



Azure NetApp Files

Trident

NetApp
February 02, 2026

Table of Contents

- Azure NetApp Files 1
 - Configure an Azure NetApp Files backend 1
 - Azure NetApp Files driver details 1
 - Managed identities for AKS 1
 - Cloud identity for AKS 2
 - Prepare to configure an Azure NetApp Files backend 4
 - Prerequisites for NFS and SMB volumes 4
 - Additional requirements for SMB volumes 6
- Azure NetApp Files backend configuration options and examples 7
 - Backend configuration options 7
 - Example configurations 10
 - Storage class definitions 17
 - Create the backend 20

Azure NetApp Files

Configure an Azure NetApp Files backend

You can configure Azure NetApp Files as the backend for Trident. You can attach NFS and SMB volumes using an Azure NetApp Files backend. Trident also supports credential management using managed identities for Azure Kubernetes Services (AKS) clusters.

Azure NetApp Files driver details

Trident provides the following Azure NetApp Files storage drivers to communicate with the cluster. Supported access modes are: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Driver	Protocol	volumeMode	Access modes supported	File systems supported
azure-netapp-files	NFS SMB	Filesystem	RWO, ROX, RWX, RWOP	nfs, smb

Considerations

- The Azure NetApp Files service does not support volumes smaller than 50 GiB. Trident automatically creates 50-GiB volumes if a smaller volume is requested.
- Trident supports SMB volumes mounted to pods running on Windows nodes only.

Managed identities for AKS

Trident supports [managed identities](#) for Azure Kubernetes Services clusters. To take advantage of streamlined credential management offered by managed identities, you must have:

- A Kubernetes cluster deployed using AKS
- Managed identities configured on the AKS kubernetes cluster
- Trident installed that includes the `cloudProvider` to specify "Azure".

Trident operator

To install Trident using the Trident operator, edit `tridentorchestrator_cr.yaml` to set `cloudProvider` to "Azure". For example:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

Helm

The following example installs Trident sets `cloudProvider` to Azure using the environment variable `$CP`:

```
helm install trident trident-operator-100.2506.0.tgz --create
--namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

`tridentctl`

The following example installs Trident and sets the `cloudProvider` flag to Azure:

```
tridentctl install --cloud-provider="Azure" -n trident
```

Cloud identity for AKS

Cloud identity enables Kubernetes pods to access Azure resources by authenticating as a workload identity instead of by providing explicit Azure credentials.

To take advantage of cloud identity in Azure, you must have:

- A Kubernetes cluster deployed using AKS
- Workload identity and `oidc-issuer` configured on the AKS Kubernetes cluster
- Trident installed that includes the `cloudProvider` to specify "Azure" and `cloudIdentity` specifying workload identity

Trident operator

To install Trident using the Trident operator, edit `tridentorchestrator_cr.yaml` to set `cloudProvider` to "Azure" and set `cloudIdentity` to `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.

For example:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxxx' # Edit
```

Helm

Set the values for **cloud-provider (CP)** and **cloud-identity (CI)** flags using the following environment variables:

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx'"
```

The following example installs Trident and sets `cloudProvider` to Azure using the environment variable `$CP` and sets the `cloudIdentity` using the environment variable `$CI`:

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

`tridentctl`

Set the values for **cloud provider** and **cloud identity** flags using the following environment variables:

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxxx"
```

The following example installs Trident and sets the `cloud-provider` flag to `$CP`, and `cloud-identity` to `$CI`:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Prepare to configure an Azure NetApp Files backend

Before you can configure your Azure NetApp Files backend, you need to ensure the following requirements are met.

Prerequisites for NFS and SMB volumes

If you are using Azure NetApp Files for the first time or in a new location, some initial configuration is required to set up Azure NetApp files and create an NFS volume. Refer to [Azure: Set up Azure NetApp Files and create an NFS volume](#).

To configure and use an [Azure NetApp Files](#) backend, you need the following:



- `subscriptionID`, `tenantID`, `clientID`, `location`, and `clientSecret` are optional when using managed identities on an AKS cluster.
- `tenantID`, `clientID`, and `clientSecret` are optional when using a cloud identity on an AKS cluster.

- A capacity pool. Refer to [Microsoft: Create a capacity pool for Azure NetApp Files](#).
- A subnet delegated to Azure NetApp Files. Refer to [Microsoft: Delegate a subnet to Azure NetApp Files](#).
- `subscriptionID` from an Azure subscription with Azure NetApp Files enabled.
- `tenantID`, `clientID`, and `clientSecret` from an [App Registration](#) in Azure Active Directory with sufficient permissions to the Azure NetApp Files service. The App Registration should use either:
 - The Owner or Contributor role [predefined by Azure](#).
 - A [custom Contributor role](#) at the subscription level (`assignableScopes`) with the following permissions that are limited to only what Trident requires. After creating the custom role, [assign the role using the Azure portal](#).

Custom contributor role

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",
          "Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",
          "Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",
          "Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/delete",
```

```

        "Microsoft.Features/features/read",
        "Microsoft.Features/operations/read",
        "Microsoft.Features/providers/features/read",

        "Microsoft.Features/providers/features/register/action",

        "Microsoft.Features/providers/features/unregister/action",

        "Microsoft.Features/subscriptionFeatureRegistrations/read"
    ],
    "notActions": [],
    "dataActions": [],
    "notDataActions": []
}
]
}
}

```

- The Azure location that contains at least one [delegated subnet](#). As of Trident 22.01, the location parameter is a required field at the top level of the backend configuration file. Location values specified in virtual pools are ignored.
- To use Cloud Identity, get the client ID from a [user-assigned managed identity](#) and specify that ID in `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx`.

Additional requirements for SMB volumes

To create an SMB volume, you must have:

- Active Directory configured and connected to Azure NetApp Files. Refer to [Microsoft: Create and manage Active Directory connections for Azure NetApp Files](#).
- A Kubernetes cluster with a Linux controller node and at least one Windows worker node running Windows Server 2022. Trident supports SMB volumes mounted to pods running on Windows nodes only.
- At least one Trident secret containing your Active Directory credentials so Azure NetApp Files can authenticate to Active Directory. To generate secret `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- A CSI proxy configured as a Windows service. To configure a `csi-proxy`, refer to [GitHub: CSI Proxy](#) or [GitHub: CSI Proxy for Windows](#) for Kubernetes nodes running on Windows.

Azure NetApp Files backend configuration options and examples

Learn about NFS and SMB backend configuration options for Azure NetApp Files and review configuration examples.

Backend configuration options

Trident uses your backend configuration (subnet, virtual network, service level, and location), to create Azure NetApp Files volumes on capacity pools that are available in the requested location and match the requested service level and subnet.

Azure NetApp Files backends provide these configuration options.

Parameter	Description	Default
version		Always 1
storageDriverName	Name of the storage driver	"azure-netapp-files"
backendName	Custom name of the storage backend	Driver name + "_" + random characters
subscriptionID	The subscription ID from your Azure subscription Optional when managed identities is enabled on an AKS cluster.	
tenantID	The tenant ID from an App Registration Optional when managed identities or cloud identity is used on an AKS cluster.	
clientID	The client ID from an App Registration Optional when managed identities or cloud identity is used on an AKS cluster.	
clientSecret	The client secret from an App Registration Optional when managed identities or cloud identity is used on an AKS cluster.	
serviceLevel	One of Standard, Premium, or Ultra	"" (random)

Parameter	Description	Default
location	Name of the Azure location where the new volumes will be created Optional when managed identities is enabled on an AKS cluster.	
resourceGroups	List of resource groups for filtering discovered resources	[] (no filter)
netappAccounts	List of NetApp accounts for filtering discovered resources	[] (no filter)
capacityPools	List of capacity pools for filtering discovered resources	[] (no filter, random)
virtualNetwork	Name of a virtual network with a delegated subnet	""
subnet	Name of a subnet delegated to Microsoft.Netapp/volumes	""
networkFeatures	Set of VNet features for a volume, may be Basic or Standard. Network Features is not available in all regions and might have to be enabled in a subscription. Specifying networkFeatures when the functionality is not enabled causes volume provisioning to fail.	""
nfsMountOptions	Fine-grained control of NFS mount options. Ignored for SMB volumes. To mount volumes using NFS version 4.1, include nfsvers=4 in the comma-delimited mount options list to choose NFS v4.1. Mount options set in a storage class definition override mount options set in backend configuration.	"nfsvers=3"
limitVolumeSize	Fail provisioning if the requested volume size is above this value	"" (not enforced by default)

Parameter	Description	Default
debugTraceFlags	Debug flags to use when troubleshooting. Example, <code>\{"api": false, "method": true, "discovery": true\}</code> . Do not use this unless you are troubleshooting and require a detailed log dump.	null
nasType	Configure NFS or SMB volumes creation. Options are <code>nfs</code> , <code>smb</code> or <code>null</code> . Setting to <code>null</code> defaults to NFS volumes.	<code>nfs</code>
supportedTopologies	Represents a list of regions and zones that are supported by this backend. For more information, refer to Use CSI Topology .	
qosType	Represents the QoS type: Auto or Manual.	Auto
maxThroughput	Sets the maximum throughput allowed in MiB/sec. Supported only for manual QoS capacity pools.	4 MiB/sec



For more information on Network Features, refer to [Configure network features for an Azure NetApp Files volume](#).

Required permissions and resources

If you receive a "No capacity pools found" error when creating a PVC, it is likely your app registration doesn't have the required permissions and resources (subnet, virtual network, capacity pool) associated. If debug is enabled, Trident will log the Azure resources discovered when the backend is created. Verify an appropriate role is being used.

The values for `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork`, and `subnet` can be specified using short or fully-qualified names. Fully-qualified names are recommended in most situations as short names can match multiple resources with the same name.



If the vNet is located in a different resource group from the Azure NetApp Files (ANF) storage account, specify the resource group for the virtual network while configuring the `resourceGroups` list for the backend.

The `resourceGroups`, `netappAccounts`, and `capacityPools` values are filters that restrict the set of discovered resources to those available to this storage backend and may be specified in any combination. Fully-qualified names follow this format:

Type	Format
Resource group	<resource group>
NetApp account	<resource group>/<netapp account>
Capacity pool	<resource group>/<netapp account>/<capacity pool>
Virtual network	<resource group>/<virtual network>
Subnet	<resource group>/<virtual network>/<subnet>

Volume provisioning

You can control default volume provisioning by specifying the following options in a special section of the configuration file. Refer to [Example configurations](#) for details.

Parameter	Description	Default
<code>exportRule</code>	Export rules for new volumes. <code>exportRule</code> must be a comma-separated list of any combination of IPv4 addresses or IPv4 subnets in CIDR notation. Ignored for SMB volumes.	"0.0.0.0/0"
<code>snapshotDir</code>	Controls visibility of the <code>.snapshot</code> directory	"true" for NFSv4 "false" for NFSv3
<code>size</code>	The default size of new volumes	"100G"
<code>unixPermissions</code>	The unix permissions of new volumes (4 octal digits). Ignored for SMB volumes.	"" (preview feature, requires whitelisting in subscription)

Example configurations

The following examples show basic configurations that leave most parameters to default. This is the easiest way to define a backend.

Minimal configuration

This is the absolute minimum backend configuration. With this configuration, Trident discovers all of your NetApp accounts, capacity pools, and subnets delegated to Azure NetApp Files in the configured location, and places new volumes on one of those pools and subnets randomly. Because `nasType` is omitted, the `nfs` default applies and the backend will provision for NFS volumes.

This configuration is ideal when you are just getting started with Azure NetApp Files and trying things out, but in practice you are going to want to provide additional scoping for the volumes you provision.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
  tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
  clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
  clientSecret: SECRET
  location: eastus
```

Managed identities for AKS

This backend configuration omits `subscriptionID`, `tenantID`, `clientID`, and `clientSecret`, which are optional when using managed identities.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    - resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/eastus-anf-subnet
```

Cloud identity for AKS

This backend configuration omits `tenantID`, `clientID`, and `clientSecret`, which are optional when using a cloud identity.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

Specific service level configuration with capacity pool filters

This backend configuration places volumes in Azure's `eastus` location in an `Ultra` capacity pool. Trident automatically discovers all of the subnets delegated to Azure NetApp Files in that location and places a new volume on one of them randomly.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
```

Backend example with manual QoS capacity pools

This backend configuration places volumes in Azure's `eastus` location with manual QoS capacity pools.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anfl
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```


Advanced configuration

This backend configuration further reduces the scope of volume placement to a single subnet, and also modifies some volume provisioning defaults.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
virtualNetwork: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: "true"
  size: 200Gi
  unixPermissions: "0777"
```

Virtual pool configuration

This backend configuration defines multiple storage pools in a single file. This is useful when you have multiple capacity pools supporting different service levels and you want to create storage classes in Kubernetes that represent those. Virtual pool labels were used to differentiate the pools based on performance.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
  - application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
  - labels:
      performance: gold
      serviceLevel: Ultra
      capacityPools:
        - application-group-1/netapp-account-1/ultra-1
        - application-group-1/netapp-account-1/ultra-2
      networkFeatures: Standard
  - labels:
      performance: silver
      serviceLevel: Premium
      capacityPools:
        - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
      serviceLevel: Standard
      capacityPools:
        - application-group-1/netapp-account-1/standard-1
        - application-group-1/netapp-account-1/standard-2
```

Supported topologies configuration

Trident facilitates provisioning of volumes for workloads based on regions and availability zones. The `supportedTopologies` block in this backend configuration is used to provide a list of regions and zones per backend. The region and zone values specified here must match the region and zone values from the labels on each Kubernetes cluster node. These regions and zones represent the list of permissible values that can be provided in a storage class. For storage classes that contain a subset of the regions and zones provided in a backend, Trident creates volumes in the mentioned region and zone. For more information, refer to [Use CSI Topology](#).

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

Storage class definitions

The following `StorageClass` definitions refer to the storage pools above.

Example definitions using `parameter.selector` field

Using `parameter.selector` you can specify for each `StorageClass` the virtual pool that is used to host a volume. The volume will have the aspects defined in the chosen pool.

```
---  
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
  name: gold  
provisioner: csi.trident.netapp.io  
parameters:  
  selector: performance=gold  
allowVolumeExpansion: true
```

```
---  
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
  name: silver  
provisioner: csi.trident.netapp.io  
parameters:  
  selector: performance=silver  
allowVolumeExpansion: true
```

```
---  
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
  name: bronze  
provisioner: csi.trident.netapp.io  
parameters:  
  selector: performance=bronze  
allowVolumeExpansion: true
```

Example definitions for SMB volumes

Using `nasType`, `node-stage-secret-name`, and `node-stage-secret-namespace`, you can specify an SMB volume and provide the required Active Directory credentials.

Basic configuration on default namespace

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Using different secrets per namespace

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

Using different secrets per volume

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```



`nasType: smb` filters for pools which support SMB volumes. `nasType: nfs` or `nasType: null` filters for NFS pools.

Create the backend

After you create the backend configuration file, run the following command:

```
tridentctl create backend -f <backend-file>
```

If the backend creation fails, something is wrong with the backend configuration. You can view the logs to determine the cause by running the following command:

```
tridentctl logs
```

After you identify and correct the problem with the configuration file, you can run the create command again.

Copyright information

Copyright © 2026 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.