response.

BoostFS client connection details

After mount points are created, you can use the `ddboost show connections` command to see details about clients that use BoostFS to connect to the Data Domain system.

The details displayed in the output include the BoostFS version number and the Boost library, as shown in the following example:

```
dduser@ddve1# ddboost show connections

Active Clients: 0

Clients:
Client                    Idle  Plugin Version  OS Version                    Application Version                                  Encrypted  DSP  Transport
------------------------  ----  --------------  ----------------------------  ---------------------------------------------------  ---------  ---  ---------
client.yourdomain.com     YES   3.4.1.2-592285  Linux 3.13.0-144-generic x86_64  BOOSTFS:1.1.1.1-000000 MongoDBApp FUSE:2.9.3        NO         YES  IPv4
------------------------  ----  --------------  ----------------------------  ---------------------------------------------------  ---------  ---  ---------

Client Connections:
Max Client Connections: 180
---------- ifgroup ----------  ---------- Client Connections ----------  - DD Connections -  --------
                               -------- Control --------                 -------- Data ---------
Group-name  Status  Interface                                  Write  Read  Src-repl  Dst-repl  Synthetic  Repl-out  Repl-in  Total
----------  ------  -----------------------------------------  -----  ----  --------  --------  ---------  --------  -------  -----
none                                          10.6.109.148     0      0     0         0         0          0         0        0
none                2620:0:170:1604:2a0:d1ff:feec:d071         0      0     0         0         0          0         0        0
----------  ------  -----------------------------------------  -----  ----  --------  --------  ---------  --------  -------  -----
                                      Total Connections:       0      0     0         0         0          0         0        0
----------  ------  -----------------------------------------  -----  ----  --------  --------  ---------  --------  -------  -----
```

See the *Data Domain Operating System Command Reference Guide* for more information about the `ddboost show connections` command.

CHAPTER 6

Troubleshooting

Log information

You can use the following log files to diagnose BoostFS problems:

- BoostFS log file
  By default, the BoostFS log file is found the directory `/opt/emc/boostfs/log`. The default name of the file is `ddboostfs_<uid>_<gid>.log`, where:
  - `<uid>`
    is the user id of boostfs user
  - `<gid>`
    is the group id of boostfs user

  A typical BoostFS log message appears in the following format:

  ```
  Date + Time + Procss-ID + Thread-ID +  [logging-leve: E - error, W - warning, I - info, D - debug) + Message-Text
  ```

  The following is an example information message:

  ```
  May 23 12:53:51 2996 4014012160 [I] bfs_close_open_nodsp: File / 00000004 opened in non-DSP mode
  ```
- DD Boost SDK precert log file
- Data Domain File System logs
  Data Domain File System logs are found on the Data Domain system in the directory `/ddr/var/log/debug`. See the *Data Domain Operating System Administration Guide* for more information.

BoostFS generates a local log file that contains its internal status, activities, warnings, and errors. You can specify the logging level in addition to the name and location of the log file by using the CLI or the BoostFS configuration file.

If you want to initiate troubleshooting during BoostFS operations, you can use the **`kill -s SIGUSR2 <boostfs_pid>`** to rotate the BoostFS log level, where *<boostfs_pid>* is the process identifier of the BoostFS process.

You might need to set a size limit on the log file to ensure that when the size of the log file reaches that limit, BoostFS will rotate log messages.

You can configure the maximize size of the BoostFS log file in the configuration file. You can also configure the number of older log files you wish to keep.

When the log file size reaches the maximum specified size (in MB), the log file is renamed by appending ".1" to the log file name. If there is already an existing log file that ends in ".1," that file is renamed to replace ".1" with ".2." As each log file reaches the maximum size, log files with numbers (n) appended are renamed .n+1 up to the maximum log rotate number.

Known issues

Unable to establish a BoostFS mount point after upgrade to 1.1

`[E] bfs_get_passphrase: unable to get passphrase - incompatible lockbox version`

There is an incompatible lockbox version. When you upgrade from BoostFS 1.0 to 1.1, you must create a new lockbox and add user credentials. See Upgrade the BoostFS client on page 21 for more details.

Unable to establish a BoostFS mount point

The following section describes other mount common errors and solutions:

The mount point mount-point is nonempty.

BoostFS cannot be mounted on a nonempty mount point. Try mounting BoostFS again on an empty mount point. This error can occur if the user already has a mount point established. You might want to check to see if this might be the case. You can either use the already established mount point or use **boostfs umount** to unmount the existing mount point and establish a new one. This error can also appear if the directory on the client being mounted to already contains files. In this case, theses file need to be removed or an empty mount point directory must be selected instead.

Cannot mount mount-point: unexpected error, please see log for details.

Most often seen if the DD Boost protocol isn't enabled and configured on the Data Domain System. You should check the BoostFS log files for more details and confirming DD Boost is enabled using the `ddboost status` command on the Data Domain System.

Invalid mount point option and value pair [option=key from config file] [value= value from config file]
/mnt/test: Configuration initialization failed

This message can appear when errors occur during the processing of the BoostFS configuration file. A best practice is to review the specific key and value printed out in the BoostFS configuration file and making any corrections.

Cannot mount mount-point: unexpected error

This error is most often seen when using Kerberos authentication and an error exists in the setup. Review the Kerberos instructions in this configuration guide and ensuring the values are set properly.

Insufficient access to or storage-unit storage-unit does not exist

This error is seen if you are using Kerberos and the Kerberos ticket has expired. A best practice is to use the `boostfs kerberos query` to confirm whether the user's ticket has expired, and using the `boostfs kerberos set` command to reconfigure the expired ticket.

Unable to unmount a BoostFS mountpoint

fusermount: failed to unmount mount-point: Invalid argument

This message can appear if the BoostFS mount point has not yet been established. There should be no issues if a mount point has already been unmounted, but this error can still appear.

Unable to access a BoostFS mount point

A permission or privileges error can appear when attempting to use the mount point. This error is most often seen when the user does not have the necessary permissions to access a mount point. By default, the only user allowed to access a mount point is the one that established it. To allow other users to share this mount point, you must include the `-o allow-others=true option` either on the command line or in the BoostFS configuration file.

Configuration values are not taking effect

Typically configuration parameters are not taking effect because the `[global]` label at the top of the BoostFS configuration file has not been uncommented. A best practice is to check the configuation file to confirm `# [global]` has been changed to `[global]`.

BoostFS does not mount after reboot

If BoostFS fails to mount after rebooting the system, you can add the `_netdev` option to `/etc/fstab` as shown in the following example:

```
10.98.88.93:/user1-stu /home/user1/boostfs boostfs
umask=0000,user,_netdev 0 0
```

APPENDIX A

Appendix

About Puppet

If you have an enterprise/remote environment, you can install and configure Puppet to distribute the BoostFS configuration file – and any future updates – to clients. This is a best practice.

Puppet is open-source software that allows you to manage clients in a master-server manner. When you install Puppet, you designate one system as the master. BoostFS uses Puppet to distribute or "push" BoostFS to different machines in a large enterprise environment.

In smaller-scale environments, you can simply install BoostFS on individual machines.

For more information about Puppet, see the Puppet Labs website at https://puppet.com/product/puppet-enterprise-and-open-source-puppet.

References

The following documents, located at Online Support, provide additional and relevant information. Access to these documents depends on your login credentials. If you do not have access to a document, contact a sales representative.

- *Data Domain BoostFS Integration Guide: Application Validation and Best Practices*, available on https://community.emc.com
- *Data Domain Operating System Version Administration Guide*
- *Data Domain Operating System Version Initial Configuration Guide*