



Add storage capacity to an NFS-enabled SVM

ONTAP 9

NetApp
November 28, 2022

Table of Contents

- Add storage capacity to an NFS-enabled SVM 1
 - Add storage capacity to an NFS-enabled SVM overview 1
 - Create an export policy 1
 - Add a rule to an export policy..... 1
 - Create a volume or qtree storage container..... 7
 - Secure NFS access using export policies 10
 - Verify NFS client access from the cluster..... 12
 - Test NFS access from client systems 13

Add storage capacity to an NFS-enabled SVM

Add storage capacity to an NFS-enabled SVM overview

To add storage capacity to an NFS-enabled SVM, you must create a volume or qtree to provide a storage container, and create or modify an export policy for that container. You can then verify NFS client access from the cluster and test access from client systems.

What you'll need

- NFS must be completely set up on the SVM.
- The default export policy of the SVM root volume must contain a rule that permits access to all clients.
- Any updates to your name services configuration must be complete.
- Any additions or modifications to a Kerberos configuration must be complete.

Create an export policy

Before creating export rules, you must create an export policy to hold them. You can use the `vserver export-policy create` command to create an export policy.

Steps

1. Create an export policy:

```
vserver export-policy create -vserver vserver_name -policyname policy_name
```

The policy name can be up to 256 characters long.

2. Verify that the export policy was created:

```
vserver export-policy show -policyname policy_name
```

Example

The following commands create and verify the creation of an export policy named `exp1` on the SVM named `vs1`:

```
vs1::> vserver export-policy create -vserver vs1 -policyname exp1

vs1::> vserver export-policy show -policyname exp1
Vserver          Policy Name
-----
vs1              exp1
```

Add a rule to an export policy

Without rules, the export policy cannot provide client access to data. To create a new export rule, you must identify clients and select a client match format, select the access

and security types, specify an anonymous user ID mapping, select a rule index number, and select the access protocol. You can then use the `vserver export-policy rule create` command to add the new rule to an export policy.

What you'll need

- The export policy you want to add the export rules to must already exist.
- DNS must be correctly configured on the data SVM and DNS servers must have correct entries for NFS clients.

This is because ONTAP performs DNS lookups using the DNS configuration of the data SVM for certain client match formats, and failures in export policy rule matching can prevent client data access.

- If you are authenticating with Kerberos, you must have determined which of the following security methods is used on your NFS clients:
 - `krb5` (Kerberos V5 protocol)
 - `krb5i` (Kerberos V5 protocol with integrity checking using checksums)
 - `krb5p` (Kerberos V5 protocol with privacy service)

About this task

It is not necessary to create a new rule if an existing rule in an export policy covers your client match and access requirements.

If you are authenticating with Kerberos and if all volumes of the SVM are accessed over Kerberos, you can set the export rule options `-rorule`, `-rwrule`, and `-superuser` for the root volume to `krb5`, `krb5i`, or `krb5p`.

Steps

1. Identify the clients and the client match format for the new rule.

The `-clientmatch` option specifies the clients to which the rule applies. Single or multiple client match values can be specified; specifications of multiple values must be separated by commas. You can specify the match in any of the following formats:

Client match format	Example
Domain name preceded by the "." character	<code>.example.com</code> or <code>.example.com, .example.net, ...</code>
Host name	<code>host1</code> or <code>host1, host2, ...</code>
IPv4 address	<code>10.1.12.24</code> or <code>10.1.12.24, 10.1.12.25, ...</code>
IPv4 address with a subnet mask expressed as a number of bits	<code>10.1.12.10/4</code> or <code>10.1.12.10/4, 10.1.12.11/4, ...</code>
IPv4 address with a network mask	<code>10.1.16.0/255.255.255.0</code> or <code>10.1.16.0/255.255.255.0, 10.1.17.0/255.255.255.0, ...</code>

Client match format	Example
IPv6 address in dotted format	::1.2.3.4 or ::1.2.3.4, ::1.2.3.5, ...
IPv6 address with a subnet mask expressed as a number of bits	ff::00/32 or ff::00/32, ff::01/32, ...
A single netgroup with the netgroup name preceded by the @ character	@netgroup1 or @netgroup1, @netgroup2, ...

You can also combine types of client definitions; for example, `.example.com, @netgroup1`.

When specifying IP addresses, note the following:

- Entering an IP address range, such as `10.1.12.10-10.1.12.70`, is not allowed.

Entries in this format are interpreted as a text string and treated as a host name.

- When specifying individual IP addresses in export rules for granular management of client access, do not specify IP addresses that are dynamically (for example, DHCP) or temporarily (for example, IPv6) assigned.

Otherwise, the client loses access when its IP address changes.

- Entering an IPv6 address with a network mask, such as `ff::12/ff::00`, is not allowed.

2. Select the access and security types for client matches.

You can specify one or more of the following access modes to clients that authenticate with the specified security types:

- `-rorule` (read-only access)
- `-rwrule` (read-write access)
- `-superuser` (root access)



A client can only get read-write access for a specific security type if the export rule allows read-only access for that security type as well. If the read-only parameter is more restrictive for a security type than the read-write parameter, the client might not get read-write access. The same is true for superuser access.

You can specify a comma-separated list of multiple security types for a rule. If you specify the security type as `any` or `never`, do not specify any other security types. Choose from the following valid security types:

When security type is set to...	A matching client can access the exported data...
any	Always, regardless of incoming security type.

When security type is set to...	A matching client can access the exported data...
none	If listed alone, clients with any security type are granted access as anonymous. If listed with other security types, clients with a specified security type are granted access and clients with any other security type are granted access as anonymous.
never	Never, regardless of incoming security type.
krb5	If it is authenticated by Kerberos 5. Authentication only: The header of each request and response is signed.
krb5i	If it is authenticated by Kerberos 5i. Authentication and integrity: The header and body of each request and response is signed.
krb5p	If it is authenticated by Kerberos 5p. Authentication, integrity, and privacy: The header and body of each request and response is signed, and the NFS data payload is encrypted.
ntlm	If it is authenticated by CIFS NTLM.
sys	If it is authenticated by NFS AUTH_SYS.

The recommended security type is `sys`, or if Kerberos is used, `krb5`, `krb5i`, or `krb5p`.

If you are using Kerberos with NFSv3, the export policy rule must allow `-rorule` and `-rwrule` access to `sys` in addition to `krb5`. This is because of the need to allow Network Lock Manager (NLM) access to the export.

3. Specify an anonymous user ID mapping.

The `-anon` option specifies a UNIX user ID or user name that is mapped to client requests that arrive with a user ID of 0 (zero), which is typically associated with the user name `root`. The default value is `65534`. NFS clients typically associate user ID `65534` with the user name `nobody` (also known as *root squashing*). In ONTAP, this user ID is associated with the user `pcuser`. To disable access by any client with a user ID of 0, specify a value of `65535`.

4. Select the rule index order.

The `-ruleindex` option specifies the index number for the rule. Rules are evaluated according to their order in the list of index numbers; rules with lower index numbers are evaluated first. For example, the rule with index number 1 is evaluated before the rule with index number 2.

If you are adding...	Then...
The first rule to an export policy	Enter 1.
Additional rules to an export policy	<p>a. Display existing rules in the policy: <code>vserver export-policy rule show -instance -policyname <i>your_policy</i></code></p> <p>b. Select an index number for the new rule depending on the order it should be evaluated.</p>

5. Select the applicable NFS access value: {nfs|nfs3|nfs4}.

`nfs` matches any version, `nfs3` and `nfs4` match only those specific versions.

6. Create the export rule and add it to an existing export policy:

```
vserver export-policy rule create -vserver vserver_name -policyname
policy_name -ruleindex integer -protocol {nfs|nfs3|nfs4} -clientmatch { text |
"text,text,..." } -rorule security_type -rwrule security_type -superuser
security_type -anon user_ID
```

7. Display the rules for the export policy to verify that the new rule is present:

```
vserver export-policy rule show -policyname policy_name
```

The command displays a summary for that export policy, including a list of rules applied to that policy. ONTAP assigns each rule a rule index number. After you know the rule index number, you can use it to display detailed information about the specified export rule.

8. Verify that the rules applied to the export policy are configured correctly:

```
vserver export-policy rule show -policyname policy_name -vserver vserver_name
-ruleindex integer
```

Examples

The following commands create and verify the creation of an export rule on the SVM named `vs1` in an export policy named `rs1`. The rule has the index number 1. The rule matches any client in the domain `eng.company.com` and the netgroup `@netgroup1`. The rule enables all NFS access. It enables read-only and read-write access to users that authenticated with `AUTH_SYS`. Clients with the UNIX user ID 0 (zero) are anonymized unless authenticated with Kerberos.

```

vs1::> vserver export-policy rule create -vserver vs1 -policyname expl
-ruleindex 1 -protocol nfs
-clientmatch eng.company.com,@netgroup1 -rorule sys -rwrule sys -anon 65534
-superuser krb5

vs1::> vserver export-policy rule show -policyname nfs_policy
Virtual      Policy      Rule      Access      Client      RO
Server       Name        Index     Protocol    Match       Rule
-----
vs1          expl        1         nfs         eng.company.com, sys
                                     @netgroup1

vs1::> vserver export-policy rule show -policyname expl -vserver vs1
-ruleindex 1

Vserver: vs1
Policy Name: expl
Rule Index: 1
Access Protocol: nfs
Client Match Hostname, IP Address, Netgroup, or Domain:
eng.company.com,@netgroup1
RO Access Rule: sys
RW Access Rule: sys
User ID To Which Anonymous Users Are Mapped: 65534
Superuser Security Types: krb5
Honor SetUID Bits in SETATTR: true
Allow Creation of Devices: true

```

The following commands create and verify the creation of an export rule on the SVM named vs2 in an export policy named expol2. The rule has the index number 21. The rule matches clients to members of the netgroup dev_netgroup_main. The rule enables all NFS access. It enables read-only access for users that authenticated with AUTH_SYS and requires Kerberos authentication for read-write and root access. Clients with the UNIX user ID 0 (zero) are denied root access unless authenticated with Kerberos.


```
vs2::> vserver export-policy rule create -vserver vs2 -policyname expol2
-ruleindex 21 -protocol nfs
-clientmatch @dev_netgroup_main -rorule sys -rwrule krb5 -anon 65535
-superuser krb5
```

```
vs2::> vserver export-policy rule show -policyname nfs_policy
```

Virtual Server	Policy Name	Rule Index	Access Protocol	Client Match	RO Rule
vs2	expol2	21	nfs	@dev_netgroup_main	sys

```
vs2::> vserver export-policy rule show -policyname expol2 -vserver vs1
-ruleindex 21
```

```

Vserver: vs2
Policy Name: expol2
Rule Index: 21
Access Protocol: nfs
Client Match Hostname, IP Address, Netgroup, or Domain:
@dev_netgroup_main
RO Access Rule: sys
RW Access Rule: krb5
User ID To Which Anonymous Users Are Mapped: 65535
Superuser Security Types: krb5
Honor SetUID Bits in SETATTR: true
Allow Creation of Devices: true
```

Create a volume or qtree storage container

Create a volume

You can create a volume and specify its junction point and other properties by using the `volume create` command.

What you'll need

The SVM security style must be UNIX, and NFS should be set up and running.

About this task

A volume must include a *junction path* for its data to be made available to clients. You can specify the junction path when you create a new volume. If you create a volume without specifying a junction path, you must *mount* the volume in the SVM namespace using the `volume mount` command.

Steps

1. Create the volume with a junction point:

```
volume create -vserver vserver_name -volume volume_name -aggregate
```

```
aggregate_name -size {integer[KB|MB|GB|TB|PB]} -security-style unix -user
user_name_or_number -group group_name_or_number -junction-path junction_path
[-policy export_policy_name]
```

The choices for `-junction-path` are the following:

- Directly under root, for example, `/new_vol`

You can create a new volume and specify that it be mounted directly to the SVM root volume.

- Under an existing directory, for example, `/existing_dir/new_vol`

You can create a new volume and specify that it be mounted to an existing volume (in an existing hierarchy), expressed as a directory.

If you want to create a volume in a new directory (in a new hierarchy under a new volume), for example, `/new_dir/new_vol`, then you must first create a new parent volume that is junctioned to the SVM root volume. You would then create the new child volume in the junction path of the new parent volume (new directory).

+ If you plan to use an existing export policy, you can specify it when you create the volume. You can also add an export policy later with the `volume modify` command.

2. Verify that the volume was created with the desired junction point:

```
volume show -vserver vs1.example.com -volume volume_name -junction
```

Examples

The following command creates a new volume named `users1` on the SVM `vs1.example.com` and the aggregate `aggr1`. The new volume is made available at `/users`. The volume is 750 GB in size, and its volume guarantee is of type `volume` (by default).

```
cluster1::> volume create -vserver vs1.example.com -volume users
-aggregate aggr1 -size 750g -junction-path /users
[Job 1642] Job succeeded: Successful

cluster1::> volume show -vserver vs1.example.com -volume users -junction
```

		Junction		Junction
Vserver	Volume	Active	Junction Path	Path Source
vs1.example.com	users1	true	/users	RW_volume

The following command creates a new volume named "home4" on the SVM "vs1.example.com" and the aggregate "aggr1". The directory `/eng/` already exists in the namespace for the vs1 SVM, and the new volume is made available at `/eng/home`, which becomes the home directory for the `/eng/` namespace. The volume is 750 GB in size, and its volume guarantee is of type `volume` (by default).

```
cluster1::> volume create -vserver vs1.example.com -volume home4
-aggregate aggr1 -size 750g -junction-path /eng/home
[Job 1642] Job succeeded: Successful
```

```
cluster1::> volume show -vserver vs1.example.com -volume home4 -junction
```

Vserver	Volume	Active	Junction Path	Junction Path Source
vs1.example.com	home4	true	/eng/home	RW_volume

Create a qtree

You can create a qtree to contain your data and specify its properties by using the `volume qtree create` command.

What you'll need

- The SVM and the volume that will contain the new qtree must already exist.
- The SVM security style must be UNIX, and NFS should be set up and running.

Steps

1. Create the qtree:

```
volume qtree create -vserver vserver_name { -volume volume_name -qtree
qtree_name | -qtree-path qtree path } -security-style unix [-policy
export_policy_name]
```

You can specify the volume and qtree as separate arguments or specify the qtree path argument in the format `/vol/volume_name/_qtree_name`.

By default, qtrees inherit the export policies of their parent volume, but they can be configured to use their own. If you plan to use an existing export policy, you can specify it when you create the qtree. You can also add an export policy later with the `volume qtree modify` command.

2. Verify that the qtree was created with the desired junction path:

```
volume qtree show -vserver vserver_name { -volume volume_name -qtree
qtree_name | -qtree-path qtree path }
```

Example

The following example creates a qtree named `qt01` located on SVM `vs1.example.com` that has a junction path `/vol/data1`:

```
cluster1::> volume qtree create -vserver vs1.example.com -qtree-path  
/vol/data1/qt01 -security-style unix  
[Job 1642] Job succeeded: Successful
```

```
cluster1::> volume qtree show -vserver vs1.example.com -qtree-path  
/vol/data1/qt01
```

```
          Vserver Name: vs1.example.com  
          Volume Name: data1  
          Qtree Name: qt01  
Actual (Non-Junction) Qtree Path: /vol/data1/qt01  
          Security Style: unix  
          Oplock Mode: enable  
          Unix Permissions: ---rwxr-xr-x  
          Qtree Id: 2  
          Qtree Status: normal  
          Export Policy: default  
Is Export Policy Inherited: true
```

Secure NFS access using export policies

Secure NFS access using export policies

You can use export policies to restrict NFS access to volumes or qtrees to clients that match specific parameters. When provisioning new storage, you can use an existing policy and rules, add rules to an existing policy, or create a new policy and rules. You can also check the configuration of export policies



Beginning with ONTAP 9.3, you can enable export policy configuration checking as a background job that records any rules violations in an error rule list. The `vserver export-policy config-checker` commands invoke the checker and display results, which you can use to verify your configuration and delete erroneous rules from the policy. The commands only validate export configuration for host names, netgroups, and anonymous users.

Manage the processing order of export rules

You can use the `vserver export-policy rule setindex` command to manually set an existing export rule's index number. This enables you to specify the precedence by which ONTAP applies export rules to client requests.

About this task

If the new index number is already in use, the command inserts the rule at the specified spot and reorders the list accordingly.

Step

1. Modify the index number of a specified export rule:

```
vserver export-policy rule setindex -vserver virtual_server_name -policyname  
policy_name -ruleindex integer -newruleindex integer
```

Example

The following command changes the index number of an export rule at index number 3 to index number 2 in an export policy named rs1 on the SVM named vs1:

```
vs1::> vserver export-policy rule setindex -vserver vs1  
-policyname rs1 -ruleindex 3 -newruleindex 2
```

Assign an export policy to a volume

Each volume contained in the SVM must be associated with an export policy that contains export rules for clients to access data in the volume.

About this task

You can associate an export policy to a volume when you create the volume or at any time after you create the volume. You can associate one export policy to the volume, although one policy can be associated to many volumes.

Steps

1. If an export policy was not specified when the volume was created, assign an export policy to the volume:

```
volume modify -vserver vserver_name -volume volume_name -policy  
export_policy_name
```

2. Verify that the policy was assigned to the volume:

```
volume show -volume volume_name -fields policy
```

Example

The following commands assign the export policy nfs_policy to the volume vol1 on the SVM vs1 and verify the assignment:

```
cluster::> volume modify -vserver vs1 -volume vol1 -policy nfs_policy  
  
cluster::> volume show -volume vol1 -fields policy  
vserver volume      policy  
-----  
vs1      vol1      nfs_policy
```

Assign an export policy to a qtree

Instead of exporting an entire volume, you can also export a specific qtree on a volume to

make it directly accessible to clients. You can export a qtree by assigning an export policy to it. You can assign the export policy either when you create a new qtree or by modifying an existing qtree.

What you'll need

The export policy must exist.

About this task

By default, qtrees inherit the parent export policy of the containing volume if not otherwise specified at the time of creation.

You can associate an export policy to a qtree when you create the qtree or at any time after you create the qtree. You can associate one export policy to the qtree, although one policy can be associated with many qtrees.

Steps

1. If an export policy was not specified when the qtree was created, assign an export policy to the qtree:

```
volume qtree modify -vserver vserver_name -qtree-path  
/vol/volume_name/qtree_name -export-policy export_policy_name
```

2. Verify that the policy was assigned to the qtree:

```
volume qtree show -qtree qtree_name -fields export-policy
```

Example

The following commands assign the export policy `nfs_policy` to the qtree `qt1` on the SVM `vs1` and verify the assignment:

```
cluster::> volume modify -vserver vs1 -qtree-path /vol/vol1/qt1 -policy  
nfs_policy  
  
cluster::>volume qtree show -volume vol1 -fields export-policy  
vserver volume qtree export-policy  
-----  
vs1      data1  qt01  nfs_policy
```

Verify NFS client access from the cluster

You can give select clients access to the share by setting UNIX file permissions on a UNIX administration host. You can check client access by using the `vserver export-policy check-access` command, adjusting the export rules as necessary.

Steps

1. On the cluster, check client access to exports by using the `vserver export-policy check-access` command.

The following command checks read/write access for an NFSv3 client with the IP address 1.2.3.4 to the

volume home2. The command output shows that the volume uses the export policy `exp-home-dir` and that access is denied.

```
cluster1::> vserver export-policy check-access -vserver vs1 -client-ip
1.2.3.4 -volume home2 -authentication-method sys -protocol nfs3 -access
-type read-write
```

Path	Policy	Policy Owner	Policy Owner Type	Rule Index	Access
/	default	vs1_root	volume	1	read
/eng	default	vs1_root	volume	1	read
/eng/home2	exp-home-dir	home2	volume	1	denied

3 entries were displayed.

2. Examine the output to determine whether the export policy works as intended and the client access behaves as expected.

Specifically, you should verify which export policy is used by the volume or qtree and the type of access the client has as a result.

3. If necessary, reconfigure the export policy rules.

Test NFS access from client systems

After you verify NFS access to the new storage object, you should test the configuration by logging in to an NFS administration host and reading data from and writing data to the SVM. You should then repeat the process as a non-root user on a client system.

What you'll need

- The client system must have an IP address that is allowed by the export rule you specified earlier.
- You must have the login information for the root user.

Steps

1. On the cluster, verify the IP address of the LIF that is hosting the new volume:

```
network interface show -vserver svm_name
```

2. Log in as the root user to the administration host client system.
3. Change the directory to the mount folder:

```
cd /mnt/
```

4. Create and mount a new folder using the IP address of the SVM:

- a. Create a new folder:

```
mkdir /mnt/folder
```

- b. Mount the new volume at this new directory:

```
mount -t nfs -o hard IPAddress:/volume_name /mnt/folder
```

- c. Change the directory to the new folder:

```
cd folder
```

The following commands create a folder named test1, mount the vol1 volume at the 192.0.2.130 IP address on the test1 mount folder, and change to the new test1 directory:

```
host# mkdir /mnt/test1
host# mount -t nfs -o hard 192.0.2.130:/vol1 /mnt/test1
host# cd /mnt/test1
```

5. Create a new file, verify that it exists, and write text to it:

- a. Create a test file:

```
touch filename
```

- b. Verify that the file exists.:

```
ls -l filename
```

- c. Enter:

```
cat > filename
```

Type some text, and then press Ctrl+D to write text to the test file.

- d. Display the content of the test file.

```
cat filename
```

- e. Remove the test file:

```
rm filename
```

- f. Return to the parent directory:

```
cd ..
```

```
host# touch myfile1
host# ls -l myfile1
-rw-r--r-- 1 root root 0 Sep 18 15:58 myfile1
host# cat >myfile1
This text inside the first file
host# cat myfile1
This text inside the first file
host# rm -r myfile1
host# cd ..
```

6. As root, set any desired UNIX ownership and permissions on the mounted volume.

7. On a UNIX client system identified in your export rules, log in as one of the authorized users who now has access to the new volume, and repeat the procedures in steps 3 to 5 to verify that you can mount the volume and create a file.

Copyright information

Copyright © 2022 NetApp, Inc. All Rights Reserved. Printed in the U.S. No part of this document covered by copyright may be reproduced in any form or by any means—graphic, electronic, or mechanical, including photocopying, recording, taping, or storage in an electronic retrieval system—without prior written permission of the copyright owner.

Software derived from copyrighted NetApp material is subject to the following license and disclaimer:

THIS SOFTWARE IS PROVIDED BY NETAPP “AS IS” AND WITHOUT ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, WHICH ARE HEREBY DISCLAIMED. IN NO EVENT SHALL NETAPP BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

NetApp reserves the right to change any products described herein at any time, and without notice. NetApp assumes no responsibility or liability arising from the use of products described herein, except as expressly agreed to in writing by NetApp. The use or purchase of this product does not convey a license under any patent rights, trademark rights, or any other intellectual property rights of NetApp.

The product described in this manual may be protected by one or more U.S. patents, foreign patents, or pending applications.

LIMITED RIGHTS LEGEND: Use, duplication, or disclosure by the government is subject to restrictions as set forth in subparagraph (b)(3) of the Rights in Technical Data -Noncommercial Items at DFARS 252.227-7013 (FEB 2014) and FAR 52.227-19 (DEC 2007).

Data contained herein pertains to a commercial product and/or commercial service (as defined in FAR 2.101) and is proprietary to NetApp, Inc. All NetApp technical data and computer software provided under this Agreement is commercial in nature and developed solely at private expense. The U.S. Government has a non-exclusive, non-transferrable, nonsublicensable, worldwide, limited irrevocable license to use the Data only in connection with and in support of the U.S. Government contract under which the Data was delivered. Except as provided herein, the Data may not be used, disclosed, reproduced, modified, performed, or displayed without the prior written approval of NetApp, Inc. United States Government license rights for the Department of Defense are limited to those rights identified in DFARS clause 252.227-7015(b) (FEB 2014).

Trademark information

NETAPP, the NETAPP logo, and the marks listed at <http://www.netapp.com/TM> are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.