Virtual File Services

Developer Kit

August 2015

Novell

Legal Notices

Novell, Inc. makes no representations or warranties with respect to the contents or use of this documentation, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to revise this publication and to make changes to its content, at any time, without obligation to notify any person or entity of such revisions or changes.

Further, Novell, Inc. makes no representations or warranties with respect to any software, and specifically disclaims any express or implied warranties of merchantability or fitness for any particular purpose. Further, Novell, Inc. reserves the right to make changes to any and all parts of Novell software, at any time, without any obligation to notify any person or entity of such changes.

Any products or technical information provided under this Agreement may be subject to U.S. export controls and the trade laws of other countries. You agree to comply with all export control regulations and to obtain any required licenses or classification to export, re-export, or import deliverables. You agree not to export or re-export to entities on the current U.S. export exclusion lists or to any embargoed or terrorist countries as specified in the U.S. export laws. You agree to not use deliverables for prohibited nuclear, missile, or chemical biological weaponry end uses. Please refer to http://www.novell.com/info/exports/ (http://www.novell.com/info/exports/) for more information on exporting Novell software. Novell assumes no responsibility for your failure to obtain any necessary export approvals.

Copyright © 2001- 2015 Novell, Inc. All rights reserved. No part of this publication may be reproduced, photocopied, stored on a retrieval system, or transmitted without the express written consent of the publisher.

Novell, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at http://www.novell.com/company/legal/patents/ (http://www.novell.com/company/legal/patents/) and one or more additional patents or pending patent applications in the U.S. and in other countries.

Novell, Inc.
1800 South Novell Place
Provo, UT 84606
U.S.A.
www.novell.com

Online Documentation: To access the online documentation for this and other Novell developer products, and to get updates, see developer.novell.com/ndk. To access online documentation for Novell products, see www.novell.com/documentation.

Novell Trademarks

For Novell trademarks, see the Novell Trademark and Service Mark list (http://www.novell.com/company/legal/trademarks/tmlist.html).

Third-Party Materials

All third-party trademarks are the property of their respective owners.

Contents

About This Guide **11**

1 Basic Concepts **13**

1.1 Overview . . . 13
1.2 VFS vs Traditional File System Access . . . 14
1.3 Virtual File Composition . . . 14
1.4 Cautions . . . 15
1.4.1 eDirectory Name Formats . . . 15
1.4.2 Multiple Readers and Writers. . . . 15
1.4.3 Maximum Lengths . . . 16
1.4.4 Read and Write Offsets . . . 16
1.4.5 Client-Side Caching . . . 16
1.4.6 Device Sharing . . . 16
1.5 Partitions . . . 16
1.5.1 Number of Partitions . . . 17
1.5.2 Partition Types . . . 17
1.5.3 Partition IDs and Media Manager Objects . . . 17
1.5.4 Mirrored Partitions . . . 18
1.5.5 Shared Clustering Partitions . . . 18
1.6 Pools . . . 19
1.6.1 Freeze and Thaw Functionality . . . 19
1.7 Volumes . . . 20
1.7.1 Encrypted Volumes . . . 20
1.7.2 EVS Tests . . . 22
1.7.3 Console Commands . . . 22
1.8 Junctions . . . 22
1.9 Command Definitions . . . 22
1.10 Statistics . . . 23

2 What's New or Changed for Virtual File Services **25**

2.1 Updates to `files.cmd` and `user.cmd` Definitions . . . 25
2.2 Updates to `TrusteeInfo.xml` and `UserInfo.xml` Files . . . 25
2.3 Updates to `manage.cmd` definitions . . . 26

3 manage.cmd Definitions **27**

3.1 AFP . . . 29
getServerConfiguration (AFP) . . . 30
setServerConfiguration (AFP) . . . 31
3.2 Adapter. . . . 32
getAdapterInfo . . . 33
listAdapters . . . 35
3.3 Authorize System . . . 36
addTrustee . . . 37
3.4 CIFS . . . 39
addContext . . . 40
addDomainACL . . . 41
addShare . . . 42
createContextList . . . 43

createDomain . . . 44
deleteDomain . . . 46
findContext . . . 47
getCreateContextListStatus . . . 48
getDomainConfiguration . . . 49
getImportWindowsUsersStatus . . . 51
getServerConfiguration . . . 52
getShareProperties . . . 55
importWindowsUsers . . . 56
joinDomain . . . 57
leaveDomain . . . 59
listContexts . . . 60
listDomainControllers . . . 61
listImportedUsers . . . 63
listShares . . . 64
modifyContextList . . . 66
modifyShare . . . 67
removeContext . . . 68
removeShare . . . 69
setDomainConfiguration . . . 70
setServerConfiguration . . . 72
3.5 Deleted Volume . . . 74
continueState . . . 75
pauseState . . . 77
purgeVolume . . . 78
salvageVolume . . . 79
3.6 Device . . . 82
getDeviceInfo . . . 83
getDeviceInfo2 . . . 85
getDeviceInfo3 (Linux) . . . 89
getPathInfo . . . 94
initializeDevice . . . 96
listDevices . . . 98
listDevicePartitions . . . 101
listDevicePools . . . 102
listMultiPaths . . . 103
modifyDevice . . . 104
multiPath . . . 106
renameDevice . . . 110
scanDevices . . . 111
3.7 DFS . . . 112
createLink . . . 113
deleteLink . . . 115
getDfsGUID . . . 116
initDFSGUIDs . . . 118
modifyLink . . . 119
readLink . . . 121
setDfsGUID . . . 123
3.8 Directory Quota . . . 125
addQuota (obsolete) . . . 126
3.9 Junction . . . 127
createJunction . . . 128
deleteJunction . . . 130
3.10 LSS . . . 131
getLSSInfo . . . 132
getLSSVolumeInfo . . . 137
3.11 Partition . . . 140
addPartition . . . 141
addPartition2 . . . 145

addPartitionToMirror . . . 146
getPartitionInfo . . . 148
getPartitionMirrorStats . . . 150
listPartitions . . . 152
modifyPartition . . . 156
removePartition . . . 158
removePartitionFromMirror . . . 160
resyncPartitionMirror . . . 161
3.12 Pool . . . 162
activatePoolSnapshot . . . 163
activatePoolSnapshot (Linux) . . . 164
addPool . . . 166
addPool2 . . . 169
addPoolSnapshot . . . 171
addPoolSnapshot (Linux) . . . 172
deactivatePoolSnapshot . . . 174
deactivatePoolSnapshot (Linux) . . . 175
expandPool . . . 176
expandPool2 . . . 178
getDefaultClusterNames . . . 179
getNDSName . . . 181
getPoolDevices . . . 183
getPoolInfo . . . 184
getPoolSnapshotInfo . . . 189
getPoolSnapshotInfo (Linux) . . . 190
getState . . . 192
listPools . . . 194
listPoolSnapshots . . . 196
listPoolSnapshots (Linux) . . . 198
modifyPoolInfo . . . 200
modifyState . . . 202
poolFreeze . . . 204
poolFreezeStatus . . . 206
poolThaw . . . 210
removePool . . . 212
removePool2 . . . 214
removePoolSnapshot . . . 215
removePoolSnapshot (Linux) . . . 216
renamePool . . . 217
renamePoolSnapshot . . . 219
3.13 RAID . . . 220
addRAID . . . 221
addRAID2 . . . 224
expandRAID . . . 226
removeRAID . . . 228
removeRAID2 . . . 229
renameRAID . . . 230
restripeRAID . . . 231
3.14 Server . . . 232
getServerFreeSpace . . . 233
listDevices (Server) . . . 235
listPartitions (Server) . . . 236
listPools . . . 239
3.15 User Space Restriction . . . 239
browseUserSpaceRestrictions (user) . . . 240
getUserSpaceRestriction (user) . . . 242
setUserSpaceRestriction (user) . . . 244
listUserSpaceRestrictions (user) . . . 246
3.16 VLDB . . . 248

addVolumeToVLDB . . . 249
createNewService . . . 251
deleteService . . . 253
getVLDBInfo . . . 255
loadVLDB . . . 263
lookup . . . 264
removeVolumeFromVLDB . . . 266
replicaAddedToVLDB . . . 267
replicaRemovedFromVLDB . . . 268
setVLDBConfiguration . . . 269
shutdownVLDB . . . 271
startRepair . . . 272
startService . . . 274
stopRepair . . . 275
stopService . . . 276
3.17 Volume Operations . . . 277
addTraditionalVolume . . . 278
addVolume . . . 281
expandTraditionalVolume . . . 284
getNDSName (Volume) . . . 286
getState . . . 287
getTraditionalVolumeInfo . . . 289
getVolumeInfo . . . 292
listVolumes . . . 300
listEvmsVolumes . . . 301
modifyState . . . 302
modifyVolumeInfo . . . 304
removeUser . . . 308
removeVolume . . . 310
renameVolume . . . 312
3.18 Volume MN Operations . . . 313
changeJobState . . . 314
createJob . . . 315
getJobList . . . 317
getJobStatus . . . 318
listSkippedFiles . . . 320

4 User Commands 323

browseUserSpaceRestrictions (user) . . . 324
getUserSpaceRestriction (user) . . . 326
setUserSpaceRestriction (user) . . . 328
listUserSpaceRestrictions (user) . . . 330

5 NDS.cmd Definitions 333

5.1 Object Operations . . . 334
getAttribute . . . 335
5.2 Pool Operations . . . 336
addPool . . . 337
removePool . . . 339
5.3 Volume Operations . . . 341
addVolume . . . 342
removeVolume . . . 344
5.4 User Operations . . . 346
addUser . . . 347
removeUser . . . 349

6 FileEvents.xml Definitions 351

changeEventEpoch . . . 352
getEFLNameSpaceID . . . 353
getInactiveEpochInterval . . . 354
listAllFiles . . . 355
listEpochs . . . 356
listFileEvents . . . 357
pingEpoch . . . 358
removeEventEpoch . . . 359
resetEventList . . . 360
setEFLNameSpaceID . . . 361
setInactiveEpochInterval . . . 362
startEventEpoch . . . 363
stopEventEpoch . . . 364

7 files.cmd Definitions 365

addQuota . . . 366
addTrustee . . . 367
getAllEffectiveRights . . . 369
getEffectiveRightsByID . . . 371
getEffectiveRightsByUser . . . 372
getFileInfo . . . 373
modifyInheritedRightsFilter . . . 379
purgeDeletedFile . . . 381
removeAllTrustees . . . 382
removeTrustee . . . 384
salvageDeletedFile . . . 385
scanSalvageableFiles . . . 388
setFileInfo . . . 394

8 Inventory.xml Definitions 397

NRMServerInventory.xml . . . 398
Volume_Inventory.xml . . . 402
Volume_Trustees.xml . . . 405

9 Archive Definitions 407

9.1 archiveAdmin.cmd Definitions . . . 408
activateJob . . . 409
deactivateJob . . . 410
getInfo . . . 411
getJobInfo . . . 413
getLogTimeRange . . . 416
listJobNames . . . 417
queryLog . . . 418
setInfo . . . 421
startJob . . . 423
stopJob . . . 424
testFilter . . . 425
9.2 archive.cmd Definitions . . . 426
deleteFile . . . 427
getContentVersions . . . 429
getDirContents . . . 432

getVersions . . . 434
restoreFile . . . 437
shutdown . . . 439

10 linux.cmd Definitions 441

poolIDToName . . . 442
uidToEquivalentGUIDs . . . 443
userIDToName . . . 444
volumeIDFileIDToPath . . . 445
volumeIDToName . . . 446
addPool . . . 447
addVolume . . . 448
poolUpdate . . . 449
getVolumeInfo . . . 450
resetIDs . . . 451

11 Advanced Concepts 453

11.1 Transformation Templates . . . 453
11.1.1 Datastreams . . . 455
11.2 Virtual I/O Commands . . . 456

12 Values 457

12.1 Device Types . . . 457
12.2 Enabled Attributes Bits . . . 457
12.3 Job Types . . . 458
12.4 Mirror Group Statuses . . . 458
12.5 NSS Volume States . . . 458
12.6 Pool States . . . 459
12.7 Pool Types . . . 459
12.8 State Values . . . 459
12.9 Traditional Volume States . . . 460
12.10 Volume States . . . 461
12.11 Volume Types . . . 461

13 Functions 463

MGMT_FindFirstElement . . . 464
MGMT_MakeCommandVirtualFile . . . 465
MGMT_MakeCommandVirtualFileWithHelp . . . 466
MGMT_MakeFunctionVirtualFile . . . 467
VIRT_AddResultData . . . 468
VIRT_AddResultElement . . . 469
VIRT_AddResultTag . . . 470
VIRT_MakeResultsImportant . . . 471
VIRT_MakeResultsNormal . . . 472
VIRT_ResetResult . . . 473
XML_BackwardFindEndTag . . . 474
XML_findEndOfNonWhiteSpace . . . 475
XML_ForwardFindTag . . . 476
XML_GetNextTag . . . 477
XML_GetTagElement . . . 478

14 Examples **479**

14.1 Creating a Virtual File 479
14.2 Accessing a Virtual File with Perl 481
14.2.1 Reading a Datastream 481
14.2.2 Writing a Datastream 482
14.2.3 Writing a Command 482

A Revision History **485**

About This Guide

Virtual File Services (VFS) provides methods that allow you to manage services such as Novell® Storage Services (NSS) and Novell eDirectory™ using standard file system functions. Using VFS and a scripting or GUI-based interface, you can view the status and statistics for your system and change the system parameters.

This guide contains the following sections:

- Chapter 1, “Basic Concepts,” on page 13
- Chapter 2, “What’s New or Changed for Virtual File Services,” on page 25
- Chapter 3, “manage.cmd Definitions,” on page 27
- Chapter 4, “User Commands,” on page 323
- Chapter 5, “NDS.cmd Definitions,” on page 333
- Chapter 6, “FileEvents.xml Definitions,” on page 351
- Chapter 7, “files.cmd Definitions,” on page 365
- Chapter 8, “Inventory.xml Definitions,” on page 397
- Chapter 9, “Archive Definitions,” on page 407
- Chapter 10, “linux.cmd Definitions,” on page 441
- Chapter 11, “Advanced Concepts,” on page 453
- Chapter 12, “Values,” on page 457
- Chapter 13, “Functions,” on page 463
- Chapter 14, “Examples,” on page 479
- Appendix A, “Revision History,” on page 485

Audience

This guide is intended for developers interested in using standard file system functions to manage Novell services.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the User Comment feature at the bottom of each page of the online documentation.

Additional Information

For the related developer support postings for Virtual File Services, see the Developer Support Forums (http://developer.novell.com/ndk/devforums.htm).

Documentation Updates

For the most recent version of this guide, see the Virtual File Services NDK page (http://developer.novell.com/ndk/vfs.htm).

Docuentation Conventions

In Novell documentation, a greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path.

A trademark symbol (®, ™, etc.) denotes a Novell trademark. An asterisk (*) denotes a third-party trademark.

1 Basic Concepts

Virtual File Services (VFS) provides file system API access to non-file system entities through six basic file system functions: create, delete, open, close, read, and write.

VFS allows not only the file system, but any service provided by a server to be managed through simple file system functions. Because Web browsers operate through simple functions, this functionality opens the ability to have virtual files that generate XML or HTML. Managing your file system is now as easy as opening your favorite browser and accessing the appropriate files.

You can also create your own virtual files using more advanced features that are discussed in later sections.

TIP: Use the CDATA element to pass strings that contain special characters. CDATA is an XML standard, and the information in a CDATA element is not parsed.

The following topics are discussed in this section:

- Section 1.1, “Overview,” on page 13
- Section 1.2, “VFS vs Traditional File System Access,” on page 14
- Section 1.3, “Virtual File Composition,” on page 14
- Section 1.4, “Cautions,” on page 15
- Section 1.5, “Partitions,” on page 16
- Section 1.6, “Pools,” on page 19
- Section 1.7, “Volumes,” on page 20
- Section 1.8, “Junctions,” on page 22
- Section 1.9, “Command Definitions,” on page 22
- Section 1.10, “Statistics,” on page 23

1.1 Overview

NSS provides a special administration volume—known as the admin volume—that exists on all servers. This volume uses no disk space and is created at startup time. Using VFS and the services provided by files that are created on the admin volume, your administrator can potentially control all server management functions.

On NetWare: At Novell, NSS uses VFS to create all of the files on the _Admin volume. Using VFS, the NSS team accepts commands and returns information dynamically. For example, they can use VFS to read a file that returns the statistics for cache buffer utilization. The contents of that file are generated in real time. They can also use files on the _Admin volume to control all of their storage management by sending XML commands and reading the resulting XML stream. Both ConsoleOne and NORM use this same method to handle their storage needs.

There are three basic virtual files provided with this release:

- _Admin/manage_NSS/manage.cmd

- _Admin/manage_NSS/nds.cmd
- _Admin/manage_OS/setparmcontrol.cmd

These files are predefined by NSS and support a predefined XML syntax for managing NSS with basic commands. Since they are predefined, these files cannot be deleted or created.

On Linux: At Novell, NSS uses VFS to create all of the files on the _admin volume. Using VFS, the NSS team accepts commands and returns information dynamically. For example, they can use VFS to read a file that returns the statistics for cache buffer utilization. The contents of that file are generated in real time. They can also use files on the _admin volume to control all of their storage management by sending XML commands and reading the resulting XML stream. Both ConsoleOne and NORM use this same method to handle their storage needs.

There are three basic virtual files provided with this release:

- _admin/Manage_NSS/manage.cmd
- _admin/Manage_NSS/nds.cmd
- _admin/Manage_OS/setparmcontrol.cmd

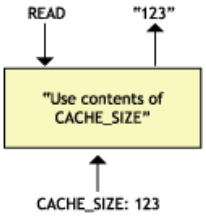
1.2 VFS vs Traditional File System Access

Normally, you write data to a file with the intention of retrieving it at a later time. The file system stores your original data to some type of persistent media and ensures that the data returned is the same data that you originally stored.

Virtual files are treated much like regular files and can be deleted, created, opened, closed, read, and written. However, virtual files do not contain data and nothing is persistently stored in these files. Instead, a virtual file is configured at creation with information that tells the file system how to generate data for read operations and how to process data for write operations.

For example, a virtual file might represent a memory location that contains the amount of memory to be used in file system caching. When a read is posted to the file, the contents of the memory location are read, converted to ASCII, and the generated number is used to satisfy the read request. Data that is written to the file is converted from ASCII into a number and used to change the contents of the memory location. This process is demonstrated in the following graphic.

Figure 1-1 *Virtual File Memory Contents*



1.3 Virtual File Composition

A virtual file has both of the following elements:

- A behavior definition, which is also known as a Transformation Template (see "Transformation Templates" on page 453) and provides instructions to the virtual file system so it can render virtual data in an appropriate manner
- Virtual contents, which is the content generated at the time the file is read

The transformation template (behavior definition) for a virtual file is created by opening the file and writing a sequence of XML-based commands to the file.

The predefined command files, `manage.cmd` and `nds.cmd`, contain predefined transformation templates, which you should not modify.

How does the system know if you are writing a transformation template or the virtual contents of the file? The `virtualIO` XML element must be written to the file before you send the transformation template. This element is used for many virtual file I/O commands on the admin volume and includes a definition for the behavior of the virtual file.

1.4 Cautions

When using virtual files, you should be aware of the following issues:

- Section 1.4.1, “eDirectory Name Formats,” on page 15
- Section 1.4.2, “Multiple Readers and Writers,” on page 15
- Section 1.4.3, “Maximum Lengths,” on page 16
- Section 1.4.4, “Read and Write Offsets,” on page 16
- Section 1.4.5, “Client-Side Caching,” on page 16
- Section 1.4.6, “Device Sharing,” on page 16

1.4.1 eDirectory Name Formats

Older XML requires eDirectory™ names in backslash format. For example

```
\novell_inc\novell\prv\nss\randys
```

However, in VFS, you can enter eDirectory names in any format and it is changed internally to the backslash format.

You can input names in the following eDirectory container format (with or without the leading dot):

```
.cn=admin.o=novell
```

However, container format is not required.

1.4.2 Multiple Readers and Writers

To manage consistency, when virtual reads reference memory or a function, the results are put in a buffer that is specific to that instance of opening the file. However, this process means that if you close and reopen a file, the results from any previous commands are no longer accessible.

Also, read operations that start reading at an offset other than zero reference the results buffer without refreshing it. Read operations at offset zero usually refresh the read information. However, if it is a function-type datastream that does not specify a read function, you must write to the file to refresh the data.

1.4.3 Maximum Lengths

The results buffer is initially set to 8K. However, read and write operations can change the size of the buffer so that responses are limited only by the amount of free memory that is available.

The transformation template is compiled at the time it is written to the file and held in memory. Generally, the template does not use very much memory. However, “data” type datastreams can consume more memory than is initially set.

1.4.4 Read and Write Offsets

Because you are writing the virtual I/O commands to a file, by default, the offset of any following read and write operations is set to occur just past the data that was supposedly written. VFS tracks the length of the virtual commands and adjusts the offset so that any succeeding read and write operations appear to start at offset zero.

You must be careful when you seek to an offset other than zero in a virtual file since the offset might not be where you expected it.

The file system examines incoming offsets. As long as these offsets are greater than the length of the original virtual I/O command, they are adjusted to account for that length. However, if an offset is ever less than the length of the virtual I/O command, the file system assumes that a seek has been performed and no longer makes an adjustment. All future seek operations to location zero start at the front of the data being rendered by the virtual file and all subsequent seek operations start at the actual position in the result buffer.

If you are going to read from a file without closing and reopening it, you should seek to location zero after writing to the same file.

1.4.5 Client-Side Caching

Due to the dynamic results of virtual files, no client-side caching should be performed on virtual files.

Any oplock requests are rejected for virtual files.

1.4.6 Device Sharing

If a device is shareable and you intend to use it in a cluster, you must manually mark the state of the device as shared (because Novell Storage Services (NSS) cannot automatically detect the intended use of a device). By marking a device “shared,” all partitions, NSS storage pools, and NSS logical volumes created on that device are automatically marked “shareable for clustering.”

1.5 Partitions

Before using any of the partition operations listed in “Partition” on page 140, you should have a basic understanding of the following topics that explain how partitions work on NetWare®:

- Section 1.5.1, “Number of Partitions,” on page 17
- Section 1.5.2, “Partition Types,” on page 17
- Section 1.5.3, “Partition IDs and Media Manager Objects,” on page 17
- Section 1.5.4, “Mirrored Partitions,” on page 18
- Section 1.5.5, “Shared Clustering Partitions,” on page 18

1.5.1 Number of Partitions

A disk can be divided into a maximum of four partitions, which is the same rule that NetWare follows with versions prior to NetWare 6.0. Starting with NetWare 6.0, the NetWare media manager abstracts NetWare partitions to include one additional level so that a single physical NetWare partition can contain an unlimited number of virtual NetWare partitions.

When you use the XML commands for "Partition" on page 140, each virtual NetWare partition appears as a real physical partition. Only when you are examining the disk by using a DOS-based partition management utility can you see that there is actually one physical partition.

1.5.2 Partition Types

NetWare supports multiple partition types. Some of these types can be managed by the XML commands in "Partition" on page 140, while other types are merely recognized by the listPartitions (page 152) command.

The following partition types can be created and/or managed by XML commands:

Type 0 (0x00) -- Free Space Partition type

This partition type is not actually a real partition. It is used in the listPartitions (page 152) command to represent any unpartitioned free space that exists on a device.

Type 101 (0x65) -- Traditional NetWare Partition type

This type of partition is used by the traditional NetWare file system. Traditional NetWare volumes contain one or more partitions (or pieces of partitions) of this type.

Type 105 (0x69) -- NSS Partition type

This type of partition is used by the NSS file system. NSS pools contain one or more partitions of this type.

Type 207 (0xCF) -- Virtual Device Partition type

This type of partition is used by the media manager to construct virtual RAID devices. A partition that is created with this type appears to the file system as a device that can be added into a software RAID configuration.

Type 127 -- iSCSI Partition type

This type of partition is used by the iSCSI target software which exposes such partition as an iSCSI target.

All other partition types, such as DOS type 4 or type 6 partitions, and clustering SBD partitions show up in the partition list returned by the listPartitions (page 152) command. However, these types cannot be managed in any other way using the XML commands.

1.5.3 Partition IDs and Media Manager Objects

A partition ID is a number that is automatically assigned to a partition by the media manager when a NetWare server is booted. The ID for a partition is not guaranteed to be the same number each time the server is booted.

The NetWare media manager maintains an in-memory database of all disk-related objects (including, but not limited to, adapters, devices, physical partitions, HotFix objects, and mirror objects). Every object in this database has a media manager ID that is assigned at boot time.

If a partition is used with mirroring, it has three different media manager objects that can be used to manage it:

- A physical partition ID, which represents the raw physical partition
- A logical HotFix partition ID
- A logical mirror group partition ID

With NetWare 6.0, a HotFix (bad block redirection) area is also required in order to implement mirroring, so the media manager maintains a HotFix object for the partition as well. This HotFix object has a “logical” HotFix partition ID, which is unique and is different from the “physical” partition ID.

Each mirror group (group of partitions that are mirrored to each other) is also represented by another media manager object that has a “logical” mirror group partition ID.

If two partitions are mirrored to each other, both have unique and separate physical partition IDs and logical HotFix partition IDs, but they share a single logical mirror group partition ID.

When you create an NSS pool or a traditional NetWare volume, you must specify the ID of the partition to be added. For the XML commands, you should always use the most abstract ID of the partition. For example, if the partition does not support HotFix and mirroring, the most abstract ID is the physical partition ID.

However, if the partition has a HotFix and mirror group object, you should use the logical partition ID. (A single mirror group object can represent multiple physical partitions that are mirrored to each other. The only unique way to represent creating a pool or volume on the mirrored group of partitions is to specify the partition ID that represents the entire mirror group.)

1.5.4 Mirrored Partitions

In order to mirror two partitions together, both partitions must have HotFix and Mirror objects. Both partitions must also have identical data sizes. If a partition is too big or too small to mirror to another partition, the data size of the partition can be controlled by increasing or decreasing the size of the HotFix area so that the data sizes are identical. A partition's HotFix area can be from 200-245,760 sectors.

When you create a partition, you can create it in its own standalone mirror group so that it can be combined with other partitions at a later time. You can also create a partition and add it directly to an existing mirror group.

1.5.5 Shared Clustering Partitions

When you create a partition on a device that is marked as “shareable for clustering,” the partition inherits that state from the device. Operations that create and delete partitions check the “shareable for clustering” state of the device and require that the clustering software be loaded and operational before allowing the partition deletion or creation on such devices.

The ignoreShareState element can be specified on these operations to prevent the state from being checked for clustering.

1.6 Pools

Before you create any NSS logical volumes, you must first create an NSS storage pool on which to place the volumes. The commands in "Pool" on page 162 create, delete, and manipulate NSS storage pools. For information about freezing and thawing pools and why this functionality is useful, see "Freeze and Thaw Functionality" on page 19.

Prior to NetWare 6.0, all volumes were directly associated with physical storage. With NetWare 6.0 and later, the physical storage is separated from the volumes, and NSS storage pools now exist.

An NSS storage pool is a group of one or more NSS partitions, while an NSS volume is a logical entity that contains user data and is assigned to reside on an NSS pool. The location of an NSS logical volume is not physically tied to a specific NSS storage pool, and a single NSS storage pool can contain multiple NSS logical volumes.

How you design an NSS logical volume allows it to be migrated from one pool to another and allows it to be replicated across multiple NSS storage pools on different servers.

1.6.1 Freeze and Thaw Functionality

The freeze and thaw functions (see poolFreeze (page 204), poolFreezeStatus (page 206), and poolThaw (page 210)) help you ensure all the data in your pool is consistent. For example, a snapshot NLM can freeze and thaw a pool to ensure that snapshots contain consistent data for snapshot and server applications.

You must register your application to receive the two following events, which inform your application when to freeze and when to thaw:

- "NSS.PoolFreeze" on page 19
- "NSS.PoolThaw" on page 20

These events are not consumable. You cannot call file system functions during the event, and you must return quickly (in less than half a second) so that other applications can be notified. Most of the work to ensure that data is synchronized and consistent is performed on an application thread and not during an event.

NSS does not prevent multiple freezes from occurring at the same time on a specific server. However, NSS prevents multiple freezes on the same pool at the same time.

NSS.PoolFreeze

The NSS.PoolFreeze event notifies your application that another application wants to synchronize all the data on a specific pool. If the freeze cannot happen quickly, your event handler must start up a thread to finish the work.

Your application must then synchronize all its data on all volumes in the specified pool. This data must remain consistent until your application receives the NSS.PoolThaw event.

To receive a list of all volumes in a pool, call getPoolInfo (page 184). To receive a list of pools on a specified server, call listPools (page 194). To retrieve the pool that a specified volume is in, call getVolumeInfo (page 292).

NSS.PoolThaw

The NSS.PoolThaw event notifies your application that you can thaw your data on the specified pool. If the thaw cannot happen quickly, your event handler must start up a thread to finish the work.

Once your application returns from handling the NSS.PoolThaw event, you can receive another NSS.PoolFreeze event on the same pool.

1.7 Volumes

For additional information about Volumes, see the following subsections:

- Section 1.7.1, “Encrypted Volumes,” on page 20
- Section 1.7.2, “EVS Tests,” on page 22
- Section 1.7.3, “Console Commands,” on page 22

The commands listed in “Volume Operations” on page 277 can be called to manipulate both NSS logical volumes and traditional NetWare volumes.

NSS logical volumes reside inside of NSS storage pools. Instead of being assigned to occupy an exact amount of physical space, they are assigned a maximum quota size. This size can be a specific amount, or it may be limited by the size of the available space in the NSS storage pool. An NSS logical volume can grow to be as big as its assigned quota, but it occupies only as much physical space in the NSS storage pool as is needed to store the current data that is owned by the volume.

NSS logical volumes can be in one of the following states:

- deactive — Indicates that the volume is not currently activated or is not currently available for use.
- active — Indicates that the volume is currently activated and is available for use by the File System Services (64-Bit) (http://developer.novell.com/wiki/index.php/File_System_Services_%2864-Bit%29).
- mounted — Indicates that the volume is currently activated and is also mounted by the traditional NetWare file system. A volume in this state can be accessed by the traditional NetWare file system APIs and NCPs and the 64-Bit File System Services functions.

NOTE: The XML commands do not report a dismounted state. If an NSS logical volume is active but not currently mounted, it is in the "active" state.

Traditional NetWare volumes exist in, and are managed by, the traditional NetWare file system. Traditional volumes directly consume physical partitions or portions of physical partitions. These volumes can be in only a mounted or dismounted state and cannot be accessed using the 64-Bit File System Services functions. However, they can be accessed using the traditional file system APIs and NCPs (see the documentation for NetWare Core Protocols (http://developer.novell.com/wiki/index.php/Category:Novell_Developer_Kit_Unsupported)).

1.7.1 Encrypted Volumes

Encrypted Volume Support (EVS) is available in NetWare 6.5 SP2 and later.

EVS provides a mechanism to store user data in an encrypted form on NSS volumes while continuing to use most applications, NLMs, and backup utilities that currently work with NSS.

The basic feature set and internal operations of EVS is as follows:

- Any NSS volume (with the exception of the sys volume) can be designated an encrypted volume at the time it is created. The encrypted volume attribute remains with the volume throughout its existence. A volume cannot be later converted to be an unencrypted volume once it is designated as encrypted. The encryption functionality must be designated at the time it is created.
- Currently, only NSSMU version 3.20, build 940 or later can be used to encrypt a volume. At volume creation time, NSS prompts you to designate the volume as encrypted. If encryption is selected, NSS prompts you for a password and accepts the designation. Passwords can be any character string 16 characters or less in length.
- When a password is detected, NSS consults with the NICI libraries to generate a 128-bit AES key that remains associated with the volume. The password is then used to wrap the key and other volume-specific cryptographic information into 128-byte packages that are persistently stored in two locations in NSS. The primary location is in the volume data block. The second location is in the volume locator beast. One the password is used to wrap the cryptographic data, the password is deleted from memory; and the volume is marked with the encrypted attribute (which is part of the volume-specific persistent data).
- When a volume is activated, its persistent data is loaded from the volume data block. If the volume has the encrypted attributed, a memory list of volume names and keys is consulted to see if this volume has a known key. If the key is present, it's used. If the key is not present, the list of volumes and passwords is consulted. If a password is available, it is used to unwrap the key from the persistent data and the new key is added to the volume and key list.
- Once the encrypted volume is activated, all encryption operations are transparent to file system applications that call file I/O functions. Data that is written to files is held in cache until the time that the data is normally written. At the time the data is physically written, the data is encrypted to a temporary write buffer, which is then delivered to the lower-level zlss function. After the data is written, the buffer is returned to an available list and the clear-text data remains in the cache. During reads, the cache is consulted to determine if a requested block is already in memory. If it is in memory, the clear-text data is transferred. If the requested block is not in cache, a physical read request is made, with the read being directed to a temporary buffer. After the read completion (but before control is returned to the calling function), the encrypted data in the temporary buffer is decrypted into a cache buffer, and the temporary buffer is assigned back to an available list. The read then proceeds as usual, and the clear-text data is made available to all future requestors.
- To see new XML tags and volume attributes that were added to accommodate EVS, see
  - addVolume (page 281)
  - getVolumeInfo (page 292)
  - modifyVolumeInfo (page 304)

Because direct I/O to an encrypted volume bypasses the encryption engine and potentially allows a mix of encrypted and non-encrypted data on the same volume, you should avoid it.

Encrypted volumes increment the media major version number when a volume is created, so that encrypted volumes cannot be activated by releases prior to NetWare 6.5 SP2. The volume cannot be rolled back from SP2 to SP1. If you attempt such a rollback, the volume fails to activate and you won't be able to repair the pool. To effectively rollback an encrypted volume, the system administrator must move the user data off the encrypted volume. For example, by backing up a volume to volume copy.

If you archive files from an encrypted volume to a unencrypted volume, those files are stored in an non-encrypted state. If you want files to be archived in an encrypted state, the destination path for your archive manager must be on an encrypted volume.

1.7.2 EVS Tests

When testing an encrypted volume, use all the tests available for testing any other volume. Encryption should be transparent above the physical read/write layer of zlss, so applications should run without any changes. All the rules of rights, trustees, ownership, sharing, visibility, locking, transactions, restrictions, etc., remain the same. The only noticeable difference between encrypted and non-encrypted volumes during run time is that encrypted volumes run slower.

1.7.3 Console Commands

You can manipulate EVS using nssmu.nlm and iManager. You can also use console command lines to display a volume's status and to activate or mount and deactivate or dismount encrypted volumes. Some example console command contexts follow:

```
MYSERVER:NSS /activate=MYVOLUME:volPasswordXYZZY
MYSERVER:NSS /volumeActivate=MYVOLUME:volPasswordXYZZY
MYSERVER:NSS /activate=MYVOLUME
```

The last console command is followed by a prompt to enter an encrypted volume password.

If an encrypted volume has been activated after the server is brought up, no further passwords are required.

Note that it is not permissible to activate encrypted volumes using ALL as the volume name.

The status of an encrypted volume can be displayed using the following console command:

```
NSS /volumes
```

1.8 Junctions

For NetWare 6.5 SP1 and later, Virtual File Services supports junctions.

For each junction operation, the UI must generate the file data. However, a generic link file format has been defined that allows for a junction link file. The UI doesn't need to know the file format.

Whenever an ndsObject element is used, a fully distinguished, untyped name is expected. This name can include a leading dot and trailing tree name, but they are not necessary. If the tree name is omitted from the ndsObject element, it can be specified in the tgtTree element.

The following operations support junctions:

- createLink (page 113)
- deleteLink (page 115)
- readLink (page 121)

1.9 Command Definitions

This documentation provides the XML element definitions for manage.cmd and nds.cmd.

The following abbreviations are used to indicate if the element is

- Opt for optional. This is not a required element.
- Req for required. This element is required and is not optional.
- Rpt for repeating. You can include as many of these elements as you need.

To present XML elements as succinctly as possible, the end tags for various elements are not shown. When you use these elements, however, you must add an appropriate value and the end tag, as shown in the following examples:

The poolName element is often documented as

```
<poolName>       <!-- The name of the volume's pool -->
```

However, to indicate the SYS pool in the poolName element, you need to substitute the following line:

```
<poolName>SYS</poolName>
```

The value attribute of the nameSpaces element is often documented as

```
<nameSpaces value=" ">
```

To indicate a volume that supports the four basic namespaces, substitute the following line:

```
<nameSpaces value="23">DOS Long Macintosh Unix</nameSpaces>
```

Note that white space and new lines were added to the all examples for readability.

1.10 Statistics

The following xml files consists of the statistics:

- GeneralStats.xml
- FileStats.xml
- CompressionStats.xml
- AuthorizationCache.xml
- BufferCache.xml
- NameCache.xml
- ObjectCache.xml
- Thread.xml

2 What's New or Changed for Virtual File Services

With support for Active Directory introduced in OES 2015, new XML elements and attributes have been added to differentiate between Active Directory and eDirectory identities. This section provides a summary of changes to the VFS NDK.

- Section 2.1, "Updates to files.cmd and user.cmd Definitions," on page 25
- Section 2.2, "Updates to TrusteeInfo.xml and UserInfo.xml Files," on page 25
- Section 2.3, "Updates to manage.cmd definitions," on page 26

2.1 Updates to `files.cmd` and `user.cmd` Definitions

For the following requests, a new XML element `<includeADIdentities/>` is introduced to include Active Directory identities in the XML reply. When a user is referred to in the XML reply, a new attribute named type will specify value of the attribute as adID or edirID to indicate an Active Directory or an eDirectory identity. For AD identities, name is specified in the sAMAccountName format.

- getFileInfo (page 373).
- getAllEffectiveRights (page 369)
- scanSalvageableFiles (page 388)
- salvageDeletedFile (page 385)
- listUserSpaceRestrictions (user) (page 246)
- browseUserSpaceRestrictions (user) (page 240)

For the following XML requests, the user type is specified using the new attribute type, which will specify the value of the attribute as adID or edirID to indicate an Active Directory or an eDirectory identity. For Active Directory identities, name is specified in the `sAMAccountName` format.

- setFileInfo (page 394)
- addTrustee (page 367)
- removeTrustee (page 384)
- getUserSpaceRestriction (user) (page 242)
- setUserSpaceRestriction (user) (page 244)
- getEffectiveRightsByID (page 371)
- getEffectiveRightsByUser (page 372)

2.2 Updates to `TrusteeInfo.xml` and `UserInfo.xml` Files

For the `TrusteeInfo.xml` and `UserInfo.xml` files, when a user is referred to in the XML reply, a new attribute named type will specify the value of the attribute as adID or edirID to indicate an Active Directory or an eDirectory identity. For AD identities name is specified in the sAMAccountName format.

2.3 Updates to `manage.cmd` definitions

adEnabledMedia: New tag added to the `addpool2` request to upgrade the pool media format to support Active Directory identities. This tag is also added to the getPoolInfo and listPools reply to indicate that the pool is Active Directory-enabled.

force: New tag added to the `addpool2` request to forcefully add a pool in a mixed node cluster.

volumeADEnabled: New tag introduced in `modifyVolumeInfo` request to modify the attributes of the volume. This tag is also added to the `getVolumeInfo` reply to indicate that the volume is Active Directory-enabled.

3 manage.cmd Definitions

Use the following path name to open the manage.cmd file:

`_admin/Manage_NSS/manage.cmd (Linux)`

`_Admin/manage_NSS/manage.cmd (NetWare)`

Every time you open the `manage.cmd` file (and before you send other commands), you must write the following sequence to the file:

```
<virtualIO><datastream name="command"/></virtualIO>
```

This sequence notifies `manage.cmd` that you are ready to write XML commands and read the XML responses to those commands.

Every command is wrapped with either nssRequest or nssReply elements, as shown in the following examples:

```
<nssRequest>
   <afp>
      <getServerConfiguration>
   </afp>
</nssRequest>

<nssReply>
   <afp>
      <getServerConfiguration>
         <isOnline>
         <result value=" ">
            <description/>
         </result>
      </getServerConfiguration>
   </afp>
</nssReply>
```

This section contains definitions for the following command categories:

- Section 3.1, “AFP,” on page 29
- Section 3.2, “Adapter,” on page 32
- Section 3.3, “Authorize System,” on page 36
- Section 3.4, “CIFS,” on page 39
- Section 3.5, “Deleted Volume,” on page 74
- Section 3.6, “Device,” on page 82
- Section 3.7, “DFS,” on page 112
- Section 3.8, “Directory Quota,” on page 125
- Section 3.9, “Junction,” on page 127
- Section 3.10, “LSS,” on page 131
- Section 3.11, “Partition,” on page 140
- Section 3.12, “Pool,” on page 162
- Section 3.13, “RAID,” on page 220
- Section 3.14, “Server,” on page 232

- Section 3.15, “User Space Restriction,” on page 239
- Section 3.16, “VLDB,” on page 248
- Section 3.17, “Volume Operations,” on page 277
- Section 3.18, “Volume MN Operations,” on page 313

3.1 AFP

This section contains the following AFP commands:

- “getServerConfiguration (AFP)” on page 30
- “setServerConfiguration (AFP)” on page 31

Each command is wrapped with either the nssRequest or nssReply element and the afp element.

getServerConfiguration (AFP)

Returns AFP configuration information for the server.

Request

```
<getServerConfiguration>
```

Reply

```
<getServerConfiguration>
   <isOnline>
   <result value=" ">
      <description/>
   </result>
</getServerConfiguration>
```

Elements

isOnline

(Optional) Specifies that AFPTCP is currently loaded and online.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

setServerConfiguration (AFP)

Adds a trustee with the specified rights.

Request

```
<setServerConfiguration>
   <isOnline enabled = "">
</setServerConfiguration>
```

Reply

```
<setServerConfiguration>
   <result value=" ">
      <description/>
   </result>
</setServerConfiguration>
```

Elements

isOnline

(Optional)

result

Specifies a zError or eDirectory error value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

enabled

Indicates whether AFP is enabled:

yes AFPSTRT.NCF is added to AUTOEXEC.NCF and the AFP NLMs are loaded.

no AFPSTRT.NCF is removed from AUTOEXEC.NCF and the AFP NLMs are unloaded from the server

3.2 Adapter

This section contains the following Adapter commands:

- “getAdapterInfo” on page 33
- “listAdapters” on page 35

Each command is wrapped with either the nssRequest or nssReply element and the adapter element.

getAdapterInfo

Returns information about the adapter that was passed in. This command is implemented only on NetWare and not on Linux.

Request

```
<getAdapterInfo>
```

Reply

```
<adapterInfo>
   <adapterID/>
   <adapterName/>
   <adapterDriverID/>
   <adapterDriverName/>
   <adapterNumber/>
   <adapterSlot/>
   <adapterSubSystemID/>
   <adapterInterruptInfo>
      <adapterInterrupt/>
   </adapterInterruptInfo>
   <adapterDMAInfo>
      <adapterDMAChannel/>
   </adapterDMAInfo/>
   <adapterMemoryInfo>
      <adapterMemoryPhysicalAddress/>
      <adapterMemoryPhysicalLength/>
      <adapterMemoryVirtualAddress/>
   </adapterMemoryInfo>
   <adapterSupportedTargetIDs/>
   <adapterSupportedUnitNumbers/>
   <adapterCardTargetID/>
   <adapterFlags/>
   <adapterType/>
</adapterInfo>
```

Elements

adapterInfo

Occurs once.

adapterName

Specifies the name of the adapter.

adapterDriverID

Specifies the adapter's driver ID that was assigned by Novell® for the driver associated with this adapter.

adapterDriverName

Specifies the name of the driver that is operating the adapter.

adapterNumber

Specifies the number assigned by the IO subsystem that defines an instance of the adapter.

adapterSlot

Specifies the slot or Hardware Interface Number (HIN) that is assigned to the adapter.

adapterSubSystemID

(Optional) Specifies the ID of the IO subsystem that created and manages the adapter. This ID is displayed only if the subsystem ID is NWPA or CIOS.

adapterInterruptInfo

(Optional) Specifies the adapter interrupt information.

adapterInterrupt

Specifies the primary and secondary interrupts that are associated with the adapter.

adapterDMAInfo

(Optional) Specifies the adapter DMA channel information.

adapterDMAChannels

Specifies the primary and secondary DMA channels that are used with the adapter.

adapterMemoryInfo

(Optional) Specifies the adapter's physical and virtual memory information.

adapterMemoryPhysicalAddress

Specifies the physical address of the primary and secondary memory addresses.

adapterMemoryPhysicalLength

Specifies the length of the adapterMemoryPhysicalAddress element.

adapterMemoryVirtualAddress

Specifies the virtual address that is associated with the physical memory address.

adapterSupportedTargetIDs

(Optional) Specifies the number of target IDs that are supported by the adapter.

adapterSupportedUnitNumbers

(Optional) Specifies the number of unit numbers that are supported by the adapter.

adapterCardTargetID

(Optional) Specifies the target ID that the adapter supports.

adapterFlags

(Optional) Specifies any flags adapter is Instance_Unloaded_Support.

adapterTypes

(Optional) Specifies the type of the adapter (or the adapter's ID):

SCSI
IDB-ATA

listAdapters

Returns a list of all of a server's adapters. This command is implemented only on NetWare and not on Linux.

Request

```
<listAdapters>
```

Reply

```
<listAdapters>
   <adapterInfo>
     <adapterID/>
     <adapterName/>
   </adapterInfo>
</listAdapters>
```

Elements

adapterInfo

Repeats for each physical or RAID device or mirror group listed.

adapterID

Specifies the adapter's ID.

adapterName

Specifies the name of the adapter.

3.3 Authorize System

This section contains the following Authorize System commands:

- “addTrustee” on page 37

Each command is wrapped with either the nssRequest or nssReply element and the authorizeNW element.

addTrustee

Adds a trustee with the specified rights. This is an outdated version for adding trustee rights. It works okay, but it can be used only by the administrator and is strict about the way object names are specified. Use addTrustee (page 367) instead.

Request

```
<addTrustee>
   <name/>
   <context/>
   <rights/>
   <fileName/>
</addTrustee>
```

Reply

```
<addTrustee>
   <result value=" ">
      <description/>
   </result>
 </addTrustee>
```

Elements

name

(Required) Specifies the name of the user.

context

(Required) Specifies the eDirectory context for the user name, including the tree name.

rights

(Required) Specifies a string of characters that represents the NetWare® trustee rights mask, where each character is a specific granted right:

a access control
c create
e delete (or erase)
f view (or file scan)
m modify
r read
w write
s supervisor

For example, to grant read, write, and file scan access, use

```
<rights>rwf</rights>
```

fileName

(Required) Specifies the target file name (including the volume).

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

```
<addTrustee>
   <name>test</name>
   <context>novell.T=MYTREE</context>
   <rights>rfce</rights>
   <fileName>SYS:data/text.txt</filename>
</addTrustee>
```

3.4 CIFS

This section contains the following CIFS commands:

- “addContext” on page 40
- “addDomainACL” on page 41
- “addShare” on page 42
- “createContextList” on page 43
- “createDomain” on page 44
- “deleteDomain” on page 46
- “findContext” on page 47
- “getCreateContextListStatus” on page 48
- “getDomainConfiguration” on page 49
- “getImportWindowsUsersStatus” on page 51
- “getServerConfiguration” on page 52
- “getShareProperties” on page 55
- “importWindowsUsers” on page 56
- “joinDomain” on page 57
- “leaveDomain” on page 59
- “listContexts” on page 60
- “listDomainControllers” on page 61
- “listImportedUsers” on page 63
- “listShares” on page 64
- “modifyContextList” on page 66
- “modifyShare” on page 67
- “removeContext” on page 68
- “removeShare” on page 69
- “setDomainConfiguration” on page 70
- “setServerConfiguration” on page 72

Each command is wrapped with either the nssRequest or nssReply element and the cifs element.

addContext

Adds an eDirectory context to the CIFS user context search list.

Request

```
<addContext>
   <context/>
</addContext>
```

Reply

```
<addContext>
   <result value=" ">
      <description/>
   </result>
</addContext>
```

Elements

context

Specifies the eDirectory name (in typeless, distinguished-name format) of the user context to add.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

addDomainACL

Adds an eDirectory ACL for the specified domain at the specified context. This ACL gives rights to the domain's controller group to be able to work with users from a different part of the eDirectory tree. The ACL gives read/write privileges to the CIFS Login Script and RID attributes in user, group, container, and profile objects, which allows the controllers to manipulate these attributes at the specified context and below it in the tree.

Request

```
<addDomainACL>
   <domain/>
   <context/>
   <unp/>
   <user/>
   <password/>
</addDomainACL>
```

Reply

```
<addDomainACL>
   <result value=" ">
      <description/>
   </result>
</addDomainACL>
```

Elements

domain

Specifies the eDirectory name (in typeless, distinguished-name format) of the domain object that represents the domain for which an ACL is to be added.

context

Specifies the eDirectory name (in typeless, distinguished-name format) of the tree context where the ACL is to be added.

unp

(Optional) Specifies the NMAS-encrypted username and password that has rights to perform the operation at the specified context.

user

(Optional) Specifies the clear-text user name of a user that has rights to perform the operation. This element is used only if the unp element is not specified.

password

(Optional/Required if user is specified) Specifies the clear-text password of a user that has rights to perform the operation.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

addShare

Creates a CIFS share on the server.

Request

```
<addShare>
   <server/>
   <tree/>
   <shareName/>
   <pathName/>
   <comment/>
</addShare>
```

Reply

```
<addShare>
   <result value=" ">
     <description/>
   </result>
</addShare>
```

Elements

server

(Optional) Specifies the distinguished name of the eDirectory server object. The name must be a fully distinguished name, including the leading dot before the object name and the trailing dot after the tree name (or with no leading or trailing dot but a complete identity of the object within the current tree). The default is the eDirectory object for the physical server.

tree

(Optional) Specifies the name of the eDirectory tree that contains the server. The default is the tree of the physical server.

shareName

Specifies the NetBios name to give to the new share.

pathName

Specifies the full path of the new share. For example, sys:/system.

comment

(Optional) Specifies a description of the new share.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

createContextList

Begins the process of automatically generating the CIFS user context search list.

Request

```
<createContextList/>
```

Reply

```
<createContextList>
   <result value=" ">
      <description/>
   </result>
</createContextList>
```

Elements

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

createDomain

Creates a new CIFS PDC domain in the eDirectory tree and designates the specified server as the starting PDC.

Request

```
<createDomain>
   <server/>
   <context/>
   <domainName/>
   <comment/>
   <unp/>
   <user/>
   <password/>
</createDomain>
```

Reply

```
<createDomain>
   <result value=" ">
      <description/>
   </result>
</createDomain>
```

Elements

server

(Optional) Specifies the distinguished name of the eDirectory server object (must be a fully distinguished name with a leading dot before the object name and a trailing dot after the tree name or an identifier of the object within the current tree). The default is the eDirectory object for the physical server.

context

Specifies the eDirectory name (in typeless, distinguished-name format) of the tree context where the domain object is to be created.

domainName

Specifies the name of the new domain.

comment

(Optional) Specifies any descriptions or text strings relevant to the domain.

unp

(Optional) Specifies the NMAS-encrypted username and password that has rights to perform the operation at the specified context.

user

(Optional) Specifies the clear-text username of a user that has rights to perform the operation. The user element is present only if the unp element is not specified.

password

(Optional) Specifies the clear-text password of a user that has rights to perform the operation. The password element is present only if the unp element is not specified.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

deleteDomain

Creates a new CIFS PDC domain in the eDirectory tree and designates the specified server as the starting PDC.

Request

```
<deleteDomain>
   <server/>
   <domain/>
   <unp/>
   <user/>
   <password/>
</deleteDomain>
```

Reply

```
<deleteDomain>
   <result value=" ">
      <description/>
   </result>
</deleteDomain>
```

Elements

server

(Optional) Specifies the distinguished name of the eDirectory server object (must be a fully distinguished name with a leading dot before the object name and a trailing dot after the tree name or an identifier of the object within the current tree). The default is the eDirectory object for the physical server.

domain

Specifies the eDirectory name (in typeless, distinguished-name format) of the new domain object that represents the domain to be added.

unp

(Optional) Specifies the NMAS-encrypted username and password that has rights to perform the operation at the specified context.

user

(Optional) Specifies the clear-text username of a user that has rights to perform the operation. The user element is present only if the unp element is not specified.

password

(Optional/Required if user is specified) Specifies the clear-text password of a user that has rights to perform the operation. The password element is present only if the unp element is not specified.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

findContext

Finds a context in the CIFS user context search list.

Request

```
<findContext>
   <context/>
</findContext>
```

Reply

```
<findContext>
   <result value=" ">
      <description/>
   </result>
</findContext>
```

Elements

context

Specifies the eDirectory name (in typeless, distinguished-name format) of the user context to locate.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

getCreateContextListStatus

Returns the status of the createContextList (page 43) command.

Request

```
<getCreateContextListStatus>
```

Reply

```
<getCreateContextListStatus>
   <status/>
   <state>
     <lastCompletionTime/>
   </status>
   <result value=" ">
      <description/>
   </result>
</getCreateContextListStatus>
```

Elements

state

Specifies the state of the context:

running
notRunning

lastCompletionTime

Specifies a string representation of the UTC last completion time.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

getDomainConfiguration

Returns the CIFS configuration information for the domain.

Request

```
<getDomainConfiguration>
   <domain/>
   <unp/>
   <user/>
   <password/>
</getDomainConfiguration>
```

Reply

```
<getDomainConfiguration>
   <domainName/>
   <comment/>
   <pdc/>
   <group/>
   <sid/>
   <nextRID/>
   <epoch/>
   <result value=" ">
      <description/>
   </result>
</getDomainConfiguration>
```

Elements

domain

(Optional) Specifies the distinguished name of the eDirectory domain object (must be a fully distinguished name with a leading dot before the object name and a trailing dot after the tree name or an identifier of the object within the current tree). The default is the domain object that is associated with the CIFS server that is being talked to.

unp

(Optional) Specifies the NMAS-encrypted username and password that has rights to perform the operation at the specified context.

user

(Optional) Specifies the clear-text username of a user that has rights to perform the operation. The user element is present only if the unp element is not specified.

password

(Optional/Required if user is specified) Specifies the clear-text password of a user that has rights to perform the operation. The password element is present only if the unp element is not specified.

domainName

Specifies the NetBios name that was advertised by the domain.

comment

(Optional) Specifies any description of text associated with the domain.

pdc

Specifies the distinguished name of the PDC server object.

group

Specifies the DN of the domain controller group object.

sid

Specifies the SID for the domain.

nextRID

Specifies the next RID value that is not reserved yet.

epoch

Specifies the epoch number for this domain.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

getImportWindowsUsersStatus

Returns the status of the ImportWindowsUsers command.

Request

```
<getImportWindowsUsersStatus>
```

Reply

```
<getImportWindowsUsersStatus>
   <status>
      <state/>
      <lastCompletionTime/>
   </status>
   <result value=" ">
      <description/>
   </result>
</getImportWindowsUsersStatus>
```

Elements

state

Specifies the state:

running
notRunning

lastCompletionTime

Specifies the string representation of the UTC last completion time.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

getServerConfiguration

Returns CIFS configuration information for the server.

Request

```
<getServerConfiguration>
   <server/>
   <tree/>
</getServerConfiguration>
```

Reply

```
<getServerConfiguration>
   <cifsServerName/>
   <comment/>
   <winsIPAddress/>
   <authMode/>
   <groupName/>
   <pdcName/>
   <pdcIPAddress/>
   <attachIPAddresses>
      <ipAddress/>
   </attachIPAddresses>
   <isVirtual/>
   <isOnline/>
   <dfs/>
   <oplocks/>
   <smbSignatureMode/>
   <lmCompatibilityLevel/>
   <cifsProxyUserInfo/>
   <loginScripts/>
   <shareVolsByDefault/>
   <domain/>
   <beginRID/>
   <endRID/>
   <result value=" ">
      <description/>
   </result>
</getServerConfiguration>
```

Elements

server

(Optional) Specifies the distinguished name of the eDirectory server object. The name must be a fully distinguished name, including the leading dot before the object name and the trailing dot after the tree name (or with no leading or trailing dot but a complete identity of the object within the current tree). The default is the eDirectory object for the physical server.

tree

(Optional) Specifies the name of the eDirectory tree that contains the server. The default is the tree of the physical server.

cifsServerName

Specifies the NetBios name advertised by the server.

comment

(Optional) Specifies a text string associated with the CIFS server.

winsIPAddress

(Optional) Specifies the WINS server IP address in dotted ASCII notation. For example, 137.65.67.72.

authMode

Specifies the authentication mode:

local
domain
domainMember
domainController

The domain mode means the domain pass-through mode.

groupName

Specifies the workgroup (if in local mode) or the domain name.

pdcName

Specifies the NetBios name of the primary domain controller. Optional if the mode element specifies local, domainMember, or domainController.

pdcIPAddress

Specifies the IP address (in dotted ASCII notation) of the primary domain controller. Optional if the mode element is local, domainMember, or domainController.

attachIPAddresses

(Optional) Specifies the attach IP addresses that are currently assigned to the server.

isVirtual

(Optional) Specifies that the server is a cluster virtual server.

isOnline

(Optional) Specifies that the server is currently loaded and online.

dfs

(Optional) Specifies that CIFS is enabled for distributed file system support.

oplocks

(Optional) Specifies that CIFS is enabled for Oplocks support.

smbSignatureMode

Specifies the SMB signing supports message authentication, which prevents active message attacks.

lmCompatibilityLevel

Specifies the NTLM compatibility level that CIFS server is configured to use.

cifsProxyUserInfo

Specifies the CIFS proxy user name for the CIFS server.

loginScripts

(Optional) Specifies that CIFS is enabled for login script support.

shareVolsByDefault

(Optional) Specifies that the server exports all mounted volumes as shares by default. The server also exports any explicitly defined shares.

domain

(Optional) Specifies the distinguished name of the domain object. Valid only if the mode element is domainController or domainMember.

beginRID

(Optional) Specifies the beginning value in the range of reserved RID values for a domain controller. Valid only if the mode element is domainController.

endRID

(Optional) Specifies the ending value in the range of reserved RID values for a domain controller. Valid only if the mode element is domainController.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

getShareProperties

Returns CIFS configuration information for the server.

Request

```
<getShareProperties>
   <server/>
   <tree/>
   <shareName/>
</getShareProperties>
```

Reply

```
<getShareProperties>
   <shareInfo>
      <shareName/>
      <pathName/>
      <comment/>
   </shareInfo>
   <result value=" ">
      <description/>
   </result>
</getShareProperties>
```

Elements

server

(Optional) Specifies the distinguished name of the eDirectory server object. The name must be a fully distinguished name, including the leading dot before the object name and the trailing dot after the tree name (or with no leading or trailing dot but a complete identity of the object within the current tree). The default is the eDirectory object for the physical server.

tree

(Optional) Specifies the name of the eDirectory tree that contains the server. The default is the tree of the physical server.

shareName

Specifies the NetBios name to give to the share.

pathName

Specifies the full path of the new share. For example, sys:/system.

comment

(Optional) Specifies a description of the new share.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

importWindowsUsers

Begins the process of importing Windows users from a Primary Domain Controller from a company other than Novell.

Request

```
<importWindowsUsers/>
```

Reply

```
<importWindowsUsers>
   <result value=" ">
      <description/>
   </result>
</importWindowsUsers>
```

Elements

result

Specifies a zError or eDirectory error or 0 (for no error).

description

Specifies a text description of the result.

joinDomain

Joins the specified server into the existing domain identified by the domain distinguished name.

Request

```
<joinDomain>
   <server/>
   <domain/>
   <mode/>
   <unp/>
   <user/>
   <password/>
</joinDomain>
```

Reply

```
<joinDomain>
   <result value=" ">
      <description/>
   </result>
</joinDomain>
```

Elements

server

(Optional) Specifies the distinguished name of the eDirectory server object. The name must be a fully distinguished name, including the leading dot before the object name and the trailing dot after the tree name (or with no leading or trailing dot but a complete identity of the object within the current tree). The default is the eDirectory object for the physical server.

domain

Specifies the eDirectory name (in typeless, distinguished-name format) of the domain object that represents the domain to be joined.

mode

Specifies the mode of the server to be set in the server configuration:

domainMember

domainController

unp

(Optional) Specifies the NMAS-encrypted username and password that has rights to perform the operation at the specified context.

user

(Optional) Specifies the clear-text user name of a user that has rights to perform the operation. This element is used only if the unp element is not specified.

password

(Optional/Required if user is specified) Specifies the clear-text password of a user that has rights to perform the operation.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

leaveDomain

Leaves the specified domain that is identified by the domain distinguished name. This command must be issued to the physical server that owns the server object. The mode of the server is left in local mode after leaving the domain. If the server is not participating in the domain, the command fails. Also, if the server is the PDC of the domain, the command fails. The PDC cannot leave a domain.

Request

```
<leaveDomain>
   <server/>
   <domain/>
   <unp/>
   <user/>
   <password/>
</leaveDomain>
```

Reply

```
<leaveDomain>
   <result value=" ">
      <description/>
   </result>
</leaveDomain>
```

Elements

server

(Optional) Specifies the distinguished name of the eDirectory server object. The name must be a fully distinguished name, including the leading dot before the object name and the trailing dot after the tree name (or with no leading or trailing dot but a complete identity of the object within the current tree). The default is the eDirectory object for the physical server.

domain

Specifies the eDirectory name (in typeless, distinguished-name format) of the domain object that represents the domain to leave.

unp

(Optional) Specifies the NMAS-encrypted username and password that has rights to perform the operation at the specified context.

user

(Optional) Specifies the clear-text user name of a user that has rights to perform the operation. This element is used only if the unp element is not specified.

password

(Optional/Required if user is specified) Specifies the clear-text password of a user that has rights to perform the operation.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

listContexts

Returns all or a portion of the CIFS user context search list.

Request

```
<listContexts>
   <startIdx/>
   <numEntries/>
</listContexts>
```

Reply

```
<listContexts>
    <context/>
    <result value=" ">
       <description/>
    </result>
</listContexts>
```

Elements

startIdx

(Optional) Specifies the starting index of the first context to return. Defaults to 0 (start at the beginning).

numEntries

(Optional) Specifies the maximum number of contexts to return in reply. Defaults to return all contexts.

context

Specifies the eDirectory name (in typeless, distinguished-name format) of the user context list entry.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

listDomainControllers

Returns all or a portion of a domain's domain controller list.

Request

```
<listDomainControllers>
   <domain/>
   <startIdx/>
   <numEntries/>
   <unp/>
   <user/>
   <password/>
</listDomainControllers>
```

Reply

```
<listDomainControllers>
   <server/>
   <result value=" ">
      <description/>
   </result>
</listDomainControllers>
```

Elements

domain

(Optional) Specifies the distinguished name of the eDirectory domain object. The name must be a fully distinguished name, including the leading dot before the object name and the trailing dot after the tree name (or with no leading or trailing dot but a complete identity of the object within the current tree). The default is the domain object that is associated with the CIFS server you are talking to.

startIdx

(Optional) Specifies the starting index of the first domain controller to return. Defaults to 0 (start at the beginning).

numEntries

(Optional) Specifies the maximum number of domain controllers to return in the reply. Defaults to return all controllers.

unp

(Optional) Specifies the NMAS-encrypted username and password that has rights to perform the operation at the specified context.

user

(Optional) Specifies the clear-text user name of a user that has rights to perform the operation. This element is used only if the unp element is not specified.

password

(Optional/Required if user is specified) Specifies the clear-text password of a user that has rights to perform the operation.

server

Specifies the eDirectory name (in typeless, distinguished-name format) of the server that is acting as a domain controller in the domain.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

listImportedUsers

Returns all or a portion of the imported Windows user list.

Request

```
<listImportedUsers>
   <startIdx/>
   <numEntries/>
</listImportedUsers>
```

Reply

```
<listImportedUsers>
   <userName/>
   <result value=" ">
      <description/>
   </result>
</listImportedUsers>
```

Elements

startIdx

(Optional) Specifies the starting index of the first user to return. Default to 0 (start at the beginning).

numEntries

(Optional) Specifies the maximum number of users to return in the reply. Defaults to return all users.

userName

Specifies the imported user name.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

listShares

Returns share information for all shares on the server.

Request

```
<listShares type=" ">
   <server/>
   <tree/>
   <startIdx/>
   <numEntries/>
</listShares>
```

Reply

```
<listShares>
   <shareInfo>
      <shareName/>
      <pathName/>
      <comment/>
   </shareInfo>
   <result value=" ">
      <description/>
   </result>
</listShares>
```

Elements

listShares

Specifies what types of information to return for the shares.

server

(Optional) Specifies the distinguished name of the eDirectory server object. The name must be a fully distinguished name, including the leading dot before the object name and the trailing dot after the tree name (or with no leading or trailing dot but a complete identity of the object within the current tree). The default is the eDirectory object for the physical server.

tree

(Optional) Specifies the name of the eDirectory tree that contains the server. Defaults to the tree of the physical server.

startIdx

(Optional) Specifies the starting index of the first share to return. Defaults to 0 (start at the beginning).

numEntries

(Optional) Specifies the maximum number of shares to return. Defaults to return all shares.

shareInfo

Specifies the share information.

shareName

Specifies the share NetBios name that is advertised by CIFS. The share name is returned for both the basic and all type.

pathName

(Optional) Specifies the share path name. For example, sys:\system\. The path name is returned for only the all type.

comment

(Optional) Specifies a text string associated with the share. Comments are returned for only the all type.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

type

(Optional) Specifies what type of information to return:

basic

all

Defaults to basic information being returned.

modifyContextList

Modifies all or a portion of the CIFS user context search list.

Request

```
<modifyContextList
   <context>
      <find/>
      <replace/>
   </context>
</modifyContextList>
```

Reply

```
<modifyContextList>
   <result value=" ">
      <description/>
   </result>
</modifyContextList>
```

Elements

context

Specifies the context to modify.

find

Specifies the original name (in typeless, distinguished-name format) of the context to change.

replace

Specifies the new context name.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

modifyShare

Modifies all or a portion of the CIFS user context search list.

Request

```
<modifyShare>
   <server/>
   <tree/>
   <shareName/>
   <newShareName/>
   <pathName/>
   <comment/>
</modifyShare>
```

Reply

```
<modifyShare>
   <result value=" ">
      <description/>
   </result>
</modifyShare>
```

Elements

server

(Optional) Specifies the distinguished name of the eDirectory server object. The name must be a fully distinguished name, including the leading dot before the object name and the trailing dot after the tree name (or with no leading or trailing dot but a complete identity of the object within the current tree). The default is the eDirectory object for the physical server.

tree

(Optional) Specifies the name of the eDirectory tree that contains the server. Defaults to the tree of the physical server.

shareName

Specifies the share NetBios name that is advertised by CIFS. The share name is returned for both the basic and all type.

newShareName

(Optional) Specifies the new name of the share.

pathName

(Optional) Specifies the share path name. For example, sys:\system\.

comment

(Optional) Specifies a text string associated with the share.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

removeContext

Removes a context from the CIFS user context search list.

Request

```
<removeContext>
   <context/>
</removeContext>
```

Reply

```
<removeContext>
   <result value=" ">
      <description/>
   </result>
</removeContext>
```

Elements

context

Specifies the context or contexts to remove.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

removeShare

Removes a CIFS share on the server.

Request

```
<removeContext>
   <context/>
</removeContext>
```

Reply

```
<removeContext>
   <result value=" ">
      <description/>
   </result>
</removeContext>
```

Elements

context

Specifies the context or contexts to remove.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

setDomainConfiguration

Updates CIFS configuration information for the domain. If this command is used to set a new PDC for the domain, the new PDC must already be a domain controller that is participating in the domain.

Request

```
<setDomainConfiguration>
   <domain/>
   <pdc/>
   <comment/>
   <unp/>
   <user/>
   <password/>
</setDomainConfiguration>
```

Reply

```
<setDomainConfiguration>
   <result value=" ">
      <description/>
   </result>
</setDomainConfiguration>
```

Elements

domain

(Optional) Specifies the distinguished name of the eDirectory domain object (must be a fully distinguished name with a leading dot before the object name and a trailing dot after the tree name or an identifier of the object within the current tree). The default is the domain object that is associated with the CIFS server that is being talked to.

pdc

(Optional) Specifies the eDirectory name (in typeless, distinguished-name format) of the NCP server object that is configured as the PDC of the domain.

comment

(Optional) Specifies a text string that is associated with the domain.

unp

(Optional) Specifies the NMAS-encrypted username and password that has rights to perform the operation at the specified context.

user

(Optional) Specifies the clear-text username of a user that has rights to perform the operation. The user element is present only if the unp element is not specified.

password

(Optional/Required if user is specified) Specifies the clear-text password of a user that has rights to perform the operation. The password element is present only if the unp element is not specified.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

setServerConfiguration

Updated CIFS configuration information for the server.

Request

```
<setServerConfiguration>
   <server/>
   <tree/>
   <cifsServerName/>
   <comment/>
   <winsIPAddress/>
   <authMode/>
   <groupName/>
   <pdcName/>
   <pdcIPAddress/>
   <attachIPAddresses>
       <ipAddress/>
   </attachIPAddresses>
   <dfs enabled=""/>
   <oplocks enabled=""/>
   <loginScripts enabled=""/>
   <isOnline enabled=""/>
   <shareVolsByDefault enabled=""/>
</setServerConfiguration>
```

Reply

```
<setServerConfiguration>
   <result value="">
      <description/>
   </result>
</setServerConfiguration>
```

Elements

server

(Optional) Specifies the distinguished name of the eDirectory server object. The name must be a fully distinguished name, including the leading dot before the object name and the trailing dot after the tree name (or with no leading or trailing dot but a complete identity of the object within the current tree). The default is the eDirectory object for the physical server.

tree

(Optional) Specifies the name of the eDirectory tree that contains the server. The default is the tree of the physical server.

cifsServerName

(Optional) Specifies the NetBios name advertised by the server.

comment

(Optional) Specifies a text string associated with the CIFS server.

winsIPAddress

(Optional) Specifies the WINS server IP address in dotted ASCII notation. For example, 137.65.67.72.

authMode

Specifies the authentication mode:

local

domain

The domainMember and domainController modes can be set only by using joinDomain (page 57).

groupName

Specifies the workgroup (if in local mode) or the domain name.

pdcName

Specifies the NetBios name of the primary domain controller. Optional if the mode element specifies local.

pdcIPAddress

Specifies the IP address (in dotted ASCII notation) of the primary domain controller. Optional if the mode element is local.

attachIPAddresses

(Optional) Specifies the attach IP addresses that are currently assigned to the server.

ipAddress

Specifies from 1-6 IP addresses, each of which is in dotted ASCII notation. For example, 137.65.67.72.

dfs

(Optional) Specifies that CIFS is enabled for distributed file system support.

oplocks

(Optional) Specifies that CIFS is enabled for Oplocks support.

loginScripts

(Optional) Specifies that CIFS is enabled for login script support.

isOnline

(Optional) Specifies whether CIFS is enabled on the server. When enabling CIFS, the cifsstrt.ncf file is added to the autoexec.ncf files and the CIFS NLMs are loaded on the server. When disabling CIFS, cifsstrt.ncf is removed from autoexec.ncf and the CIFS NLMs are unloaded from the server.

shareVolsByDefault

(Optional) Specifies that the server exports all mounted volumes as shares by default. The server also exports any explicitly defined shares.

result

Specifies a zError or eDirectory value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

enabled

Specifies yes or no to indicate whether a specific feature is enabled.

3.5 Deleted Volume

The following commands can be used to manipulate deleted NSS logical volumes:

- “continueState” on page 75
- “pauseState” on page 77
- “purgeVolume” on page 78
- “salvageVolume” on page 79

Each command is wrapped with either the nssRequest or nssReply element and the deletedVolume element.

NSS logical volumes reside inside of NSS storage pools. When a logical volume is deleted, it is not immediately purged from the system. Instead, it goes into a deleted state, is renamed to an encoded name, and is assigned a time when it is permanently purged from the system. At any time before the deleted volume is automatically purged, it can be salvaged or manually purged.

NOTE: Do not confuse the deleted functionality of entire volumes with the salvage feature for files and directories.

---

continueState

Continues the current state of the deleted logical volume. If the current state is purging paused, it changes the state back to purging and resumes the purge of the deleted volume. If the current state is auto purging paused, it has the effect of changing the deleted volume back to the salvageable state and allows a future auto purge of the volume.

Request

```
<continueState>
   <deletedVolumeName/>
</continueState>
```

Reply

```
<continueState>
   <result value=" ">
      <description/>
   </result>
</continueState>
```

Elements

deletedVolumeName

(Required) Specifies the internal name of the deleted volume.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to continue a volume's state is as follows:

```
<nssRequest>
   <deletedVolume>
      <continueState>
         <deletedVolumeName>2FNC78F4IHL3_DV
         </deletedVolumeName>
      </continueState>
   </deletedVolume>
</nssRequest>
```

A nssReply packet to the continue state command follows:

```
<nssReply>
   <deletedVolume>
      <continueState>
         <result value="0">
            <description/>success</description>
         </result>
      </continueState>
   </deletedVolume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

pauseState

Pauses the current state of a deleted logical volume. If the current state is salvageable, it has the effect of preventing the volume from being auto purged. If the current state is purging, it pauses the purge process.

Request

```
<pauseState>
   <deletedVolumeName/>
</pauseState>
```

Reply

```
<pauseState>
   <result value=" ">
      <description/>
   </result>
</pauseState>
```

Elements

deletedVolumeName

(Required) Specifies the internal name of the deleted volume that is being paused.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to pause a volume's state is as follows:

```
<nssRequest>
   <deletedVolume>
      <pauseState>
         <deletedVolumeName>2FNC78F4IHL3_DV
         </deletedVolumeName>
      </pauseState>
   </deletedVolume>
</nssRequest>
```

A nssReply packet to the pause state command follows:

```
<nssReply>
   <deletedVolume>
      <pauseState>
         <result value="0">
            <description/>success</description>
         </result>
      </pauseState>
   </deletedVolume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

purgeVolume

Purges a deleted NSS logical volume.

Request

```
<purgeVolume>
   <deletedVolumeName/>
</purgeVolume>
```

Reply

```
<purgeVolume>
   <result value=" ">
      <description/>
   </result>
</purgeVolume>
```

Elements

deletedVolumeName

(Required) Specifies the internal name of the deleted volume that is being purged.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to purge a deleted volume is as follows:

```
<nssRequest>
   <deletedVolume>
      <purgeVolume>
         <deletedVolumeName>417F2724IHL3_DV
         </deletedVolumeName>
      </purgeVolume>
   </deletedVolume>
</nssRequest>
```

A nssReply packet to the purge volume command follows:

```
<nssReply>
   <deletedVolume>
      <purgeVolume>
         <result value="0">
            <description/>success</description>
         </result>
      </purgeVolume>
   </deletedVolume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

salvageVolume

Salvages or undeletes a deleted NSS logical volume. salvageVolume can also optionally re-add the eDirectory volume object for the undeleted volume.

Request

```
<salvageVolume state=" ">
   <deletedVolumeName/>
   <volumeName/>
   <ndsName/>
   <context/>
   <poolName/>
   <ndsPoolName/>
   <noNDSObject/>
   <updateVLDB/>
</salvageVolume>
```

Reply

```
<salvageVolume>
   <result value=" ">
      <description/>
   </result>
</salvageVolume>
```

Elements

salvageVolume

Specifies the volume to be salvaged.

deletedVolumeName

(Required) Specifies the internal name of the deleted volume to salvage.

volumeName

(Required) Specifies the new name to assign to the salvaged volume.

ndsName

(Required unless noNDSObject is used) Specifies the name of the eDirectory volume object that represents the volume in eDirectory. If NULL is specified, the name of the eDirectory volume object is generated by pre-pending the server name and an underscore to the name specified in the volumeName element.

context

(Required unless noNDSOjbect is used) Specifies where the eDirectory volume object is created. If no context is specified, defaults to be the same as the server object.

poolName

Specifies the value to use as the nssfsPool attribute of the eDirectory volume object. If NULL is specified, the pool's actual eDirectory name is retrieved from eDirectory and used.

ndsPoolName

(Required unless noNDSObject is used) Specifies the value to use as the nssfsPool attribute of the eDirectory volume object. If NULL is specified, the pool's actual eDirectory name is retrieved from eDirectory and used.

noNDSObject

(Optional) Specifies that no eDirectory objects should be created for the salvaged volume. If used, the ndsName, context, and ndsPoolName elements are ignored.

updateVLDB

Specifies that the DFS Volume Location Database (VLDB) is updated by the XML processing code. This element is used for backward compatibility with ConsoleOne, which does not know about this element but does its own VLDB updating. New code should include this element.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

state

(Optional) Specifies what state the salvaged volume is set to:

deactive

active

mounted

Defaults to whatever state the current file system policies dictate (see "NSS Volume States" on page 458).

Example

The following example salvages a deleted volume, which is currently named "M8C6CTP2IHL3_DV" on MYPOOL, and renames it MYPOOL. It also creates an eDirectory volume object for NSS2 using the default names.

```
<nssRequest>
   <deletedVolume>
      <salvageVolume state="mounted">
         <deletedVolumeName>M8C6CTP2IHL3_DV
         </deletedVolumeName>
         <volumeName>NSS2</volumeName>
         <poolName>MYPOOL</poolName>
         <ndsName>
         <context>
         <ndsPoolName>
      </salvageVolume>
   </deletedVolume>
</nssRequest>
```

A nssReply packet to the salvage volume command follows:

```
<nssReply>
   <deletedVolume>
      <salvageVolume>
         <result value="0">
            <description/>success</description>
         </result>
      </salvageVolume>
   </deletedVolume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

3.6 Device

The following commands allow you to manipulate devices on your server. (For a caution on shared devices, see “Device Sharing” on page 16.)

- “getDeviceInfo” on page 83
- “getDeviceInfo2” on page 85
- “getDeviceInfo3 (Linux)” on page 89
- “getPathInfo” on page 94
- “initializeDevice” on page 96
- “listDevices” on page 98
- “listDevicePartitions” on page 101
- “listDevicePools” on page 102
- “listMultiPaths” on page 103
- “modifyDevice” on page 104
- “multiPath” on page 106
- “renameDevice” on page 110
- “scanDevices” on page 111

Each command is wrapped with either the nssRequest or nssReply element and the device element.

getDeviceInfo

Returns information about the passed in object ID, which can be a physical device, a RAID device, or a mirror virtual device.

Request

```
<getDeviceInfo>
   <objectID/>
</getDeviceInfo>
```

Reply

```
<deviceSimpleInfo>
   <name/>
   <objectID/>
   <type/>
   <size/>
   <freeSize/>
   <shared/>
   <deviceRAID>
      <raidType/>
      <elementSize/>
      <stripeSize/>
      <restripeFlag/>
      <segmentInfo>
         <numSegments/>
         <segmented/>
      </segmentInfo>
   </deviceRAID>
   <mirrored/>
   <deviceMirror>
      <mirrorGroupStatus/>
      <mirrorGroupPercent/>
      <numMirrors/>
      <mirrorInfo>
         <id/>
         <mirrorPercent/>
      </mirrorInfo>
   </deviceMirror>
</deviceSimpleInfo>
```

Elements

name

Specifies the name of the device as assigned by Media Manager.

objectID

Specifies the device ID (for a non-mirrored virtual device) or specifies the mirror ID (for a mirrored virtual device).

type

Specifies the type of the object as assigned by Media Manager.

size

Specifies the total size of the object.

freeSize

Specifies the object's available size.

shared

Specifies if this is a shared device.

deviceRAID

Specifies this is a software RAID device and describes the RAID configuration.

raidType

Specifies the type of RAID device, such as RAID 0 or RAID 5.

elementSize

Specifies the size (in bytes) of the RAID segments. All segments must be of identical size.

stripeSize

Specifies the stripe size (in bytes) of the RAID device.

restripeFlag

Specifies the restripe status of the RAID device. A non0 value indicates that the RAID device is in the process of restriping.

segmentInfo

Specifies information about each segment.

numSegments

Specifies the number of segments in the RAID.

segmented

Specifies the ID of each segment as assigned by Media Manager.

mirrored

Specifies if the device is mirrored.

deviceMirror

Specifies if the device is a mirrored device (RAID 1- Mirroring) and describes the mirrored device.

mirrorGroupStatus

Specifies the status bits for the entire mirror device.

mirrorGroupPercent

Specifies the lowest remirror percentage of any segment in the entire mirror device. If the device is fully synchronized, the percentage is 100. If one segment is 63% sychronized and another is 77% synchronized, the percentage is 63.

numMirrors

Specifies the number of segments in the mirror device.

mirrorInfo

Specifies information about each segment's mirror.

id

Specifies the segment ID as assigned by Media Manager.

mirrorPercent

Specifies the percentage of how complete each remirror is for each segment.

getDeviceInfo2

Returns information about the passed in object ID, which can be a physical device or a RAID device (RAID 0, RAID 1, RAID 5).

Request

```
<getDeviceInfo2>
   <objectID/>
</getDeviceInfo2>
```

Reply

```
<getDeviceInfo2>
   <name/>
   <objectID/>
   <type/>
   <size/>
   <freeSize/>
   <majorVersion/>
   <minorVersion/>
   <partitions>
      <partition>
        <partitionID/>
         <partitionType/>
         <mountPoint/>
         <hasSYS/>
         <bootable/>
      </partition>
   </partitions>
   <hasDOS/>
   <multiPath/>
   <shared/>
   <removable/>
   <deviceRAID>
      <raidType/>
      <elementSize/>
      <elementCount/>
      <stripeSize/>
      <restripeFlag/>
      <restripeEnabled/>
      <segmentInfo>
         <numSegments/>
         <segment>
            <mirrorPercent/>
            <deviceName/>
            <segmentName/>
            <deviceID/>
            <segmentID/>
         </segment>
      </segmentInfo>
   </deviceRAID>
   <mirrored/>
   <deviceMirror>
      <mirrorGroupStatus/>
      <mirrorGroupPercent/>
      <numMirrors/>
      <remirrorEnabled/>
```

```
      <mirrorInfo>
         <segment>
            <mirrorPercent/>
            <deviceName/>
            <segmentName/>
            <deviceID/>
            <segmentID/>
         </segment>
      </mirrorInfo>
   </deviceMirror>
   <result value=" ">
      <description/>
   </result>
</getDeviceInfo2>
```

Elements

objectID

(Required) On input for NetWare, specifies the device ID received from Media Manager. On output for NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device object name.

name

NetWare only. Specifies the name of the device assigned by Media Manager.

type

(Optional) Specifies the RAID type of the device assigned by Media Manager.

size

Specifies the total size of the object in bytes.

freeSize

Specifies the object's available size in bytes.

majorVersion

Linux only. Specifies the major number of the device.

minorVersion

Linux only. Specifies the minor number of the device.

partition

Repeats for each partition on the device.

partitionID

On NetWare, specifies the partition ID number received from Media Manager. On Linux, specifies the partition object name.

partitionType

(Required) Specifies the type of the partition. The following types are for Linux:

0x00 freespace
0x82 Linux Swap
0x83 general Linux

mountPoint

Linux only. Specifies the partition's mount point.

hasSYS

(Optional) Specifies that the partition contains the SYS pool.

bootable

Linux only. Specifies that the partition contains the boot partition.

hasDOS

NetWare only. Specifies that this is a DOS partition. On Linux, this information can be retrieved from the partition type.

multiPath

(Optional) Specifies that the device has multipath.

shared

(Optional) Specifies that the device is shared.

removable

(Optional) NetWare only. Specifies that the deivce is removable.

raidType

Specifies the RAID type: 0, 1, or 5.

elementSize

Specifies the size in bytes.

elementCount

(Required) If deviceRAID is used, specifies the number of segments that are present. If raidType is 5 and elementCount equals the numSegments, one segment can be deleted.

stripeSize

Specifies the size in bytes.

restripeEnabled

Linux only. Specifies if the restripe is enabled.

numSegments

Specifies the number of segments in the RAID. If raidType is 1 (mirroring) and numSegments is greater than one, segments can be deleted (down to one remaining segment).

segment

Repeats for each segment (partition) that makes up the RAID device.

deviceName

NetWare only. Specifies the device name assigned by Media Manager. If there's a missing segment in a mirror device, no information is returned for deviceName and deviceID.

segmentName

NetWare only. Specifies the partition name assigned by Media Manager.

deviceID

On NetWare, specifies the segment's (or mirror segment's) device ID number received by Media Manager. On Linux, specifies the segment's (or mirror segment's) device name. If there's a missing segment in a mirror device, no information is returned for deviceName and deviceID.

deviceRAID

(Optional) If exists, specifies the device is a RAID device.

mirrored

(Optional) For RAID 1 devices only.

deviceMirror

(Optional) For RAID 1 devices only. If there's a missing segment in a mirror device, the missing segment name is returned as missing_raid1_1.

remirrorEnabled

Linux only. Specifies if re-mirroring is enabled.

mirrorActive

Linux only. Specifies if the mirror segment is active:

1 in synchronization
0 not in synchronization

mirrorStatus

Linux only. Specifies the status of the mirror segment:

1 in synchronization
0 not in synchronization

getDeviceInfo3 (Linux)

Returns information about the object ID, which can be a physical device or a RAID device (RAID 0, RAID 1, and RAID 5).

Request

```
<getDeviceInfo3 value =" "/>
   <objectID/>
</getDeviceInfo3>
```

NOTE: The current implementation of this command is capable of returning the details for only 32 devices at a time. If you specify more than 32 devices in the "`LIST`" option or use "`ALL`" when there are more than 32 devices, the command returns an error.

Reply

```
<getDeviceInfo3>
   <deviceInfo>
      <name/>
      <objectID/>
      <type/>
      <size/>
      <freeSize/>
      <majorVersion/>
      <minorVersion/>
      <partitions>
         <partition>
            <partitionID/>
            <partitionType/>
            <mountPoint/>
            <hasSYS/>
            <bootable/>
         </partition>
        </partitions>
      <multiPath/>
      <shared/>
      <deviceRAID>
         <raidType/>
         <elementSize/>
         <elementCount/>
         <stripeSize/>
         <restripeFlag/>
         <restripeEnabled/>
         <segmentInfo>
            <numSegments/>
            <segment>
               <mirrorPercent/>
               <deviceName/>
               <segmentName/>
               <deviceID/>
               <segmentID/>
            </segment>
         </segmentInfo>
      </deviceRAID>
      <mirrored/>
      <deviceMirror>
         <mirrorGroupStatus/>
         <mirrorGroupPercent/>
         <numMirrors/>
         <remirrorEnabled/>
         <mirrorInfo>
```

```
            <segment>
               <mirrorPercent/>
               <deviceName/>
               <segmentName/>
               <deviceID/>
               <segmentID/>
            </segment>
         </mirrorInfo>
      </deviceMirror>
      <result value=" ">
         <description/>
      </result>
   </deviceInfo>
</getDeviceInfo3>
```

Elements

objectID

Specifies the device object name.

device info

Specifies the information of the device.

name

Specifies the name of the device assigned by Media Manager.

type

(Optional) Specifies the RAID type of the device assigned by Media Manager.

size

Specifies the total size of the object in bytes.

freeSize

Specifies the object's available size in bytes.

majorVersion

Specifies the major number of the device.

minorVersion

Specifies the minor number of the device.

partition

Repeats for each partition on the device.

partitionID

Specifies the partition object name.

partitionType

(Required) Specifies the following type of the partition:

0x00 freespace
0x82 Linux Swap
0x83 general Linux

mountPoint

Specifies the partition's mount point.

hasSYS

(Optional) Specifies that the partition contains the SYS pool.

bootable

Specifies that the partition contains the boot partition.

multiPath

(Optional) Specifies that the device has multipath.

shared

(Optional) Specifies that the device is shared.

raidType

Specifies the RAID type: 0, 1, or 5.

elementSize

Specifies the size in bytes.

elementCount

(Required) If deviceRAID is used, specifies the number of segments that are present. If raidType is 5 and elementCount equals the numSegments, one segment can be deleted.

stripeSize

Specifies the size in bytes.

restripeEnabled

Specifies if the restripe is enabled.

numSegments

Specifies the number of segments in the RAID. If raidType is 1 (mirroring) and numSegments is greater than one, segments can be deleted (down to one remaining segment).

segment

Repeats for each segment (partition) that makes up the RAID device.

deviceID

Specifies the segment's (or mirror segment's) device name. If there's a missing segment in a mirror device, no information is returned for deviceName and deviceID.

deviceRAID

(Optional) If exists, specifies the device is a RAID device.

mirrored

(Optional) For RAID 1 devices only.

deviceMirror

(Optional) For RAID 1 devices only. If there's a missing segment in a mirror device, the missing segment name is returned as missing_raid1_1.

remirrorEnabled

Specifies if re-mirroring is enabled.

mirrorActive

Specifies if the mirror segment is active:

1 in synchronization

0 not in synchronization

mirrorStatus

Specifies the status of the mirror segment:

1 in synchronization

0 not in synchronization

result

Specifies the result value only if the data for the device is not retrieved. It also shows the complete error information along with the result value.

Specifies the information of all the queried object IDs in the sequential order.

Attribute

value

Specify "`ALL`" or "`LIST`" based on your requirement. If you give "`ALL`", you get the information of all the object IDs, maximum up to 32 object IDs only at a time. If you give "`LIST`", you get the information of the specified object IDs. You can select maximum up to 32 object IDs at a time.

Example

An example for the `LIST` option is as follows:

Request

```
<nssRequest>
   <device>
      <getDeviceInfo3 value="LIST">
         <objectID>sdb</objectID>
         <objectID>sdc</objectID>
      </getDeviceInfo3>
   </device>
</nssRequest>
```

Reply

```
<nssReply>
   <device>
      <getDeviceInfo3>
        <deviceInfo>
              <name>sdb</name>
              <objectID>sdb</objectID>
              <majorVersion>8</majorVersion>
              <minorVersion>16</minorVersion>
              <type>0</type>
              <size>5368709120</size>
              <freeSize>4850696192</freeSize>
           <partitions>
              <partition>
                 <partitionID>sdb1.1</partitionID>
                 <partitionType>105</partitionType>
                 <mountPoint>/opt/novell/nss/mnt/.pools/AC</mountPoint>
              </partition>
              <partition>
                 <partitionID>sdb1.2</partitionID>
                 <partitionType>105</partitionType>
```

```
                <mountPoint>/opt/novell/nss/mnt/.pools/AB</mountPoint>
             </partition>
          </partitions>
       </deviceInfo>
       <deviceInfo>
             <name>sdc</name>
             <objectID>sdc</objectID>
             <majorVersion>8</majorVersion>
             <minorVersion>32</minorVersion>
             <type>0</type>
             <size>1073741824</size>
             <freeSize>1073725440</freeSize>
          <partitions>
             <partition>
                <partitionID>sdc_freespace1</partitionID>
                <partitionType>0</partitionType>
             </partition>
          </partitions>
       </deviceInfo>
     </getDeviceInfo3>
   </device>
 <result value="0"><description>zOK</description></result>
</nssReply>
```

getPathInfo

Returns information about a multipath.

Request

```
<multiPath>
   <getPathInfo>
      <deviceID/>
      <pathID/>
   </getPathInfo>
</multiPath>
```

Reply

```
<multiPath>
   <getPathInfo>
      <pathInfo>
         <deviceName/>
         <pathName/>
         <priority/>
         <adaptorID/>
         <port/>
         <status>
            <up/>
            <selected/>
            <loadBalance/>
         </status>
      <pathInfo>
   </getPathInfo>
</multiPath>
<result value=" ">
   <description/>
</result>
```

Elements

deviceID

On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device object name.

pathID

On NetWare, specifies the path ID received from Media Manager. On Linux, specifies the path name.

deviceName

NetWare only.

adaptorID

NetWare only.

up

(Optional)

selected

(Optional)

loadBalance

(Optional)

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

initializeDevice

Re-initializes a device. Use this command with extreme caution because it destroys all data on the device. It also destroys all NSS storage pools and volumes that occupy any portion of the device that is being initialized. If a pool spans multiple partitions, including one on the device that is being initialized, the entire pool is destroyed. Note that when NSS storage pools and volumes are deleted by initializeDevice, their corresponding eDirectory objects are not deleted from the directory. For cleanup, it is best to first delete all pool and volumes that occupy a device before initializing the device itself.

Request

```
<initializeDevice>
   <deviceID/>
   <deviceInitFlag/>
</initializeDevice>
```

Reply

```
<initializeDevice>
   <result value=" ">
      <description/>
   </result>
</initializeDevice>
```

Elements

deviceID

On NetWare, specifies the device ID assigned by Media Manager. On Linux, specifies the device object name.

(Optional) deviceInitFlag

Initializes the device.

If the value is set to TRUE the devices are forcefully initialized and if the value is set to FALSE, the device that contains a system volume (such as /boot, swap, and /(root)) is not initialized.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to initialize a device is as follows:

```
<nssRequest>
   <device>
      <initializeDevice>
         <deviceID>3</deviceID>
         <deviceInitFlag>deviceInitFlag</deviceInitFlag>
      </initializeDevice>
   </device>
</nssRequest>
```

A nssReply packet to the initialize device command follows:

```
<nssReply>
   <device>
      <initializeDevice>
         <result value="0">
            <description/>success</description>
         </result>
      </initializeDevice>
   </device>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

listDevices

Obtains a detailed list of all devices on the server.

Request

```
<listDevices/>
```

Reply

```
<listDevices>
   <deviceInfo>
      <deviceName/>
      <deviceID/>
      <deviceType/>
      <unitSize/>
      <sectors/>
      <capacity/>
      <alignment/>
      <deviceShared/>
      <removable/>
      <deviceRAID>
         <raidType/>
         <elementSize/>
         <stripeSize/>
         <restripeFlag/>
      </deviceRAID>
      <result value=" ">
         <descriptionn>
      </result>
   </deviceInfo>
   <result value=" ">
      <description/>
   </result>
</listDevices>
```

Elements

deviceInfo

Repeats for each device being listed. Specifies information for each device.

deviceName

Specifies the name of the device as assigned by Media Manager and by the disk driver.

deviceID

Specifies the ID of the device as assigned by Media Manager.

deviceType

Specifies the type of the device (see Section 12.1, “Device Types,” on page 457).

unitSize

Specifies the size (in bytes) of the sector.

sectors

Specifies the number of sectors on a track. Because NetWare 6 partitions do not need to be aligned on cylinder boundaries, this number is not very useful.

capacity

Specifies the capacity (in sectors) of the drive.

alignment

Specifies the number of sectors on a cylinder. Because NetWare 6 partitions do not need to be aligned on cylinder boundaries, this number is not very useful.

deviceShared

Specifies that the device is flagged as a shared device.

removable

Specifies that the device is removable.

deviceRAID

Specifies the device is a software RAID device and contains information about the RAID configuration.

raidType

Specifies the type of the RAID device, such as RAID 0.

elementSize

Specifies the size (in bytes) of the segments in the device. Each segment is a partition of type Virtual Device Partition Type. All segments must be identical in size.

stripeSize

Specifies the stripe size of the device

restripeFlag

Specifies the restripe status of the device. A nonzero value indicates that the device is in the process of restriping.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to list the devices is as follows:

```
<nssRequest>
   <device>
      <listDevices>
   </device>
</nssRequest>
```

The following nssReply response shows two devices. The first device is a physical device while the second is a software RAID device.

```
<nssReply>
   <device>
      <deviceInfo>
         <deviceName>
             [V312-A0-D0:0] WDIGTL WDE4360-1807A3 rev:1.80
         </deviceName>
         <deviceID>1</deviceID>
         <deviceType>0</deviceType>
         <unitSize>512</unitSize>
         <sectors>63</sectors>
         <capacity>8385930</capacity>
         <alignment>16065</alignment>
         <result value="0">
            <description/>success</description>
         </result>
      </deviceInfo>

      <deviceInfo>
         <deviceName>
             [V043-A99-D0:0] RAID 0 Device 0
         </deviceName>
         <deviceID>16</deviceID>
         <deviceType>0</deviceType>
         <unitSize>512</unitSize>
         <sectors>32</sectors>
         <capacity>204800</capacity>
         <alignment>32</alignment>
         <deviceRAID>
            <raidType>0</raidType>
            <elementSize>104857600</elementSize>
            <stripeSize>65536</stripeSize>
            <restripeFlag>0</restripeFlag>
         </deviceRAID>
         <result value="0">
            <description/>success</description>
         </result>
      </deviceInfo>

      <result value="0">
         <description/>success</description>
      </result>
   </device>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

listDevicePartitions

Returns a list of partitions for a device.

Request

```
<listDevicePartitions>
      <deviceID/>
</listDevicePartitions>
```

Reply

```
<listDevicePartitions>
   <partitions>
      <partition>
         <partitionID/>
         <partitionType/>
      </partition>
   </partitions>
</listDevicePartitions>
```

Elements

deviceID

(Required) On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device object name.

partition

One for each partition.

partitionID

On NetWare, specifies the partition ID number received from Media Manager. On Linux, specifies the partition object name.

listDevicePools

Returns the pool list on a device.

Request

```
<listDevicePools>
   <objectID/>
</listDevicePools>
```

Reply

```
<listDevicePools>
   <poolSimpleInfo>
      <poolName/>
      <poolState/>
      <shared/>
   </poolSimpleInfo>
</listDevicePools>
```

Elements

objectID

On NetWare, specifies the device ID assigned from Media Manager. On Linux, specifies the device object name.

poolSimpleInfo

Repeats for each pool.

poolName

Specifies the name of the pool.

poolState

Specifies active, deactive, or mount.

shared

(Optional) Specifies the pool is shared.

listMultiPaths

Returns a list of multipaths for a device.

Request

```
<multiPath>
   <listMultiPaths>
      <deviceID/>
   </listMultiPaths>
</multiPath>
```

Reply

```
<multiPath>
   <listMultiPaths>
      <pathID/>
   </listMultiPaths>
</multiPath>
```

Elements

deviceID

On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device object name.

pathID

On NetWare, specifies the path ID received from Media Manager. On Linux, specifies the path name.

modifyDevice

Modifies the "shared" state of the device. The shared state is a manually set flag that should be set by the user on all devices that participate as shared devices in a cluster. The software has no mechanism for automatically detecting which devices the user desires to have participate in a cluster. Once a device is flagged as "shared," other XML commands allow partitions, NSS storage pools, and NSS logical volumes to be created only if the proper clustering software is installed and running.

Request

```
<modifyDevice>
   <deviceID/>
   <shared state=" "/>
</modifyDevice>
```

Reply

```
<modifyDevice>
   <result value=" ">
      <description/>
   </result>
</modifyDevice>
```

Elements

deviceID

On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device object name.

shared

Specifies if the device should be marked as shareable for clustering.

description

Specifies a text description of the returned result.

Attributes

state

Specifies a lowercase yes or no.

value

Specifies an error value or 0 (for no error).

Example

A nssRequest packet to set the "shared" state on a device is as follows:

```
<nssRequest>
   <device>
      <modifyDevice>
         <deviceID>3</deviceID>
         <shared state="yes">
      </modifyDevice>
   </device>
</nssRequest>
```

A nssReply packet to the modify device command follows:

```
<nssReply>
   <device>
      <modifyDevice>
         <result value="0">
            <description/>success</description>
         <result>
      </modifyDevice>
   </device>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

multiPath

Allows the user to control the behavior of multiple adaptors that are connected to the same device(s). The controllable behaviors include listing the current configuration, setting priorities for each path, setting path on-line and off-line, and selecting the path to use. Multiple paths can exist due to multiple adaptors and/or ports to the same device.

Request

```
<multiPath>
   <resetRegistry/>
   <multiPathInfo/>
   <setPathPriority>
      <pathID/>
      <priority/>
      <insert/>
   </setPathPriority>
   <selectPath>
      <pathID/>
   </selectPath>
   <setPathState>
      <pathID/>
      <state/>
   </setPathState>
   <selectDefaultPath>
      <deviceID/>
   </selectDefaultPath>
</multiPath>
```

Reply

```
<multiPath>
   <resetRegistry>
      <result value=" "/>
      <description/>
   </resetRegistry>
   <multiPathInfo>
      <deviceName/>
      <deviceID/>
      <pathInfo>
         <pathName/>
         <pathID/>
         <priority/>
         <adapterID/>
         <port/>
         <status>
            <up/>
            <selected/>
            <loadBalance/>
         </status>
         <result value=" ">
            <description/>
         </result>
      </pathInfo>
      <result value=" ">
         <description/>
      </result>
   </multiPathInfo>
   <setPathPriority>
      <result value=" ">
         <description/>
      </result>
   </setPathPriority>
   <selectPath>
```

```
      <result value=" ">
         <description/>
   </selectPath>
   <setPathState>
      <result value=" ">
         <description/>
      </result>
   </setPathState>
   <selectDefaultPath>
      <deviceID/>
      <result value=" ">
         <description/>
   </selectDefaultPath>
   <result value=" ">
      <description/>
   </result>
</multiPath>
```

Elements

resetRegistry

Specifies to reset the Media Manager failover registry entries.

multiPathInfo

Specifies to request all paths for all devices.

setPathPriority

Specifies the path priority.

pathID

(Required) On NetWare, specifies the path ID received from Media Manager. On Linux, specifies the path name.

priority

(Required) Specifies the priority value (0 equals highest priority).

insert

(Optional) Specifies to bump all entries with an equal priority to a priority lower than the one being set.

selectPath

Specifies the path to be used to get to a device.

setPathState

Specifies to set a path either up or down.

state

Specifies up or down.

selectDefaultPath

Specifies to select the path with the highest priority.

deviceID

On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device object name.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

deviceName

NetWare only.

pathInfo

Repeats for each path to a device.

adapterID

NetWare only.

status

Specifies the set states.

Example

The following example returns information from a system where two different devices are connected to the same two adaptors:

```
<nssRequest>
   <device>
      <multiPath>
         <multiPathInfo/>
      </multipath>
   </device>
</nssRequest>
```

A nssReply packet to the multiple path command follows:

```
<nssReply>
   <device>
      <multiPath>
         <multiPathInfo>
            <deviceName>
               [V345-A2-D2:0] SEAGATE ST34573W
               rev:5764
            </deviceName>
            <deviceID>23</deviceID>
            <pathInfo>
               <pathName>
                  [V345-A2-D2:0] SEAGATE ST34573W
                  rev:5764
               </pathName>
               <pathID>8</pathID>
               <priority>0</priority>
               <adaptorID>6</adaptorID>
               <port>0</port>
               <status><up><selected></status>
               <result value="0">
                  <description/>success
                  </description>
               </result>
            </pathInfo>
            <pathInfo>
               <pathName>
                  [V345-A3-D2:0] SEAGATE ST34573W
                  rev:5764
               </pathName>
               <pathID>15</pathID>
```

```
            <priority>0</priority>
            <adaptorID>13</adaptorID>
            <port>0</port>
            <status><up></status>
            <result value="0">
               <description/>success
               </description>
            </result>
         </pathInfo>

         <result value="0">
            <description/>success
            </description>
         </result>
      </multiPathInfo>

      <multiPathInfo>
         <deviceName>
            [V345-A3-D0:0] SEAGATE ST34573W
            rev:5764
         </deviceName>
         <deviceID>32</deviceID>
         <pathInfo>
            <pathName>
               [V345-A2-D0:0] SEAGATE ST34573W
               rev:5764
            </pathName>
            <pathID>7</pathID>
            <priority>2</priority>
            <adaptorID>6</adaptorID>
            <port>0</port>
            <status><up></status>
            <result value="0">
               <description/>success
               </description>
            </result>
         </pathInfo>
         <pathInfo>
            <pathName>
               [V345-A3-D0:0] SEAGATE ST34573W
               rev:5764
            </pathName>
            <pathID>14</pathID>
            <priority>1</priority>
            <adaptorID>13</adaptorID>
            <port>0</port>
            <status><up><selected></status>
            <result value="0">
               <description/>success
               </description>
            </result>
         </pathInfo>

         <result value="0">
            <description/>success
            </description>
         </result>
      </multiPathInfo>

      <result value="0">
         <description/>success
         </description>
      </result>
   </multiPath>
</device>
</nssreply>
```

renameDevice

Renames a device.

Request

```
<renameDevice>
   <objectID/>
   <name/>
</renameDevice>
```

Reply

```
<renameDevice>
   <result value=" ">
       <description/>
   </result>
</renameDevice>
```

Elements

objectID

Specifies the ID of the object to rename.

name

Specifies the new name of the object.

result

Specifies an error or 0 (for no error).

description

Specifies a text description of the result.

scanDevices

Requests that the media manager re-scan the server to look for any new devices.

Request

```
<scanDevices/>
```

Reply

```
<scanDevices>
   <result value=" ">
      <description/>
   </result>
</scanDevices>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to scan for new devices is as follows:

```
<nssRequest>
   <device>
      <scanDevices></scanDevices>
   </device>
</nssRequest>
```

A nssReply packet to the scan device command follows:

```
<nssReply>
   <device>
      <scanDevices>
         <result value="0">
            <description/>success</description>
         </result>
      </scanDevices>
   </device>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

3.7 DFS

The following Distributed File System (DFS) commands can be called to get and set the DFS globally unique ID (GUID) for a volume:

- “createLink” on page 113
- “deleteLink” on page 115
- “getDfsGUID” on page 116
- “initDFSGUIDs” on page 118
- “modifyLink” on page 119
- “readLink” on page 121
- “setDfsGUID” on page 123

Each command is wrapped with either the nssRequest or nssReply element and the dfs element.

Every volume that participates in the DFS needs to have a GUID assigned. This is the ID by which it is known to DFS.

When volume replication and volume moves are implemented in future NetWare versions, this DFS GUID is the same on all replicated instances of the volume, no matter where they physically reside.

createLink

Creates a link to a junction.

Request

```
<createLink>
   <pathName/>
   <junction>
      <managementContext>
         <ndsObject/>
         <tgtTree/>
      </managementContext>
      <dfsGUID/>
      <ndsVolume>
         <ndsObject/>
         <tgtTree/>
      </ndsVolume>
      <volumeInfo>
         <server/>
         <tgtTree/>
         <volumeName/>
      </volumeInfo>
   </junction>
   <symlink>
      <nameSpace/>
      <pathName/>
   </symlink>
   <unc>
      <pathName/>
   </unc>
   <url>
      <pathName/>
   </url>
</createLink>
```

Reply

```
<createLink>
   <result value=" ">
      <description/>
   </result>
</createLink>
```

Elements

pathName

Specifies the link file to be created.

tgtTree

(Optional)

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Remarks

When you create a junction, the target volume needs to be identified by either a DFS GUID, an eDirectory Volume object, or an eDirectory NCP Server object and Host Resource Name (or the physical volume name). Supply only one of these forms. The combination of an eDirectory NCP Server object and Host Resource Name applies only to NetWare 6 or later. Other forms can be used to create a junction to a pre NetWare 6 server.

For NetWare 6.5 SP1, the managementContext element is required.

For NetWare 6.5 SP2, the managementContext element is option if ndsVolume or volumeInfo is used. If the managementContext field is not supplied, the server that is creating the junction determines the context from the ndsObject or server elements that you supplied in the request. If the managementContext element is supplied with one of these forms, the supplied value is used and the server does not need to determine the context.

Example

```
<nssRequest>
   <dfs>
      <createLink>
         <pathName>VOL1:\foo\junction</pathName>
         <junction>          <!-- Creating a junction -->
            <managementContext>
               <ndsObject>nss.prv.novell</ndsObject>
               <tgtTree>novell_inc</tgtTree># optional
            </managementContext>
            <dfsGUID>C2EAAA00-3211-11D6-B7-C7-00C04FA33547</dfsGUID>
            <ndsVolume>
               <ndsObject>VLDB-MASTER_VOL1.novell</ndsObject>
               <tgtTree>novell_inc</tgtTree># optional
            </ndsVolume>
            <volumeInfo>
               <server>vldb-master.novell</server>
               <tgtTree>novell_inc</tgtTree>#optional
               <volumeName>VOL1</volumeName>
            </volumeInfo>
         </junction>
         <symlink>            <!-- Creating a symbolic link -->
            <nameSpace>long</nameSpace>
            <pathName>abc/def</pathName>
         </symlink>
         <unc>                <!-- Creating a UNC link -->
            <pathName>\\ServName\VolName\foo.bar</pathName>
         </unc>
         <url>                <!-- Creating a URL link -->
            <pathName>http://nss.provo.novell.com/dfs</pathName>
         </url>
      </createLink>
   </dfs>
</nssRequest>
```

deleteLink

Deletes a link to a junction.

Request

```
<deleteLink>
   <pathName/>
</deleteLink>
```

Reply

```
<deleteLink>
   <result value=" ">
      <description/>
   </result>
</deleteLink>
```

Elements

pathName

Specifies the link file to be created.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

```
<nssRequest>
   <dfs>
      <deleteLink>
         <pathName>VOL1:\foo\junction</pathName>
      </deleteLink>
   </dfs>
</nssRequest>
```

getDfsGUID

Retrieves the currently assigned DFS GUID for a volume.

Request

```
<getDfsGUID>
   <volumeName/>
</getDfsGUID>
```

Reply

```
<getDfsGUID>
   <dfsGUID/>
   <result value=" ">
      <description/>
   </result>
</getDfsGUID>
```

Elements

volumeName

Specifies the name of the volume from which to get the GUID.

dfsGUID

Specifies the actual GUID that was assigned to this volume for use with DFS.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to return the DFS GUID is as follows:

```
<nssRequest>
   <dfs>
      <getDfsGUID>
         <volumeName>NSS1</volumeName>
      </getDfsGUID>
   </dfs>
</nssRequest>
```

A nssReply packet to the get DFS GUID command follows:

```
<nssReply>
   <dfs>
      <getDfsGUID>
         <dfsGUID>C212F8B4-3223-01D6-80-00-FBDA22AE6917
         </dfsGUID>
         <result value="0">
            <description/>success</description>
         </result>
      </getDfsGUID>
   </dfs>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

initDFSGUIDs

Assigns a DFS GUID if the volume does not already have a GUID assigned (for all mounted volumes) and adds the volume to the VLDB if there is a management context defined and the volume is not already included in the VLDB.

Request

```
<initDFSGUIDs/>
```

Reply

```
<initDFSGUIDs>
   <result value=" ">
      <description/>
   </result>
</initDFSGUIDs>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

modifyLink

Renames or changes the contents of a link file. The newName element is optional and allows the file to be renamed. If a file is being renamed, the contents elements (junction, symlink, unc, or url) are not necessary. However, by including these elements, you can rename and modify the contents of a link file in the same call to modifyLink.

Request

```
<modifyLink>
   <pathName/>
   <newName/>
   <junction>
      <managementContext>
         <ndsObject/>
         <tgtTree/>
      </managementContext>
      <dfsGUID/>
      <ndsVolume>
         <ndsObject/>
         <tgtTree/>
      </ndsVolume>
      <volumeInfo>
          <server/>
          <tgtTree/>
          <volumeName/>
      </volumeInfo>
   </junction>
   <symlink>
      <nameSpace/>
      <pathName/>
   </symlink>
   <unc>
      <pathName/>
   </unc>
   <url>
      <pathName/>
   </url>
</modifyLink>
```

Reply

```
<modifyLink>
   <result value="">
      <description/>
   </result>
</modifyLink>
```

Elements

pathName

Specifies the link file to modify.

newName

(Optional) Specifies the new name of the file.

tgtTree

(Optional)

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

```
<nssRequest>
   <dfs>
      <modifyLink>
         <pathName>VOL1:\foo\junction</pathName>
         <newName>VOL1:\foo\newjunc</newName>

         <junction>          <!-- Modifying a junction -->
            <managementContext>
               <ndsObject>nss.prv.novell</ndsObject>
               <tgtTree>novell_inc</tgtTree>
            </managementContext>
            <dfsGUID>C2EAAA00-3211-11D6-B7-C7-00C04FA33547</dfsGUID>
            <ndsVolume>
               <ndsObject>VLDB-MASTER_VOL1.novell</ndsObject>
               <tgtTree>novell_inc</tgtTree>
            </ndsVolume>
            <volumeInfo>
               <server>vldb-master.novell</server>
               <tgtTree>novell_inc</tgtTree>
               <volumeName>VOL1</volumeName>
            </volumeInfo>
         </junction>

         <symlink>           <!-- Modifying a symbolic link -->
            <nameSpace>long</nameSpace>
            <pathName>abc/def</pathName>
         </symlink>

         <unc>               <!-- Modifying a UNC link -->
            <pathName>\\ServName\VolName\foo.bar</pathName>
         </unc>

         <url>               <!-- Modifying a URL link -->
            <pathName>http://nss.provo.novell.com/dfs</pathName>
         </url>
      </modifyLink>
   </dfs>
</nssRequest>
```

readLink

Reads a junction link and returns a list of physical volume instances. Note that there can be multiple volumeInfo elements in the response.

Request

```
<readLink>
   <pathName/>
</readLink>
```

Reply

```
<readLink>
   <result value="">
      <description/>
   </result>
   <junction>
      <managementContext>
         <ndsObject/>
         <tgtTree/>
      </managementContext>
      <dfsGUID/>
      <volumeInfo>
         <server/>
         <tgtTree/>
         <volumeName/>
      </volumeInfo>
   </junction>
   <symlink>
      <nameSpace/>
      <pathName/>
   </symlink>
   <unc>
      <pathName/>
   </unc>
   <url>
      <pathName/>
   </url>
</readLink>
```

Elements

pathName

Specifies the link file to modify.

tgtTree

(Optional)

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

```
<nssRequest>
   <dfs>
      <readLink>
         <result value="0">
            <description/>success</description>
         </result>
         <junction>            <!-- File is a junction -->
            <managementContext>
               <ndsObject>nss.prv.novell</ndsObject>
               <tgtTree>novell_inc</tgtTree>
            </managementContext>
            <dfsGUID>C2EAAA00-3211-11D6-B7-C7-00C04FA33547</dfsGUID>
            <volumeInfo>
               <server>vldb-master.novell</server>
               <tgtTree>novell_inc</tgtTree>
               <volumeName>VOL1</volumeName>
            </volumeInfo>
         </junction>

         <symlink>            <!-- File is a symlink -->
            <nameSpace>long</nameSpace>
            <pathName>abc/def</pathName>
         </symlink>

         <unc>               <!-- File is a UNC link -->
            <pathName>\\ServName\VolName\foo.bar</pathName>
         </unc>

         <url>               <!-- File is a URL link -->
            <pathName>http://nss.provo.novell.com/dfs</pathName>
         </url>
      </readLink>
   </dfs>
</nssReply>
```

setDfsGUID

Assigns a DFS GUID to a volume. If you specify an exact DFS GUID, it is stored as specified. Otherwise, if you specify the dfsGUID element with no content, a DFS GUID is generated for you by setDfsGUID.

Request

```
<setDfsGUID>
   <volumeName/>
   <dfsGUID/>
</setDfsGUID>
```

Reply

```
<setDfsGUID>
   <dfsGUID/>
   <result value=" ">
      <description/>
   </result>
</setDfsGUID>
```

Elements

volumeName

Specifies the volume on which to set the GUID.

dfsGUID

(Required) Specifies the GUID. If NULL is specified, a DFS GUID is generated.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to set the GUID is as follows:

```
<nssRequest>
   <dfs>
      <setDfsGUID>
         <volumeName>NSS1</volumeName>
         <dfsGUID>
      </setDfsGUID>
   </dfs>
</nssRequest>
```

A nssReply packet to the set DFS GUID command follows:

```
<nssReply>
   <dfs>
      <setDfsGUID>
          <dfsGUID>C212F8B4-3223-01D6-80-00-FBDA22AE6917
          </dfsGUID>
          <result value="0">
             <description/>success</description>
          </result>
      </setDfsGUID>
   </dfs>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

3.8 Directory Quota

This section contains the following Directory Quota commands:

- “addQuota (obsolete)” on page 126

Each command is wrapped with either the nssRequest or nssReply element and the directoryQuota element.

addQuota (obsolete)

is obsolete. Call addQuota (page 366) instead.

3.9 Junction

The following commands can be called to manipulate file system junctions:

- "createJunction" on page 128
- "deleteJunction" on page 130

Each command is wrapped with either the nssRequest or nssReply element and the junction element.

A junction is a special type of file that is used in Distributed File System (DFS).

To a client that is DFS-aware, a junction appears as a directory. The directory contains the entire volume subtree to which the junction points.

To a non-DFS-aware client, a junction appears as a file that cannot be opened, modified, or deleted.

createJunction

Creates a file system junction.

Request

```
<createJunction>
   <junctionPath/>
   <junctionName/>
   <junctionDefinition/>
   <nameSpace/>
</createJunction>
```

Reply

```
<createJunction>
   <result value=" ">
      <description/>
   <result>
</createJunction>
```

Elements

junctionPath

Specifies the full path to the directory where the new junction is created.

junctionName

Specifies the name of the junction to create.

junctionDefinition

Specifies the definition to write to the newly created junction. The format of this data is defined in the DFS documentation.

nameSpace

Specifies the name space ID in which the junction name is created.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following example creates a junction named "myJunction" in the SYS:/temp directory. The junction name is to be created using the LONG (4) name space. The junction references volume NSS1 on MYSERVER in MY_TREE.

```
<nssRequest>
   <junction>
      <createJunction>
         <junctionPath>SYS:/tmp</junctionPath>
         <junctionName>myJunction</junctionName>
         <junctionDefinition>
            /../junction/.MYSERVER_NSS1.novell.MY_TREE./
            0xCC507D5C2732D6018002FBDA22AE6917
         </junctionDefinition>
         <nameSpace>4</nameSpace>
      </createJunction>
   </junction>
</nssRequest>
```

A nssReply packet to the create junction command follows:

```
<nssReply>
   <junction>
      <createJunction>
         <result value="0">
            <description/>success</description>
         </result>
      </createJunction>
   </junction>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

deleteJunction

Deletes a file system junction.

Request

```
<deleteJunction>
   <junctionPath/>
</deleteJunction>
```

Reply

```
<deleteJunction>
   <result value=" ">
      <description/>
   <result>
</deleteJunction>
```

Elements

junctionPath

Specifies the full path of the junction to delete.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to delete a junction is as follows:

```
<nssRequest>
   <junction>
      <deleteJunction>
         <junctionPath>SYS:/tmp/myJunction</junctionPath>
      </deleteJunction>
   </junction>
</nssRequest>
```

A nssReply packet to the delete junction command follows:

```
<nssReply>
   <junction>
      <deleteJunction>
         <result value="0">
            <description/>success</description>
         </result>
      </deleteJunction>
   </junction>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

3.10 LSS

This section contains the following Loadable Storage Systems (LSS) commands:

- "getLSSInfo" on page 132
- "getLSSVolumeInfo" on page 137

Each command is wrapped with either the nssRequest or nssReply element and the lss element.

NSS supports LSS, which provides the ability to add additional storage modules that expose and make available different types of storage systems to NSS.

The main LSS for NSS is the ZLSS, which is the native storage system of NSS. Other LSS modules include DOSFAT and CD9660, which allow NSS to access DOS partitions and CD devices, respectively.

getLSSInfo

Returns information about the Loadable Storage Systems on a server. This command is implemented only on NetWare and not on Linux.

Request

```
<getLssInfo/>
```

Reply

```
<getLssInfo>
   <lssInfo>
      <lssName/>
      <lssID/>
      <createAllowed/>
      <poolSupportedFeatures value="">
         <readonly/>
         <shared/>
      </poolSupportedFeatures>
      <poolDefaultFeatures value="">
         <readonly/>
         <shared/>
      </poolDefaultFeatures>
      <poolChangeableFeatures value="">
         <readonly/>
         <shared/>
      </poolChangeableFeatures>
      <volSupportedFeatures>
         <readonly/>
         <salvage/>
         <compression/>
         <directoryQuota/>
         <userQuota/>
         <flushFiles/>
         <mfl/>
         <snapshot/>
         <backup/>
         <shredding/>
         <userTransaction/>
         <migration/>
      </volSupportedFeatures>
      <volDefaultFeatures>
         <readonly/>
         <salvage/>
         <compression/>
         <directoryQuota/>
         <userQuota/>
         <flushFiles/>
         <mfl/>
         <snapshot/>
         <backup/>
         <shredding/>
         <userTransaction/>
         <migration/>
      </volDefaultFeatures>
      <volChangeableFeatures>
         <readonly/>
         <salvage/>
         <compression/>
         <directoryQuota/>
```

```
        <userQuota/>
        <flushFiles/>
        <mfl/>
        <snapshot/>
        <backup/>
        <shredding/>
        <userTransaction/>
        <migration/>
      </volChangeableFeatures>
   </lssInfo>
   <result value=" ">
      <description/>
   </result>
</getLssInfo>
```

Elements

lssInfo

Repeats for each LSS on the system.

lssName

Specifies the name of the LSS.

lssID

Specifies the ID that is associated with the LSS.

createAllowed

Specifies that the LSS allows new volumes and pools to be created.

poolSupportedFeatures

Specifies a list of elements that represent pool features that are supported by the LSS type.

readonly

Specifies that read only is supported, enabled by default, or changeable for the pool or volume.

shared

Specifies that shared is supported, enabled by default, or changeable for the pool or volume.

poolDefaultFeatures

Specifies a list of elements that represent pool features that are enabled by default when pools are created on the LSS type.

poolChangeableFeatures

Specifies a list of elements that represent pool features that can be changed for the LSS type.

volSupportedFeatures

Specifies a list of elements that represent volume features that are supported by the LSS type.

salvage

Specifies that salvage is supported, enabled by default, or changeable on the volume.

compression

Specifies that compression is supported, enabled by default, or changeable on the volume.

directoryQuota

Specifies that a directory quota is supported, enabled by default, or changeable on the volume.

userQuota

Specifies that a user quota is supported, enabled by default, or changeable on the volume.

flushFiles

Specifies that flushing files is supported, enabled by default, or changeable on the volume.

mfl

Specifies that mfl is supported, enabled by default, or changeable on the volume.

snapshot

Specifies that snapshots are supported, enabled by default, or changeable on the volume.

backup

Specifies that back ups are supported, enabled by default, or changeable on the volume.

shredding

Specifies that shredding is supported, enabled by default, or changeable on the volume.

userTransaction

Specifies that user transactions are supported, enabled by default, or changeable on the volume.

migration

Specifies that migration is supported, enabled by default, or changeable on the volume.

volDefaultFeatures

Specifies a list of elements that represent volume features that are enabled by default when volumes are created on the LSS type.

volChangeableFeatures

Specifies a list of elements that represent volume features that are changeable for volumes on the LSS type.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Atrributes

value

Specifies the decimal value of the LSS's supported, default, or changeable pool or volume features bit mask.

Example

A nssRequest packet to return the LSS information is as follows:

```
<nssRequest>
   <lss>
      <getLSSInfo>
      </getLSSInfo>
   </lss>
</nssRequest>
```

In the following example, the server has the standard ZLSS loaded, as well as DOSFAT and CD9660

```
<nssReply>
   <lss>
      <getLSSInfo>
         <lssInfo>
            <lssName>ZLSS</lssName>
            <lssID>20</lssID>
            <createAllowed>
            <poolSupportedFeatures value="123">
               <shared>
            </poolSupportedFeatures>
            <poolDefaultFeatures value="57">
            </poolDefaultFeatures>
            <poolChangeableFeatures value="123">
               <shared>
            </poolChangeableFeatures>
            <volSupportedFeatures value="469762043">
               <salvage/>
               <compression>
               <directoryQuota>
               <userQuota>
               <flushFiles>
               <mfl>
               <snapshot>
               <shredding>
               <userTransaction>
               <migration>
            </volSupportedFeatures>
            <volDefaultFeatures value="262129">
               <salvage/>
               <backup>
            </volDefaultFeatures>
            <volChangeableFeatures value="468975627">
               <salvage/>
               <compression>
               <directoryQuota>
               <userQuota>
               <flushFiles>
               <mfl>
               <snapshot>
               <shredding>
               <userTransaction>
               <migration>
            </volChangeableFeatures>
         </lssInfo>

         <lssInfo>
            <lssName>DOSFAT</lssName>
            <lssID>30</lssID>
            <poolSupportedFeatures value="0">
            </poolSupportedFeatures>
            <poolDefaultFeatures value="0">
            </poolDefaultFeatures>
            <poolChangeableFeatures value="0">
            </poolChangeableFeatures>
            <volSupportedFeatures value="64">
               <backup>
            </volSupportedFeatures>
            <volDefaultFeatures value="64">
               <backup>
            </volDefaultFeatures>
            <volChangeableFeatures value="0">
               <backup>
            </volChangeableFeatures>
         </lssInfo>

         <lssInfo>
            <lssName>CD9660</lssName>
```

```
            <lssID>40</lssID>
            <poolSupportedFeatures value="0">
            </poolSupportedFeatures>
            <poolDefaultFeatures value="0">
            </poolDefaultFeatures>
            <poolChangeableFeatures value="0">
            </poolChangeableFeatures>
            <volSupportedFeatures value="68">
               <readOnly>
               <backup>
            </volSupportedFeatures>
            <volDefaultFeatures value="68">
               <readOnly>
               <backup>
            </volDefaultFeatures>
            <volChangeableFeatures value="0">
               <backup>
            </volChangeableFeatures>
         </lssInfo>

      <result value="0">
         <description/>success</description>
      </result>
      </getLSSInfo>
   </lss>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

getLSSVolumeInfo

Returns the supported, default, and chageable LSS features for the specified LSS type.

Request

```
<getLSSVolumeInfo>
   <lssName/>
</getLSSVolumeInfo>
```

Reply

```
<getLSSVolumeInfo>
         <lssVolumeInfo>
            <volSupportedFeatures value=" ">
               <salvage/>
               <compression/>
               <directoryQuota/>
               <userQuota/>
               <flushFiles/>
               <mfl/>
               <snapshot/>
               <backup/>
               <shredding/>
               <userTransaction/>
               <migration/>
               <backup/>
            </volSupportedFeatures>
            <volDefaultFeatures value=" ">
               <salvage/>
               <compression/>
               <directoryQuota/>
               <userQuota/>
               <flushFiles/>
               <mfl/>
               <snapshot/>
               <backup/>
               <shredding/>
               <userTransaction/>
               <migration/>
               <backup/>
            </volDefaultFeatures>
            <volChangeableFeatures value=" ">
               <salvage/>
               <compression/>
               <directoryQuota/>
               <userQuota/>
               <flushFiles/>
               <mfl/>
               <snapshot/>
               <backup/>
               <shredding/>
               <userTransaction/>
               <migration/>
               <backup/>
            </volChangeableFeatures>
         </lssVolumeInfo>
         <result value="0">
            <description/>
         </result>
      </getLSSVolumeInfo>
```

Elements

lssName

Specifies the name of the LSS.

lssVolumeInfo

Specifies information for the volume.

volSupportedFeatures

Specifies a list of elements that represent volume features that are supported by the LSS type.

salvage

Specifies that salvage is supported, enabled by default, or changeable on the volume.

compression

Specifies that compression is supported, enabled by default, or changeable on the volume.

directoryQuota

Specifies that a directory quota is supported, enabled by default, or changeable on the volume.

userQuota

Specifies that a user quota is supported, enabled by default, or changeable on the volume.

flushFiles

Specifies that flushing files is supported, enabled by default, or changeable on the volume.

mfl

Specifies that mfl is supported, enabled by default, or changeable on the volume.

snapshot

Specifies that snapshots are supported, enabled by default, or changeable on the volume.

backup

Specifies that back ups are supported, enabled by default, or changeable on the volume.

shredding

Specifies that shredding is supported, enabled by default, or changeable on the volume.

userTransaction

Specifies that user transactions are supported, enabled by default, or changeable on the volume.

migration

Specifies that migration is supported, enabled by default, or changeable on the volume.

volDefaultFeatures

Specifies a list of elements that represent volume features that are enabled by default when volumes are created on the LSS type.

volChangeableFeatures

Specifies a list of elements that represent volume features that are changeable for volumes on the LSS type.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Atrributes

value

Specifies the decimal value of the LSS's supported, default, or changeable volume features bit mask.

3.11 Partition

This section contains partition commands. Before using the following commands, you should understand the basic partition concepts that are explained in “Partitions” on page 16.

- “addPartition” on page 141
- “addPartition2” on page 145
- “addPartitionToMirror” on page 146
- “getPartitionInfo” on page 148
- “getPartitionMirrorStats” on page 150
- “listPartitions” on page 152
- “modifyPartition” on page 156
- “removePartition” on page 158
- “removePartitionFromMirror” on page 160
- “resyncPartitionMirror” on page 161

Each command is wrapped with either the nssRequest or nssReply element and the partition element.

addPartition

Creates a new partition on a device.

Request

```
<addPartition>
   <deviceID/>
   <partitionType/>
   <startingSector/>
   <numSectors/>
   <freeSpaceID/>
   <hotFixSize/>
   <mirrorID/>
   <label/>
   <ignoreShareState/>
</addPartition>
```

Reply

All elements in partitionInfo are exact duplicates of the elements in listPartitions (page 152). The only difference is that addPartition returns the partitionInfo for the partition that was just created.

```
<addPartition>
   <partitionInfo>
      <deviceName/>
      <partitionName/>
      <deviceID/>
      <partitionType/>
      <partitionID/>
      <label/>
      <startingSector/>
      <numSectors/>
      <logicalPartitionID/>
      <logicalPartitionCapacity/>
      <mirrorID/>
      <hotFixID/>
      <hotFixSize/>
      <hotFixAvailSize/>
      <poolName/>
      <volumes>
         <volumeInfo>
            <volumeName/>
            <volStartingSector/>
            <volNumSectors/>
         </volumeInfo>
      </volumes>
      <raidID/>
      <result value=" ">
         <description/>
      </result>
   </partitionInfo>
</addPartition>
```

Elements

deviceID

(Required) Specifies the ID of the device on which the new partition should be created. On reply, specifies the device on which the partitions resides, as assigned by Media Manager at boot time.

partitionType

(Required) Specifies the type of partition (see "Partition Types" on page 17).

startingSector

(Required) Specifies the starting sector where the partition begins on the device. If freeSpaceID is specified, startingSector is ignored. Otherwise, Media Manager uses startingSector to locate the free space on the device that contains the sector. Once the free space is loacted, the new partition is created at the beginning of the free space.

numSectors

(Required) Specifies the size (in sectors) of the new partition.

freeSpaceID

(Required) Specifies the partition ID, as returned by listPartitions (page 152), of the free space in which the partition is created. If freeSpaceID is nonzero, startingSector is ignored.

hotFixSize

(Required) Specifies the number of sectors to use as the HotFix area for the new partition. If 0, no HotFix is created. If nonzero, the specified number of sectors on the new partition is reserved for HotFix overhead.

On reply, specifies the size (in sectors) of the HotFix area that is reserved to track bad block redirection. If the partition does not have a HotFix object, the size is 0. This element is filled only if partitionType is NSS or traditional NetWare.

mirrorID

(Optional) Specifies the ID of the mirror group to add the partition to. If not included, a mirror object is not created for the partition and the partition can never participate in a mirror group. If NULL is passed, a new mirror group is created for the partition and the new partition is the only member of the group.

On reply, specifies the ID of the mirror group. If the partition does not belong to a mirror group, the ID is 0. This element is filled in the reply only if partitionType is NSS or traditional NetWare.

label

(Optional) Specifies that no check is done to see if the device is marked shareable for clustering. Usually, the server checks to validate that the clustering software is loaded and operational before it allows partitions to be created on a device that is marked shareable.

On reply, specifies an optional label that was assigned to the partition when it was created. If there is no label, an empty element exists in the reply.

ignoreShareState

Specifies that no check is done to see if the device is shared.

deviceName

Specifies the name of the device on which the partition resides, as assigned by Media Manager and the disk driver.

partitionName

Specifies the name of the partition, as assigned by Media Manager and the disk driver.

partitionID

Specifies the physical partition ID, as assigned by Media Manager. The ID represents the partition itself but does not represent any HotFix or mirror objects on the partition.

logicalPartitionID

Specifies the logical partition ID that is used when creating NSS pools or traditional volumes on the partition. If the partition does not have HotFix and mirroring, the ID is the same as the physical partitionID. Otherwise, the ID is the same as the mirrorID. This element is used only if partitionType is NSS or traditional NetWare.

logicalPartitionCapacity

Specifies the actual capacity (in sectors) of the logical partition. If the partition contains HotFix and mirroring objects, the capacity is smaller than the size of the physical partition due to the overhead associated with HotFix. This element is used only if partitionType is NSS or traditional NetWare.

hotFixID

Specifies the ID of the HotFix object. If the partition does not have a HotFix object, the ID is 0. This element is used only is partitionType is NSS or traditional NetWare.

hotFixAvailableSize

Specifies the useable size (in sectors) of the HotFix area that is reserved to track bad block redirection. HotFix has some overhead, so this size is smaller than the size in hotFixSize. If the partition does not have a HotFix object, the size is 0. This element is used only if partitionType is NSS or traditional NetWare.

poolName

Specifies the pool name for an NSS partition that has a pool that uses the partition. This element is used only if partitionType is NSS and if the partition is currently owned by an NSS pool.

volumes

Specifies the traditional volume segments residing on a partition. This element is used only if partitionType is traditional NetWare.

volumeInfo

Specifies information for each volume segment on the partition.

volumeName

Specifies the name of the traditional NetWare volume that owns this segment of the partition.

volStartingSector

Specifies the starting sector number of the piece of the partition that is owned by the volume.

volNumSectors

Specifies the number of sectors that are owned by the volume at the starting sector offset.

raidID

Specifies the media manager device ID of the RAID device that consumes the virtual device partition. This element is used only if partitionType is virtual device.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to add an NSS partition is as follows:

```
<nssRequest>
   <partition>
      <addPartition>
         <deviceID>2</deviceID>
         <partitionType>105</partitionType>
         <startingSector>63</startingSector>
         <numSectors>204800</numSectors>
         <freeSpaceID>12</freeSpaceID>
         <hotFixSize>200</hotFixSize>
         <mirrorID></mirrorID>
      </addPartition>
   </partition>
</nssRequest>
```

A nssReply packet to the add partition command follows:

```
<nssReply>
   <partition>
      <addPartition>
         <partitionInfo>
            <deviceName>
               [V312-A0-D2:0] HP 2.13 GB #A2 rev:0180
            </deviceName>
            <partitionName>
               [V312-A0-D2:0-P0] NSS Partition
            </partitionName>
            <deviceID>2</deviceID>
            <partitionType>105</partitionType>
            <partitionID>14</partitionID>
            <label></label>
            <startingSector>63</startingSector>
            <numSectors>204800</numSectors>
            <logicalPartitionID>17</logicalPartitionID>
            <logicalPartitionCapacity>
               204600
            </logicalPartitionCapacity>
            <mirrorID>17</mirrorID>
            <hotFixID>16</hotFixID>
            <hotFixSize>200</hotFixSize>
            <hotFixAvailSize>8</hotFixAvailSize>
         </partitionInfo>
         <result value="0">
            <description/>success</description>
         </result>
      </addPartition>
   </partition>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

addPartition2

Creates a new partition on a device.

Request

```
<addPartition2>
   <deviceID/>
   <partitionType/>
   <size/>
   <label/>
   <ignoreShareState/>
</addPartition2>
```

Reply

```
<addPartition2>
   <result value=" ">
      <description/>
   </result>
</addPartition2>
```

Elements

deviceID

(Required) On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device object name.

partitionType

(Required) Specifies the type of partition (see Section 1.5.2, “Partition Types,” on page 17.)

size

(Required) Specifies the size of the new partition in bytes.

label

(Optional) Specifies the user-defined partition label.

ignoreShareState

(Optional) If exists, specifies that no check is done to see if the device is shared.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

addPartitionToMirror

Adds one or more partitions to another existing mirror group. In order to add a partition to a mirror group, the partition must have been created with both a HotFix object and a mirror object. The data size of the partition must be exactly the same as the data size of the partitions that already exist in the mirror group. Also, the partition being added must not currently be a part of any other mirror group.

Request

```
<addPartitionToMirror>
   <mirrorID/>
   <partitions>
      <partitionID/>
   </partitions>
</addPartitionToMirror>
```

Reply

```
<addPartitionToMirror>
   <result value=" ">
      <description/>
   </result>
</addPartitionToMirror>
```

Elements

mirrorID

(Required) Specifies the ID of the existing mirror group to add the partition to.

partitions

Specifies a list of partitions that need to be added to the mirror group.

partitionID

Repeats for each instance of a partition that needs to be added to the mirror group.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to add partitionID 14 to an existing mirror group (ID 20) is as follows:

```
<nssRequest>
   <partition>
      <addPartitionToMirror>
         <mirrorID>20</mirrorID>
         <partitions>
            <partitionID>14</partitionID>
         </partitions>
      </addPartitionToMirror>
   </partition>
</nssRequest>
```

A nssReply packet to the add partition to mirror command follows:

```
<nssReply>
   <partition>
      <addPartitionToMirror>
         <result value="0">
            <description/>success</description>
         </result>
      </addPartitionToMirror>
   </partition>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

getPartitionInfo

Returns information about the specified partition.

Request

```
<getPartitionInfo>
   <partitionID/>
</getPartitionInfo>
```

Reply

```
<getPartitionInfo>
   <deviceName/>
   <partitionName/>
   <deviceID/>
   <partitionType/>
   <partitionID/>
   <label/>
   <startingSector/>
   <numSectors/>
   <logicalPartitionID/>
   <logicalPartitionCapacity/>
   <mirrorID/>
   <hotFixID/>
   <hotFixSize/>
   <hotFixAvailSize/>
   <poolName/>
   <raidID/>
   <result value=" ">
      <description/>
   </result>
</getPartitionInfo>
```

Elements

deviceName

NetWare only.

deviceID

(Required) On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device name.

partitionID

(Required) On NetWare, specifies the partition ID received from Media Manager. On Linux, specifies the partition name.

label

(Optional)

numSectors

Specifies the size of the partition.

logicalPartitionID

On NetWare, specifies the logical partition ID received from Media Manager. On Linux, specifies the logical partition name.

mirrorID

On NetWare, specifies the mirror ID received from Media Manager. On Linux, specifies the mirror name.

hotFixID

On NetWare, specifies the hot fix ID received from Media Manager. On Linux, specifies the hot fix name.

raidID

(Optional) If exists, specifies that the device is a RAID virtual device. On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device name.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

getPartitionMirrorStats

Returns statistics for the specified mirror group.

Request

```
<getPartitionMirrorState>
   <mirrorID/>
</getPartitionMirrorState>
```

Reply

```
<getPartitionMirrorStats>
   <result value=" ">
      <description/>
   </result>
   <mirrorGroupStatus/>
   <mirrorGroupPercent/>
   <numMirrors/>
   <mirrorInfo>
      <hotFixID/>
      <mirrorPercent/>
   </mirrorInfo>
</getPartitionMirrorStats>
```

Elements

mirrorID

Specifies the mirror group ID for which you want the statistics.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

mirrorGroupStatus

Specifies the status bits for the entire mirror group (see “Mirror Group Statuses” on page 458).

mirrorGroupPercent

Specifies the lowest remirror percentage of any partition in the entire mirror group. If the entire group is fully synchronized, this is 100. If one of the partitions is 63% synchronized and another partition is 77% synchronized, the percentage returned is 63.

numMirrors

Specifies the number of partitions in the mirror group.

mirrorInfo

Repeats for each partition in the mirror group.

hotFixID

Specifies the HotFix ID of the partition.

mirrorPercent

Specifies the remirror complete percentage for the partition.

Example

A nssRequest packet to get the partition mirror statistics on mirror group 20 is as follows:

```
<nssRequest>
   <partition>
      <getPartitionMirrorStats>
         <mirrorID>20</mirrorID>
      </getPartitionMirrorStats>
   </partition>
</nssRequest>
```

The following nssReply packet to the get partition statistics command has two partitions with HotFix IDs of 19 and 16, and both partitions are 100% synchronized with the mirror group:

```
<nssReply>
   <partition>
      <getPartitionMirrorStats>
         <mirrorGroupStatus>7</mirrorGroupStatus>
         <mirrorGroupPercent>100</mirrorGroupPercent>
         <numMirrors>2</numMirrors>
         <mirrorInfo>
            <hotFixID>19</hotFixID>
            <mirrorPercent>100</mirrorPercent>
         </mirrorInfo>

         <mirrorInfo>
            <hotFixID>16</hotFixID>
            <mirrorPercent>100</mirrorPercent>
         </mirrorInfo>
         <result value="0">
            <description/>success</description>
         </result>
      </getPartitionMirrorStats>
   </partition>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

listPartitions

Obtains a detailed list of all partitions. listPartitions can be used to list either all of the partitions on the entire server or all the partitions that exist on a single device.

Request

```
<listPartitions>
   <deviceID/>
</listPartitions>
```

Reply

```
<listPartition>
   <partitionInfo>
      <deviceName/>
      <partitionName/>
      <deviceID/>
      <partitionType/>
      <partitionID/>
      <label/>
      <startingSector/>
      <numSectors/>
      <logicalPartitionID/>
      <logicalPartitionCapacity/>
      <mirrorID/>
      <hotFixID/>
      <hotFixSize/>
      <hotFixAvailSize/>
      <poolName/>
      <volumes>
         <volumeInfo>
            <volumeName/>
            <volStartingSector/>
            <volNumSectors/>
         </volumeInfo>
      </volumes>
      <raidID/>
      <result value=" ">
         <description/>
      </result>
   </partitionInfo>
</listPartition>
```

Elements

deviceName

Specifies the name of the device on which the partition resides, as assigned by Media Manager and the disk driver.

partitionName

Specifies the name of the partition, as assigned by Media Manager and the disk driver.

deviceID

On input, specifies that only the partitions on this device are returned. If not used, all partition on all devices are returned.

On output, specifies the device on which the partition resides, as assigned by Media Manager at boot time.

partitionID

Specifies the physical partition ID, as assigned by Media Manager. The ID represents the physical partition itself but does not represent any HotFix or mirror objects of the partition.

label

(Optional) Specifies a label that was assigned to the partition when it was created. If there is no label, an empty element is returned.

startingSector

Specifies the starting sector offset where the partitions begins on the device.

numSectors

Specifies the length (in sectors) of the partition.

logicalPartitionID

Specifies the logical partition ID that should be used when creating NSS pools or traditional volumes on the partition. If the partition does not have HotFix and mirroring, this ID is the same as the physical partition ID. Otherwise, this ID is the same as the mirrorID. This element is filled only if partitionType is NSS or traditional NetWare.

logicalPartitionCapacity

Specifies the actual capacity (in sectors) of the logical partition. If the partition contains HotFix and mirroring objects, the capacity is smaller than the size of the physical partition due to the overhead associated with HotFix. This element is filled only if partitionType is NSS or traditional NetWare.

mirrorID

Specifies the ID of the mirror group. If the partition does not belong to a mirror group, the ID is 0. This element is filled only if partitionType is NSS or traditional NetWare.

hotFixID

Specifies the ID of the HotFix object. If the partition does not have a HotFix object, the ID is 0. This element is filled only if partitionType is NSS or traditional NetWare.

hotFixSize

Specifies the size (in sectors) of the HotFix area that is reserved to track bad block redirection. If the partition does not have a HotFix object, the size is 0. This element is filled only if partitionType is NSS or traditional NetWare.

hotFixAvailSize

Specifies the useable size (in sectors) of the HotFix area that is reserved to track bad block redirection. HotFix has some overhead, so this size is smaller than the size in hotFixSize. If the partition does not have a HotFix object, the size is 0. This element is filled only if partitionType is NSS or traditional NetWare.

poolName

Specifies the name of the pool if the partition is an NSS partition and a pool has been created that uses the partition. This element is filled only if partitionType is NSS and if the partition is currently owned by an NSS pool.

volumes

Specifies that the partition has one or more traditional volume segments on it. This element is filled only if partitionType is traditional NetWare.

volumeInfo

Repeats for each volume segment on the partition.

volumeName

Specifies the name of the traditional NetWare volume that owns this segment of the partition.

volStartingSector

Specifies the starting sector number of the piece of the partition that is owned by the volume.

volNumSectors

Specifies the number of sectors that are owned by the volume at the starting sector offset.

raidID

Specifies the Media Manager device ID of the RAID device that consumes this virtual device partition. This element is filled only if partitionType is virtual device.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to list partitions is as follows:

```
<nssRequest>
   <partition>
      <listPartitions>
   </partition>
</nssRequest>
```

The following nssReply packet shows three partitions, all on deviceID one. The first partitionInfo represents a DOS partition, the second partitionInfo represents an NSS partition, and the third partitionInfo represents an unpartitioned free space at the end of the device.

```
<nssReply>
   <partition>
      <partitionInfo>
         <deviceName>
            [V312-A0-D0:0] WDIGTL WDE4360-1807A3 rev:1.80
         </deviceName>
         <partitionName>
            [V312-A0-D0:0-P0] Big DOS; OS/2; Win95 Partition
         </partitionName>
         <deviceID>1</deviceID>
         <partitionType>6</partitionType>
         <partitionID>10</partitionID>
         <label></label>
         <startingSector>63</startingSector>
         <numSectors>417627</numSectors>
         <result value="0">
            <description/>success</description>
         </result>
      </partitionInfo>

      <partitionInfo>
         <deviceName>
            [V312-A0-D0:0] WDIGTL WDE4360-1807A3 rev:1.80
         </deviceName>
         <partitionName>
```

```
            [V312-A0-D0:0-PCB] NSS Partition
         <partitionName>
         <deviceID>1</deviceID>
         <partitionType>105</partitionType>
         <partitionID>11</partitionID>
         <label></label>
         <startingSector>417690</startingSector>
         <numSectors>4116480</numSectors>
         <logicalPartitionID>23</logicalPartitionID>
         <logicalPartitionCapacity>
            4108200
         </logicalPartitionCapacity>
         <mirrorID>23</mirrorID>
         <hotFixID>22</hotFixID>
         <hotFixSize>8280</hotFixSize>
         <hotFixAvailSize>8056</hotFixAvailSize>
         <poolName>SYS</poolName>
         <result value="0">
            <description/>success</description>
         </result>
      </partitionInfo>

      <partitionInfo>
         <deviceName>
            [V312-A0-D0:0] WDIGTL WDE4360-1807A3 rev:1.80
         </deviceName>
         <partitionName>
            [V312-A0-D0:0-P8A5] Free Partition Space
         </partitionName>
         <deviceID>1</deviceID>
         <partitionType>0</partitionType>
         <partitionID>13</partitionID>
         <label></label>
         <startingSector>4534170</startingSector>
         <numSectors>3851760</numSectors>
         <result value="0">
            <description/>success</description>
         </result>
      </partitionInfo>
   </partition>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

modifyPartition

Modifies the partition label and size.

Request

```
<modifyPartition>
   <partitionID/>
   <label/>
   <growSize/>
   <shrinkSize/>
</modifyPartition>
```

Reply

```
<modifyPartition>
   <result value=" ">
      <description/>
   </result>
</modifyPartition>
```

Elements

partitionID

(Required) On NetWare, specifies the partition ID received from Media Manager. On Linux, specifies the partition name.

label

(Optional) If exists, specifies the new value for the partition label.

growSize

(Optional) If exists, specifies that the partition size is increasing. Specifies the amount to grow the partition by in bytes (not the desired total partition size). If growSize exists, shrinkSize should not exist.

shrinkSize

(Optional) If exists, specifies that the partition size is decreasing. Specifies the amount to shrink the partition by in bytes (not the desired total partition size). If shrinkSize exists, growSize should not exist.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

Example

A nssRequest packet to remove an NSS partition is as follows:

```
<nssRequest>
   <partition>
      <removePartition>
         <partitionID>14</partitionID>
      </removePartition>
   </partition>
</nssRequest>
```

A nssReply packet to the remove partition command follows:

```
<nssReply>
   <partition>
      <removePartition>
         <result value="0">
            <description/>success</description>
         </result>
      </removePartition>
   </partition>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

removePartition

Deletes an existing partition on a device. If the partition being deleted contains a portion of NSS pools or traditional NetWare volumes, removePartition causes those pools and volumes to also be deleted. However, removePartition does not deleted the eDirectory objects for those pools and volumes. Before deleting a partition, you should use other XML commands to delete the pools and volumes to ensure that all eDirectory cleanup occurs correctly. If the partition being deleted is currently part of an active mirror group, you should first remove the partition from the mirror group before deleting it.

Request

```
<removePartition>
   <partitionID/>
   <ignoreShareState/>
</removePartition>
```

Reply

```
<removePartition>
   <result value=" ">
      <description/>
   </result>
</removePartition>
```

Elements

partitionID

On NetWare, specifies the partition ID received from Media Manager. On Linux, specifies the partition name.

ignoreShareState

(Optional) If exists, no check is done to see if the device is marked shareable for clustering. Usually, the server checks to validate that the clustering software is loaded and operational before allowing partition deletions on a device that is marked for clustering.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

Example

A nssRequest packet to remove an NSS partition is as follows:

```
<nssRequest>
   <partition>
      <removePartition>
         <partitionID>14</partitionID>
      </removePartition>
   </partition>
</nssRequest>
```

A nssReply packet to the remove partition command follows:

```
<nssReply>
   <partition>
      <removePartition>
         <result value="0">
            <description/>success</description>
         </result>
      </removePartition>
   </partition>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

removePartitionFromMirror

Removes a partition from another existing mirror group. Use removePartitionFromMirror only if the mirror group contains more than one partition. If the partition is the only partition in the group, there is no need to remove it.

Request

```
<removePartitionFromMirror>
    <partitionID/>
</removePartitionFromMirror>
```

Reply

```
<remmovePartitionFromMirror>
   <result value=" ">
      <description/>
   </result>
</removePartitionFromMirror>
```

Elements

partitionID

Specifies the physical partition ID of the partition to be removed.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to remove a partition from a mirror group is as follows:

```
<nssRequest>
   <partition>
      <removePartitionFromMirror>
         <partitionID>14</partitionID>
      </removePartitionFromMirror>
   </partition>
</nssRequest>
```

A nssReply packet to the remove partition from a mirror group command follows:

```
<nssReply>
   <partition>
      <removePartitionFromMirror>
         <result value="0">
            <description/>success</description>
         </result>
      </removePartitionFromMirror>
   </partition>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

resyncPartitionMirror

Causes a mirror group to resynchronize.

Request

```
<resyncPartitionMirror>
   <mirrorID/>
</resyncPartitionMirror>
```

Reply

```
<resyncPartitionMirror>
   <result value=" ">
      <description/>
   </result>
</resyncPartitionMirror>
```

Elements

mirrorID

Specifies the ID of the mirror group that needs to be synchronized.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to resynchronize mirror group 20 is as follows:

```
<nssRequest>
   <partition>
      <resyncPartitionMirror>
         <mirrorID>20</mirrorID>
      </resyncPartitionMirror>
   </partition>
</nssRequest>
```

A nssReply packet to the resynchronize command follows:

```
<nssReply>
   <partition>
      <resyncPartitionMirror>
         <result value="0">
            <description/>success</description>
         </result>
      </resyncPartitionMirror>
    </partition>

   <result value="0">
     <description/>zOK</description>
   </result>
</nssReply>
```

3.12 Pool

This section contains the following Pool commands. For more information on NSS pools, see Section 1.6, “Pools,” on page 19.

- “activatePoolSnapshot” on page 163
- “activatePoolSnapshot (Linux)” on page 164
- “addPool” on page 166
- “addPool2” on page 169
- “addPoolSnapshot” on page 171
- “addPoolSnapshot (Linux)” on page 172
- “deactivatePoolSnapshot” on page 174
- “deactivatePoolSnapshot (Linux)” on page 175
- “expandPool” on page 176
- “expandPool2” on page 178
- “getDefaultClusterNames” on page 179
- “getNDSName” on page 181
- “getPoolDevices” on page 183
- “getPoolInfo” on page 184
- “getPoolSnapshotInfo” on page 189
- “getPoolSnapshotInfo (Linux)” on page 190
- “getState” on page 192
- “listPools” on page 194
- “listPoolSnapshots” on page 196
- “listPoolSnapshots (Linux)” on page 198
- “modifyPoolInfo” on page 200
- “modifyState” on page 202
- “poolFreeze” on page 204
- “poolFreezeStatus” on page 206
- “poolThaw” on page 210
- “removePool” on page 212
- “removePool2” on page 214
- “removePoolSnapshot” on page 215
- “removePoolSnapshot (Linux)” on page 216
- “renamePool” on page 217
- “renamePoolSnapshot” on page 219

Each command is wrapped with either the nssRequest or nssReply element and the pool element.

activatePoolSnapshot

Activates a pool snapshot.

Request

```
<activatePoolSnapshot>
   <snapName/>
</activatePoolSnapshot>
```

Reply

```
<activatePoolSnapshot>
   <result value=" ">
      <description/>
   </result>
</activatePoolSnapshot>
```

Elements

snapName

Specifies the name of the pool snapshot.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

activatePoolSnapshot (Linux)

Mounts a pool snapshot.

Request

```
<activatePoolSnapshot>
   <snapName/>
   <shared/>
</activatePoolSnapshot>
```

Reply

```
<activatePoolSnapshot>
   <result value=" ">
      <description/>
   </result>
</activatePoolSnapshot>
```

Elements

snapName

Specifies the name of the pool snapshot.

shared

(optional) Specifies whether the snapshot should be mounted with the shared flag.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following is an example to activate pool snapshot named MYPOOL_S1 .

An nss request to activate pool snapshot

```
<nssRequest>
  <pool>
    <activatePoolSnapshot>
      <snapName>MYPOOL_S1</snapName>
    </activatePoolSnapshot>
  </pool>
</nssRequest>
```

An nss reply packet to activate pool snapshot

```
<nssReply>
  <pool>
    <activatePoolSnapshot>
      <result value="0">
        <description>success</description>
      </result>
    </activatePoolSnapshot>
  </pool>
    <result value="0">
      <description>zOK</description>
    </result>
</nssReply>
```

addPool

Creates an NSS storage pool on the server. This command is implemented only on NetWare and not on Linux.

Request

```
<addPool state=" ">
   <poolName/>
   <ndsName/>
   <context/>
   <noNDSOjbect/>
   <partitions>
      <partitionID/>
   </partitions>
   <lssType/>
   <ignoreShareState/>
   <cluster>
      <advertisingProtocol/>
      <ipAddress/>
      <virtualServerName/>
      <cifsVirtualServerName/>
   </cluster>
</addPool>
```

Reply

```
<addPool>
   <result value=" ">
      <description/>
   </result>
</addPool>
```

Elements

poolName

Specifies the name to be given to the new pool.

ndsName

(Required unless noNDSObject is used) Specifies the name of the eDirectory pool object that represents the pool. If NULL is passed, the name of the eDirectory pool object is generated by prepending the server name and an underscore to the poolName value and adding “_POOL” to the end of the name.

context

(Required unless noNDSObject is used) Specifies the eDirectory context in which the eDirectory pool object is created. If NULL is passed, the eDirectory pool object is created in the same eDirectory context where the server object resides.

noNDSObject

(Optional) Specifies that no eDirectory objects should be created for the pool. If used, the ndsName and context elenments are ignored.

partitions

Specifies a list of one or more partitionID elements.

partitionID

Repeats for each partition being added to the new pool. Specifies the logical ID. If the partition has HotFix and mirroring, the ID is the mirror group ID for the partition. If the partition does not have HotFix and mirroring, the ID is the raw physical partition ID.

lssType

Specifies the loadable storage system type to be used by the new pool. Currently, the only supported type is “ZLSS.”

ignoreShareState

(Optional) Specifies that no check is performed to see if the device is marked shareable for clustering. Usually, the server checks to validate that the clustering software is loaded and operaitonal before allowing pools to be created on a device that is marked shareable.

cluster

(Optional) Specifies that the NSS pool should be auto-enabled to work with clustering software. Use this element only if the device on which the pool is being created is marked shareable for clustering.

advertisingProtocol

(Optional) Specifies a space-separated list of protocols on which the clustered pool advertises its clustering services:

cifs
afp
ncp

ipAddress

(Required) Specifies the IP address to assign to the virtual server for the clustered pool.

virtualServerName

(Required) Specifies the name to assign to the eDirectory virtual server object. If you do not choose your own name, call getDefaultClusterNames (page 179) to retrieve the suggested default virtualServerName before calling addPool.

cifsVirtualServerName

(Required if advertisingProtocol is cifs) Specifies the name to be used to advertise the cluster virtual server on the CIFS protocol. If advertisingProtocol does not specify cifs, this element is ignored.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

state

(Optional) Specifies what state the pool is set to after it is created:

active
deactive

Example

The following is an example of adding a pool named MYPOOL. addPool creates the pool and adds a pool object into eDirectory in the same eDirectory container as the server. The name of the eDirectory pool object is <serverName>_MYPOOL. The pool consumes two partition with IDs 20 and 23.

```
<nssRequest>
   <pool>
      <addPool state="active">
         <poolName>MYPOOL</poolName>
         <ndsName>
         <context>
         <lssType>ZLSS</lssType>
         <partitions>
            <partitionID>20</partitionID>
            <partitionID>23</partitionID>
         </partitions>
      </addPool>
   </pool>
</nssRequest>
```

A nssReply packet to the add pool command follows:

```
<nssReply>
   <pool>
      <addPool>
         <result value="0">
            <description/>success</description>
         </result>
      </addPool>
   </pool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

addPool2

Creates an NSS storage pool on the server.

Request

```
<addPool2 state=" ">
   <poolName/>
   <ndsName/>
   <context/>
   <noNDSOjbect/>
   <devices>
      <addPoolDeviceInfo>
         <objectID/>
         <size/>
      </addPoolDeviceInfo>
   </devices>
   <lssType/>
   <ignoreShareState/>
   <cluster>
      <advertisingProtocols/>
      <ipAddress/>
      <virtualServerName/>
      <cifsVirtualServerName/>
   </cluster>
</addPool2>
```

Reply

```
<addPool2>
   <result value=" ">
      <description/>
   </result>
</addPool2>
```

Elements

poolName

Specifies the name to give to the new pool.

ndsName

(Required, unless noNDSObject is used) Specifies the name of the eDirectory object for the pool. If no name is given, the name is generated by appending the server name and an underscore to the pool name and adding POOL to the end of the name.

context

(Required, unless noNDSObject is used) Specifies the context where the eDirectory pool object is created. If no context is given, the context is assumed to be the same as the server object.

noNDSObject

(Optional) Specifies that only the pool is created (not the eDirectory object). If noNDSOjbect is specified, ndsName, context, and ndsPoolName are ignored.

devices

Specifies the list of devices whose segments are included in the pool.

addPoolDeviceInfo

(Repeating) Specifies a device whose segments need to be included in the pool.

objectID

On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device name.

size

Specifies the size for the device.

lssType

Specifies the LSS type for the pool. Currently, always zLSS.

ignoreShareState

If exists, specifies that the partition is not checked to see if it's shared.

advertisingProtocols

(Optional) Specifies a space-separated list of protocols on which the clustered pool advertises its clustering services. Valid protocols include: cifs, aft, and ncp.

ipAddress

(Required) Specifies the address to be assigned to the virtual server for the clustered pool.

virtualServerName

(Required) Specifies the name to give to the virtual server object in eDirectory.

cifsVirtualServerName

(Required if advertisingProtocols specifies cifs) Specifies the name to use when advertising this virtual server object in the CIFS protocol.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

addPoolSnapshot

Creates a snapshot (snapName) of a pool and designates another pool (snapPoolName) as a snapshot data repository.

Request

```
<addPoolSnapshot>
   <poolName/>
   <snapPoolName/>
   <snapName/>

</addPoolSnapshot>
```

Reply

```
<addPoolSnapshot>
   <result value=" ">
      <description/>
   </result>
</addPoolSnapshot>
```

Elements

poolName

Specifies the name of the pool to take a snapshot of.

snapPoolName

Specifies the pool name on which to store the snapshot data.

snapName

Specifies the name of the pool snapshot.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

addPoolSnapshot (Linux)

Creates a snapshot (snapName) of a pool and designates another pool (snapPoolName) as a snapshot data repository.

Request

```
<addPoolSnapshot>
   <poolName/>
   <freeSpaceID/>
   <numSectors/>
   <snapName/>
</addPoolSnapshot>
```

Reply

```
<addPoolSnapshot>
   <result value=" ">
      <description/>
   </result>
</addPoolSnapshot>
```

Elements

poolName

Specifies the name of the pool to take a snapshot of.

freeSpaceID

Specifies the ID of the free space object to use to store the snapshot data.

snapName

Specifies the name of the pool snapshot.

numSectors

Specifies the number of sectors of the free space to use to store the snapshot data.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following is an example to add a pool snapshot named MYPOOL_S1 to a pool named MYPOOL .

An nss request to add pool snapshot:

```
<nssRequest>
  <pool>
    <addPoolSnapshot>
      <poolName>/dev/evms/MYPOOL</poolName>
        <freeSpaceID>sdb</freeSpaceID>
        <numSectors>102400</numSectors>
        <snapName>MYPOOL_S1</snapName>
    </addPoolSnapshot>
  </pool>
</nssRequest>
```

An nss reply packet to add pool snapshot:

```
<nssReply>
  <pool>
    <addPoolSnapshot>
      <result value="0">
        <description>success</description>
      </result>
    </addPoolSnapshot>
  </pool>
      <result value="0">
        <description>zOK</description>
      </result>
</nssReply>
```

deactivatePoolSnapshot

Deactivates a pool snapshot.

Request

```
<deactivatePoolSnapshot>
   <snapName/>
</deactivatePoolSnapshot>
```

Reply

```
<deactivatePoolSnapshot>
   <result value=" ">
      <description/>
   </result>
</deactivatePoolSnapshot>
```

Elements

snapName

Specifies the name of the pool snapshot.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

deactivatePoolSnapshot (Linux)

Dismounts a pool snapshot.

Request

```
<deactivatePoolSnapshot>
   <snapName/>
</deactivatePoolSnapshot>
```

Reply

```
<deactivatePoolSnapshot>
   <result value=" ">
      <description/>
   </result>
</deactivatePoolSnapshot>
```

Elements

snapName

Specifies the name of the pool snapshot.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following is an example to deactivate a pool snapshot named MYPOOL_S1.

An nss request to deactivate pool snapshot:

```
<nssRequest>
  <pool>
    <deactivatePoolSnapshot>
      <snapName>MYPOOL_S1</snapName>
    </deactivatePoolSnapshot>
  </pool>
</nssRequest>
```

An nss reply packet to deactivate pool snapshot:

```
<nssReply>
  <pool>
    <deactivatePoolSnapshots>
      <result value="0">
        <description>success</description>
      </result>
    </deactivatePoolSnapshots>
  </pool>
      <result value="0">
        <description>zOK</description>
      </result>
</nssReply>
```

expandPool

Expands the size of (adds additional partitions to) an NSS storage pool. This command is implemented only on NetWare and not on Linux.

Request

```
<expandPool>
   <poolName/>
   <partitions>
      <partitionID/>
   </partitions>
</expandPool>
```

Reply

```
<expandPool>
   <result value=" ">
      <description/>
   </result>
</expandPool>
```

Elements

poolName

Specifies the name of the pool to expand.

partitions

Specifies one or more partitions.

partitionID

Repeats for each partition. Specifies the logical ID of a partition to add to the pool. If the pool has HotFix and mirroring, the ID is the mirror group ID for the partition. If the partition does not have HotFix and mirroring, it is the ID of the raw physical partition.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following is an example of expanding the size of MYPOOL by adding another partition to it:

```
<nssRequest>
   <pool>
      <expandPool>
         <poolName>MYPOOL</poolName>
         <partitions>
            <partitionID>18</partitionID>
         </partitions>
      </expandPool>
   </pool>
</nssRequest>
```

A nssReply packet to the expand pool command follows:

```
<nssReply>
   <pool>
      <expandPool>
         <result value="0">
            <description/>success</description>
         </result>
      </expandPool>
   </pool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

expandPool2

Expands the size of (adds additional partitions to) an NSS storage pool.

Request

```
<expandPool2>
   <poolName/>
   <devices>
      <addPoolDeviceInfo>
         <objectID/>
         <size/>
      </addPoolDeviceInfo>
   </devices>
</expandPool2>
```

Reply

```
<expandPool2>
   <result value=" ">
      <description/>
   </result>
</expandPool2>
```

Elements

devices

Specifies the list of devices whose segments are in the pool.

addPoolDeviceInfo

(Repeating) Specifies a device whose segments need to be included in the pool.

objectID

On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device name.

size

Specifies the user-defined size for the device.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

getDefaultClusterNames

Returns the default names the clustering software would assign for a given pool name. Before calling the addPool (page 166) command with the cluster element, you should call this command to acquire the suggested default names for the clustering parameters. Then pass the values returned by getDefaultClusterNames as input element values to the addPool command.

Request

```
<getDefaultClusterNames>
   <poolName/>
</getDefaultClusterNames>
```

Reply

```
<getDefaultClusterNames>
   <poolName/>
   <ndsName/>
   <virtualServerName/>
   <cifsVirtualServerName/>
   <result value=" ">
      <description/>
   </result>
</getDefaultClusterNames>
```

Elements

poolName

Specifies the NSS storage pool name for which default names are to be returned. The named pool does not need to exist.

ndsName

Specifies the suggested default name for the pool's eDirectory object.

virtualServerName

Specifies the suggested default name for the pool's virtual server eDirectory object.

cifsVirtualServerName

Specifies the suggested default name for the pool's CIFS virtual server (that's used in advertising on the CIFS protocol).

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following is an example of calling getDefaultClusterNames on the "ibm_cluster" server on the "MYPOOL" pool:

```
<nssRequest>
   <pool>
      <getDefaultClusterNames>
         <poolName>MYPOOL</poolName>
      </getDefaultClusterNames>
   </pool>
</nssRequest>
```

A nssReply packet to the get default cluster names command follows:

```
<nssReply>
   <pool>
      <getDefaultClusterNames>
         <poolName>MYPOOL</poolName>
         <ndsName>ibm_cluster_MYPOOL_POOL</ndsName>
         <virtualServerName>
            ibm_cluster_MYPOOL_SERVER
         </virtualServerName>
         <cifsVirtualServerName>
            ibm_clust_MYPOO
         </cifsVirtualServerName>
         <result value="0">
            <description/>success</description>
         </result>
      </getDefaultClusterNames>
   </pool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

getNDSName

Returns what the eDirectory name is for an existing NSS storage pool.

Request

```
<getNDSName>
   <poolName/>
</getNDSName>
```

Reply

```
<getNDSName>
   <ndsName/>
   <context/>
   <result value=" ">
      <description/>
   </result>
</getNDSName>
```

Elements

poolName

Specifies the name of the pool for which to find the eDirectory name.

ndsName

Specifies the name of the eDirectory pool object that represents the NSS storage pool.

context

Specifies the eDirectory context of the returned ndsName.

result

Specifies an error value or 9 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to return the eDirectory name is as follows:

```
<nssRequest>
   <pool>
      <getNDSName>
         <poolName>SYS</poolName>
      </getNDSName>
   </pool>
</nssRequest>
```

A nssReply packet to the get name command follows:

```
<nssReply>
   <pool>
      <getNDSName>
         <ndsName>MYSERVER_SYS_POOL</ndsName>
         <context>\MYSERVER_TREE\novell</context>
         <result value="0">
            <description/>success</description>
         </result>
      </getNDSName>
   </pool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

getPoolDevices

Returns the device occupied by the specified pool.

Request

```
<getPoolDevices>
   <poolName/>
<getPoolDevice>
```

Reply

```
<getPoolDevices>
   <deviceSimpleInfo>
      <objectID/>
      <mirrored/>
      <name/>
      <size/>
      <shared/>
   </deviceSimpleInfo>
</getPoolDevices>
```

Elements

poolName

Specifies the name of the pool to return information about.

deviceSimpleInfo

Repeats. Specifies a device (physical or RAID) or a mirror group.

objectID

On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device name.

mirrored

(Optional) Specifies that the device is mirrored.

size

Specifies the size of the device (in bytes).

shared

(Optional) Specifies that the device is shared.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

getPoolInfo

Returns detailed information about an existing NSS storage pool on the server. Also returns the pool's segment information.

Request

```
<getPoolInfo type=" ">
   <poolName/>
<getPoolInfo>
```

Reply

```
<getPoolInfo>
   <basicInfo>
      <mountPoint/>
      <poolName/>
      <ndsPoolName/>
      <ndsPoolGUID/>
      <poolGUID/>
      <poolState/>
      <nameSpaces value=" "/>
      <blockSize/>
      <size/>
      <usedSize/>
      <createdTime value=" "/>
      <modifiedTime value=" "/>
      <lssType/>
   </basicInfo>
   <attributeInfo>
      <supportedAttributes value=" ">
         <readonly/>
         <shared/>
      </supportedAttributes>
      <enabledAttributes value=" ">
         <readonly/>
         <shared/>
      </enabledAttributes>
   </attributeInfo>
   <salvageInfo>
      <freeableSize/>
      <nonFreeableSize
   </salvageInfo/>
   <volumeInfo>
      <volumeName/>
   </volumeInfo>
   <deletedVolumeInfo>
      <volumeName/>
   </deletedVolumeInfo>
   <segmentInfo>
      <segment>
         <deviceName/>
         <segmentName/>
         <deviceID/>
         <segmentID/>
         <label/>
         <offset/>
         <size/>
      </segment>
   </segmentInfo>
   <result value=" ">
      <description/>
   </result>
</getPoolInfo>
```

Elements

poolName

Specifies the name of the pool.

basicInfo

Specifies that the type was either all or basic.

mountPoint

Linux only. Specifies the pool's mount point.

ndsPoolName

Specifies the name of the pool's eDirectory object.

ndsPoolGUID

Specifies the globally unique ID (GUID) of the eDirectory pool object.

poolGUID

Specifies the GUID of the NSS pool.

poolState

Specifies the state of the pool:

mounted
active
deactive
maintenance
unknown

nameSpaces

Specifies a list of name spaces, separated by spaces:

DOS
Long
Macintosh
Unix

blockSize

Specifies the size of the pool's block.

size

Specifies the total size of the pool (in bytes).

usedSize

Specifies the total number of bytes used by the pool. This number also includes all files that are contained in the salvage system.

createdTime

Specifies a string representation of the UTC time when the pool was created.

modifiedTime

Specifies a string representation of the UTC time when the pool was modified.

lddType

Specifies the LSS type for the pool:

ZLSS
CD9660
unknown

attributeInfo

Specifies the type was all or attributes

supportedAttributes

Specifies a list of tags that represent the attributes that are supported by the pool.

readonly

If exists, specifies that the read only feature is supported on the pool.

shared

If exists, specifies that the shareable-for-clustering feature is supported on the pool.

enabledAttributes

Specifies a list of elements that represent supported pool attributes.

salvageInfo

Specifies that the type is all or salvage.

freeableSize

Specifies the number of purgeable bytes on the pool. A purgeable file is one that is deleted but which still exists in the file system so it can be either purged or salvaged.

nonFreeableSize

Specifies the number of nonpurgeable bytes on the pool. A nonpurgeable file is one that is deleted and still exists in the file system but has not yet met the criteria for being purged automatically.

volumeInfo

Specifies that the type is all or volumes.

volumeName

(Repeating) Specifies a list of all NSS logical volume names that are contained in the pool. Each NSS logical volume is represented by one instance.

deletedVolumeInfo

Specifies that the type is all or deletedVolumes.

segmentInfo

Specifies that the type is all or segments.

deviceName

NetWare only. Specifies the name of the device on which the segment resides as assigned by Media Manager and the disk driver.

segmentName

NetWare only. Specifies the name of the segment assigned by Media Manager and the disk driver.

deviceID

On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device name.

segmentID

On NetWare, specifies the partition ID received from Media Manager. On Linux, specifies the partition name.

label

(Optional) Specifies a label that was assigned to the partition when it was created. If there is no label, an empty element is returned.

offset

Specifies the starting offset (in bytes) where the segment begins on the device.

size

Specifies the length of the segment (in bytes).

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

Attributes

type

(Optional) Specifies what type of information is returned. (See Section 12.7, “Pool Types,” on page 459.) If not specified, all information is returned.

nameSpaces value

Specifies the decimal value of the name space mask for the pool.

value

(For createdTime and modifiedTime) Specifies the decimal UTC time.

value

(For supportedAttributes and enabledAttributes) Specifies the decimal value of the pool's bit masks.

Example

A nssRequest packet to return information on the SYS pool is as follows:

```
<nssRequest>
   <pool>
      <getPoolInfo type="all">
         <poolName>SYS</poolName>
      </getPoolInfo>
   </pool>
</nssRequest>
```

A nssReply packet to the get pool information command follows:

```
<nssReply>
   <pool>
      <getPoolInfo>
         <basicInfo>
            <poolName>SYS</poolName>
            <ndsPoolName>
               .CN=BRENDAL_SYS_POOL.O=novell.T=BRENDAL_TREE.
            </ndsPoolName>
            <ndsPoolGUID>
               8ABF1880-E425-11D5-B7-C1-00C04FA33547
            </ndsPoolGUID>
            <poolGUID>
               C252E4EE-E3E8-01D5-80-00-F26070A1467D
            </poolGUID>
            <poolState>active</poolState>
            <nameSpaces value="0"></nameSpaces>
            <blockSize>4096</blockSize>
            <size>2102394880</size>
            <usedSize>1199800320</usedSize>
            <createdTime value="1015426180">
               Mar 6, 2002   7:49:40 am
            </createdTime>
            <modifiedTime value="1015426180">
               Mar 6, 2002   7:49:40 am
            </modifiedTime>
            <lssType>ZLSS</lssType>
         </basicInfo>

         <attributeInfo>
            <supportedAttributes value="123">
               <shared>
            </supportedAttributes>
            <enabledAttributes value="57">
            </enabledAttributes>
         </attributeInfo>

         <salvageInfo>
            <freeableSize>569831424</freeableSize>
            <nonFreeableSize>0</nonFreeableSize>
         </salvageInfo>

         <volumeInfo>
            <volumeName>SYS</volumeName>
         </volumeInfo>

         <deletedVolumeInfo>
         </deletedVolumeInfo>

      <result value="0">
         <description/>success</description>
      </result>
      </getPoolInfo>
   </pool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

getPoolSnapshotInfo

Returns information for a specified pool snapshot.

Request

```
<getPoolSnapshotInfo>
   <snapName/>
</getPoolSnapshotInfo>
```

Reply

```
<getPoolSnapshotInfo>
   <poolSnapshotInfo>
      <snapName/>
      <poolName/>
      <snapPoolName/>
      <poolSize/>
      <time/>
      <allocatedSize/>
      <state/>
   </poolSnapshotInfo>
   <result value="">
      <description/>
   </result>
</getPoolSnapshotInfo>
```

Elements

snapName

Specifies the name of the pool's snapshot.

poolName

Specifies the name of the pool of which to take a snapshot.

snapPoolName

Specifies the pool name on which to store the snapshot data.

time

Specifies the UTC time when the pool snapshot is created.

allocatedSize

Specifies the snapshot's allocated size.

state

Specifies the state of the pool:

active
deactive

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

getPoolSnapshotInfo (Linux)

Returns information for a specified pool snapshot.

Request

```
<getPoolSnapshotInfo>
   <snapName/>
</getPoolSnapshotInfo>
```

Reply

```
<listPoolSnapshots>
   <poolSnapshotInfo>
      <snapName/>
      <poolName/>
      <snapPoolName/>
      <poolSize/>
      <allocatedSize/>
      <poolSize/>
      <percentFull/>
      <state/>
      <mountPoint/>
      <writeable/>
      <result value="">
         <description/>
      </result>
   </poolSnapshotInfo>
   <result value="">
      <description/>
   </result>
</listPoolSnapshots>
```

Elements

snapName

Specifies the name of the pool snapshot.

poolName

Specifies the name of the pool of which to take a snapshot.

snapPoolName

Specifies the name of the segment or partition on which to store the snapshot data.

allocatedSize

Specifies the allocated size of the segment or partition on which to store the snapshot data.

poolSize

Specifies the size of the pool of which to take a snapshot.

percentFull

Specifies how full (as a percentage) the segment or partition on which to store the snapshot is.

state

Specifies the state of the pool:

active
deactive

mountPoint

Specifies the mount point of the snapshot.

writeable

Specifies that the snapshot is writeable.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following is an example to get a pool snapshot named MYPOOL_S1.

An nss request to get pool snapshot information:

```
<nssRequest>
  <pool>
    <getPoolSnapshotInfo>
      <snapName>MYPOOL_S1</snapName>
    </getPoolSnapshotInfo>
  </pool>
</nssRequest>
```

An nss reply packet to get pool snapshot information:

```
<nssReply>
  <pool>
    <getPoolSnapshotInfo>
    <poolSnapshotInfo>
      <snapName>MYPOOL_S1</snapName>
      <poolName>MYPOOL</poolName>
      <snapPoolName>sdb2</snapPoolName>
      <allocatedSize>102400</allocatedSize>
      <poolSize>200672</poolSize>
      <percentFull>1</percentFull>
      <state>Active</state>
      <mountPoint>/opt/novell/nss/mnt/.pools/SNAP1</mountPoint>
      <writeable/>
    </poolSnapshotInfo>      <result value="0">
        <description>success</description>
      </result>
    </getPoolSnapshotInfo>  </pool>      <result value="0">
        <description>zOK</description>
      </result>
</nssReply>
```

getState

Retrieves the current state of an existing NSS storage pool. This command is implemented only on NetWare and not on Linux.

Request

```
<getState>
   <poolName/>
</getState>
```

Reply

```
<getState>
   <poolName/>
   <poolState/>
   <result value=" ">
      <description/>
   </result>
</getState>
```

Elements

poolName

Specifies the name of an existing NSS storage pool for which to return the state.

poolState

Specifies the state of an NSS storage pool:

active
deactive
maintenance

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to returns the state is as follows:

```
<nssRequest>
   <pool>
      <getState>
         <poolName>SYS</poolName>
      </getState>
   </pool>
</nssRequet>
```

A nssReply packet to the get state command follows:

```
<nssReply>
   <pool>
      <getState>
         <poolName>SYS</poolName>
         <poolState>active</poolState>
         <result value="0">
            <description/>success</description>
         </result>
      </getState>
   </pool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

listPools

Lists basic information about all of the NSS storage pools residing on the server. This command is implemented only on NetWare and not on Linux.

Request

```
<listPools>
```

Reply

```
<listPools>
   <poolInfo>
      <poolName/>
      <poolState/>
      <totalBytes/>
      <freeBytes/>
      <enabledAttributeBits/>
      <result value=" ">
         <description/>
      </result>
   </poolInfo>
   <result value=" ">
      <description/>
   </result>
</listPools>
```

Elements

poolInfo

Repeats for each pool.

poolName

Specifies the name of the pool.

poolState

Specifies a numeric value that represents the internal state of the pool (see “Pool States” on page 459).

totalBytes

Specifies the total size (in bytes) of the pool.

freeBytes

Specifies the total number of bytes that are available for use in the pool. This value is calculated by taking the total size of the pool, subtracting how many bytes are actually in use, adding back the number of used bytes in the salvage system that are currently purgeable, and subtracting a free space adjustment amount (which is used by the LSS to reserve a certain amount of free space for internal use so that it always has enough free space to logging and transaction operations).

enabledAttributeBits

Specifies a hex bit value, such as 0x39 (see “Enabled Attributes Bits” on page 457).

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to list pools is as follows:

```
<nssRequest>
   <pool>
      <listPools>
      </listPools>
   </pool>
</nssRequest>
```

A nssReply packet to the modify device command follows:

```
<nssReply>
   <pool>
      <listPools>
         <poolInfo>
            <poolName>SYS</poolName>
            <poolState>6</poolState>
            <totalBytes>2102394880</totalBytes>
            <freeBytes>1471926272</freeBytes>
            <enabledAttributeBits>
               0X39
            </enabledAttributeBits>
            <result value="0">
               <description/>success</description>
            </result>
         </poolInfo>

         <poolInfo>
            <poolName>MYPOOL</poolName>
            <poolState>6</poolState>
            <totalBytes>103809024</totalBytes>
            <freeBytes>94285824</freeBytes>
            <enabledAttributeBits>
               0X39
            </enabledAttributeBits>
            <result value="0">
               <description/>success</description>
            </result>
         </poolInfo>

         <result value="0">
            <description/>success</description>
         </result>
      </listPools>
   </pool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

listPoolSnapshots

Lists all existing pool snapshots and their related information.

Request

```
<listPoolSnapshots>
   <poolName/>
   <details/>
</listPoolSnapshots>
```

Reply

```
<listPoolSnapshots>
   <poolSnapshotInfo>
      <snapName/>
      <poolName/>
      <snapPoolName/>
      <poolSize/>
      <time/>
      <allocatedSize/>
      <state/>
      <result value="">
         <description/>
      </result>
   </poolSnapshotInfo>
   <result value="">
      <description/>
   </result>
</listPoolSnapshots>
```

Elements

poolName

Specifies the name of the pool to return information for.

details

Specifies to return the detailed information for the existing pools. If this element isn't included, listPoolSnapshots returns only a list of snapPoolName elements.

snapName

Specifies the name of the pool snapshot.

poolName

Specifies the name of the pool of which to take a snapshot.

snapPoolName

Specifies the name of the pool on which to store the snapshot data.

time

Specifies the UTC time when the pool snapshot was created.

allocatedSize

Specifies the number of sectors allocated to the snapshot.

state

Specifies the state of the pool:

active
deactive

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

listPoolSnapshots (Linux)

Lists all existing pool snapshots and their related information.

Request

```
<listPoolSnapshots>
   <details/>
   <poolName/>
</listPoolSnapshots>
```

Reply

```
<listPoolSnapshots>
   <poolSnapshotInfo>
      <snapName/>
      <poolName/>
      <snapPoolName/>
      <poolSize/>
      <allocatedSize/>
      <poolSize/>
      <percentFull/>
      <state/>
      <mountPoint/>
      <writeable/>
      <result value="">
         <description/>
      </result>
   </poolSnapshotInfo>
   <result value="">
      <description/>
   </result>
</listPoolSnapshots>
```

Elements

details

(optional) Specifies whether to return all information. Otherwise, only the snapshot names are returned.

poolName

(optional) Specifies to return only the snapshots of the specified pool.

snapName

Specifies the name of the pool snapshot.

poolName

Specifies the name of the pool of which to take a snapshot.

snapPoolName

Specifies the name of the segment or partition on which to store the snapshot data.

allocatedSize

Specifies the allocated size of the segment or partition on which to store the snapshot data.

poolSize

Specifies the size of the pool of which to take a snapshot.

percentFull

Specifies how full (as a percentage) the segment or partition on which to store the snapshot is.

state

Specifies the state of the pool:

active
deactive

mountPoint

Specifies the mount point of the snapshot.

writeable

Specifies that the snapshot is writeable.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following is an example to list all pool snapshots and their related information.

An nss request to list pool snapshots and their information:

```
<nssRequest>
  <pool>
    <listPoolSnapshots>
      <details/>
    </listPoolSnapshots>
  </pool>
</nssRequest>
```

An nss reply packet to list pool snapshots and their information:

```
<nssReply>
  <pool>
    <listPoolSnapshots>
    <poolSnapshotInfo>
      <snapName>MYPOOL_S1</snapName>
      <poolName>MYPOOL</poolName>
      <snapPoolName>sdb2</snapPoolName>
      <allocatedSize>102400</allocatedSize>
      <poolSize>200672</poolSize>
      <percentFull>1</percentFull>
      <state>Active</state>
      <mountPoint>not mounted</mountPoint>
      <writeable/>
    </poolSnapshotInfo>
    </listPoolSnapshots>      <result value="0">
        <description>success</description>
      </result>
    </listPoolSnapshots>
    </listPoolSnapshots>  </pool>      <result value="0">
        <description>zOK</description>
      </result>
</nssReply>
```

modifyPoolInfo

Modifies the properties of an existing NSS storage pool on the server. Note that no properties can currently be modified with modifyPoolInfo. However, the command is documented with examples of how it works in the future when modifications are allowed. This command is implemented only on NetWare and not on Linux.

Request

```
<modifyPoolInfo>
   <poolName/>
   <basicInfo>
      <mountPoint/>
   </basicInfo>
   <enabledAttributes>
      <shared enabled=" "/>
   </enabledAttributes>
</modifyPoolInfo>
```

Reply

```
<modifyPoolInfo>
   <result value=" ">
      <description/>
   </result>
</modifyPoolInfo>
```

Elements

modifyPoolInfo

(Optional, Repeating) Specifies to modify the properties of an existing pool. Currently, no properties can be modified.

poolName

Specifies the name of the pool for which information is to be modifieds.

basicInfo

Specifies the basic information for the pool.

mountPoint

Specifies the pool's mount point for Linux only.

enabledAttributes

Is not used.

shared

Specifies an example only. The shareable-for-clustering state cannot be modified for an NSS storage pool. If modifications were allowed, the enabled attribute would state either yes or no.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

Example

Because the shared attribute cannot currently be modified, the following examples are not functional.

A nssRequest packet to modify pool information is as follows:

```
<nssRequest>
   <pool>
      <modifyPoolInfo>
         <poolName>MYPOOL</poolName>
         <enabledAttributes>
            <shared enabled="yes">
         </enabledAttributes>
      </modifyPoolInfo>
   </pool>
</nssRequest>
```

A nssReply packet to the modify pool information command follows:

```
<nssReply>
   <pool>
      <modifyPoolInfo>
         <result value="0">
            <description/>success</description>
         </result>
      </modifyPoolInfo>
   </pool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

modifyState

Activates or deactivates an existing NSS storage pool.

Request

```
<modifyState>
   <poolName/>
   <poolState/>
   <ignoreShareState/>
</modifyState>
```

Reply

```
<modifyState>
   <result value=" ">
      <description/>
   </result>
</modifyState>
```

Elements

poolName

Specifies the name of an existing NSS storage pool for which the state is modified.

poolState

Specifies the desired target state to which the pool should be set:

active

deactive

ignoreShareState

(Optional) Specifies that a pool can be activated when the clustering software is not present. Usually, the server checks to validate that the clustering software is loaded and operational before allowing pools to be activated on a device that is marked shareable for clustering. Use this switch in rare circumstances where you can ensure that the pool is not in use on another node and that it is not started or used elsewhere. If a shared pool is accidentally activated on two nodes, the pool becomes corrupted.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to modify the state of a pool is as follows:

```
<nssRequest>
   <pool>
      <modifyState>
         <poolName>MYPOOL</poolName>
         <poolState>deactive</poolState>
      </modifyState>
   </pool>
</nssRequest>
```

A nssReply packet to the modify device command follows:

```
<nssReply>
   <pool>
      <modifyState>
         <result value="0">
            <description/>success</description>
         </result>
      </modifyState>
   </pool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

poolFreeze

Freezes the specified pool (as an asynchronous request). Call poolFreezeStatus (page 206) to check on the freeze status. Call poolThaw (page 210) to indicate that you are finished with the freeze. This command is implemented only on NetWare and not on Linux.

Request

```
<poolFreeze>
   <poolName/>
   <timeout/>
   <holdSeconds/>
</poolFreeze>
```

Reply

```
<poolFreeze>
   <userKey value=" "/>
   <poolName value=" "/>
   <result value=" ">
      <description/>
   </result>
</poolFreeze>
```

Elements

poolName

Specifies the name of the pool.

timeout

Specifies the number of seconds to wait before a timeout is generated.

holdSeconds

Specifies the number of seconds to hold the results for the command.

userKey

Specifies for which freeze the results match.

result

Specifies an error value or 0 (for no error). The result applies to the freeze command.

description

Specifies a text description of the result.

Remarks

poolFreeze causes NSS to inform all registered applications to synchronize their data. After all the data for registered applications is synchronized, NSS flushes all user and system-cached data on the pool.

Only active pools can be frozen. If the specified pool is not active when poolFreeze is called, the function returns an NSS error.

If poolFreeze returns success, you must call poolThaw in a timely manner. However, if poolFreeze returns failure, do not call poolThaw.

Example

The nssRequest packet to freeze a pool follows:

```
<nssRequest>
   <pool>
      <poolFreeze>
         <poolName>SM</poolName>
         <timeOut>40</timeOut>
         <holdSeconds>70</holdSeconds>
      </poolFreeze>
   </pool>
</nssRequest>
```

The nssReply packet to freeze a pool follows:

```
<nssReply>
   <pool>
      <poolFreeze>
         <userKey value="28B95AB0-DBB4-01D6-80-00-B78907429F66">
         <poolName value="SM">
         <result value="0">
            <description/>success</description>
         </result>
      </poolFreeze>
   </pool>
   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

poolFreezeStatus

Returns the status of a freeze request. This command is implemented only on NetWare and not on Linux.

Request

```
<poolFreezeStatus>
   <userKey/>
</poolFreezeStatus>
```

Reply

```
<poolFreezeStatus>
   <userKey value=" "/>
   <poolName value=" "/>
   <timeOut value=" "/>
   <holdSeconds value=" "/>
   <thawDone value=" "/>
   <thawStatus value=" "/>
   <thawOperationReturnCode value=" "/>
   <freezeDone value=" "/>
   <freezeStatus value=" "/>
   <results/>
   <count used=" " needed=" "/>
   <application>
      <status value=" "/>
      <source value=" "/>
      <sourceMessage value=" "/>
   </application>
</results>
<result value=" ">
   <description/>
</result>
</poolFreezeStatus>
```

Elements

userKey

Specifies the key (from the poolFreeze (page 204) command) that indicates from which freeze to return the status.

poolName

Specifies the name of the pool.

timeOut

Specifies the number of seconds before the freeze times out.

holdSeconds

Specifies the number of seconds before the status is no longer available.

thawDone

Specifies whether the thaw is complete:

TRUE
FALSE

thawStatus

Specifies the zERR code of the thaw command (only if the thaw is complete and thawDone is TRUE). Otherwise, this element is not used.

thawOperationReturnCode

Specifies the zERR code that was supplied to the thaw command (only if the thaw is complete and thawDone is TRUE). This value might not be the same value that was passed to poolThaw (page 210). For example, if the thaw was automatically performed because of a timeout, this value is set to indicate that condition. If thawDone is FALSE, this element is not used.

freezeDone

Specifies whether the freeze portion is complete:

TRUE
FALSE

freezeStatus

Specifies the zERR code of the freeze command (only if the freeze is complete and freezeDone is TRUE). To determine if the applications have frozen their data, see the results element. Zero indicates that the freeze completed successfully. If freezeDone is FALSE, this element is not used.

results

Specifies application-specific results. This element is used only if freezeDone is TRUE.

count

Specifies the number of application results.

application

Repeats for each application that is registered to freeze its data.

status

Specifies the zERR code from the specified application freeze attempt.

source

Specifies the application name.

sourceMessage

Specifies an application help message. This message can contain information as to why an application cannot freeze its data.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

used

Specifies how many application results have been returned.

needed

Specifies the total number of application results. If this value does not equal the number specified in the used attribute, some results are not present. You can correct this state by performing another freeze and thaw on the pool.

Example

Before Calling a Thaw

The nssRequest packet to return the status of a freeze before calling a thaw follows:

```
<nssRequest>
   <pool>
      <poolFreezeStatus>
         <userKey>28B95AB0-DBB4-01D6-80-00-B78907429F66</userKey>
      </poolFreezeStatus>
   </pool>
</nssRequest>
```

The nssReply packet containing the freeze status follows:

```
<nssReply>
   <pool>
      <poolFreezeStatus>
         <userKey value="28B95AB0-DBB4-01D6-80-00-B78907429F66">
         <poolName value="SM">
         <timeOut value="39">
         <holdSeconds value="69">
         <thawDone value="FALSE">
         <freezeDone value="TRUE">
         <freezeStatus value="0">
         <results>
            <count used="2" needed="2">
            <application>
               <status value="0">
               <source value="Novell.FreezeEventUnitTest.Handler2">
               <sourceMessage value="Hi mom">
            </application>
            <application>
               <status value="0">
               <source value="Novell.FreezeEventUnitTest.Handler1">
               <sourceMessage value="Hello world">
            </application>
         </results>
         <result value="0">
            <description/>success</description>
         </result>
      </poolFreezeStatus>
   </pool>
   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

After Calling a Thaw

The nssRequest packet to return the status of a freeze after calling a thaw follows:

```
<nssRequest>
   <pool>
      <poolFreezeStatus>
         <userKey>28B95AB0-DBB4-01D6-80-00-B78907429F66</userKey>
      </poolFreezeStatus>
   </pool>
</nssRequest>
```

The nssReply packet containing the freeze status follows:

```
<nssReply>
   <pool>
      <poolFreezeStatus>
         <userKey value="28B95AB0-DBB4-01D6-80-00-B78907429F66">
         <poolName value="SM">
         <timeOut value="34">
         <holdSeconds value="64">
         <thawDone value="TRUE">
         <thawStatus value="0">
         <thawOperationReturnCode value="0">
         <freezeDone value="TRUE">
         <freezeStatus value="0">
         <results>
            <count used="2" needed="2">
            <application>
               <status value="0">
               <source value="Novell.FreezeEventUnitTest.Handler2">
               <sourceMessage value="Hi mom">
            </application>
            <application>
               <status value="0">
               <source value="Novell.FreezeEventUnitTest.Handler1">
               <sourceMessage value="Hello world">
            </application>
         </results>
         <result value="0">
            <description/>success</description>
         </result>
      </poolFreezeStatus>
   </pool>
   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

poolThaw

Allows a pool to thaw. After poolThaw is called, the following tags from poolFreezeStatus (page 206) must have a value of 0 to indicate a successful freeze/thaw operation: freezeStatus, thawStatus, and thawOperationReturnCode. In addition, the results content should indicate that all applications were successful in the quiescent attempts.

Request

```
<poolThaw>
   <userKey/>
   <thawOperationReturnCode/>
</poolThaw>
```

Reply

```
<poolThaw>
    <userKey value=" "/>
    <poolName value=" "/>
    <result value=" ">
       <description/>
    </result>
</poolThaw>
```

Elements

userKey

Specifies a key that indicates from which freeze to return the status.

thawOperationReturnCode

Specifies a zERR code that indicates any errors. zOK indicates there were no errors.

poolName

Specifies the name of the pool.

result

Specifies an error value or 0 (for no error). The result applies directly to the thaw operation.

description

Specifies a text description of the result.

Example

The nssRequest packet to thaw a pool follows:

```
<nssRequest>
   <pool>
      <poolThaw>
         <userKey>28B95AB0-DBB4-01D6-80-00-B78907429F66</userKey>
      </poolThaw>
   </pool>
</nssRequest>
```

The nssReply packet from a pool thaw follows:

```
<nssReply>
   <pool>
      <poolThaw>
         <userKey value="28B95AB0-DBB4-01D6-80-00-B78907429F66">
         <poolName value="SM">
         <result value="0">
            <description/>success</description>
         </result>
      </poolThaw>
   </pool>
   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

removePool

Deletes an NSS storage pool on the server. Unless instructed not to, removePool also deletes the eDirectory object for the pool. Deleting an NSS storage pool implicitly deletes all NSS logical volumes on the pool. However, is does not delete the eDirectory objects for all of those NSS logical volumes. If you want the volume's eDirectory objects to be properly cleaned up, first delete all of the logical volumes that are contained on the pool before deleting the pool itself. This command is implemented only on NetWare and not on Linux.

Request

```
<removePool>
   <poolName/>
   <dontRemoveNDSObject/>
   <ignoreShareState/>
</removePool>
```

Reply

```
<removePool>
   <result value=" ">
      <description/>
   </result>
</removePool>
```

Elements

poolName

Specifies the name of the pool to delete.

dontRemoveNDSObject

Specifies not to delete the eDirectory object for the pool.

ignoreShareState

(Optional) Specifies not to check and see if the device is marked shareable for clustering. Usually, the server checks to validate that the clustering software is loaded and operational before allowing pools to be deleted on a device that is marked shareable.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to remove a pool is as follows:

```
<nssRequest>
   <pool>
      <removePool>
         <poolName>MYPOOL</poolName>
      </removePool>
   </pool><
 </nssRequest>
```

A nssReply packet to the remove pool command follows:

```
<nssReply>
   <pool>
      <removePool>
         <result value="0">
            <description/>success</description>
         </result>
      </removePool>
   </pool>
   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

removePool2

Deletes the specified pool.

Request

```
<removePool2>
   <poolName/>
</removePool2>
```

Reply

```
<removePool2>
   <result value=" ">
      <description/>
   </result>
</removePool2>
```

Elements

poolName

Specifies the name of the pool to delete.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

removePoolSnapshot

Removes a specified pool snapshot.

Request

```
<removePoolSnapshot>
   <snapName/>
</removePoolSnapshot>
```

Reply

```
<removePoolSnapshot>
   <result value=" ">
      <description/>
   </result>
</removePoolSnapshot>
```

Elements

snapName

Specifies the name of the pool's snapshot.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

removePoolSnapshot (Linux)

Removes a specified pool snapshot named MYPOOL_S1.

Request

```
<removePoolSnapshot>
   <snapName/>
</removePoolSnapshot>
```

Reply

```
<removePoolSnapshot>
   <result value=" ">
      <description/>
   </result>
</removePoolSnapshot>
```

Elements

snapName

Specifies the name of the pool's snapshot.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following is an example to remove a pool snapshot:

An nss request to remove a pool snapshot:

```
<nssRequest>
  <pool>
    <removePoolSnapshot>
      <snapName>MYPOOL_S1</snapName>
    </removePoolSnapshot>
  </pool>
</nssRequest>
```

An nss reply packet to remove a pool snapshot:

```
<nssReply>
  <pool>
    <removePoolSnapshot>
      <result value="0">
        <description>success</description>
      </result>
    </removePoolSnapshot>
  </pool>
      <result value="0">
        <description>zOK</description>
      </result>
</nssReply>
```

renamePool

Renames an NSS storage pool on the server (and optionally renames the corresponding eDirectory pool object).

Request

```
<renamePool>
   <poolName/>
   <newPoolName/>
   <newNDSName/>
   <dontRenameNDSObject/>
</renamePool>
```

Reply

```
<renamePool>
   <result value=" ">
      <description/>
   </result>
</renamePool>
```

Elements

poolName

Specifies the current name of the pool.

newPoolName

Specifies the new name the pool is known by on the server.

newNDSName

(Required unless dontRenameNDSObject is specified) Specifies the new name to assign to the pool's eDirectory object. If NULL is passed, a default name is generated by prepending the server name and adding “_POOL” to the end of the name.

dontRenameNDSObject

Specifies not to rename the eDirectory object for the pool. If this element is used, newNDSName is ignored.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following example renames MYPOOL to YOURPOOL and also renames the eDirectory pool object to the new default name:

```
<nssRequest>
   <pool>
      <renamePool>
         <poolName>MYPOOL</poolName>
         <newPoolName>YOURPOOL</newPoolName>
         <newNDSName>
      </renamePool>
   </pool>
</nssRequest>
```

A nssReply packet to the rename pool command follows:

```
<nssReply>
   <pool>
      <renamePool>
         <result value="0">
            <description/>success</description>
         </result>
      </renamePool>
   </pool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

renamePoolSnapshot

Renames a pool snapshot. This command is implemented only on NetWare and not on Linux.

Request

```
<renamePoolSnapshot>
   <snapName/>
   <newSnapName/>
</renamePoolSnapshot>
```

Reply

```
<renamePoolSnapshot>
   <result value=" ">
      <description/>
   </result>
</renamePoolSnapshot>
```

Elements

snapName

(Required) Specifies the current snapshot name of the pool.

newSnapName

(Required) Specifies the new snapshot name for the pool.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

3.13 RAID

The following commands can be called to manipulate software RAID devices on a server:

- “addRAID” on page 221
- “addRAID2” on page 224
- “expandRAID” on page 226
- “removeRAID” on page 228
- “removeRAID2” on page 229
- “renameRAID” on page 230
- “restripeRAID” on page 231

Each command is wrapped with either the nssRequest or nssReply element and the raid element.

NSS currently supports software RAID 0. In the future, it will support other RAID types. A software RAID device is made up of virtual device partitions, which have a partition type of 207 (0xCF). These virtual device partitions are created for you by the addRAID (page 221) and expandRAID (page 226) commands.

This section does not apply to RAID hardware configurations, which are fully supported in the NSS environment.

addRAID

Creates a software RAID device form one or more free spaces. Currently, only RAID type 0 is supported. However, other RAID types will be supported in the future. Note that all segments in a software RAID device must be exactly the same size.

Request

```
<addRAID>
   <stripeSize/>
   <raidType/>
   <raidSegments>
      <segment>
         <deviceID/>
         <freeSpaceID/>
         <startingSector/>
         <numSectors/>
      </segment>
   </raidSegments>
</addRAID>
```

Reply

```
<addRAID>
   <result value=" ">
      <description/>
   </result>
   <partialResult value=" ">
      <description/>
      <partitionsRequested/>
      <partitionsCreated/>
      <partitionsAdded/>
      <partitionsStranded/>
      <maxPartitions/>
   </partialResult>
</addRAID>
```

Elements

stripeSize

Specifies the strip size for the RAID device.

raidType

Specifies the type of RAID. For example, 0 for a RAID 0 device.

raidSegments

Specifies the list of RAID segments to add to the RAID device.

segment

Repeats for each RAID segment to add.

deviceID

Specifies the device ID on which the segment is created.

freeSpaceID

Specifies the ID of the free space partition of the device from which to consume space.

startingSector

Specifies the starting sector offset within the free space partition of the segment to include.

numSectors

Specifies the number of sectors to include.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

partialResult

(Optional) Specifies what happened and why not all of the segments were added. It is possible that the RAID device was created but not all of the requested segments were successfully added. In this case, result is zOK and this element specifies more details about what occurred.

partitionRequested

Specifies the number of partitions that were originally supposed to be added to the RAID device.

partitionsCreated

Specifies the number of partitions that were successfully created with the intent to add them to the RAID device.

partitionsAdded

Specifies the number of partitions that were successfully added to the RAID device.

partitionsStranded

Specifies the number of partitions that were created but could not be successfully added to the RAID device. These partitions were, for some reason, not able to be deleted either.

maxPartitions

(Optional) Specifies that one or more partitions were not added to the RAID device because the maximum number of RAID segments was exceeded on the device.

Attributes

value

Specifies the first reason why the operation partially failed.

Example

The following example creates a RAID 0 device with two segments: one on device 2 and one on device 3:

```
<nssRequest>
   <raid>
      <addRAID>
         <stripeSize>65536</stripeSize>
         <raidType>0</raidType>
         <raidSegments>
            <segment>
               <deviceID>2</deviceID>
               <freeSpaceID>7</freeSpaceID>
               <startingSector>208845</startingSector>
               <numSectors>102400</numSectors>
            </segment>
            <segment>
               <deviceID>3</deviceID>
               <freeSpaceID>14</freeSpaceID>
               <startingSector>197321</startingSector>
               <numSectors>102400</numSectors>
            </segment>
         </raidSegments>
      </addRAID>
   </raid>
</nssRequest>
```

A nssReply packet to the add RAID command follows:

```
<nssReply>
   <raid>
      <addRAID>
         <result value="0">
            <description/>success</description>
         </result>
      </addRAID>
   </raid>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

addRAID2

Creates a RAID 0, RAID 1 mirror group, or a RAID 5 device.

Request

```
<addRAID2>
   <stripeSize/>
   <raidName/>
   <raidType/>
   <raidSegments>
      <segment>
         <deviceID/>
         <freeSpaceID/>
      </segment>
   </raidSegments>
</addRAID2>
```

Reply

```
<addRAID2>
   <raidID/>
   <result value=" ">
      <description/>
   </result>
</addRAID2>
```

Elements

stripeSize

(Optional, but required for RAID 0 and 5)

raidName

(Optional)

raidType

(Required)

numSectors

(Required) Specifies the RAID segment size.

segment

(Repeating) Repeat for all segments.

deviceID

(Required) On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device name.

freeSpaceID

(Required) On NetWare, specifies the free space ID received from Media Manager. On Linux, specifies the free space name.

raidID

(Required) On NetWare, specifies the RAID ID received from Media Manager. On Linux, specifies the RAID name.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

expandRAID

Adds additional segments to a software RAID device.

Request

```
<expandRAID>
   <raidID/>
   <raidSegments>
      <segment>
         <deviceID/>
         <freeSpaceID/>
         <startingSector/>
         <numSectors/>
      </segment>
   </raidSegments>
</expandRAID>
```

Reply

```
<expandRAID>
   <result value=" ">
      <description/>
   </result>
</expandRAID>
```

Elements

raidID

(Required) On NetWare, specifies the RAID ID received from Media Manager. On Linux, specifies the RAID name.

raidSegments

Specifies a list of RAID segments (partitions) to add.

segment

(Repeating) Specifies a RAID segment to add.

deviceID

(Required) On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device name.

freeSpaceID

(Required) On NetWare, specifies the free space ID received from Media Manager. On Linux, specifies the free space name.

startingSector

(Optional) Specifies the starting sector offset within the free space partition of the segment to include.

numSectors

(Optional) Specifies the number of sectors to include.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

Example

The following example adds one additional segment from device 2 to the existing RAID device with an ID of 21:

```
<nssRequest>
   <raid>
      <expandRAID>
         <raidID>21</raidID>
         <raidSegments>
            <segment>
               <deviceID>2</deviceID>
               <freeSpaceID>7</freeSpaceID>
               <startingSector>321300</startingSector>
               <numSectors>102400</numSectors>
            </segment>
         </raidSegments>
      </expandRAID>
   </raid>
</nssRequest>
```

A nssReply packet to the expand RAID command follows:

```
<nssReply>
   <raid>
      <expandRAID>
         <result value="0">
            <description/>success</description>
         </result>
      </expandRAID>
   </raid>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

removeRAID

Deletes a software RAID device.

Request

```
<removeRAID>
   <raidID/>
</removeRAID>
```

Reply

```
<removeRAID>
    <result value=" ">
       <description/>
    <result>
</removeRAID>
```

Elements

raidID

Specifies the ID of the RAID device to be deleted.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to remove a RAID device is as follows:

```
<nssRequest>
   <raid>
      <removeRAID>
         <raidID>21</raidID>
      </removeRAID>
   </raid>
</nssRequest>
```

A nssReply packet to the remove RAID command follows:

```
<nssReply>
   <raid>
      <removeRAID>
         <result value="0">
            <description/>success</description>
         </result>
      </removeRAID>
   </raid>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply
```

removeRAID2

Deletes the specified RAID and deletes all of the partitions in the RAID, including any pool and volume that resides on the RAID.

Request

```
<raid>
   <removeRAID2>
      <raidID/>
   </removeRAID2>
</raid>
```

Reply

```
<raid>
   <removeRAID2>
      <result value=" ">
         <description/>
      <result>
   </removeRAID2>
</raid>
```

Elements

raidID

On NetWare, specifies the RAID ID received from Media Manager. On Linux, specifies the RAID name.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

renameRAID

Renames a RAID 0, RAID 1 - mirror group, or a RAID 5 device.

Request

```
<renameRAID>
   <raidID/>
   <name/>
</renameRAID>
```

Reply

```
<renameRAID>
   <result value=" ">
      <description/>
   <result>
</renameRAID>
</raid>
```

Elements

raidID

(Required) On NetWare, specifies the RAID ID received from Media Manager. On Linux, specifies the RAID name.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

restripeRAID

Initiates a restripe command on a software RAID device.

Request

```
<restripeRAID>
   <raidID/>
</restripeRAID>
```

Reply

```
<restripeRAID>
   <result value=" ">
      <description/>
   <result>
</restripeRAID>
```

Elements

raidID

(Required) On NetWare, specifies the RAID ID received from Media Manager. On Linux, specifies the RAID name.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

Example

A nssRequest packet to restripe a RAID device is as follows:

```
<nssRequest>
   <raid>
      <restripeRAID>
         <raidID>21</raidID>
      </restripeRAID>
   </raid>
</nssRequest>
```

A nssReply packet to the restripe RAID command follows:

```
<nssReply>
   <raid>
      <restripeRAID>
         <result value="0">
            <description/>success</description>
         </result>
      </restripeRAID>
   </raid>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

3.14 Server

This section contains the following Server commands:

- “getServerFreeSpace” on page 233
- “listDevices (Server)” on page 235
- “listPartitions (Server)” on page 236
- “listPools” on page 239

Each command is wrapped with either the nssRequest or nssReply element and the server element.

getServerFreeSpace

Returns free space available on a server. It also returns mirror groups (the mirror ID and the available space in bytes) and virtual RAID devices (device ID and the available free space in bytes).

Request

```
<getServerFreeSpace type="detail"/>
```

Reply

```
<getServerFreeSpace>
   <deviceSimpleInfo>
      <objectID/>
      <mirrored/>
      <name/>
      <size/>
      <shared/>
      <freeSpaces>
         <freeSpaceInfo>
            <freeSpaceID/>
            <size/>
            <offset/>
         </freeSpaceInfo>
      </freeSpaces>
   </deviceSimpleInfo>
</getServerFreeSpace>
```

Elements

deviceSimpleInfo

Repeats for each device (physical or RAID) or mirror group being listed.

objectID

Specifies the NetWare device ID from Media Manager or the Linux device object name.

mirrored

(Optional) Specifies that the device is a mirrored virtual device.

name

(Optional) Specifies the name of the device as assigned by Media Manager (for NetWare only).

size

(Optional) Specifies the free space (in bytes) for this device or mirror group.

shared

(Optional) Specifies that the device is flagged as a shared device.

freeSpaces

(Optional) Specifies the detailed free space information

freeSpaceInfo

Repeats for each free space on the device.

freeSpaceID

(Required) Specifies the NetWare device ID received from Media Manager or the Linux device object name.

size

(Required)

offset

(Required)

Attributes

type

(Optional)

listDevices (Server)

Returns a list of all of a server's devices. The difference between this command and the listDevices (page 98) command is the outer tag and whether each command is called as a device or a server command.

Request

```
<listDevices type=" "/>
```

Reply

```
<listDevices>
   <serverDeviceInfo>
      <objectID/>
      <name/>
      <mirrored/>
      <shared/>
   </serverDeviceInfo>
</listDevices>
```

Elements

serverDeviceInfo

Repeats for each device (physical or raid) or mirror group being listed.

objectID

Specifies the NetWare device ID received from Media Manager or the Linux device object name.

name

Specifies the name of the device, as assigned by Media Manager (for NetWare only).

mirrored

(Optional) Specifies that the device is a mirrored virtual device.

shared

(Optional) Specifies that the device is flagged as a shared device.

Attributes

type

Specifies the types of devices to list:

all
physical
raid
mirror
virtual (which includes raid and mirror)

listPartitions (Server)

Lists all partitions on a server.

Request

```
<listPartitions type=" "/>
```

Reply

```
<listPartitions>
   <partitionInfo>
      <partitionID/>
      <type/>
      <details>
         <partitionName/>
         <state/>
         <label/>
         <deviceName/>
         <deviceID/>
         <poolName/>
         <startingSector/>
         <size/>
         <logicalPartitionID/>
         <logicalPartitionCapacity/>
         <mirrorID/>
         <hotFixID/>
         <hotFixSize/>
         <hotFixAvailSize/>
         <raidID/>
         <growable/>
         <growSize/>
         <shrinkable/>
      </details>
   </partitionInfo>
   <result value=" ">
      <description/>
   </result>
</listPartitions>
```

Elements

partitionInfo

Repeats for each partition on the server.

partitionID

(Required) On NetWare, specifies the partition ID number as received from Media Manager. On Linux, specifies the partition name.

type

(Optional) Specifies the partition type. The only valid type is detail. Any other value is treated as if you left this attribute out.

details

(Optional) Specifies to return the child elements.

partitionName

(Optional) Specifies the name of the partition, as assigned by Media Manager (for NetWare only).

state

Specifies the state of the partition:

used
free

label

(Optional) Specifies the user-defined partition label.

deviceName

(Optional) Specifies the name of the device, as assigned by Media Manager (for NetWare only).

deviceID

On NetWare, specifies the device ID received from Media Manager. On Linux, specifies the device object name.

poolName

Specifies that the pool name (if the partitions contains a pool).

startingSector

Specifies the partition's starting sector.

size

Specifies the partition size (in bytes).

logicalPartitionID

On NetWare, specifies the logical partition ID received from Media Manager. On Linux, specifies the logical partition name.

logicalPartitionCapacity

Specifies the logical partition capacity in bytes.

mirrorID

(Optional) On NetWare, specifies the mirror ID received from Media Manager.

hotFixID

(Optional) On NetWare, specifies the hot fix ID received from Media Manager. On Linux, specifies the hot fix name.

hotFixSize

(Optional) Specifies the hot fix size (in bytes).

hotFixAvailSize

(Optional) Specifies the hot fix available size (in bytes).

raidID

(Optional) Specifies that this partition is part of a RAID. On NetWare, specifies the RAID ID received from Media Manager. On Linux, specifies the RAID name.

growable

(Optional) Specifies that the partition can grow (for Linux only).

growSize

(Optional) Specifies the total number of bytes the partition can grow to (for Linux only).

shrinkable

(Optional) Specifies that the partition can shrink (for Linux only).

listPools

Returns the pool list on a server.

Request

```
<listPools/><noSnapshot/>
```

Reply

```
<listPools>
    <PoolSimpleInfo>
       <poolName/>
       <poolState/>
       <shared/>
   </poolSimpleInfo>
</listPools>
```

Elements

noSnapshot

(Optional) Snapshot pools are excluded from the displayed list of the pools .

poolName

Repeats for each pool.

poolState

Specifies the name of the pool.

shared

(Optional) Specifies that the pool is shared.

3.15 User Space Restriction

The commands in this section are intended for user operations that can be performed by non-administrative users.

Use the following path name to open the user.cmd file:

```
_admin/Manage_NSS/manage.cmd
```

Every command is wrapped with either the nssRequest and userSpaceRestrictions elements or the nssReply element.

This section contains definitions for the following command categories:

- “browseUserSpaceRestrictions (user)” on page 240
- “getUserSpaceRestriction (user)” on page 242
- “setUserSpaceRestriction (user)” on page 244
- “listUserSpaceRestrictions (user)” on page 246

browseUserSpaceRestrictions (user)

Returns the list of users with space restrictions on a specified volume and the quotas for those users.

Request

```
<browse>
   <includeADIdentities/>
   <volumeName/>
   <allUsers/>
</browse>
```

Reply

```
<browse>
   <userList>
      <user>
         <id/>
         <userName/>
         <quota/>
         <spaceUsed/>
      </user>
   </userList>
   <result value="">
      <description/>
   </result>
<browse>
```

Elements

includeADIdentities

Specifies to include AD identities as part of the trustees in the XML reply. The user type is specified using a new attribute `type`, which will specify the value of the attribute as `adID` to indicate an Active Directory identity.

volumeName

(Required) Specifies the volume's name for which to return the user space restriction.

allUsers

(Optional) Specifies to return all users that are using storage but have no restrictions. This functionality is useful in listing the storage in use for all users.

user

Specifies user quota for the users who are assigned user quota. Attribute `type` specifies `adID` to indicate an Active Directory identity.

id

Specifies the unique ID for the user.

userName

Specifies the user name for the restricted user.

quota

Specifies the quota for the restricted user.

spaceUsed

Specifies the space in use by the restricted user.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to list space restrictions for the MYVOL volume is as follows:

```
<nssRequest>
   <userSpaceRestrictions>
      <browse>
         <volumeName>MYVOL</volumeName>
      </browse>
   </userSpaceRestrictions>
</nssRequest>
```

A nssReply packet to the list space restrictions command follows:

```
<nssReply>
   <userSpaceRestrictions>
      <browse>
         <userList>
            <user>
               <userName>somebody.somedept.someorg</username>
               <quota>1048576</quota>
               <spaceUsed>401524</spaceUsed>
            </user>
            <user>
               <userName>someone.somedept.someorg</username>
               <quota>262144</quota>
               <spaceUsed>65534</spaceUsed>
            </user>
         </userList>
      </browse>
   </userSpaceRestrictions>
   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

getUserSpaceRestriction (user)

Returns the user space restriction for a supplied user on a specified volume. Attribute `type` in the request specifies `adID` to indicate an Active Directory identity.

Request

```
<get>
   <volumeName/>
   <id/>
   <userName/>
</get>
```

Reply

```
<get>
   <quota/>
   <noQuota/>
   <fullyRestricted/>
   <spaceUsed/>
   <result value="">
      <description/>
   </result>
</get>
```

Elements

volumeName

(Required) Specifies the volume's name for which to return the user space restrictions.

id

Specifies the unique ID for the user as returned from browseUserSpaceRestrictions (user) (page 240). Either the user's ID or name must be specified.

userName

Specifies the DN of the user for which to return restrictions. Either the user's name or ID must be specified.

quota

Specifies the quota for the requested user.

noQuota

Specifies that the user has no limit quota on the specified volume.

fullyRestricted

Specifies that the user is limited to no space or new usage on the specified volume.

spaceUsed

Specifies the space in use by the restricted user.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to get space restrictions for a volume is as follows:

```
<nssRequest>
   <userSpaceRestrictions>
      <get>
         <volumeName>MYVOL</volumeName>
         <userName>somebody.somedept.someorg</userName>
      </get>
   </userSpaceRestrictions>
</nssRequest>
```

A nssReply packet to the get space restrictions command follows:

```
<nssReply>
   <userSpaceRestrictions>
      <get>
          <quota>1048576</quota>
          <spaceUsed>401524</spaceUsed>
       </get>
   </userSpaceRestrictions>
   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

setUserSpaceRestriction (user)

Adds the user space restriction for a supplied user DN to a specified volume. Attribute `type` in the request specifies `adID` to indicate an Active Directory identity

Request

```
<set>
   <volumeName/>
   <id/>
   <userName/>
   <quota/>
   <noQuota/>
   <fullyRestricted/>
</set>
```

Reply

```
<set>
   <result value="">
      <description/>
   </result>
</set>
```

Elements

volumeName

(Required) Specifies the volume's name to which to add a user space restriction.

id

Specifies the unique ID of the user. Either the user's ID or name must be specified.

userName

Specifies the DN of the user to add. Either the user's name or ID must be specified.

quota

(Required unless noQuota or fullyRestricted is specified) Specifies the quota for the specified user.

noQuota

(Required unless quota or fullyRestricted is specified) Specifies that the user has no limit and removes any existing restriction.

fullyRestricted

(Required unless quota or noQuota is specified) Specifies that the user is limited to no more space.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to set space restrictions for a volume is as follows:

```
<nssRequest>
   <userSpaceRestrictions>
      <set>
         <volumeName>MYVOL</volumeName>
         <userName>somebody.somedept.someorg</userName>
         <quota>1048576</quota>
      </set>
   </userSpaceRestrictions>
 </nssRequest>
```

A nssReply packet to the set space restrictions command follows:

```
<nssReply>
   <userSpaceRestrictions>
      <set>
          <result value="0">
               <description/>success</description>
            </result>
       </set>
   </userSpaceRestrictions>
   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

listUserSpaceRestrictions (user)

Lists the user space restrictions for n number of users, based on the last user information provided.

Request

```
<list>
   <includeADIdentities/>
   <volumeName/>
   <lastUserID/>
   <numEntries/>
   <allUsers/>
</list>
```

Reply

```
<list>
<userList>
      <user>
         <id/>
         <userName/>
         <quota/>
         <spaceUsed/>
      </user>
   </userList>
   <result value="">
      <description/>
   </result>
</list>
```

Elements

includeADIdentities

Specifies to include AD identities as part of the trustees in the XML reply. The user type is specified using a new attribute `type`, which will specify the value of the attribute as `adID` to indicate an Active Directory identity.

volumeName

(Required) Specifies the volume's name to which to add a user space restriction.

lastUserID

Specifies the unique ID of the last user. Either the user's ID or name must be specified. Passing of an invalid GUID `00000000-0000-0000-00-00-000000000000` indicates that the user ID should begin with the first user

numEntries

Specifies the numebr of entries from the last user ID.

allUsers

(Optional) Specifies to return all users that are using storage but have no restrictions. This functionality is useful in listing the storage in use for all users.

fullyRestricted

Specifies that the user is limited to no more space.

noQuota

Specifies that the user has no limit.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

user

Specifies user quota for the users who are assigned user quota. Attribute `type` specifies `adID` to indicate an Active Directory identity.

Example

A nssRequest packet to list user quota for a volume is as follows:

```
<nssRequest>
   <userSpaceRestrictions>
      <list>
         <volumeName><![CDATA[VOL2]]></volumeName>
         <lastUserID>00000000-0000-0000-00-00-000000000000</lastUserID>
         <numEntries>20</numEntries>
      </list>
   </userSpaceRestrictions>
 </nssRequest>
```

A nssReply packet to list user quota for a volume is as follows:

```
<nssReply>
   <userSpaceRestrictions>
      <list>
         <userList>
            <user>
              <id>ab959229-a26c-4c58-16-ba-299295ab6ca2</id>
              <userName>wwwrun.novell</userName>
              <quota>1048576</quota>
              <spaceUsed>0</spaceUsed>
            </user>
         </userList>
        <result value="0">
         <description/>zOK</description>
        </result>
      </list>
    </userSpaceRestrictions>
</nssReply>
```

3.16 VLDB

This section contains the following commands that can be used to manipulate Volume Location Data Base (VLDB):

- “addVolumeToVLDB” on page 249
- “createNewService” on page 251
- “deleteService” on page 253
- “getVLDBInfo” on page 255
- “loadVLDB” on page 263
- “lookup” on page 264
- “removeVolumeFromVLDB” on page 266
- “replicaAddedToVLDB” on page 267
- “replicaRemovedFromVLDB” on page 268
- “setVLDBConfiguration” on page 269
- “shutdownVLDB” on page 271
- “startRepair” on page 272
- “startService” on page 274
- “stopRepair” on page 275
- “stopService” on page 276

VLDBs are used by Distributed File System (DFS) to track the physical location of volumes that are stored in NSS. The DFS GUID for each volume is stored in eDirectory. VLDBs can look up that GUID and determine which server(s) contain the volume in question.

Each command is wrapped with either the nssRequest or nssReply element and the vldb element.

addVolumeToVLDB

Adds a volume to a VLDB.

Request

```
<addVolumeToVLDB>
   <dfsGUID/>
   <ndsServerName/>
   <volumeName/>
</addVolumeToVLDB>
```

Reply

```
<addVolumeToVLDB>
   <result value=" ">
      <description/>
   <result>
</addVolumeToVLDB>
```

Elements

dfsGUID

Specifies the DFS GUID of the volume to add.

ndsServerName

Specifies the eDirectory name, in relative distinguished-name format, of the server which owns the volume to add.

volumeName

Specifies the name of the volume to add.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to add a volume to a VLDB is as follows:

```
<nssRequest>
   <vldb>
      <addVolumeToVLDB>
         <dfsGUID>
            906D22B6-32DD-01D6-80-06-FBDA22AE6917
         </dfsGUID>
         <ndsServerName>
            MYSERVER.novell.MY_TREE.
         </ndsServerName>
         <volumeName>NSS3</volumeName>
      </addVolumeToVLDB>
   </vldb>
</nssRequest>
```

A nssReply packet to the add volume to VLDB command follows:

```
<nssReply>
   <vldb>
      <addVolumeToVLDB>
         <result value="0">
            <description/>success</description>
         </result>
      </addVolumeToVLDB>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

createNewService

Creates a new VLDB service on a server.

Request

```
<createNewService>
   <dataBasePath/>
   <autoLoadVLDB/>
   <managementContext>
      <ndsObject/>
      <tgtTree/>
   </managementContext>
   <dbInfo>
      <backEndDB/>
      <dataBasePath/>
   </dbInfo>
   <user/>
   <password/>
   <unp/>
</createNewService>
```

Reply

```
<createNewService>
   <result value=" ">
      <description/>
   <result>
</createNewService>
```

Elements

dataBasePath

Specifies a path string that identifies the directory in which to store the VLDB.

autoLoadVLDB

Specifies whether to configure autoexec.ncf and nssvlbd.ncf to automatically load vldb.nlm when the server is loaded:

true (default)

false

tgtTree

(Optional)

user

Specifies the user. The leading dot is optional. Do not include the tree name.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Remarks

This command allows ConsoleOne to continue running without changes while allowing iManager to perform DFS commands without having to access eDirectory.

For backward compatibility, the dataBasePath element is available. However, if the new form is used, dataBasePath should not appear at the top level.

The new form uses the management context container, which is created (by adding the DFS attributes to the specified container, and this container must exist) if it doesn't previously exist.

Example

A nssRequest packet to create a new VLDB service is as follows:

```
<nssRequest>
   <vldb>
      <createNewService>
            <dataBasePath>sys:\etc</dataBasePath>
            <autoLoadVLDB/>
         <managementContext>
            <ndsObject>nss.prv.novell</ndsObject>
            <tgtTree>novell_inc</tgtTree>
         </managementContext>
         <dbInfo>
            <backEndDB>vdqad</backEndDB>
            <dataBasePath>sys:\etc</dataBasePath>
         </dbInfo>
         <user>.admin.novell</user>
         <password>junk</password>
         <unp>0123456789ABCDEF</unp>
      </createNewService>
   </vldb>
</nssRequest>
```

A nssReply packet to the create new VLDB service command follows:

```
<nssReply>
   <vldb>
      <createNewService>
         <result value="0">
            <description/>success</description>
         </result>
      </createNewService>
   </vldb>

   <result value="0">
     <description/>zOK</description>
   </result>
</nssReply>
```

deleteService

Removes the VLDB service from the server and updates eDirectory. If the last (or only) VLDB server is removed from a management context, the management context is automatically deleted.

Request

```
<deleteService>
   <user/>
   <password/>
   <unp/>
</deleteService>
```

Reply

```
<deleteService>
   <result value=" ">
      <description/>
   <result>
</deleteService>
```

Elements

user

Specifies the user. The leading dot is optional. Do not include the tree name.

unp

Specifies the protected credentials (encoded as a hex string). Either unp or the user and password must be supplied.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Remarks

This command allows ConsoleOne to continue running without changes while allowing iManager to perform DFS commands without having to access eDirectory.

For backward compatibility, the dataBasePath element is available. However, if the new form is used, dataBasePath should not appear at the top level.

The new form uses the management context container, which is created (by adding the DFS attributes to the specified container, and this container must exist) if it doesn't previously exist.

Example

A nssRequest packet to create a new VLDB service is as follows:

```
<nssRequest>
   <vldb>
      <deleteService>
         <user>.admin.novell</user>
         <password>junk</password>
         <unp>0123456789ABCDEF</unp>
      </deleteService>
   </vldb>
</nssRequest>
```

A nssReply packet to the create new VLDB service command follows:

```
<nssReply>
   <vldb>
      <deleteService>
         <result value="0">
            <description/>success</description>
         </result>
      </deleteService>
   </vldb>

   <result value="0">
     <description/>zOK</description>
   </result>
</nssReply>
```

getVLDBInfo

Returns detailed information about a VLDB.

Request

```
<getVLDBInfo type=" "/>
```

Reply

```
<getVLDBInfo>
   <basicInfo>
      <vldbState/>
      <version>
         <majorVersion/>
         <minorVersion/>
         <releaseNumber/>
         <buildNumber/>
      </version>
      <backEndVersion>
         <majorVersion/>
         <minorVersion/>
         <releaseNumber/>
         <buildNumber/>
      </backEndVersion>
      <vldbBuildDate value=" "/>
      <vldbLoadTime value=" "/>
      <numProcessThreads/>
      <numRunningThreads/>
      <backEndDatabasePath/>
      <autoLoadVLDB/>
   </basicInfo>
   <statisticsInfo>
      <vldbCreateCount/>
      <vldbDeleteCount/>
      <vldbModifyCount/>
      <vldbLookupCount/>
      <vldbTotalRequests/>
      <vldbErrorCount/>
      <vldbAuthErrorCount/>
   </statisticsInfo>
   <repairInfo>
      <repairState/>
      <repairPercentComplete/>
      <repairLevel/>
      <repairStartTime value = " "/>
      <repairEndTime value=" "/>
      <repairStatus/>
      <repairCompletionCode/>
   </repairInfo>
   <result value=" ">
      <description/>
   </result>
</getVLDBInfo>
```

Elements

basicInfo

Specifies the basic information if the type specifies all or basic.

vldbState

Specifies the state of the VLDB:

initializing
stopped
running
broken
underRepair
unknown

version

Specifies the version of the VLDB.

majorVersion

Specifies the major version number.

minorVersion

Specifies the minor version number.

buildNumber

Specifies the build number.

backEndVersion

Specifies the version of the VLDB backend.

releaseNumber

Specifies the release number.

vldbBuildDate

Specifies the date that the vldb.nlm file was built.

vldbLoadTime

Specifies the time that the vldb.nlm file was loaded.

numProcessThreads

Specifies the number of threads the VLDB requested to use.

numRunningThreads

Specifies the number of threads that the VLDB is currently using.

backEndDatabasePath

Specifies the path where the backend database is located.

autoLoadVLDB

Specifies that the VLDB is set up to be auto loaded when the server is started.

statisticsInfo

Specifies the statistical information if the type specifies all or statistics.

vldbCreateCount

Specifies the number of volumes that were added to the VLDB.

vldbDeleteCount

Specifies the number of volumes that were deleted from the VLDB.

vldbModifyCount

Specifies the number of volumes that were modified in the VLDB.

vldbLookupCount

Specifies the number of volumes that were looked up in the VLDB.

vldbTotalRequests

Specifies the total number of requests made to the VLDB.

vldbErrorCount

Specifies the number of errors in VLDB requests.

vldbAuthErrorCount

Specifies the number of authorization errors.

repairInfo

Specifies the repair information if the type specifies all or repair.

repairState

Specifies the state of repairs:

notRepairing
repairing
complete
failed
unknown

repairPercentComplete

Specifies how complete (as a percentage) the current repair is.

repairLevel

Specifies the level of the current repair:

none
lowLevel
refresh
rebuild
unknown

repairStartTime

Specifies the time the current or last repair started.

repairEndTime

Specifies the time the current or last repair ended.

repairStatus

Specifies the status of the current repair:

success
failed
aborted
unknown

repairCompletionCode

Specifies the error code that was returned by the last repair operation.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

type

Specifies the properties of the VLDB to return:

all
basic
statistics
repair

If all is specified, all the information for the VLDB is returned. Otherwise, only the specified type of requested information is returned.

value

Specifies the decimal value of the date. The high 16 bits is the year, followed by 8 bits for the month. The last 8 bits is the day.

or

Specifies the UTC decimal time.

Example

A nssRequest packet to returns VLDB information is as follows:

```
<nssRequest>
   <vldb>
      <getVLDBInfo type="all">
      </getVLDBInfo>
   </vldb>
</nssRequest>
```

Depending on the management context, the following are examples of reply packets that can be returned:

- “Usual Reply” on page 258
- “No Management Context” on page 259
- “Management Context: VLDB Server Running” on page 260
- “Management Context: VLDB Server Not Running” on page 261
- “Management Context: Not a VLDB Server” on page 261

Usual Reply

A nssReply packet to the get VLDB information command follows:

```
<nssReply>
   <vldb>
      <getVLDBInfo>
         <basicInfo>
            <vldbState>running</vldbState>
            <version>
               <majorVersion>3</majorVersion>
               <minorVersion>1</minorVersion>
               <releaseNumber>0</releaseNumber>
               <buildNumber>37</buildNumber>
            </version>
            <backEndVersion>
               <majorVersion>3</majorVersion>
               <minorVersion>1</minorVersion>
               <releaseNumber>0</releaseNumber>
               <buildNumber>33</buildNumber>
            </backEndVersion>
            <vldbBuildDate value="131203344">
               2002-1-16
            </vldbBuildDate>
            <vldbLoadTime value="1015620810">
               Mar 8, 2002   1:53:30 pm
            </vldbLoadTime>
            <numProcessThreads>1</numProcessThreads>
            <numRunningThreads>1</numRunningThreads>
            <backEndDatabasePath>
               sys:\etc</backEndDatabasePath>
            <autoLoadVLDB>true</autoLoadVLDB>
         </basicInfo>

         <statisticsInfo>
            <vldbCreateCount>4</vldbCreateCount>
            <vldbDeleteCount>0</vldbDeleteCount>
            <vldbModifyCount>0</vldbModifyCount>
            <vldbLookupCount>0</vldbLookupCount>
            <vldbTotalRequests>10</vldbTotalRequests>
            <vldbErrorCount>0</vldbErrorCount>
            <vldbAuthErrorCount>0</vldbAuthErrorCount>
         </statisticsInfo>

         <repairInfo>
            <repairState>notRepairing</repairState>
             <repairPercentComplete>0</repairPercentComplete>
            <repairLevel>rebuild</repairLevel>
            <repairStartTime value="1015620814">
               Mar 8, 2002   1:53:34 pm
            </repairStartTime>
            <repairEndTime value="1015620815">
               Mar 8, 2002   1:53:35 pm
            </repairEndTime>
            <repairStatus>success</repairStatus>
            <repairCompletionCode>0</repairCompletionCode>
         </repairInfo>

         <result value="0">
            <description/>success</description>
         </result>
      </getVLDBInfo>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```


No Management Context

If there is no management context defined, the following response returns:

```
<nssReply>
   <vldb>
      <getVLDBInfo>
         <result value="0">
            <description/>success</description>
         </result>
         <basicInfo>
         <statisticsInfo>
         <repairInfo>
      </getVLDBInfo>
   </vldb>
</nssReply>
```

Management Context: VLDB Server Running

If a management context is defined for a VLDB server, the following response returns:

Note that for NetWare 6.5 SP2, replica server names are relative to the management context. For NetWare 6.5 SP1, replica server names are fully distinguished eDirectory names.

```
<nssReply>
   <vldb>
      <getVLDBInfo>
         <result value="0">
            <description/>success</description>
         </result>
         <basicInfo>
            <managementContext>
               <ndsObject>nss.prv.novell</ndsObject>
               <tgtTree>novell_inc</tgtTree>
            </managementContext>
            <backEndDB>vdqad</backEndDB>
            <serverName>vldb-master</serverName>
            <serverName>vldb-other.nss.prv</serverName>
            <vldbState>running</vldbState>
            <version>
               <majorVersion>3</majorVersion>
               <minorVersion>20</minorVersion>
               <releaseNumber>0</releaseNumber>
               <buildNumber>157</buildNumber>
            </version>
            <backEndVersion>
               <majorVersion>3</majorVersion>
               <minorVersion>20</minorVersion>
               <releaseNumber>0</releaseNumber>
               <buildNumber>152</buildNumber>
            </backEndVersion>
            <vldbBuildDate value="131203344">2002-1-16</vldbBuildDate>
            <vldbLoadTime value="1015620810">Mar 8, 2002  1:53:30 pm
               </vldbLoadTime>
            <numProcessThreads>1</numProcessThreads>
            <numRunningThreads>1</numRunningThreads>
            <backEndDatabasePath>sys:\etc</backEndDatabasePath>
            <autoLoadVLDB>true</autoLoadVLDB>
         </basicInfo>
         <statisticsInfo>
            <vldbCreateCount>4</vldbCreateCount>
            <vldbDeleteCount>0</vldbDeleteCount>
            <vldbModifyCount>0</vldbModifyCount>
            <vldbLookupCount>0</vldbLookupCount>
            <vldbTotalRequests>10</vldbTotalRequests>
            <vldbErrorCount>0</vldbErrorCount>
            <vldbAuthErrorCount>0</vldbAuthErrorCount>
         </statisticsInfo>
```

```
         <repairInfo>
            <repairState>notRepairing</repairState>
            <repairPercentComplete>0</repairPercentComplete>
            <repairLevel>rebuild</repairLevel>
            <repairStartTime value="1015620814>Mar 8, 2002 1:53:34 pm
               </repairStartTime>
            <repairEndTime value="1015620815">Mar 8, 2002 2:53:35 pm
               </repairEndTime>
            <repairStatus>success</repairStatus>
            <repairCompletionCode>0</repairCompletionCode>
         </repairInfo>
      </getVLDBInfo>
   </vldb>
</nssReply>
```

Management Context: VLDB Server Not Running

If a management context is defined for a VLDB server but the server isn't running, the following response returns:

```
<nssReply>
   <vldb>
      <getVLDBInfo>
         <result value="0">
            <description/>success</description>
         </result>
         <basicInfo>
            <managementContext>
               <ndsObject>nss.prv.novell</ndsObject>
               <tgtTree>novell_inc</tgtTree>
            </managementContext>
            <backEndDB>vdqad</backEndDB>
            <serverName>vldb-master</serverName>
            <serverName>vldb-other.nss.prv</serverName>
            <vldbState>notLoaded</vldbState>
         </basicInfo>
         <statisticsInfo/>
         <repairInfo/>
      </getVLDBInfo>
   </vldb>
</nssReply>
```

Management Context: Not a VLDB Server

If a management context is defined and the server isn't a VLDB server, the following response returns:

```
<nssReply>
   <vldb>
      <getVLDBInfo>
         <result value="0">
            <description/>success</description>
         </result>
         <basicInfo>
            <managementContext>
               <ndsObject>nss.prv.novell</ndsObject>
               <tgtTree>novell_inc</tgtTree>
            </managementContext>
            <backEndDB>vdqad</backEndDB>
            <serverName>vldb-master</serverName>
            <serverName>vldb-other.nss.prv</serverName>
            <vldbState>notVLDBServer</vldbState>
         </basicInfo>
         <statisticsInfo/>
         <repairInfo/>
      </getVLDBInfo>
   </vldb>
</nssReply>
```

loadVLDB

Loads and starts up the VLDB software on a server where the VLDB resides.

Request

```
<loadVLDB/>
```

Reply

```
<loadVLDB>
   <result value=" ">
      <description/>
   <result>
</loadVLDB>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to load VLDB software is as follows:

```
<nssRequest>
   <vldb>
      <loadVLDB>
      </loadVLDB>
   </vldb>
</nssRequest>
```

A nssReply packet to the load VLDB command follows:

```
<nssReply>
   <vldb>
      <loadVLDB>
         <result value="0">
            <description/>success</description>
         </result>
      </loadVLDB>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

lookup

Returns the physical volume mapping for a DFS GUID as returned by a read link request. As an administrator, don't worry about what GUID a junction references; but note what volume on which server the junction is pointing to.

Request

```
<lookup>
   <dfsGUID/>
</lookup>
```

Reply

```
<lookup>
   <result value=" ">
      <description/>
   </result>
   <dfsGUID/>
   <volumeInfo>
      <server/>
      <tgtTree/>
      <volumeName/>
      <type/>
   </volumeInfo>
</lookup>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

volumeInfo

Repeats for each instance of the volume.

Example

A nssRequest packet to load VLDB software is as follows:

```
<nssRequest>
   <vldb>
      <lookup>
         <dfsGUID>C2EAAA00-3211-11D6-B7-C7-00C04FA33547</dfsGUID>
      </lookup>
   </vldb>
</nssRequest>
```

A nssReply packet to the load VLDB command follows:

```
<nssReply>
   <vldb>
      <lookup>
         <result value="0">
            <description/>success</description>
         </result>
         <dfsGUID>C2EAAA00-3211-11D6-B7-C7-00C04FA33547</dfsGUID>
         <volumeInfo>
            <server>prv-psst.nss.prv.novell</server>
            <tgtTree>novell_inc</tgtTree>
            <volumeName>VOL1</volumeName>
            <type>RW</type>
         </volumeInfo>
      </lookup>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

removeVolumeFromVLDB

Removes a volume from a VLDB.

Request

```
<removeVolumeFromVLDB>
   <dfsGUID/>
</removeVolumeFromVLDB>
```

Reply

```
<removeVolumeFromVLDB>
   <result value=" ">
      <description/>
   <result>
</removeVolumeFromVLDB>
```

Elements

dfsGUID

Specifies the DFS GUID of the volume to remove.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to remove a volume from a VLDB is as follows:

```
<nssRequest>
   <vldb>
      <removeVolumeFromVLDB>
         <dfsGUID>
            906D22B6-32DD-01D6-80-06-FBDA22AE6917
         </dfsGUID>
      </removeVolumeFromVLDB>
   </vldb>
</nssRequest>
```

A nssReply packet to the remove volume from VLDB command follows:

```
<nssReply>
   <vldb>
      <removeVolumeFromVLDB>
         <result value="0">
            <description/>success</description>
         </result>
      </removeVolumeFromVLDB>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

replicaAddedToVLDB

Informs a VLDB server that another VLDB replica server has been added.

Request

```
<replicaAddedToVLDB>
   <ndsServerName/>
</replicaAddedToVLDB>
```

Reply

```
<replicaAddedToVLDB>
   <result value=" ">
      <description/>
   <result>
</replicaAddedToVLDB>
```

Elements

ndsServerName

Specifies the eDirectory name, in relative distinguished-name format, of the server to which a replica was added.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following example tells the server that another VLDB replica was just added to MYSERVER:

```
<nssRequest>
   <vldb>
      <replicaAddedToVLDB>
         <ndsServerName>
            .MYSERVER.novell.MY_TREE.
         </ndsServerName>
      </replicaAddedToVLDB>
   </vldb>
</nssRequest>
```

A nssReply packet to the replica added to VLDB command follows:

```
<nssReply>
   <vldb>
      <replicaAddedToVLDB>
         <result value="0">
            <description/>success</description>
         </result>
      </replicaAddedToVLDB>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

replicaRemovedFromVLDB

Informs a VLDB server that another VLDB replica server has been removed.

Request

```
<replicaRemovedFromVLDB>
   <ndsServerName/>
</replicaRemovedFromVLDB>
```

Reply

```
<replicaRemovedFromVLDB>
   <result value=" ">
      <description/>
   <result>
</replicaRemovedFromVLDB>
```

Elements

ndsServerName

Specifies the eDirectory name, in relative distinguished-name format, of the server from which a replica was removed.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following example informs a server that the VLDB replica on MYSERVER has been removed:

```
<nssRequest>
   <vldb>
      <replicaRemovedFromVLDB>
         <ndsServerName>
            .MYSERVER.novell.MY_TREE.
         </ndsServerName>
      </replicaRemovedFromVLDB>
   </vldb>
</nssRequest>
```

A nssReply packet to the modify device command follows:

```
<nssReply>
   <vldb>
      <replicaRemovedFromVLDB>
         <result value="0">
            <description/>success</description>
         </result>
      </replicaRemovedFromVLDB>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
<nssReply>
```

setVLDBConfiguration

Sets the configuration parameters for a VLDB.

Request

```
<setVLDBConfiguration>
   <vldbNumThreads/>
   <dataBasePath/>
   <autoLoadVLDB/>
</setVLDBConfiguration/>
```

Reply

```
<setVLDBConfiguration>
   <result value=" ">
      <description/>
   <result>
</setVLDBConfiguration>
```

Elements

vldbNumThreads

(Optional) Specifies that the number of threads the VLDB should use is modified.

dataBasePath

(Optional) Specifies the new directory to move the VLDB database to.

autoLoadVLDB

(Optional) Specifies whether the state of autoloading is modified:

true (default)
false

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following example sets the number of threads to two and turns on the auto-load option:

```
<nssRequest>
   <vldb>
      <setVLDBConfiguration>
         <numThreads>2</numThreads>
         <autoLoadVLDB>true</autoLoadVLDB>
      </setVLDBConfiguration>
   </vldb>
</nssRequest>
```

A nssReply packet to the set VLDB configuration command follows:

```
<nssReply>
   <vldb>
      <setVLDBConfiguration>
         <result value="0">
            <description/>success</description>
         </result>
      </setVLDBConfiguration>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

shutdownVLDB

Shuts down and unloads the VLDB software on a server where the VLDB resides.

Request

```
<shutdownVLDB/>
```

Reply

```
<shutdownVLDB>
   <result value=" ">
      <description/>
   <result>
</shutdownVLDB>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to shutdown the VLDB software is as follows:

```
<nssRequest>
   <vldb>
      <shutdownVLDB>
      </shutdownVLDB>
   </vldb>
</nssRequest>
```

A nssReply packet to the shutdown VLDB command follows:

```
<nssReply>
   <vldb>
      <shutdownVLDB>
         <result value="0">
            <description/>success</description>
         </result>
      </shutdownVLDB>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

startRepair

Starts a repair command on the VLDB.

Request

```
<startRepair>
   <vldbRepairLevel/>
   <user/>
   <password/>
   <unp/>
</startRepair>
```

Reply

```
<startRepair>
   <result value=" ">
      <description/>
   <result>
</startRepair>
```

Elements

vldbRepairLevel

Specifies the type of repair to perform:

lowLevel
refresh
rebuild

user

Specifies the user. The leading dot is optional. Do not include the tree name.

unp

Specifies the protected credentials, encoded as a hex string. Either unp or both user and password can be supplied. If neither is specified, the repair runs logged in as the file server (for ConsoleOne compatibility).

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following example does a "rebuild" type of repair on the VLDB:

```
<nssRequest>
   <vldb>
      <startRepair>
         <repairLevel>rebuild</repairLevel>
         <user>.admin.novell</user>
         <password>junk</password>
         <unp>0123456789ABCDEF</unp>
      </startRepair>
   </vldb>
</nssRequest>
```

A nssReply packet to the start VLDB repair command follows:

```
<nssReply>
   <vldb>
      <startRepair>
         <result value="0">
            <description/>success</description>
         </result>
      </startRepair>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

startService

Starts the VLDB service on a server.

Request

```
<startService/>
```

Reply

```
<startService>
   <result value=" ">
      <description/>
   <result>
</startService>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to start the VLDB service is as follows:

```
<nssRequest>
   <vldb>
      <startService>
      </startService>
   </vldb>
</nssRequest>
```

A nssReply packet to the start VLDB service command follows:

```
<nssReply>
   <vldb>
      <startService>
         <result value="0">
            <description/>success</description>
         </result>
      </startService>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

stopRepair

Stops a repair command on the VLDB.

Request

```
<stopRepair/>
```

Reply

```
<stopRepair>
   <result value=" ">
      <description/>
   <result>
</stopRepair>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to stop a VLDB repair is as follows:

```
<nssRequest>
   <vldb>
      <stopRepair>
         <repairLevel>rebuild</repairLevel>
      </stopRepair>
   </vldb>
</nssRequest>
```

A nssReply packet to the stop VLDB repair command follows:

```
<nssReply>
   <vldb>
      <stopRepair>
         <result value="0">
            <description/>success</description>
         </result>
      </stopRepair>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

stopService

Stops the VLDB service on a server.

Request

```
<stopService/>
```

Reply

```
<stopService>
   <result value=" ">
      <description/>
   <result>
</stopService>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to stop the VLDB service is as follows:

```
<nssRequest>
   <vldb>
      <stopService>
      </stopService>
   </vldb>
</nssRequest>
```

A nssReply packet to the stop VLDB service command follows:

```
<nssReply>
   <vldb>
      <stopService>
         <result value="0">
            <description/>success</description>
         </result>
      </stopService>
   </vldb>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

3.17 Volume Operations

This section contains the following Volume commands:

- “addTraditionalVolume” on page 278
- “addVolume” on page 281
- “expandTraditionalVolume” on page 284
- “getNDSName (Volume)” on page 286
- “getState” on page 287
- “getTraditionalVolumeInfo” on page 289
- “getVolumeInfo” on page 292
- “listVolumes” on page 300
- “listEvmsVolumes” on page 301
- “modifyState” on page 302
- “modifyVolumeInfo” on page 304
- “removeUser” on page 308
- “removeVolume” on page 310
- “renameVolume” on page 312

For more information on volumes, see “Volumes” on page 20.

Each command is wrapped with either the nssRequest or nssReply element and the volume element.

addTraditionalVolume

Creates a traditional NetWare volume on the server.

Request

```
<addTraditionalVolume state=" ">
   <volumeName/>
   <ndsName/>
   <context/>
   <noNDSObject/>
   <noDFSGUID/>
   <dfsGUID/>
   <blockSize/>
   <compression/>
   <suballocation/>
   <migration/>
   <volSegments>
      <segment>
         <partitionID/>
         <volStartingSector/>
         <volNumSectors/>
      </segment>
   </volSegments>
   <updateVLDB>
</addTraditionalVolume>
```

Reply

```
<addTraditionalVolume>
   <result value=" ">
      <description/>
   </result>
</addTraditionalVolume>
```

Elements

volumeName

(Required) Specifies the name to give the volume being created.

ndsName

(Required unless noNDSObject is specified) Specifies the name of the eDirectory volume object to create that represents the created volume. If NULL is specified, the name of the eDirectory volume object is generated by prepending the server name and an underscore to the volumeName element.

context

(Required unless noNDSObject is specified) Specifies the eDirectory context in which the eDirectory volume object is created. If NULL is specified, the eDirectory volume object is created in the same eDirectory context where the server object resides.

noNDSObject

(Optional) Specifies that no eDirectory objects should be created for the volume. The ndsName and context elements are ignored.

noDFSGUID

(Optional) Specifies that no DFS GUID is assigned to the volume when it is created. If specified, dfsGUID and updateVLDB might not be used.

dfsGUID

(Optional) Specifies the value of the DFS GUID that is used to identify the volume in distributed file system (DFS) operations. If not specified, the file system generates a new DFS GUID for each volume being created.

blockSize

(Required) Specifies the block size (in bytes) of the traditional volume:

4096
8192
16384
32768
65536

compression

Specifies that the traditional volume is created with the compression feature enabled.

suballocation

Specifies that the traditional volume is created with the suballocation feature enabled.

migration

Specifies that the traditional volume is created with the data migration feature enabled.

volSegments

Repeats for each volume segment on the traditional volume. A volume segment can be either a whole partition or a portion of a partition.

segment

Repeats for each volume segment that needs to be included in the traditional volume.

partitionID

Specifies the logical ID of a partition that's being added to the new traditional volume. If the partition has HotFix and mirroring, it is the mirror group ID for the partition. If the partition does not have HotFix and mirroring, it is the ID of the raw physical partition.

volStartingSector

Specifies the starting sector offset within the partition of the segment to include.

volNumSectors

Specifies the number of sectors to include.

updateVLDB

Specifies that the DFS Volume Location Database (VLDB) is updated by the XML processing code. This element is added for backward-compatibility with ConsoleOne, which does not know about this element but does its own VLDB updating. All new code should include this element.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

state

(Optional) Specifies what state the volume should be set to after it is created:

mounted

dismounted

If a state is not specified, the volume defaults to whatever state the file system policies dictate.

Example

The following is an example of adding a traditional volume called "VOL1" on partition 28. It adds an eDirectory volume object using the default name of "servername_VOL1" and puts the eDirectory volume object in the same eDirectory context as the server itself.

```
<nssRequest>
   <volume>
      <addTraditionalVolume state="mounted">
         <volumeName>VOL1</volumeName>
         <ndsName>
         <context>
         <blockSize>65536</blockSize>
         <suballocation>
         <volSegments>
            <segment>
               <partitionID>28</partitionID>
               <volStartingSector>0</volStartingSector>
               <volNumSectors>202752</volNumSectors>
            </segment>
         </volSegments>
      </addTraditionalVolume>
   </volume>
</nssRequest>
```

A nssReply packet to the add traditional volume command follows:

```
<nssReply>
   <volume>
      <addTraditionalVolume>
         <result value="0">
            <description/>success</description>
         </result>
      </addTraditionalVolume>
   </volume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

addVolume

Creates an NSS logical volume on an existing NSS storage pool. By default, addVolume also creates the corresponding eDirectory volume object (unless instructed not to). To create a traditional NetWare volume, use the addTraditionalVolume (page 278) command.

Request

```
<addVolume state=" ">
   <volumeName/>
   <poolName/>
   <volumePassword/>
   <ndsName/>
   <context/>
   <ndsPoolName/>
   <noNDSOjbect/>
   <noDFSGUID/>
   <updateVLDB/>
   <volumeGUID/>
   <dfsGUID/>
</addVolume>
```

Reply

```
<addVolume>
   <result value=" ">
      <description/>
   </result>
</addVolume>
```

Elements

volumeName

(Required) Specifies the name of the volume to create.

poolName

(Required) Specifies the name of the NSS storage pool on which the volume is created.

volumePassword

Specifies to create an ecrypted volume.

ndsName

(Required unless noNDSObject is specified) Specifies the name of the eDirectory volume object that is created to represent the volume. If NULL is specified, the name of the eDirectory volume object is generated by prepending the server name and an underscore to the value of volumeName.

context

(Required unless noNDSObject is specified) Specifies the eDirectory context in which the eDirectory volume object is created. If NULL is specified, the eDirectory volume object is created in the same eDirectory context where the server object resides.

ndsPoolName

(Required unless noNDSObject is specified) Specifies the value to use as the nssfsPool attribute for the eDirectory volume object. If NULL is specified, the pool's eDirectory name is retrieved and is used for the nssfsPool attribute.

noNDSObject

(Optional) Specifies that no eDirectory objects should be created for the volume. If this element is specified, the ndsName, context, and ndsPoolName elements are ignored.

noDFSGUID

(Optional) Specifies that no DFS GUID is assigned to the volume when it is created. If specified, the dfsGUID and updateVLDB elements might not be included.

updateVLDB

Specifies that the DFS Volume Location Database (VLDB) is updated by the XML processing code. This element was added for backward-compatibility with ConsoleOne, which does not know about this element but does its own VLDB updating. All new code should include this element.

volumeGUID

(Optional) Specifies a globally unique ID (GUID) to be assigned to the volume that is being created. Usually, the file system assigns a new GUID to each volume that is being created.

dfsGUID

(Optional) Specifies the value of the GUID that is used to identify the volume in DFS operations. If not specified, the file system generates a new DFS GUID for each volume that is being created.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

state

(Optional) Specifies what state the volume shoudl be set to after it is created:

deactive
active
mounted

If the state is not specified, the pool defaults to whatever the file system policies dictate.

Remarks

An encrypted volume uses the volumePassword field.

Example

The following is an example of the addVolume command that creates a volume called "MYVOL" on the "MYPOOL" pool. It adds an eDirectory volume object using the default name of "servername_MYVOL" and puts the eDirectory volume object in the same eDirectory context as the server itself. This example uses the actual eDirectory pool object name for the nssfsPool attribute of the volume object.

```
<nssRequest>
   <volume>
      <addVolume state="mounted">
         <volumeName>MYVOL</volumeName>
         <poolName>MYPOOL</poolName>
         <ndsName>
         <context>
         <ndsPoolName>
      </addVolume>
   </volume><
/nssRequest>
```

A nssReply packet to the add volume command follows:

```
<nssReply>
   <volume>
      <addVolume>
         <result value="0">
            <description/>success</description>
         </result>
      </addVolume>
   </volume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

expandTraditionalVolume

Expands the size of a traditional NetWare volume on the server. To create an NSS logical volume, call the addVolume (page 281) command.

Request

```
<expandTraditionalVolume>
   <volumeName/>
   <volSegments>
      <segment>
         <partitionID/>
         <volStartingSector/>
         <volNumSectors/>
      </segment>
   </volSegment>
</expandTraditionalVolume>
```

Reply

```
<expandTraditionalVolume>
   <result value=" ">
      <description/>
   </result>
</expandTraditionalVolume>
```

Elements

volumeName

(Required) Specifies the name of the traditional volume to expand.

volSegments

Specifies a list of volume segments to add to the traditional volume.

segment

Repeats for every volume segment that needs to add.

partitionID

Specifies the logical ID of the partition to add to the existing traditional volume. If the partition has HotFix and mirroring, it is the mirror group ID for the partition. If the partition does not have HotFix and mirroring, it is the ID of the raw physical partition.

volStartingSector

Specifies the starting sector offset within the partition of the segment to include.

volNumSectors

Specifies the number of sectors to include.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to expand a traditional volume is as follows:

```
<nssRequest>
   <volume>
      <expandTraditionalVolume>
         <volumeName>VOL1</volumeName>
         <volSegments>
            <segment>
               <partitionID>31</partitionID>
               <volStartingSector>0</volStartingSector>
               <volNumSectors>100352</volNumSectors>
            </segment>
         </volSegments>
      </expandTraditionalVolume>
   </volume>
</nssRequest>
```

A nssReply packet to the expand a traditional volume command follows:

```
<nssReply>
   <volume>
      <expandTraditionalVolume>
         <result value="0">
            <description/>success</description>
         </result>
      </expandTraditionalVolume>
   </volume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

getNDSName (Volume)

Returns the eDirectory name for an existing volume.

Request

```
<getNDSName>
   <volumeName/>
</getNDSName>
```

Reply

```
<getNDSName>
    <ndsName/>
    <context/>
</getNDSName>
```

Elements

volumeName

Specifies the name of the volume for which to find the eDirectory name.

ndsName

Specifies the name of the eDirectory volume object.

context

Specifies the eDirectory context of the returned ndsName.

getState

Returns the state of either an NSS logical volume or a traditional NetWare volume.

Request

```
<getState>
   <volumeName/>
</getState>
```

Reply

```
<getState>
   <volumeName/>
   <volumeState/>
   <result value=" ">
      <description/>
   </result>
</getState>
```

Elements

volumeName

Specifies the name of the volume for which to return the state.

volumeState

Specifies the state of the volume (see “Volume States” on page 461).

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to get a volume's state is as follows:

```
<nssRequest>
   <volume>
      <getState>
         <volumeName>NSS1</volumeName>
      </getState>
   </volume>
</nssRequest>
```

A nssReply packet to the get state command follows:

```
<nssReply>
   <pool>
      <getState>
         <volumeName>SYS</volumeName>
         <volumeState>active</volumeState>
         <result value="0">
            <description/>success</description>
         </result>
      </getState>
   </pool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

getTraditionalVolumeInfo

Returns detailed information about an existing traditional NetWare volume. To get information about an NSS logical volume, call the getVolumeInfo (page 292) command.

Request

```
<getTraditionalVolumeInfo>
   <volumeName/>
</getTraditionalVolumeInfo>
```

Reply

```
<getTraditionalVolumeInfo>
   <volumeName/>
   <volumeState/>
   <nameSpaces value=" "/>
   <compression/>
   <suballocation/>
   <migration/>
   <blockSize/>
   <numBlocks/>
   <freeBytes/>
   <totalFiles/>
   <deletedFiles/>
   <compressedFiles/>
   <compressedDeletedFiles/>
   <createdTime value=" "/>
   <modifiedTime value=" "/>
   <archivedTime value=" "/>
   <result value = " ">
      <description/>
   </result>
</getTraditionalVolumeInfo>
```

Elements

volumeName

Specifies the name of the volume.

volumeState

Specifies the state of the volume:

mounted
dismounted

nameSpaces

Specifies a list of name space names, separated by spaces:

DOS
Long
Macintosh
Unix

compression

Specifies that compression is enabled.

suballocation

Specifies that suballocation is enabled.

migration

Specifies that migration is enabled.

blockSize

Specifies the volume's block size.

numBlocks

Specifies the number of blocks in the volume.

freeBytes

Specifies the number of bytes that are free.

totalFiles

Specifies the total number of file objects that are stored on the volume.

deletedFiles

Specifies the number of deleted files on the volume.

compressedFiles

Specifies the number of compressed files.

compressedDeletedFiles

Specifies the number of compressed deleted files.

createdTime

Specifies a string representation of the UTC time when the volume was created.

modifiedTime

Specifies a string representation of the UTC time when the volume was last modified.

archivedTime

Specifies a string representation of the UTC time when the volume was last archived.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

value

Specifies the decimal value of the name space mask for the volume.

or

Specifies the decimal UTC time.

Example

A nssRequest packet to return traditional volume information is as follows:

```
<nssRequest>
   <volume>
      <getTraditionalVolumeInfo>
         <volumeName>VOL1</volumeName>
      </getTraditionalVolumeInfo>
   </volume>
</nssRequest>
```

A nssReply packet to the get traditional volume information command follows:

```
<nssReply>
   <volume>
      <getTraditionalVolumeInfo>
         <volumeName>VOL1</volumeName>
         <volumeState>mounted</volumeState>
         <nameSpaces value="17">DOS Long </nameSpaces>
         <suballocation>
         <blockSize>65536</blockSize>
         <numBlocks>1582</numBlocks>
         <createdTime value="744977753">
            07-Mar-2002 14:42:50
         </createdTime>
         <modifiedTime value="744977754">
            07-Mar-2002 14:42:52
         </modifiedTime>
         <archivedTime value="0">Invalid DOS Time
         </archivedTime>
         <result value="0">
            <description/>success</description>
         </result>
      </getTraditionalVolumeInfo>
   </volume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

getVolumeInfo

Returns detailed information about an existing NSS logical volume. To get information about a traditional NetWare volume, use the getTraditionalVolumeInfo (page 289) command.

Request

```
<getVolumeInfo type=" ">
   <volumeName/>
</getVolumeInfo>
```

Reply

```
<getVolumeInfo>
   <basicInfo>
      <mountPoint/>
      <volumeName/>
      <poolName/>
      <ndsVolumeName/>
      <ndsVolumeGUID/>
      <volumeGUID/>
      <owner/>
      <volumeState/>
      <nameSpaces value=" "/>
      <blockSize/>
      <volumeQuota/>
      <usedSize/>
      <totalObjects/>
      <totalFiles/>
      <createdTime value=" "/>
      <modifiedTime value=" "/>
      <archivedTime value=" "/>
      <volumeReadAhead/>
   </basicInfo>
   <attributeInfo>
      <supportedAttributes value=" ">
         <volumeEncrypted/>
         <readonly/>
         <salvage/>
         <compression/>
         <directoryQuota/>
         <userQuota/>
         <flushFiles/>
         <mfl/>
         <snapshot/>
         <backup/>
         <shredding/>
         <userTransaction/>
         <migration/>
      </supportedAttributes>
      <enabledAttributes value=" ">
         <readonly/>
         <salvage/>
         <compression/>
         <directoryQuota/>
         <userQuota/>
         <flushFiles/>
         <mfl/>
         <snapshot/>
         <backup/>
         <shredding/>
         <userTransaction/>
         <migration/>
      </enabledAttributes>
   </attributeInfo>
```

```
   <salvageInfo>
      <freeableSize/>
      <nonFreeableSize/>
      <deletedFiles/>
      <oldestDeletedTime value=" "/>
      <minKeepTime/>
      <maxKeepTime/>
      <lowWaterMark/>
      <highWaterMark/>
   </salvageInfo>
   <compressionInfo>
      <compressedFiles/>
      <compressedDeletedFiles/>
      <uncompressibleFiles/>
      <precompressionBytes/>
      <compressedBytes/>
   </compressionInfo>
   <deletedVolumeInfo>
      <deleteState value=" ">
      <originalVolumeName/>
      <originalVolumeGUID/>
      <deletedTime value=" "/>
      <scheduledPurgeTime value=" "/>
      <lastStatus/>
      <lastStatusSetter/>
   </deletedVolumeInfo>
   <result value=" ">
      <description/>
   </result>
</getVolumeInfo>
```

Elements

volumeName

Specifies the name of the volume to return information for.

basicInfo

Specifies that type is all or basic.

mountPoint

Linux only. Specifies the pool's mount point.

poolName

Specifies the name of the NSS storage pool on which the volume resides.

ndsVolumeName

Specifies the fully distinguished name of the eDirectory volume object.

ndsVolumeGUID

Specifies the GUID that identifies the eDirectory volume object.

volumeGUID

Specifies the GUID that identifies the NSS volume.

owner

Specifies the user's name of the volume's owner.

volumeState

Specifies the current state of the volume (see “Volume States” on page 461).

nameSpaces

Specifies a list of name spaces. The lookup value can be the following:

dos
long
mac
unix

blockSize

Specifies the size of the volume's block.

volumeQuota

Specifies the size quota assigned to the NSS logical volume. If a number, the volume is restricted not to grow larger than the specified size. If none is returned, the volume has no assigned quota and can grow as big as the physical size of the NSS storage pool allows.

usedSize

Specifies the total number of used bytes on the volume.

totalObjects

Specifies the total number of objects that are stored on the volume.

totalFiles

Specifies the total number of file objects that are stored on the volume.

createdTime

Specifies a string representation of the UTC time when the volume was created.

modifiedTime

Specifies a string representation of the UTC time when the volume was last modified.

archivedTime

Specifies a string representation of the UTC time when the volume was archived.

volumeReadAhead

Specifies the current read ahead setting for the volume.

attributeInfo

Specifies that type is all or attributes.

supportedAttributes

Specifies a combination of the attribute elements that are supported by the volume.

volumeEncrypted

Specifies if the volume is encrypted (zATTR_ENCRYPTED).

readonly

Specifies the read only feature is supported or enabled.

salvage

Specifies the salvage feature is supported or enabled.

compression

Specifies the compression feature is supported or enabled.

directoryQuota

Specifies the directory quota feature is supported or enabled.

userQuota

Specifies the user quota feature is supported or enabled.

flushFiles

Specifies the flush files feature is supported or enabled.

mfl

Specifies the modified file list feature is supported or enabled.

snapshot

Specifies the snapshot feature is supported or enabled.

backup

Specifies the backup bit is supported or enabled.

shredding

Specifies the data shredding feature is supported or enabled.

userTransaction

Specifies the user transaction feature is supported or enabled.

migration

Specifies the migration feature is supported or enabled.

enabledAttributes

Specifies a list of attribute elements that are currently enabled on the volume.

salvageInfo

Specifies that the type is all or salvage.

freeableSize

Specifies the number of purgeable bytes on the volume.

nonFreeableSize

Specifies the number of nonpurgeable bytes on the volume.

deletedFiles

Specifies the number of deleted files on the volume.

oldestDeletedTime

Specifies a string representation of the UTC time.

minKeepTime

Specifies the number of seconds a file must remain in a salvageable state before the file system is allowed to automatically purge the file (if free space is needed). A value of 0 indicates that all deleted files are immediate candidates for automatic purging.

lowWaterMark

Specifies the low water mark percentage for the volume. If the amount of free space on the volume falls below this percentage, an automatic purge process is initiated.

highWaterMark

Specifies the high water mark percentage for the volum.e If the amount of free space on the volume rises above this percentage, an automatic purge process is initiated.

compressionInfo

Specifies that the type is all or compression.

compressedFiles

Specifies the number of compressed files.

compressedDeletedFiles

Specifies the number of compressed deleted files.

uncompressibleFiles

Specifies the number of uncompressible files.

precompressionBytes

Specifies the number of precompression bytes.

compressedBytes

Specifies the number of compressed bytes.

deletedVolumeInfo

Specifies that type is all or deletedVolume. This information is returned only if the volume is a volume that has been deleted but is not yet purged from the NSS storage pool.

deleteState

Specifies the current state of the volume.

originalVolumeName

Specifies the original name of the volume before it was deleted.

originalVolumeGUID

Specifies the original GUID of the volume before it was deleted.

deletedTime

Specifies a string representation of the UTC time.

scheduledPurgeTime

Specifies a string representation of the UTC time.

lastStatus

Specifies an error value or 0 (for no error) for the last reported error for the volume before it was deleted.

lastStatusSetter

Specifies the source (which is the NSS standard source file line number format) that reported the lastStatus error.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

Attributes

type

(Optional) Specifies what type of information to return. (See "Volume Types" on page 461.) The default value is all.

nameSpaces value

Specifies the decimal value of the name space mask for the volume.

value

(For createdtime, modifiedTime, archivedTime, oldestDeletedTime, deletedTime, and scheduledPurgeTime) Specifies the decimal UTC time.

value

(For supportedAttributes and enabledAttributes) Specifies the decimal value of the volume's bit masks.

value

(For deleteState) Specifies the decimal value of the delete state of the volume.

Remarks

An encrypted volume uses the volumeEncrypted element, which cannot be modified on an existing volume. If you try to modify the volumeEncrypted element, the volume is not changed.

deleteState can have the following values:

unknown
salvageable
purging
purging paused
auto purging paused
purge error
salvaging
salvage error
salvaged

First Example

A nssRequest packet to return the volume information is as follows:

```
<nssRequest>
   <volume>
      <getVolumeInfo type="all">
         <volumeName>MYVOL</volumeName>
      </getVolumeInfo>
   </volume>
</nssRequest>
```

A nssReply packet to the get volume information command follows:

```
<nssReply>
   <volume>
      <getVolumeInfo>
         <basicInfo>
            <volumeName>MYVOL</volumeName>
            <poolName>MYPOOL</poolName>
            <ndsVolumeName>
               .CN=MYSERVER_MYVOL.O=novell.T=MY_TREE.
            </ndsVolumeName>
            <ndsVolumeGUID>
               C2EAAA00-3211-11D6-B7-C7-00C04FA33547
            </ndsVolumeGUID>
            <volumeGUID>
               C2C86990-3211-01D6-80-00-BD09318C30CA
            </volumeGUID>
            <owner>.[Supervisor].</owner>
            <volumeState>mounted</volumeState>
            <nameSpaces value="23">
               DOS Long Macintosh Unix
            </nameSpaces>
            <blockSize>4096</blockSize>
            <volumeQuota>none</volumeQuota>
            <usedSize>36864</usedSize>
            <totalObjects>7</totalObjects>
            <totalFiles>7</totalFiles>
            <createdTime value="1015536291">
               Mar 7, 2002   2:24:51 pm
            </createdTime>
            <modifiedTime value="1015536291">
               Mar 7, 2002   2:24:51 pm
            </modifiedTime>
            <archivedTime value="0">Invalid UTC Time
            </archivedTime>
         </basicInfo>
         <attributeInfo>
            <supportedAttributes value="469762043">
               <salvage/>
               <compression>
               <directoryQuota>
               <userQuota>
               <flushFiles>
               <mfl>
               <snapshot>
               <shredding>
               <userTransaction>
               <migration>
            </supportedAttributes>
            <enabledAttributesvalue="50593779">
               <salvage/>
               <directoryQuota>
               <userQuota>
               <backup>
               <shredding>2</shredding>
            </enabledAttributes>
         </attributeInfo>
         <salvageInfo>
            <freeableSize>0</freeableSize>
            <nonFreeableSize>0</nonFreeableSize>
            <deletedFiles>0</deletedFiles>
            <oldestDeletedTime value="0">
               Invalid UTC Time
            </oldestDeletedTime>
            <minKeepTime>0</minKeepTime>
            <maxKeepTime>0</maxKeepTime>
            <lowWaterMark>10</lowWaterMark>
            <highWaterMark>20</highWaterMark>
         </salvageInfo>
         <compressionInfo>
            <compressedFiles>0</compressedFiles>
            <compressedDeletedFiles>0
```

```
               </compressedDeletedFiles>
            <uncompressibleFiles>0</uncompressibleFiles>
            <preCompressionBytes>0</preCompressionBytes>
            <compressedBytes>0</compressedBytes>
         </compressionInfo>
         <result value="0">
            <description/>success</description>
         </result>
      </getVolumeInfo>
   </volume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

Second Example

A nssRequest packet to return the volume information is as follows:

```
<nssRequest>
   <volume>
      <getVolumeInfo type="deletedVolume">
         <volumeName>2FNC78F4IHL3_DV</volumeName>
      </getVolumeInfo>
   </volume>
</nssRequest>
```

A nssReply packet to the get volume information command follows:

```
<nssReply>
   <volume>
      <getVolumeInfo>
          <deletedVolumeInfo>
             <deleteState value="3">auto purging paused
             </deleteState>
             <originalVolumeName>NSS2</originalVolumeName>
             <originalVolumeGUID>
                CA55AC24-3223-01D6-80-01-FBDA22AE6917
             </originalVolumeGUID>
             <deletedTime value="1015544045">
                Mar 7, 2002   4:34:05 pm
             </deletedTime>
             <scheduledPurgeTime value="1015889645">
                Mar 11, 2002   4:34:05 pm
             </scheduledPurgeTime>
             <lastStatus>0</lastStatus>
             <lastStatusSetter>zlssLogicalVolume.c[8326]
             </lastStatusSetter>
            </deletedVolumeInfo>
         <result value="0">
            <description/>success</description>
         </result>
      </getVolumeInfo>
   </volume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

listVolumes

Lists the volumes of a specified type.

Request

```
<listVolumes type=" "/>
```

Reply

```
<listVolumes>
   <volumeName type=" "/>
</listVolumes>
```

Elements

volumeName

Specifies the name of the volume.

Attributes

type

(Optional) Specifies what type of information is returned:

all (default)
nss
traditional

listEvmsVolumes

Lists the volumes in Evms that can be used to take a snap shot.

Request

```
<listEvmsVolumes/>
```

Reply

```
<listEvmsVolumes>
   <volumeInfo>
      <volumeName/>
      <volumeState/>
   </volumeInfo>
   <result value=" ">
      <description/>
   </result>
</listEvmsVolumes>
```

Elements

volumeName

Specifies the name of the volume.

volumeState

Specifies whether the state of the volume is shared (Yes or No).

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

modifyState

Modifies the state of either an NSS logical volume or a traditional NetWare volume.

Request

```
<modifyState>
   <volumeName/>
   <volumeState/>
   <mountPoint/>
</modifyState>
```

Reply

```
<modifyState>
   <result value=" ">
      <description/>
   </result>
</modifyState>
```

Elements

volumeName

Specifies the name of the volume for which to modify the state.

volumeState

Specifies the state to which the volume should be set (see "NSS Volume States" on page 458 and "Traditional Volume States" on page 460).

(Optional) mountPoint

Specifies the mount point for the volume (to support the Linux platform).

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to modify the state on a volume is as follows:

```
<nssRequest>
   <volume>
      <modifyState>
         <volumeName>NSS1</volumeName>
         <volumeState>deactive</volumeState>
         <mountPoint>/media/nss/volname</mountPoint>
      </modifyState>
   </volume>
</nssRequest>
```

A nssReply packet to the modify device command follows:

```
<nssReply>
   <volume>
      <modifyState>
         <result value="0">
            <description/>success</description>
         </result>
      </modifyState>
   </volume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

modifyVolumeInfo

Modifies the quota or enabled attributes on an NSS logical volume.

Request

```
<modifyVolumeInfo>
   <volumeName/>
   <basicInfo>
      <mountPoint/>
   </basicInfo>
   <volumeQuota quota=" "/>
   <volumeReadAhead=" "/>
   <enabledAttributes>
      <salvage enabled=" "/>
      <compression enabled=" "/>
      <directoryQuota enabled=" "/>
      <userQuota enabled=" "/>
      <flushFiles enabled=" "/>
      <mfl enabled=" "/>
      <snapshot enabled=" "/>
      <backup enabled=" "/>
      <shredding count=" "/>
      <migration enabled=" "/>
      <userTransaction enabled=" "/>
      <noAccessTime enabled=" "/>
   </enabledAttributes>
   <mountPoint/>
   <mountPointRename/>
   <nameSpace/>
</modifyVolumeInfo>
```

Reply

```
<modifyVolumeInfo>
   <result value=" ">
      <description/>
   </result>
</modifyVolumeInfo>
```

Elements

volumeName

Specifies the name of the NSS logical volume whose properties need to be modified.

basicInfo

Specifies the basic information for the volume.

mountPoint

Linux only. Specifies the volume's mount point.

volumeQuota

(Optional) Specifies the NSS logical volume's quota is to be modified.

volumeReadAhead

Specifies the number of read ahead blocks (multiple of 4096 bytes) to be used when reading data from the volume.

enabledAttributes

(Optional) Specifies that the state of one or more enabled attributes of a volume is to be modified.

salvage

(Optional) Specifies to enable or disable salvage.

compression

(Optional) Specifies to enable or disable compression.

directoryQuota

(Optional) Specifies to enable or disable directory quotas.

userQuota

(Optional) Specifies to enable or disable user quotas.

flushFiles

(Optional) Specifies to enable or disable flush on close.

mfl

(Optional) Specifies to enable or disable the modified files list.

snapshot

(Optional) Specifies to enable or disable file level snapshotting.

backup

(Optional) Specifies to enable or disable the backup flag.

shredding

(Optional) Specifies to enable or disable data shredding.

migration

(Optional) Specifies to enable or disable migration. Not currently implemented.

userTransaction

(Optional) Specifies to enable to disable the user transaction feature.

mountPoint

(Optional) Specifies the mount point for the volume (to support the Linux platform). For example

```
<mountPoint>/media/nss/volname</mountPoint>
```

mountPointRename

(Optional) Specifies that the mount point is renamed if the volume is renamed (to support the Linux platform). This feature works only if the volume is mounted in its default location.

nameSpace

Specifies the default namespace for requests to the volume (to support the Linux platform):

long
unix
mac
dos

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the returned result.

Attributes

quota

Specifies the maximum size (in bytes) to which the volume is allowed to grow. A value of none specifies that the volume is allowed to use any free space in the pool.

enabled

Specifies whether to enable the option:

“yes” Enable the option
“no” Disable the option

The quotes are required.

count

Specifies whether to enable the data shredding option:

1-7 Enable the option
0 Disable the option

Remarks

Note that volumes cannot be encrypted or unencrypted after they are created.

Example

The following command sets the quota for the volume to none, which allows it to grow to the size of the pool and sets up the various enabled attributes:

```
<nssRequest>
   <volume>
      <modifyVolumeInfo>
         <volumeName>MYVOL</volumeName>
         <volumeQuota quota="none"/>
         <enabledAttributes>
            <salvage enabled="yes"/>
            <compression enabled="no"/>
            <directoryQuota enabled="yes"/>
            <userQuota enabled="yes"/>
            <flushFiles enabled="no"/>
            <mfl enabled="no"/>
            <snapshot enabled="no"/>
            <backup enabled="yes"/>
            <shredding count="2"/>
            <migration enabled="no"/>
            <userTransaction enabled="no"/>
         </enabledAttributes>
      </modifyVolumeInfo>
   </volume>
 </nssRequest>
```

A nssReply packet to modify volume information command follows:

```
<nssReply>
   <volume>
      <modifyVolumeInfo>
         <result value="0">
            <description/>success</description>
         </result>
      </modifyVolumeInfo>
   </volume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

removeUser

Removes a user from the NSS user ID database.

Request

```
<removeUser>
   <volumeName/>
   <id/>
</removeUser>
```

Reply

```
<removeUser>
   <result value=" ">
      <description/>
   </result>
</removeUser>
```

Elements

volumeName

Specifies the name of the volume on the server (not the eDirectory name).

id

Specifies the ID of the user, in complete form with dashes, where each x represents a hexadecimal digit.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to remove a user is as follows:

```
<nssRequest>
   <volume>
      <removeUser>
         <volumeName>volName</volumeName>
         <id>xxxxxxxx-xxxx-xxxx-xx-xx-xxxxxxxxxxxx</id>
     </removeUser>
   </volume>
</nssRequest>
```

A nssReply packet to the modify device command follows:

```
<nssReply>
   <volume>
      <removeUser>
         <result value="0">
            <description/>success</description>
         <result>
   </volume>
   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

removeVolume

Deletes either an NSS logical volume or a traditional NetWare volume. When a traditional NetWare volume is deleted, it is immediately deleted from the system. However, when an NSS logical volume is deleted, it remains in the NSS storage pool in a salvageable state until it is either manually purged or until its scheduled auto purge time expires.

Request

```
<removeVolume>
   <volumeName/>
   <dontRemoveNDSObject/>
   <updateVLDB/>
</removeVolume>
```

Reply

```
<removeVolume>
   <result value=" ">
      <description/>
   </result>
</removeVolume>
```

Elements

volumeName

Specifies the name of the volume to remove.

dontRemoveNDSObject

Specifies that the volume's eDirectory object should not be removed. If this element is not specified, the eDirectory name is removed.

updateVLDB

Specifies that the DFS Volume Location Database (VLDB) is updated by the XML processing code. This element is for backward-compatibility with ConsoleOne, which does not know about this element but does its own VLDB updating. All new code should include this element.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to remove a volume is as follows:

```
<nssRequest>
   <volume>
      <removeVolume>
         <volumeName>NSS1</volumeName>
      </removeVolume>
   </volume>
</nssRequest>
```

A nssReply packet to the remove volume command follows:

```
<nssReply>
   <volume>
      <removeVolume>
         <result value="0">
            <description/>success</description>
         </result>
      </removeVolume>
   </volume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

renameVolume

Renames either a NSS logical volume or a traditional NetWare volume.

Request

```
<renameVolume>
   <volumeName/>
   <newVolumeName/>
   <newNDSName/>
   <dontRenameNDSObject/>
   <updateVLDB/>
</renameVolume>
```

Reply

```
<renameVolume>
   <result value=" ">
      <description/>
   </result>
</renameVolume>
```

Elements

volumeName

Specifies the name that the volume is known by on the server.

newVolumeName

Specifies the new name that the volume is known by.

newNDSName

(Required unless dontRenameNDSObject is specified) Specifies the new name to which the eDirectory volume object is renamed. If NULL is specified, the new name of the eDirectory volume object is generated by prepending the server name and an underscore to the value of the volumeName element.

dontRenameNDSObject

(Optional) Specifies not to rename the eDirectory object. If specified, the newNDSName and context elements are ignored.

updateVLDB

(Optional) Specifies that the DFS Volume Location Database (VLDB) is updated by the XML processing code. This element is for backward-compatibility with ConsoleOne, which does not know about the element but does its own VLDB updating. All new code should include this element.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to rename a volume is as follows:

```
<nssRequest>
   <volume>
      <renameVolume>
         <volumeName>MYVOL</volumeName>
         <newVolumeName>NSS1</newVolumeName>
         <newNDSName>
      </renameVolume>
   </volume>
</nssRequest>
```

A nssReply packet to the rename volume command follows:

```
<nssReply>
   <volume>
      <renameVolume>
         <result value="0">
            <description/>success</description>
         </result>
      </renameVolume>
   </volume>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

3.18 Volume MN Operations

This section contains the following Volume MN commands:

- “changeJobState” on page 314
- “createJob” on page 315
- “getJobList” on page 317
- “getJobStatus” on page 318
- “listSkippedFiles” on page 320

Each command is wrapped with either the nssRequest or nssReply element and the volMN element.

changeJobState

Modifies the state of a job.

Request

```
<changeJobState>
   <ID/>
   <time/>
   <pause/>
   <abort/>
   <resume/>
</changeJobState>
```

Reply

```
<changeJobState/>
```

Elements

time

Specifies the current time or GeneralizedTime value.

createJob

Returns a list of jobs.

Request

```
<createJob>
   <srcVol/>
   <srcPath/>
   <tgtServer/>
   <tgtVol/>
   <time/>
   <user/>
   <password/>
   <unp/>
   <copy/>
</createJob>
```

Reply

```
<createJob>
   <result value=" ">
      <description/>
   </result>
   <ID/>
</createJob>
```

Elements

srcVol

Specifies a host resource name for the volume (rather than the eDirectory resource name). Do not use a colon in the name.

srcPath

Specifies that the volume is split.

tgtServer

Specifies the name of the target server in the following format:

```
.CommonName.ContainerNamesDelimitedWithDots.TreeName.
```

time

Specifies the current time or the GeneralizedTime value.

user

Specifies the name of the user. The leading dot is required.

unp

Specifies the user's protected credentials, encoded as a hex string. Either unp or user and password must be supplied.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

ID

Specifies the ID for the job.

Example

One example of the request portion of createJob is as follows:

```
<nssRequest>     <!-- Main element for NSS requests -->
   <volMN>      <!-- (Opt, Rpt) Volume MN operations -->
      <createJob>
         <srcVol>Vol</srcVol>
         <srcPath>\a\b\c</srcPath>
         <tgtServer>DSSvrObjectName</tgtServer>
         <tgtVol>Vol</tgtVol>
         <time>20021225060000</time>
         <user>.admin.novell</user>
         <password>junk</password>
         <unp>0123456789ABCDEF</unp>
         <copy/>
      </createJob>
   </volMN>
</nssRequest>
```

getJobList

Returns a list of jobs.

Request

```
<getJobList/>
```

Reply

```
<getJobList>
   <result value=" ">
      <description/>
   </result>
   <ID/>
</getJobList>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

ID

Specifies the ID of each job. One ID is returned per job that is known to Volume Manager.

getJobStatus

Returns the status for a specified job.

Request

```
<getJobStatus>
   <ID/>
</getJobStatus>
```

Reply

```
<getJobStatus>
   <job>
      <ID/>
      <jobType/>
      <srcPath/>
      <state/>
      <pctComplete/>
      <time/>
      <comment/>
      <retryCount/>
      <skippedFileCount/>
      <totalFiles/>
   </job>
   <result value=" ">
      <description/>
   </result>
</getJobStatus>
```

Elements

ID

Specifies the ID of each job to return the status of.

jobType

Specifies the type of job (see “Job Types” on page 458).

state

Specifies the state of the job (see “State Values” on page 459).

time

Specifies the UTC time of the job.

comment

Specifies that the job is paused (only if the state is ReplayingEFL).

retryCount

Specifies the number of EFL epochs that have been replayed so far. Volume Manager keeps replaying indefinitely. If this number reaches a determined point, the administrator should prevent any further changes.

skippedFileCount

Specifies the number of files that were skipped (only if the state is Completed or FilesSkipped).

totalFiles

Specifies the running count while the job is in the Scanning state (which shows the scan is making progress). This element is used if state is not Completed or Failed. Once the job is out of the Scanning state, this element shows the total number of data sets that need to be moved until the job completes or fails.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Remarks

getJobStatus returns the following:

```
<nssReply>
   <volMn>
      <getJobStatus>
         <job>
            <id>1234</id>
            <jobType>MOVE</jobType>
            <srcPath>SYS:\</srcPath>
            <state>Running</state>
            <pctComplete>45</pctComplete>
            <time>12452542346</time>
            <comment>Cool</comment>
            <retryCount>1</retryCount>
            <skippedFileCount>0</skippedFileCount>
            <totalFiles>6894</totalFiles>
         </job>
      </getJobStatus>
   </volMn>
</nssReply>
```

listSkippedFiles

Returns a list of files that are in the FILES_SKIPPED state.

Request

```
<listSkippedFiles>
   <ID/>
   <cookie/>
</listSkippedFiles>
```

Reply

```
<listSkippedFiles>
   <result value=" ">
      <description/>
   </result>
   <ID/>
   <cookie/>
   <fileName/>
   <fileName/>
</listSkippedFiles>
```

Elements

ID

Specifies the operation ID of the move or split job that is in the FILES_SKIPPED state.

cookie

Specifies the current value of the last request. The initial value is 0. Keep issuing the request with the previous returned value to retrieve all files.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to expand a traditional volume is as follows:

```
<nssRequest>
   <volMN>
      <listSkippedFiles>
         <ID>12345</ID>
         <cookie>0</cookie>
      </listSkippedFiles>
   </volMN>
</nssRequest>
```

A nssReply packet to the expand a traditional volume command follows:

```
<nssReply>
   <volMN>
      <listSkippedFiles>
         <result value=" ">
            <description/>
         </result>
         <ID>12345</ID>
         <cookie>1234</cookie>
         <fileName>VOL1:FOO/BAR.TXT</fileName>
         <fileName>VOL1:FOO/BAR.XYZ</fileName>
      </listSkippedFiles>
   </volume>
</nssReply>
```

4 User Commands

The commands in this section are similar to the commands listed in Section 3.15, “User Space Restriction,” on page 239. However, these commands are intended for user operations that can be performed by non-administrative users.

Use the following path name to open the `user.cmd` file:

```
_admin/Manage_NSS/user.cmd
```

Every command is wrapped with either the userRequest and userQuota elements or the userReply element.

This section contains definitions for the following command categories:

- “browseUserSpaceRestrictions (user)” on page 324
- “getUserSpaceRestriction (user)” on page 326
- “setUserSpaceRestriction (user)” on page 328
- “listUserSpaceRestrictions (user)” on page 330

browseUserSpaceRestrictions (user)

Returns the list of users with space restrictions on a specified volume and the quotas for those users.

Request

```
<browse>
   <volumeName/>
   <allUsers/>
</browse>
```

Reply

```
<browse>
   <userList>
      <user>
         <id/>
         <userName/>
         <quota/>
         <spaceUsed/>
      </user>
   </userList>
   <result value="">
      <description/>
   </result>
<browse>
```

Elements

volumeName

(Required) Specifies the volume's name for which to return the user space restriction.

allUsers

(Optional) Specifies to return all users that are using storage but have no restrictions. This functionality is useful in listing the storage in use for all users.

user

Repeats for each user.

id

Specifies the unique ID for the user.

userName

Specifies the user name for the restricted user.

quota

Specifies the quota for the restricted user.

spaceUsed

Specifies the space in use by the restricted user.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A userRequest packet to list space restrictions for the MYVOL volume is as follows:

```
<userRequest>
      <browse>
         <volumeName>MYVOL</volumeName>
      </browse>
</userRequest>
```

A userReply packet to the list space restrictions command follows:

```
<userReply>
        <browse>
         <userList>
            <user>
               <userName>somebody.somedept.someorg</username>
               <quota>1048576</quota>
               <spaceUsed>401524</spaceUsed>
            </user>
            <user>
               <userName>someone.somedept.someorg</username>
               <quota>262144</quota>
               <spaceUsed>65534</spaceUsed>
            </user>
         </userList>
      </browse>
      <result value="0">
      <description/>zOK</description>
   </result>
</userReply>
```

getUserSpaceRestriction (user)

Returns the user space restriction for a supplied user on a specified volume.

Request

```
<get>
   <volumeName/>
   <id/>
   <userName/>
</get>
```

Reply

```
<get>
   <quota/>
   <noQuota/>
   <fullyRestricted/>
   <spaceUsed/>
   <result value="">
      <description/>
   </result>
</get>
```

Elements

volumeName

(Required) Specifies the volume's name for which to return the user space restrictions.

id

Specifies the unique ID for the user as returned from browseUserSpaceRestrictions (user) (page 324). Either the user's ID or name must be specified.

userName

Specifies the DN of the user for which to return restrictions. Either the user's name or ID must be specified.

quota

Specifies the quota for the requested user.

noQuota

Specifies that the user has no limit quota on the specified volume.

fullyRestricted

Specifies that the user is limited to no space or new usage on the specified volume.

spaceUsed

Specifies the space in use by the restricted user.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A userRequest packet to get space restrictions for a volume is as follows:

```
<userRequest>
         <get>
         <volumeName>MYVOL</volumeName>
         <userName>somebody.somedept.someorg</userName>
         </get>
</userRequest>
```

A userReply packet to the get space restrictions command follows:

```
<userReply>
       <get>
          <quota>1048576</quota>
          <spaceUsed>401524</spaceUsed>
       </get>
   <result value="0">
      <description/>zOK</description>
   </result>
</userReply>
```

setUserSpaceRestriction (user)

Adds the user space restriction for a supplied user DN to a specified volume.

Request

```
<set>
   <volumeName/>
   <id/>
   <userName/>
   <quota/>
   <noQuota/>
   <fullyRestricted/>
</set>
```

Reply

```
<set>
   <result value="">
      <description/>
   </result>
</set>
```

Elements

volumeName

(Required) Specifies the volume's name to which to add a user space restriction.

id

Specifies the unique ID of the user. Either the user's ID or name must be specified.

userName

Specifies the DN of the user to add. Either the user's name or ID must be specified.

quota

(Required unless noQuota or fullyRestricted is specified) Specifies the quota for the specified user.

noQuota

(Required unless quota or fullyRestricted is specified) Specifies that the user has no limit and removes any existing restriction.

fullyRestricted

(Required unless quota or noQuota is specified) Specifies that the user is limited to no more space.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A userRequest packet to set space restrictions for a volume is as follows:

```
<userRequest>
      <set>
         <volumeName>MYVOL</volumeName>
         <userName>somebody.somedept.someorg</userName>
         <quota>1048576</quota>
      </set>
</userRequest>
```

A userReply packet to the set space restrictions command follows:

```
<userReply>
       <set>
          <result value="0">
               <description/>success</description>
            </result>
       </set>
   <result value="0">
      <description/>zOK</description>
   </result>
</userReply>
```

listUserSpaceRestrictions (user)

Lists the user space restrictions for n number of users, based on the last user information stored provided.

Request

```
<list>
   <volumeName/>
   <lastUserID/>
   <numEntries/>
   <allUsers/>
</list>
```

Reply

```
<list>
<userList>
      <user>
         <id/>
         <userName/>
         <quota/>
         <spaceUsed/>
      </user>
   </userList>
   <result value="">
      <description/>
   </result>
</list>
```

Elements

volumeName

(Required) Specifies the volume's name to which to add a user space restriction.

lastUserID

Specifies the unique ID of the last user. Either the user's ID or name must be specified. Passing of an invalid GUID `00000000-0000-0000-00-00-000000000000` indicates that the user ID should begin with the first user

numEntries

Specifies the numebr of entries from the last user ID.

allUsers

(Optional) Specifies to return all users that are using storage but have no restrictions. This functionality is useful in listing the storage in use for all users.

fullyRestricted

Specifies that the user is limited to no more space.

noQuota

Specifies that the user has no limit.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A userRequest packet to list user quota for a volume is as follows:

```
<userRequest>
   <userQuota>
      <list>
         <volumeName><![CDATA[VOL2]]></volumeName>
         <lastUserID>00000000-0000-0000-00-00-000000000000</lastUserID>
         <numEntries>20</numEntries>
      </list>
   </userQuota>
 </userRequest>
```

A userReply packet to list user quota for a volume is as follows:

```
<userReply>
   <userQuota>
      <list>
         <userList>
            <user>
              <id>ab959229-a26c-4c58-16-ba-299295ab6ca2</id>
              <userName>wwwrun.novell</userName>
              <quota>1048576</quota>
              <spaceUsed>0</spaceUsed>
            </user>
         </userList>
   <result value="0">
      <description/>zOK</description>
   </result>
      </list>
    </userQuota>
</userReply>
```

5 NDS.cmd Definitions

This documentation provides the XML element definitions for `nds.cmd`.

To open the `nds.cmd` file, type the following:

```
_admin/Manage_NSS/NDS.cmd (Linux)
```

```
_Admin/manage_NSS/NDS.cmd (NetWare)
```

Every time you open the nds.cmd file and before you send other commands, you must type the following to write to the file:

```
<virtualIO><datastream name="command"></virtualIO>
```

The `nds.cmd` file contains XML element definitions for the following operations:

- Section 5.1, “Object Operations,” on page 334
- Section 5.2, “Pool Operations,” on page 336
- Section 5.3, “Volume Operations,” on page 341
- Section 5.4, “User Operations,” on page 346

5.1 Object Operations

This section contains the following command, which is called to return the value of a specific attribute on an eDirectory object:

- “getAttribute” on page 335

Each command is wrapped with either the ndsRequest or ndsReply element and the ndsObject element.

getAttribute

Returns the value of a specific attribute that belongs to a specific eDirectory object.

Request

```
<getAttribute>
   <name/>
   <context/>
   <attributeName/>
</getAttribute>
```

Reply

```
<getAttribute>
   <ndsAttribute name=" " syntax=" ">
      <value/>
   </ndsAttribute>
</getAttribute>
```

Elements

name

(Required) Specifies the eDirectory object name for which one of the attribute values is returned.

context

(Required) Specifies the eDirectory context where the object is located.

attributeName

(Required) Specifies the name of the attribute whose value is returned.

ndsAttribute

Specifies the attribute's name and syntax.

value

Repeats for each attribute.

Attributes

name

Specifies the attribute's name.

syntax

Specifies the eDirectory syntax value. For example, 3 is case insensitive string.

5.2 Pool Operations

This section contains the following commands that you can call to manipulate eDirectory objects for NSS storage pools:

- “addPool” on page 337
- “removePool” on page 339

Call these commands if you want to manipulate eDirectory objects without affecting the actual NSS storage pools.

Each command is wrapped with either the ndsRequest or ndsReply element and the ndsPool element.

addPool

Creates an eDirectory pool object for an already existing NSS storage pool.

Request

```
<addPool>
   <name/>
   <context/>
   <poolName/>
   <shared/>
   <linkVolumes/>
</addPool>
```

Reply

```
<addPool>
   <result value=" ">
      <description/>
   </result>
</addPool>
```

Elements

name

(Required) Specifies the name of the eDirectory pool object that will be created to represent the pool. If NULL is specified, the name of the eDirectory pool object is generated by prepending the server name and an underscore to the poolName element and adding “_POOL” to the end of the name.

context

(Required) Specifies the eDirectory context in which the eDirectory pool object is created. If NULL is specified, the eDirectory pool object is created in the same context where the server object resides.

poolName

(Require) Specifies the name of the pool on the server.

shared

(Optional) Specifies that a flag is set in the eDirectory pool object that indicates the pool is on a shareable-for-clustering partition. This element does not contain any content.

linkVolumes

Specifies that the eDirectory context is searched for volume objects. Each object associated with the pool is then linked to the pool object. This element does not contain any content.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

The following example uses the default name and context for the eDirectory pool object:

```
<ndsRequest>
   <ndsPool>
      <addPool>
         <poolName>MYPOOL</poolName>
         <name>
         <context>
         <linkVolumes>
      </addPool>
   </ndsPool>
</ndsRequest>
```

A nssReply packet to the add pool command follows:

```
<nssReply>
   <ndsPool>
      <addPool>
         <result value="0">
            <description/>success</description>
         </result>
      </addPool>
   </ndsPool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

removePool

Removes an eDirectory pool object.

Request

```
 <removePool>
   <name/>
   <context/>
</removePool>
```

Reply

```
<removePool>
   <result value=" ">
      <description/>
   </result>
</removePool>
```

Elements

name

(Required) Specifies the eDirectory name of the pool object to remove.

context

(Required) Specifies the eDirectory context where the object is found.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to remove a pool is as follows:

```
<ndsRequest>
   <ndsPool>
      <removePool>
         <name>MYSERVER_MYPOOL_POOL</name>
         <context>
      </removePool>
   </ndsPool>
</ndsRequest>
```

A nssReply packet to the remove pool command follows:

```
<nssReply>
   <ndsPool>
      <removePool>
         <result value="0">
            <description/>success</description>
         </result>
      </removePool>
   </ndsPool>

   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

5.3 Volume Operations

This section contains the following commands that can be called to manipulate eDirectory objects for NSS logical and traditional NetWare volumes:

- “addVolume” on page 342
- “removeVolume” on page 344

Each command is wrapped with either the ndsRequest or ndsReply element and the ndsVolume element.

addVolume

Creates an eDirectory pool object for an already existing NSS storage pool.

Request

```
<addVolume type=" ">
   <name/>
   <context/>
   <volumeName/>
   <poolName/>
   <ndsPoolName/>
   <dfsGUID/>
</addVolume>
```

Reply

```
<addVolume>
   <result value=" ">
      <description/>
   </result>
</addVolume>
```

Elements

name

(Required) Specifies the name of the eDirectory volume object to create. If NULL is specified, the name of the eDirectory volume object is generated by prepending the server name and an underscore to the volume name.

context

(Required) Specifies the eDirectory context in which the eDirectory volume object is created. If NULL is specified, the eDirectory volume object is created in the same context where the server object resides.

volumeName

Specifies the name of the volume on the server.

poolName

(Required for NSS) Specifies the name of the pool on the server. This element is ignored for traditional volumes.

ndsPoolName

(Required for NSS) Specifies the value to be use as the nssfsPool attribute name of the eDirectory volume object. If NULL is specified, the pool's actual eDirectory name is retrieved and used for the nssfsPool attribute. This element is ignored for traditional volumes.

dfsGUID

Specifies the value used as the dfs-volume-guid attribute for the eDirectory volume object.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to add a volume is as follows:

```
<ndsRequest>
   <ndsVolume>
      <addVolume>
         <volumeName>NSS1</volumeName>
         <poolName>MYPOOL</poolName>
         <dfsGUID>
            D428AB28-3216-01D6-80-01-BD09318C30CA
         </dfsGUID>
         <name>
         <context>
         <ndsPoolName>
      </addVolume>
   </ndsVolume>
</ndsRequest>
```

A nssReply packet to the add volume command follows:

```
<nssReply>
   <ndsVolume>
      <addVolume>
         <result value="0">
            <description/>success</description>
         </result>
      </addVolume>
   </ndsVolume>
   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

removeVolume

Removes an eDirectory volume object.

Request

```
<removeVolume>
   <name/>
   <context/>
</removeVolume>
```

Reply

```
<removeVolume>
   <result value=" ">
      <description/>
   </result>
</removeVolume>
```

Elements

name

(Required) Specifies the eDirectory name of the volume object to remove.

context

(Required) Specifies the eDirectory context where the object is found.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

A nssRequest packet to remove a volume is as follows:

```
<ndsRequest>
   <ndsVolume>
      <removeVolume>
         <name>MYSERVER_NSS1</name>
         <context>
      </removeVolume>
   </ndsVolume>
</ndsRequest>
```

A nssReply packet to the remove volume command follows:

```
<nssReply>
   <ndsVolume>
      <removeVolume>
         <result value="0">
            <description/>success</description>
         </result>
      </removeVolume>
   </ndsVolume>
   <result value="0">
      <description/>zOK</description>
   </result>
</nssReply>
```

5.4 User Operations

eDirectory user operations include the following commands:

- “addUser” on page 347
- “removeUser” on page 349

Each command is wrapped with either the ndsRequest or ndsReply element and the ndsUser element.

addUser

Adds a user.

Request

```
<addUser>
   <name/>
   <context/>
   <surname/>
   <userDescription/>
   <securityEquals/>
   <fullName/>
   <givenName/>
   <password/>
</addUser>
```

Reply

```
<addUser>
   <result value =" ">
      <description/>
   </result>
</addUser>
```

Elements

name

(Required) Specifies the eDirectory name of the user.

context

(Required) Specifies the eDirectory context where the user object is created. Unlike other VFS commands, the context must be in backslash format. For example

```
\novell_inc\novell\prv\nss\randys
```

surname

(Required)

userDescription

(Optional)

securityEquals

(Optional)

fullName

(Optional)

givenName

(Optional)

password

(Required)

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

```
<virtualIO><datastream name="command"/></virtualIO>

<ndsRequest>
   <ndsUser>
      <addUser>
         <name>testuser</name>
         <context>\TESTTREE\testorg\testou</context>
         <surname>testuser surname</surname>
         <fullName>testuser fullName</fullName>
         <givenName>testuser givenName</givenName>
         <userDescription>testuser description</userDescription>
         <password>secret</password>
      </addUser>
   </ndsUser>
</ndsRequest>
```

removeUser

Removes a user.

Request

```
<removeUser>
   <name/>
   <context/>
</removeUser>
```

Reply

```
<removeUser>
   <result value =" ">
      <description/>
   </result>
</removeUser>
```

Elements

name

(Required) Specifies the eDirectory name of the user object to remove.

context

(Required) Specifies the eDirectory context where the object is found.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

6 FileEvents.xml Definitions

The event file list (EFL) logs file changes for each active epoch on a specific volume. It uses the following admin volume file:

```
_admin:manage_nss\volume\(volumename)\FileEvents.xml
```

Each of the following commands is wrapped with either the nssRequest or nssReply element and the fileEventList element:

- “changeEventEpoch” on page 352
- “getEFLNameSpaceID” on page 353
- “getInactiveEpochInterval” on page 354
- “listAllFiles” on page 355
- “listEpochs” on page 356
- “listFileEvents” on page 357
- “pingEpoch” on page 358
- “removeEventEpoch” on page 359
- “resetEventList” on page 360
- “setEFLNameSpaceID” on page 361
- “setInactiveEpochInterval” on page 362
- “startEventEpoch” on page 363
- “stopEventEpoch” on page 364

changeEventEpoch

Stops an existing active EFL epoch and starts a new active epoch.

Request

```
<changeEventEpoch epochNumber=" "/>
```

Reply

```
<changeEventEpoch>
   <epoch value=" "/>
   <newEpoch value=" "/>
   <result value=" ">
      <description/>
   </result>
</changeEventEpoch>
```

Elements

epoch

Specifies the number of the epoch.

newEpoch

Specifies the new number of the epoch.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

epochNumber

Specifies the number of an epoch.

value

Specifies a number.

getEFLNameSpaceID

Returns the name space for the specified EFL. The returned value represents the name space in which the EFL's full path is represented (default is 4, zNSPACE_LONG).

Request

```
<getEFLNameSpaceID/>
```

Reply

```
<getEFLNameSpaceID>
   <namespace id=" "/>
</getEFLNameSpaceID>
```

Elements

namespace

Specifies the ID of the event list's name space.

getInactiveEpochInterval

Returns the inactive interval (in seconds) for the specified EFL. The returned value determines whether the epoch stays inactive for too long and needs to be removed. The default value is two weeks (1,209,600 seconds). In order for an epoch to stay active longer than this value, the user needs to ping the epoch periodically.

Request

```
<getInactiveEpochInterval/>
```

Reply

```
<getInactiveEpochInterval>
   <interval value=" "/>
   <result value=" ">
      <description/>
   </result>
</getInactiveEpochInterval>
```

Elements

interval

Specifies the value of the interval.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

listAllFiles

Enumerates all files on the specified volume, gives out their full path name and ID, and indicates whether each is a directory.

Request

```
<listAllFiles/>
```

Reply

```
<listAllFiles>
   <file>
      <name/>
      <id/>
      <directory/>
   </file>
   <result value=" ">
      <description/>
   </result>
</listAllFiles>
```

Elements

name

Specifies the name of the file.

id

Specifies the ID of the file.

directory

(Optional) Specifies that the file is a directory.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

listEpochs

Lists all EFL epochs on a specific volume and indicates whether they are active or stopped (used).

Request

```
<listEpochs/>
```

Reply

```
<listEpochs>
   <activeEpochs>
      <epoch value=" "/>
   </activeEpochs>
   <usedEpochs>
      <epoch value=" "/>
   </usedEpochs>
   <result value=" ">
      <description/>
   </result>
</listEpochs>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

listFileEvents

Enumerates the EFL of a specific epoch.

Request

```
<listFileEvents epochNumber=" "/>
```

Reply

```
<listFileEvents>
   <file>
      <action/>
      <name/>
      <id/>
      <directory/>
   </file>
   <result value=" ">
      <description/>
   </result>
</listFileEvents>
```

Elements

action

Specifies the action that was performed (created, renamed, etc.).

Return Value	Action Performed
1	A file is Created
2	A file is Deleted
3	Modified file data
4	Modified metadata
5	File rename
6	File undeleted

name

Specifies the full path name of the file.

id

Specifies the ID of the file.

directory

(Optional) Specifies that the file is a directory.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

pingEpoch

Pings a specific EFL epoch and prevents the epoch from removal during the period of time specifed by the EFL epoch inactive interval.

Request

```
<pingEpoch epochNumber=" "/>
```

Reply

```
<pingEpoch>
   <epoch value=" "/>
   <result value=" ">
      <description/>
   </result>
</pingEpoch>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

removeEventEpoch

Removes an EFL epoch and its file event list.

Request

```
<removeEventEpoch epochNumber=" "/>
```

Reply

```
<removeEventEpoch>
   <epoch value=" "/>
   <result value=" ">
      <description/>
   </result>
</removeEventEpoch>
```

Elements

epoch

Specifies the number of the epoch.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

resetEventList

Removes all epochs (active and used) and their EFLs on the specified volume.

Request

```
<resetEventList/>
```

Reply

```
<resetEventList>
   <result value=" ">
      <description/>
   </result>
</resetEventList>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

setEFLNameSpaceID

Sets the name space for the specified EFL and determines the name space in which the EFL's full path is represented.

Request

```
<setEFLNameSpaceID id=" "/>
```

Reply

```
<setEFLNameSpaceID>
   <result value=" ">
      <description/>
   </result>
</setEFLNameSpaceID>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

setInactiveEpochInterval

Sets the inactive interval (in seconds) for the specified EFL.

Request

```
<setInactiveEpochInterval value=" "/>
```

Reply

```
<setInactiveEpochInterval>
   <result value=" ">
      <description/>
   </result>
</setInactiveEpochInterval>
```

Elements

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

startEventEpoch

Starts a new active EFL epoch. All file changes (both metadata and user data) on the specific volume that are made after this epoch is started result in an entry in this epoch's event file list.

Request

```
<startEventEpoch/>
```

Reply

```
<startEventEpoch>
   <newEpoch value=" "/>
   <result value=" ">
      <description/>
   </result>
</startEventEpoch>
```

Elements

newEpoch

Specifies the number of the new epoch.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

stopEventEpoch

Stops an existing active EFL epoch. It keeps its file event list around, but does not add any new entry.

Request

```
<stopEventEpoch epochNumber=" "/>
```

Reply

```
<stopEventEpoch>
   <epoch value=" "/>
   <result value=" ">
      <description/>
   </result>
</stopEventEpoch>
```

Elements

epoch

Specifies the number of the epoch.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

7 files.cmd Definitions

This documentation provides the following XML element definitions for files.cmd that are available in NetWare® 6.5 and OES Linux:

- “addQuota” on page 366
- “addTrustee” on page 367
- “getAllEffectiveRights” on page 369
- “getEffectiveRightsByID” on page 371
- “getEffectiveRightsByUser” on page 372
- “getFileInfo” on page 373
- “modifyInheritedRightsFilter” on page 379
- “purgeDeletedFile” on page 381
- “removeAllTrustees” on page 382
- “removeTrustee” on page 384
- “salvageDeletedFile” on page 385
- “scanSalvageableFiles” on page 388
- “setFileInfo” on page 394

To open the files.cmd file, type the following:

```
_admin/Manage_NSS/files.cmd
```

Every time you open the files.cmd file and before you send other commands, you must type the following to write to the file:

```
<virtualIO><datastream name="command"></virtualIO>
```

You can combine multiple commands inside of one <fileRequest> element.

addQuota

Adds a directory quota.

Request

```
<directoryQuotas>
   <addQuota>
      <fileName/>
      <nameSpace/>
      <quotaAmount/>
   </addQuota>
   <symlink/>
</directoryQuotas>
```

Reply

```
<directoryQuotas>
   <addQuota>
      <result value=" ">
         <description/>
      </result>
   </addQuota>
</directoryQuotas>
```

Elements

fileName

Specifies the path name of the directory where the quota is to be set.

nameSpace

Specifies the name space of the file. nameSpace is an optional tag and is required only when the file name is specified as NSS path.

quotaAmount

Specifies the size of the quota (in bytes).

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

(Optional) symLink

Specifies not to follow the link.

NOTE: If the quota value is not assigned to the directory, 9223372036854775807 is returned. This value means quota is not set for the directory.

---

addTrustee

Adds a trustee to a file or directory. Attribute `type` in the request specifies `adID` to indicate an Active Directory identity.

Request

```
<trustees>
   <addTrustee>
      <name/>
      <userID/>
         <rights>
            <supervisor/>
            <read/>
            <write/>
            <create/>
            <erase/>
            <modify/>
            <fileScan/>
            <accessControl/>
            <salvage/>
            <secure/>
            <symlink/>
         </rights>
         <fileName/>
         <nameSpace/>
      </addTrustee>
   </trustees>
```

Reply

```
<trustees>
   <addTrustee>
      <result value=" ">
         <description/>
      </result>
   </addTrustee>
</trustees>
```

Elements

name

Specifies a name with the full context (including the tree name). The name can be delimited with either dots or slashes. If user ID is mentioned, user name is optional.

userID

Specifies the unique NDS User GUID. If both the userID and user name are passed, both the values should refer to the same user. If user name is mentioned, user ID is optional.

rights

Specifies the rights to assign to the file for the specified user.

fileName

Specifies the file name to add rights to.

nameSpace

Specifies the name space of the file. nameSpace is an optional tag and is required only when the file name is specified as NSS path.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

(Optional) symLink

Specifies not to follow the link.

Example

```
<virtualIO>
   <datastream name="command"/>
</virtualIO>

<fileRequest>
   <trustees>
      <addTrustee>
         <name><![CDATA[.CN=admin.O=novell.T=MYTREE.]]></name>
         <userID>78c19abc-d416-4eba-2f-81-bc9ac17816d4</userID>
            <rights>
               <supervisor/>
            </rights>
         <fileName>sys:\setup\test.txt</fileName>
         <nameSpace>unix</nameSpace>
      </addTrustee>
   </trustees>
</fileRequest>
```

getAllEffectiveRights

Returns all effective rights assigned to a file or directory.

Request

```
<fileInfo>
   <getAllEffectiveRights>
      <includeADIdentities/>
      <fileName/>
      <symlink/>
   </getAllEffectiveRights>
</fileInfo>
```

Reply

```
<fileInfo>
   <getAllEffectiveRights>
      <allAccessRights count="">
         <accessRights>
            <name/>
            <id/>
            <rights>
               <read/>
               <write/>
               <create/>
               <erase/>
               <accessControl/>
               <fileScan/>
               <modify/>
               <supervisor/>
            </rights>
         </accessRights>
      </allAccessRights>
      <result value="">
         <description/>
      </result>
   </getAllEffectiveRights>
</fileInfo>
```

Elements

includeADIdentities

Specifies to include AD identities as part of the trustees in the XML reply. The user type is specified using a new attribute `type`, which will specify the value of the attribute as `adID` to indicate an Active Directory identity.

fileName

Specifies a fully qualified name for the file. The context doesn't have to be included if a fully qualified name is specified.

accessRights

Repeats for each entry.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

(Optional) symLink

Specifies not to follow the link.

Attributes

count

Specifies the number of entries.

Example

The file request to return all effective rights is as follows:

```
<fileRequest>
   <fileInfo>
      <getAllEffectiveRights>
         <fileName>sys:\public
         </fileName>
      </getAllEffectiveRights>
   </fileInfo>
</fileRequest>
```

The file reply in which all rights are returned is as follows:

```
<fileReply>
   <fileInfo>
      <getAllEffectiveRights>
         <allAccessRights count="1">
            <accessRights>
               <name>.O=novell.T=test_tree
               </name>
               <id>083A4080-C752-11D7-A5-B7-888888888888
               </id>
               <rights>
                  <read/>
                  <write/>
                  <create/>
                  <erase/>
                  <accessControl/>
                  <fileScan/>
                  <modify/>
                  <supervisor/>
               </rights>
            </accessRights>
         <allAccessRights>
         <result value="0">
            <description/>success
            </description>
         </result>
      </getAllEffectiveRights>
   </fileInfo>
   <result value="0">
      <description/>zOK
      </description>
   </result>
   </fileInfo>
</fileReply>
```

getEffectiveRightsByID

Returns effective rights based on GUID for eDirectory users and SID for Active Directory users. Attribute `type` for request specifies `adID` to indicate an Active Directory identity.

Request

```
<fileInfo>
    <getEffectiveRightsByID>
        <userID/>
        <fileName/>
    </getEffectiveRightsByID>
</fileInfo>
```

Reply

```
<fileInfo>
    <getEffectiveRightsByID>
        <effectiveRights>
        <read/>
        <write/>
        <create/>
        <erase/>
        <accessControl/>
        <fileScan/>
        <modify/>
        <supervisor/>
        <visibility/>
        </effectiveRights>
        <result value="">
            <description/>
        </result>
    </getEffectiveRightsByID>
</fileInfo>
```

Elements

fileName

Specifies a fully qualified name for the file. The context doesn't have to be included if a fully qualified name is specified.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

getEffectiveRightsByUser

Returns effective rights by User. Attribute `type` for request specifies `adID` to indicate an Active Directory identity.

Request

```
<fileInfo>
    <getEffectiveRightsByUser>
        <name/>
        <fileName/>
    </getEffectiveRightsByUser>
</fileInfo>
```

Reply

```
  <fileInfo>
      <getEffectiveRightsByUser>
       <effectiveRights>
<read/>
<write/>
<create/>
<erase/>
<accessControl/>
<fileScan/>
<modify/>
<supervisor/>
</effectiveRights>
<result value="">
<description/>
</result>
</getEffectiveRightsByUser>
</fileInfo>
```

Elements

fileName

Specifies a fully qualified name for the file. The context doesn't have to be included if a fully qualified name is specified.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

getFileInfo

Returns the information (properties of a file).

Request

```
<fileInfo>
   <getEffectiveRightsByUser>
      <context/>
      <name/>
      <filename/>
      <nameSpace/>
      <nameType/>
   </getEffectiveRightsByUser>
   <getFileInfo>
      <includeADIdentities/>
      <fileName/>
      <typeOfInfo>
         <rightsInfo/>
         <standardInfo/>
         <timeInfo/>
         <idInfo/>
         <nfsInfo/>
         <visibilityInfo/>
         <symlink/>
         <directoryQuotaInfo>
            <quotaAmount/>
            <usedAmount/>
         </directoryQuotaInfo>
      </typeOfInfo>
   </getFileInfo>
</fileInfo>
```

Reply

```
<fileInfo>
   <getEffectiveRightsByUser>
      <effectiveRights>
         <read/>
         <write/>
         <create/>
         <erase/>
         <accessControl/>
         <fileScan/>
         <modify/>
         <visibility/>
         <supervisor/>
      </effectiveRights>
   </getEffectiveRightsByUser>
   <getFileInfo>
      <rightsInfo>
         <trusteeList>
            <trusteeInfo>
               <trustee>
                  <rights>
                     <supervisor/>
                     <read/>
                     <write/>
                     <create/>
                     <erase/>
                     <modify/>
                     <fileScan/>
                     <accessControl/>
                     <salvage/>
                     <secure/>
```

```
            </rights>
         </trustee>
      </trusteeInfo>
   </trusteeList>
   <inheritedRightsFilter>
      <supervisor/>
      <read/>
      <write/>
      <create/>
      <erase/>
      <modify/>
      <fileScan/>
      <accessControl/>
      <salvage/>
      <secure/>
   </inheritedRightsFilter>
   <effectiveRights>
     <read/>
     <write/>
     <create/>
     <erase/>
     <accessControl/>
     <fileScan/>
     <modify/>
     <supervisor/>
</effectiveRights>
   <result value=" ">
      <description/>
   </result>
</rightsInfo>
<standardInfo>
   <volumeName/>
   <id/>
   <parentID/>
   <logicalEOF/>
   <physicalEOF/>
   <attributes>
      <readOnly/>
      <hidden/>
      <system/>
      <subdirectory/>
      <archive/>
      <shareable/>
      <noSuballoc/>
      <transaction/>
      <notVirtual/>
      <immediatePurge/>
      <renameInhibit/>
      <deleteInhibit/>
      <copyInhibit/>
      <adminLink/>
      <link/>
      <remoteDataAccess/>
      <remoteDataInhibit/>
      <compressImmediate/>
      <dataStreamCompress/>
      <doNotCompress/>
      <noStreamCompress/>
      <attrArchive/>
      <volatile/>
   </attributes>
</standardInfo>
<timeInfo>
   <createdTime>
      <utc/>
      <string/>
   </createdTime>
   <archivedTime>
      <utc/>
      <string/>
```

```
         </archivedTime>
         <modifiedTime>
            <utc/>
            <string/>
         </modifiedTime>
         <accessedTime>
            <utc/>
            <string/>
         </accessedTime>
         <metaDataModifiedTime>
            <utc/>
            <string/>
         </metaDataModifiedTime>
      </timeInfo>
      <idInfo>
         <creator/>
         <archiver/>
         <modifier/>
         <metaDataModifier/>
      </idInfo>
      <nfsInfo>
          <nfsName/>
          <mode/>
          <symLink/>
          <user>
             <id/>
             <name/>
          </user>
          <group>
              <id/>
              <name/>
           </group>
           <rights/>
      </nfsInfo>
      <visibilityInfo>
         <visibilityEntry>
            <name/>
            <id/>
         </visibilityEntry>
      </visibilityInfo>
      <directoryQuotaInfo>
         <quotaAmount/>
         <usedAmount/>
      </directoryQuotaInfo>
         <result value=" ">
            <description/>
         </result>
      </getFileInfo>
   </fileInfo>
   <result value=" ">
      <description/>
   </result>
</fileReply>
```

Elements

context

Specifies the context of the user. For example, novell.server.tree. You can omit the context and specify a fully qualified name instead.

name

Specifies the name of the user. For example, admin.

fileName

Specifies the file name to retrieve the information for.

nameSpace

Specifies the name space of the file. nameSpace is an optional tag and is required only when the file name is specified as NSS path.

includeADIdentities

Specifies to include AD identities as part of the trustees in the XML reply. The user type is specified using a new attribute `type`, which will specify the value of the attribute as `adID` to indicate an Active Directory identity.

nameType

Specifies the type of the file name such as File, DataStream, ExtendedAttribute and DeletedFile.

typeOfInfo

Specifies the category of information to return.

rightsInfo

Specifies the trustee rights information.

standardInfo

Specifies the general information about the file (EOF, ID, attributes, etc.).

timeInfo

Specifies the creation, modification, archival, and access time.

directoryQuotaInfo

Specifies the directory quota information.

quotaAmount

Specifies the amount of the quota (in bytes). If there is no quota, quotaAmount is -1 and usedAmount is 0.

usedAmount

Specifies the number of bytes that are used in the current directory and its children. If -1 is specified, the quota is removed.

effectiveRights

Specifies the rights for the user that is making the request.

trusteeInfo

Repeats for each assigned trustee. Attribute `type` specifies `adID` to indicate an Active Directory identity.

trustee

Specifies the right-rooted, dot-delimited trustee name.

rights

Specifies the assigned rights for the trustee.

inheritedRightsFilter

Specifies the rights in the current filter.

standardInfo

Specifies the generic information for the file.

volumeName

Specifies the name of the logical volume that contains the file.

id

Specifies the unique ID for a file on a volume.

(Optional) symLink

Specifies not to follow the link.

parentID

Specified the ID of the primary parent.

logicalEOF

Specifies the location of the end of useful data in the file.

physicalEOF

Specifies the number of allocated bytes for the file (unless it is sparse).

timeInfo

Specifies various time stamps for the file.

utc

Specifies a number representing the UTC time.

string

Specifies the date, in string format.

idInfo

Specifies the IDs of various users. For creator, archiver, modifier, and metaDataModifier, attribute `type` specifies `adID` to indicate an Active Directory.

creator

Specifies the right-rooted, dot-delimited directory name of the file's creator.

archiver

Specifies the right-rooted, dot-delimited directory name of the person who last archived the file.

modifier

Specifies the right-rooted, dot-delimited directory name of the person who last modified the file.

metaDataModifier

Specifies the right-rooted, dot-delimited directory name.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

```
<virtualIO>
   <datastream name="command"/>
</virtualIO>

<fileRequest>
   <fileInfo>
      <getFileInfo>
      <fileName>test:\testdir</fileName>
      <nameSpace>unix</nameSpace>
         <typeOfInfo>
            <rightsInfo/>
            <standardInfo/>
            <timeInfo/>
            <idInfo/>
            <directoryQuotaInfo>
               <quotaAmount/>
               <usedAmount/>
           </directoryQuotaInfo>
        </typeOfInfo>
     </getFileInfo>
   </fileInfo>
</fileRequest>
```

modifyInheritedRightsFilter

Sets the inherited rights filter on a file or directory.

Request

```
<trustees>
   <modifyInheritedRightsFilter>
      <inheritedRightsFilter>
         <supervisor/>
         <read/>
         <write/>
         <create/>
         <erase/>
         <symlink/>
         <modify/>
         <fileScan/>
         <accessControl/>
         <salvage/>
         <secure/>
      </inheritedRightsFilter>
      <fileName/>
   </modifyInheritedRightsFilter>
</trustees>
```

Reply

```
<trustees>
   <modifyInheritedRightsFilter>
      <result value=" ">
         <description/>
      </result>
   </modifyInheritedRightsFilter>
</trustees>
```

Elements

inheritedRightsFilter

Specifies the rights to allow through the filter.

fileName

Specifies the name of the file on which to remove rights.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

(Optional) symLink

Specifies not to follow the link.

Example

```
<virtualIO>
   <datastream name="command"/>
</virtualIO>

<fileRequest>
   <trustees>
      <modifyInheritedRightsFilter>
         <inheritedRightsFilter>
            <supervisor/>
            <read/>
            <write/>
            <create/>
         </inheritedRightsFilter>
         <fileName>test:\testdir</fileName>
      </modifyInheritedRightsFilter>
   </trustees>
</fileRequest>
```

purgeDeletedFile

Permanently removes a deleted file from the salvage directory.

Request

```
<salvage>
   <purgeDeletedFile>
      <volumeName/>
      <id/>
      <symlink/>
   </purgeDeletedFile>
</salvage>
```

Reply

```
<salvage>
   <purgeDeletedFile>
      <result value=" ">
         <description/>
      </result>
   </purgeDeletedFile>
</salvage>
```

Elements

volumeName

Specifies the name of the volume.

id

Specifies the zID of the deleted file (in string format of a 64-bit unsigned integer).

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

(Optional) symLink

Specifies not to follow the link.

removeAllTrustees

Removes all trustee from a file or directory.

Request

```
<trustees>
   <removeAllTrustees>
      <fileName/>
      <nameSpace/>
      <symlink/>
   </removeAllTrustees>
</trustees>
```

Reply

```
<trustees>
   <removeAllTrustees>
      <result value=" ">
         <description/>
      </result>
   </removeAllTrustees>
</trustees>
```

Elements

fileName

Specifies a full context name, including the tree name. The name can be delimited with either dots or slashes.

nameSpace

Specifies the name space of the file. nameSpace is an optional tag and is required only when the file name is specified as NSS path.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

(Optional) symLink

Specifies not to follow the link.

Example

The request for removing all trustees is as follows:

```
<fileRequest>
   <trustees>
      <removeAllTrustees>
         <fileName>sys:\public</fileName>
         <nameSpace>unix</nameSpace>
      </removeAllTrustees>
   </trustees>
</fileRequest>
```

The reply to removing all trustees follows:

```
<fileReply>
   <trustees>
      <removeAllTrustees>
         <result value="0">
            <description/>success
            </description>
         </result>
      </removeAllTrustees>
   </trustees>
   <result value="0">
      <description/>zOK
      </description>
   </result>
</fileReply>
```

removeTrustee

Removes a trustee from a file or directory. Attribute `type` in the request specifies `adID` to indicate an Active Directory identity.

Request

```
<trustees>
   <removeTrustee>
      <name/>
      <symlink/>
      <userID/>
      <fileName/>
      <nameSpace/>
   </removeTrustee>
</trustees>
```

Reply

```
<trustees>
   <removeTrustee>
      <result value=" ">
         <description/>
      </result>
   </removeTrustee>
</trustees>
```

Elements

(Optional) name

Optional if userID is mentioned. Specifies a full context name, including the tree name. The name can be delimited with either dots or slashes.

fileName

Specifies the file name on which to remove rights.

nameSpace

Specifies the name space of the file. nameSpace is an optional tag and is required only when the file name is specified as NSS path.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

(Optional) symLink

Specifies not to follow the link.

userID

Optional if user name is mentioned. Specifies the unique NDS User GUID. If both the userID and user name are passed, both the values should refer to the same user.

salvageDeletedFile

Restores a deleted file.

Request

```
<salvage>
   <salvageDeletedFile>
      <volumeName/>
      <includeADIdentities/>
      <id/>
      <dstParentID/>
      <dstParentPath/>
      <dstParentFullPath/>
      <symlink/>
      <fileName/>
      <nameSpace/>
      <typeOfInfo>
         <rightsInfo/>
      </typeOfInfo>
   </salvageDeletedFile>
</salvage>
```

Reply

```
<salvage>
   <salvageDeletedFile>
      <typeOfInfo>
         <rightsInfo>
            <trusteeList>
               <trusteeInfo>
                  <trustee>
                  <rights>
                     <supervisor/>
                     <read/>
                     <write/>
                     <create/>
                     <erase/>
                     <modify/>
                     <fileScan/>
                     <accessControl/>
                     <salvage/>
                     <secure/>
                  </rights>
                  </trustee>
               </trusteeInfo>
            </trusteeList>
            <inheritedRightsFilter>
               <supervisor/>
               <read/>
               <write/>
               <create/>
               <erase/>
               <modify/>
               <fileScan/>
               <accessControl/>
               <salvage/>
               <secure/>
            </inheritedRightsFilter>
            <effectiveRights>
               <supervisor/>
               <read/>
               <write/>
               <create/>
```

```
                <erase/>
                <modify/>
                <fileScan/>
                <accessControl/>
                <salvage/>
                <secure/>
             </effectiveRights>
          </rightsInfo>
       </typeOfInfo>
       <result value=" ">
          <description/>
       </result>
    </salvageDeletedFile>
</salvage>
```

Elements

volumeName

Specifies the name of the volume.

includeADIdentities

Specifies to include AD identities as part of the trustees in the XML reply. The user type is specified using a new attribute `type`, which will specify the value of the attribute as `adID` to indicate an Active Directory identity.

id

Specifies the zID of the deleted file (in string format of a 64-bit unsigned integer).

dstParentID

Specifies the zID of the file's parent (in string format of a 64-bit unsigned integer).

dstParentPath

Specifies the path of the file's parent. An empty string indicates the volume root directory.

dstParentFullPath

Specifies the parent path of the salvaged file. This path can be a rooted Linux path or a NetWare path that includes the volume name.

fileName

Specifies the file name of the file to salvage.

nameSpace

Specifies the name space of the file name.

typeOfInfo

Specifies the categories of information to return.

rightsInfo

Specifies to return information on trustee rights.

trusteeInfo

Repeats for each assigned trustee. Attribute `type` specifies `adID` to indicate an Active Directory identity.

trustee

Specifies the right-rooted, dot-delimited trustee name.

rights

Specifies the assigned rights.

inheritedRightsFilter

Specifies the rights in the current filter.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

(Optional) symLink

Specifies not to follow the link.

scanSalvageableFiles

Checks for files that can be salvaged.

Request

```
<salvage>
   <scanSalvageableFiles>
      <includeADIdentities/>
      <volumeName/>
      <scanSequence/>
      <nameSpace/>
      <parentID/>
      <parentPath/>
      <parentFullPath/>
      <typeOfInfo>
         <rightsInfo/>
         <standardInfo/>
         <timeInfo/>
         <idInfo/>
         <symlink/>
         <dataStreamInfo/>
         <nameSpaceInfo/>
         <deletedInfo/>
      </typeOfInfo>
   </scanSalvageableFiles>
</salvage>
```

Reply

```
<salvage>
   <scanSalvageableFiles>
      <nextScanSequence/>
      <fileName/>
      <typeOfInfo>
         <rightsInfo>
            <trusteeList>
               <trusteeInfo>
                  <trustee>
                  <rights>
                     <supervisor/>
                     <read/>
                     <write/>
                     <create/>
                     <erase/>
                     <modify/>
                     <fileScan/>
                     <accessControl/>
                     <salvage/>
                     <secure/>
                  </rights>
                  </trustee>
               </trusteeInfo>
            </trusteeList>
            <inheritedRightsFilter>
               <supervisor/>
               <read/>
               <write/>
               <create/>
               <erase/>
               <modify/>
               <fileScan/>
               <accessControl/>
               <salvage/>
               <secure/>
```

```
   </inheritedRightsFilter>
   <effectiveRights>
      <supervisor/>
      <read/>
      <write/>
      <create/>
      <erase/>
      <modify/>
      <fileScan/>
      <accessControl/>
      <salvage/>
      <secure/>
   </effectiveRights>
</rightsInfo>
<standardInfo>
   <volumeName/>
   <id/>
   <parentID/>
   <logicalEOF/>
   <physicalEOF/>
   <attributes>
      <readOnly/>
      <hidden/>
      <system/>
      <subdirectory/>
      <archive/>
      <shareable/>
      <noSuballoc/>
      <transaction/>
      <notVirtual/>
      <renameInhibit/>
      <deleteInhibit/>
      <copyInhibit/>
      <adminLink/>
      <link/>
      <remoteDataAccess/>
      <remoteDataInhibit/>
      <compressImmediatePurge/>
      <dataStreamCompress/>
      <doNotCompress/>
      <noStreamCompress/>
      <attrArchive/>
      <volatile/>
   </attributes>
</standardInfo>
<timeInfo>
   <createdTime>
      <utc/>
      <string/>
   </createdTime>
   <archivedTime>
      <utc/>
      <string/>
   </archivedTime>
   <modifiedTime>
       <utc/>
       <string/>
   </modifiedTime>
   <accessedTime>
      <utc/>
      <string/>
   </accessedTime>
   <metaDataModifiedTime>
      <utc/>
      <string/>
   </metaDataModifiedTime>
</timeInfo>
<idInfo>
   <creator>
      <name/>
```

```
            <id/>
         </creator>
         <archiver>
            <name/>
            <id/>
         </archiver>
         <modifier>
            <name/>
            <id/>
         </modifier>
         <metaDataModifier>
            <name/>
            <id/>
         </metaDataModifier>
      </idInfo>
      <dataStreamInfo/>
         <count/>
         <totalDataSize>
      </dataStreamInfo>
      <nameSpaceInfo>
      <deletedInfo>
         <deletedTime>
            <utc/>
            <string/>
         </deletedTime>
         <deletorID>
      </deletedInfo>
   </typeOfInfo>
   <result value=" ">
      <description/>
   </result>
</scanSalvageableFiles>
</salvage>
```

Elements

includeADIdentities

Specifies to include AD identities as part of the trustees in the XML reply. The user type is specified using a new attribute `type`, which will specify the value of the attribute as `adID` to indicate an Active Directory identity.

volumeName

Specifies the name of the volume.

scanSequence

Specifies where to start the scan (in string format of a 64-bit unsigned integer). -1 specifies to start from the beginning.

nameSpace

Specifies the name space of the file name.

parentID

Specifies the zID of the directory to be scanned (in string format of a 64-bit unsigned integer).

parentPath

Specifies the path of the directory to scan. An empty string indicates the volume root directory.

parentFullPath

Specifies the path of the directory to scan. This path can be a rooted Linux path or a NetWare path that includes the volume name.

typeOfInfo

Specifies the categories of information to return.

rightsInfo

Specifies to return information on trustee rights.

standardInfo

Specifies to return general information about the file (EOF, ID, attributed, etc.).

timeInfo

Specifies to return the creation, modification, archival, and access times.

idInfo

Specifies to return the creation, modification, and archiver user names and IDs. If the ID cannot be found, the name returns Unknown User.

dataStreamInfo

Specifies to return the non-primary data stream count, size, etc.

nameSpaceInfo

Specifies to return the primary name space information.

deletedInfo

Specifies to return information about deleted files.

nextScanSequence

Specifies the next scan sequence (in 64-bit unsigned integer format).

fileName

Specifies the deleted file name (in CDATA format).

rightsInfo

Specifies the rights information (only if rightsInfo was specified in the request).

trusteeInfo

Repeats for each assigned trustee. Attribute `type` specifies `adID` to indicate an Active Directory identity.

trustee

Specifies the right-rooted, dot-delimited trustee name.

rights

Specifies the assigned rights.

inheritedRightsFilter

Specifies the rights in the current filter.

standardInfo

Specifies the generic information for the file.

volumeName

Specifies the name of the logical volume that contains the file.

id

Specifies the unique ID for a file or a volume (in string format of a 64-bit unsigned integer).

(Optional) symLink

Specifies not to follow the link.

parentID

Specifies the primary parent ID (in string format of a 64-bit unsigned integer).

logicalEOF

Specifies the location of the end of good data in the file.

physicalEOF

Specifies the number of allocated bytes for the file (unless the file is sparse).

timeInfo

Specifies various time stamps for the file.

utc

Specifies the UTC time.

string

Specifies the date, converted to a string.

idInfo

Specifies various user IDs. For creator, archiver, modifier, and metaDataModifier, attribute `type` specifies `adID` to indicate an Active Directory.

creator

Specifies the right-rooted, dot-delimited eDirectory name of the creator.

archiver

Specifies the right-rooted, dot-delimited eDirectory name of the archiver.

modifier

Specifies the right-rooted, dot-delimited eDirectory name of the modifier.

metaDataModifier

Specifies the right-rooted, dot-delimited eDirectory name of the metadata modifier.

dataStreamInfo

Specifies the non-primary data stream information.

count

Specifies the number of data streams.

totalDataSize

Specifies the total size (in bytes) of the data stream.

nameSpaceInfo

Specifies the primary name space:

DOS
Long

Macintosh

Unix

deletorID

Specifies the right-rooted, dot-delimited eDirectory name.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Example

```
<virtualIO>
   <datastream name="command"/>
</virtualIO>

<fileRequest>
   <salvage>
      <scanSalveageableFiles>
         <volumeName>test</volumeName>
         <scanSequence>-1</scanSequence>
         <nameSpace>DOS</nameSpace>
         <parentID/>
         <parentPath>test:\testdir</parentPath>
         <typeOfInfo>
            <rightsInfo/>
            <standardInfo/>
            <timeInfo/>
            <idInfo/>
            <dataStreamInfo/>
            <nameSpaceInfo/>
            <deletedInfo/>
         </typeOfInfo>
      </scanSalvageableFiles>
   </salvage>
```

setFileInfo

Modifies a file's information (properties).

Request

```
<fileInfo>
   <setFileInfo>
      <fileName/>
      <nameType/>
      <owner/>
      <attributes>
         <readOnly enabled=" "/>
         <hidden enabled=" "/>
         <system enabled=" "/>
         <execute enabled=" "/>
         <archive enabled=" "/>
         <sharable enabled=" "/>
         <transaction enabled=" "/>
         <immediatePurge enabled=" "/>
         <renameInhibit enabled=" "/>
         <deleteInhibit enabled=" "/>
         <copyInhibit enabled=" "/>
         <link enabled=" "/>
         <remoteDataAccess enabled=" "/>
         <remoteDataInhibit enabled=" "/>
         <compressImmediate enabled=" "/>
         <dataStreamCompress enabled=" "/>
         <doNotCompress enabled=" "/>
         <noStreamCompress enabled=" "/>
         <attrArchive enabled=" "/>
         <volatile enabled=" "/>
      </attributes>
      <symlink/>
      <nfsInfo>
         <user>
            <id/>
            <owner/>
         </user>
         <group>
            <id/>
            <owner/>
         </group>
         <rights>
      </nfsInfo>
      <idInfo>
         <creator/>
         <archiver/>
         <modifier/>
         <metaDataModifier/>

      </idInfo>
      <standardInfo>
         <attributes/>
      </standardInfo>
      <timeInfo>
         <createTime/>
         <archivedTime/>
         <accessedTime/>
         <modifiedTime/>
         <metaDataModifiedTime/>
      </timeInfo>
   </setFileInfo>
</fileInfo>
```

Reply

```
<fileInfo>
   <setFileInfo>
      <result value=" ">
         <description/>
      </result>
   </setFileInfo>
</fileInfo>
```

Elements

fileName

(Required) Specifies the name of the file.

nameType

Specifies the type of the file name such as File, DataStream, ExtendedAttribute and DeletedFile.

owner

(Optional) Specifies the name of the new owner (in either dot or slash form).

attributes

(Optional) Specifies a list of attributes to change. Each listed attribute also lists whether to enable or disable the specific attribute.

nfsInfo

(Optional) Specifies NFS information to change.

user

(Optional) Specifies either the ID or the name of the file owner.

id

Specifies the ID of the Unix owner.

(Optional) symLink

Specifies not to follow the link.

owner

Specifies the dot- or slash-delimited name from which to get the ID.

group

(Optional) Specifies either the ID or the name of the group assigned to the file.

rights

(Optional) Specifies a Unix octal value that represents the permissions to set.

idInfo

(Optional) Specifies a list of IDs to change for the file. For creator, archiver, modifier, and metaDataModifier, attribute `type` specifies `adID` to indicate an Active Directory.

creator

(Optional) Specifies the creator of the file (in dot or slash format).

archiver

(Optional) Specifies the archiver of the file (in dot or slash format).

modifier

(Optional) Specifies the modifier of the file (in dot or slash format).

metaDataModifier

(Optional) Specifies the metadata modifier of the file (in dot or slash format).

standardInfo

(Optional) Specifies the basic file attributes to change. This attributes list is the same as the preceding attributes list contained in the attributes element.

timeInfo

(Optional) Specifies the time stamps to change for the file.

createTime

(Optional) Specifies a new created time for the file (as a string in the format of YYYYMMDDHHMMSS).

archivedTime

(Optional) Specifies a new archived time for the file (as a string in the format of YYYYMMDDHHMMSS).

accessedTime

(Optional) Specifies a new accessed time for the file (as a string in the format of YYYYMMDDHHMMSS).

modifiedTime

(Optional) Specifies a new modified time for the file (as a string in the format of YYYYMMDDHHMMSS).

metaDataModifiedTime

(Optional) Specifies a new metadata modified time for the file (as a string in the format of YYYYMMDDHHMMSS).

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

Attributes

enabled

(Optional) Specifies if the current element should be enabled:

yes
no

8 Inventory.xml Definitions

NetWare® Remote Manager (NRM) Storage Resource Managment (SRM) files track inventory changes and uses the following files:

- "NRMServerInventory.xml" on page 398
- "Volume_Inventory.xml" on page 402
- "Volume_Trustees.xml" on page 405

`_admin:Novell\NRM\NRMServerInventory.xml`

Reading the NRMServerInventory.xml file causes NRM to generate an inventory for each volume and then combine each volume's inventory into a server inventory.

By generating a server inventory, you cause XML inventory files to be generated for each volume (see Volume_Inventory.xml and Volume_Trustees.xml). However, if an inventory was generated in the last hour, that inventory is used (rather than generating a new inventory). With this functionality, you can start an inventory and come back later to process that inventory's data.

When you first read the data, the Inventory_Status tags might indicate "Scanning," which tells you that NRM is currently generating the inventory. Keep trying until you receive a Done status. By processing inventories in this manner, you can asynchronously inventory multiple servers without having to wait for each inventory to complete (and without blocking any threads).

NRMServerInventory.xml

Generates an inventory for each volume and combines each volume's inventory into a server inventory.

Syntax

```
<Server_Inventory name=""/>
<Inventory_Status/>
<Space_Available/>
<Volume_Count/>
<Space_Used/>
<Directory_Count/>
<File_Count/>
<Space_Change_Last_Day/>
<Space_Change_Last_Week/>
<Space_Change_Last_Month/>
<Estimated_Months_Left/>
<File_Type_Count/>
<File_Owner_Inventory>
   <Owner Global_DN="" Local_ID="">
      <Space_Used/>
      <File_Count/>
   </Owner>
</File_Owner_Inventory>
<File_Type_Inventory>
   <Extension name="">
      <Space_Used/>
      <File_Count/>
   </Extension>
</File_Type_Inventory>
<File_Modified_Inventory>
   <Group range="">
      <Space_Used/>
      <File_Count/>
   </Group>
</File_Modified_Inventory>
<File_Access_Inventory>
   <Group range="">
      <Space_Used/>
      <File_Count/>
   </Group>
</File_Access_Inventory>
<File_Create_Inventory>
   <Group range="">
      <Space_Used/>
      <File_Count/>
   </Group>
</File_Create_Inventory>
<File_Size_Inventory>
   <Group range="">
      <Space_Used/>
      <File_Count/>
   </Group>
</File_Size_Inventory>
```

Elements

Server_Inventory

Specifies the root element for NRM inventory requests.

Inventory_Status

Specifies the status of the inventory:

Done

Scanning

Space_Available

Specifies the amount of free space on the server.

Volume_Count

Specifies the number of volumes on the server (excluding the _ADMIN volume).

Space_Used

Specifies the byte space in use by all files on the server.

Directory_Count

Specifies the number of subdirectories on the server.

File_Count

Specifies the number of files on the server.

Space_Change_Last_Day

Specifies the change in terms of available space (MB) for the last day, as a signed integer.

Space_Change_Last_Week

Specifies the change in terms of available space (MB) for the last week, as a signed integer.

Space_Change_Last_Month

Specifies the change in terms of available space (MB) for the last month, as a signed integer.

Estimated_Months_Left

Specifies the number of months before the server will have a volume run out of available space (in consideration of last month's rate).

File_Type_Count

Specifies the number of file types that are being tracked in the inventory. If the Inventory_Status is Scanning, it specifies the number of files that have been scanned thus far.

File_Owner_Inventory

Specifies information about the users that own files on the server.

Owner_Global_DN

Specifies the full eDirectory Distinguished Name.

Space_Used

Specifies the amount of byte space in use.

File_Count

Specifies the number of files.

File_Type_Inventory

Specifies information about the different types of files that are stored on the server. This element tracks only the first 2000 file types.

Extension

Specifies the space used and file count for all extensions that were not counted in the first 2000 file types. NRM tracks files that have no extensions and reports them under the No Extension name.

File_Modified_Inventory

Specifies information about when the files were last modified.

Group

Specifies the data about the number of files and associated space that were fall within the specified range.

File_Access_Inventory

Specifies information about when the files were last accessed.

File_Create_Inventory

Specifies information about when the files were created.

File_Size_Inventory

Specifies information about the size of the files.

Attributes

name

Specifies the name of the server.

Local_ID

Specifies the object ID (valid on the local server only).

range

Specifies the range to return information about:

Within Last Day
1 Day - 1 Week
1 Week - 2 Weeks
2 Weeks - 1 Month
1 Month - 2 Months
2 Months - 4 Months
4 Months - 6 Months
6 Months - 1 Year
1 Year - 2 Years
More than 2 Years

OR

Less than 1KB
1 KB - 4 KB
4 KB - 16KB
16 KB - 64 KB
64 KB - 256 KB
256 KB - 1 MB
1 MB - 4 MB

4 MB - 16 MB
16 MB - 64 MB
64 MB - 256 MB
More than 256 MB

Remarks

Reading the preceding file causes NRM to generate an inventory for each volume and then combine each volume's inventory into a server inventory. By generating a server inventory, you cause XML inventory files to be generated for each volume (see Volume_Inventory.xml and Volume_Trustees.xml). However, if an inventory was generated in the last hour, that inventory is used (rather than generating a new inventory). With this functionality, you can start an inventory and come back later to process that inventory's data.

When you first read the data, the Inventory_Status tags might indicate “Scanning,” which tells you that NRM is currently generating the inventory. Keep trying until you receive a “Done” status. By processing inventories in this manner, you can asynchronously inventory multiple servers without having to wait for each inventory to complete (and without blocking any threads).

Note that integer values are in base 10.

Volume_Inventory.xml

Contains information about the files located on the specified volume.

Syntax

```
<Volume_Inventory name=""/>
<Space_Used/>
<Directory_Count/>
<File_Count/>
<File_Type_Count/>
<File_Owner_Inventory>
   <Owner Global_DN="" Local_ID="">
      <Space_Used/>
      <File_Count/>
   </Owner>
</File_Owner_Inventory>
<File_Type_Inventory>
   <Extension name="">
      <Space_Used/>
      <File_Count/>
   </Extension>
</File_Type_Inventory>
<File_Modified_Inventory>
   <Group range="">
      <Space_Used/>
      <File_Count/>
   </Group>
</File_Modified_Inventory>
<File_Access_Inventory>
   <Group range="">
      <Space_Used/>
      <File_Count/>
   </Group>
</File_Access_Inventory>
<File_Create_Inventory>
   <Group range="">
      <Space_Used/>
      <File_Count/>
   </Group>
</File_Create_Inventory>
<File_Size_Inventory>
   <Group range="">
      <Space_Used/>
      <File_Count/>
   </Group>
</File_Size_Inventory>
```

Elements

Volume_Inventory

Specifies the root element for NRM inventory requests.

Space_Used

Specifies the byte space in use by all files on the volume.

Directory_Count

Specifies the number of subdirectories on the volume.

File_Count

Specifies the number of files on the volume.

File_Type_Count

Specifies the number of file types that are being tracked in the inventory. If the Inventory_Status is Scanning, it specifies the number of files that have been scanned thus far.

File_Owner_Inventory

Specifies information about the users that own files on the volume.

Owner_Global_DN

Specifies the full eDirectory Distinguished Name.

Space_Used

Specifies the amount of byte space in use.

File_Count

Specifies the number of files.

File_Type_Inventory

Specifies information about the different types of files that are stored on the server. This element tracks only the first 2000 file types.

Extension

Specifies the space used and file count for all extensions that were not counted in the first 2000 file types. NRM tracks files that have no extensions and reports them under the No Extension name.

File_Modified_Inventory

Specifies information about when the files were last modified.

Group

Specifies the data about the number of files and associated space that were fall within the specified range.

File_Access_Inventory

Specifies information about when the files were last accessed.

File_Create_Inventory

Specifies information about when the files were created.

File_Size_Inventory

Specifies information about the size of the files.

Attributes

name

Specifies the name of the server.

Local_ID

Specifies the object ID (valid on the local server only).

range

Specifies the range to return information about:

Within Last Day
1 Day - 1 Week

1 Week - 2 Weeks
2 Weeks - 1 Month
1 Month - 2 Months
2 Months - 4 Months
4 Months - 6 Months
6 Months - 1 Year
1 Year - 2 Years
More than 2 Years

OR

Less than 1KB
1 KB - 4 KB
4 KB - 16KB
16 KB - 64 KB
64 KB - 256 KB
256 KB - 1 MB
1 MB - 4 MB
4 MB - 16 MB
16 MB - 64 MB
64 MB - 256 MB
More than 256 MB

Remarks

This data file is placed at the root of a volume after an inventory is performed. The file contains data that was created the last time a volume or server inventory was performed.

Volume_Trustees.xml

Contains information about the trustee assignments to file and directories on the specified volume.

Syntax

```
<Volume_Trustee_Report name=""> <!-- Root element. The root attribute is the
volume's name. -->
   <Trustee_List type="">    <!-- The type attribute is either
                               "subdirectory" or "file". A Trustee_List
                              can contain one or more User elements. -->
      <Path>                 <!-- The full path to the entry from the
                             root of the volume. -->
      <User rights="">  <!-- A character string with the following
                        posibilities: SRWCEMFA. Each position can
                        contain an underline or a letter for the right
                         to be assigned, according to the following list:
                                    S Supervisor
                                    R Read Files
                                    W Write Files
                                    C Create Entries
                                    E Delete Entries
                                    M Modify Entries
                                    F File Scan
                                    A Access Control      -->
      </trustee>
   </trustees>
```

Elements

Volume_Trustee_Report

Specifies the root element.

Trustee_List

Specifies information about each user.

Path

Specifies the full path to the entry from the root of the volume.

User

Specifies a character string with the following possibilities: SRWCEMFA. Each position contains an underline or a letter that represents the rights to be assigned.

Attributes

name

Specifies the volume's name.

type

Specifies the type of the trustee list:

subdirectory
file

rights

Specifies the rights to be assigned:

S Supervisor
R Read Files
W Write Files
C Create Entries
E Delete Entries
M Modify Entries
F File Scan
A Access Control

Remarks

This data is placed at the root of a volume after an inventory is performed. The file contains data that was created the last time a volume or server inventory was performed.

9 Archive Definitions

NetWare® 6.5 Archive and Versioning Service (ArkManager) uses the following command files to handle archived files:

- archiveAdmin.cmd (see Section 9.1, “archiveAdmin.cmd Definitions,” on page 408)
- archive.cmd (see Section 9.2, “archive.cmd Definitions,” on page 426)

9.1 archiveAdmin.cmd Definitions

This section contains the following archive and versioning commands:

- “activateJob” on page 409
- “deactivateJob” on page 410
- “getInfo” on page 411
- “getJobInfo” on page 413
- “getLogTimeRange” on page 416
- “listJobNames” on page 417
- “queryLog” on page 418
- “setInfo” on page 421
- “startJob” on page 423
- “stopJob” on page 424
- “testFilter” on page 425

Every command is wrapped with either archiveAdminRequest or archiveAdminReply elements, as shown in the following examples:

```
<archiveAdminRequest>
   <arkConfigInfo>
      <getInfo type=" ">
         <jobName/>
      </getInfo>
   </arkConfigInfo>
</archiveAdminRequest>

<archiveAdminReply>
   <arkConfigInfo>
      <getInfo>
         <arkConfig>
            <basic/>
         </arkConfig>
      </getInfo>
   </arkConfigInfo >
   <result value="">
      <description/>
   </result>
</archiveAdminReply>
```

activateJob

Activates a job.

Request

```
<jobControl>
   <activateJob>
      <name/>
   </activateJob>
</jobControl>
```

Reply

```
<jobControl>
   <activateJob>
     <result value=" ">
         <description/>
      </result>
   </activateJob>
</jobControl>
```

Elements

name

(Required) Specifies the job name.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

deactivateJob

Deactivates a job.

Request

```
<jobControl>
   <deactivateJob>
      <name/>
   </deactivateJob>
</jobControl>
```

Reply

```
<jobControl>
   <deactivateJob>
     <result value=" ">
         <description/>
      </result>
   </deactivateJob>
</jobControl>
```

Elements

name

(Required) Specifies the job name.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

getInfo

Retrieves ArkManager's overall information.

Request

```
<arkConfigInfo>
   <getInfo type=" ">
      <jobName/>
   </getInfo>
</arkConfigInfo>
```

Reply

```
<arkConfigInfo>
   <getInfo>
      <arkConfig>
         <basic/>
         <defaults/>
         <job/>
      </arkConfig>
   </getInfo>
</arkConfigInfo>
<result value="">
   <description/>
</result>
```

Elements

jobName

(Optional) Specifies the name of the job. You can pass the word “defaults” to this element. Multiple jobName elements are acceptable. If no jobName is specified, all arkConfig information (including information defined by the basic element) is returned.

basic

Specifies that jobName wasn't specified in the request, so all basic information is returned.

defaults

Specifies that defaults was passed as the value to jobName in the request, so all default information is returned.

job

(Repeating) Specifies the job information for each requested job.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

Attributes

type

(Optional) Specifies the type of information to return: full or simple. The default value is full. If full is specified, information defined by the job and defaults elements is returned. If simple is specified, only information defined by the job element is returned.

getJobInfo

Retrieves ArkManager's job control-related information (such as activate, deactivate, start or stop a job).

Request

```
<jobControl>
   <getJobInfo/>
</jobControl>
```

Reply

```
<jobControl>
   <getJobInfo>
      <job>
         <name/>
         <state>
            <running/>
            <scheduled/>
            <stopped/>
         </state>
         <lastStartDate/>
         <nextStartDate/>
         <scheduledInterval/>
            <dayOfWeek>
               <days>
                  <monday/>
                  <tuesday/>
                  <wednesday/>
                  <thursday/>
                  <friday/>
                  <saturday/>
                  <sunday/>
               </days>
               <time>
            </dayOfWeek>
            <interval>
               <unit>
                  <seconds/>
                  <minutes/>
                  <hours/>
                  <days/>
               </unit>
               <value/>
            </interval>
         </scheduledInterval>
         <srcServer/>
         <srcVol/>
      </job>
      <result value=" ">
         <description/>
      </result>
   </getJobInfo>
</jobControl>
```

Elements

job

(Required) Specifies the job. Repeat for each defined job.

name

(Required) Specifies the name of the job.

state

(Required) Specifies the job state from the last time it ran. Either running, scheduled, or stopped must be specified.

running

(Optional) Specifies that the job state is running.

scheduled

(Optional) Specifies that the job is scheduled to run in the future.

stopped

(Optional) Specifies that the job stopped.

lastStartDate

(Required) Specifies the generalized time, in YYYYMMDDHHMMSS format, of the last time that the job ran. If this date is unknown, zeroes are passed back (00000000000000).

nextStartDate

(Required) Specifies the generalized time, in YYYYMMDDHHMMSS format, of the next start time that the job runs. If this date is unknown, zeroes are passed back (00000000000000).

scheduledInterval

(Required) Specifies either the dayOfWeek element or the interval element.

dayOfWeek

(Optional) Specifies the day of the week and time of day that the job is scheduled to run.

days

(Optional) Specifies the days of the week.

monday

(Optional) Specifies that the job is scheduled to run on Monday.

tuesday

(Optional) Specifies that the job is scheduled to run on Tuesday.

wednesday

(Optional) Specifies that the job is scheduled to run on Wednesday.

thursday

(Optional) Specifies that the job is scheduled to run on Thursday.

friday

(Optional) Specifies that the job is scheduled to run on Friday.

saturday

(Optional) Specifies that the job is scheduled to run on Saturday.

sunday

(Optional) Specifies that the job is scheduled to run on Sunday.

time

(Optional) Specifies the time, in HHMMSS format, of the next scheduled time that the job runs.

interval

(Optional) Specifies the unit interval when the job runs.

unit

(Optional) Specifies the unit in seconds, minutes, hours, and days. If interval is defined, unit contains one of these elements.

seconds

(Optional) Specifies the seconds of the unit interval.

minutes

(Optional) Specifies the minutes of the unit interval.

hours

(Optional) Specifies the hours of the unit interval.

days

(Optional) Specifies the days of the unit interval.

value

(Optional) Specifies a number.

srcServer

(Required) Specifies the source server that the job is backing up.

srcVol

(Required) Specifies the source volume that the job is backing up.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

getLogTimeRange

Returns the range of log time.

Request

```
<archiveLog>
   <getLogTimeRange/>
</archiveLog>
```

Reply

```
<archiveLog>
   <getLogTimeRange>
      <newestTime/>
      <oldestTiime/>
      <result value=" ">
         <description/>
      </result>
   </getLogTimeRange>
</archiveLog>
```

Elements

newestTime

(Required) Specifies the generalized time, in YYYYMMDDHHMMSS format, for the newest log time.

oldestTime

(Required) Specifies the generalized time, in YYYYMMDDHHMMSS format, for the oldest log time.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

listJobNames

Returns a list of job names.

Request

```
<archiveLog>
   <listJobNames/>
</archiveLog>
```

Reply

```
<archiveLog>
   <listJobNames>
      <jobNames>
         <name/>
      </jobNames>
      <result value=" ">
         <description/>
      </result>
   </listJobNames>
</archiveLog>
```

Elements

name

Specifies the job name. Repeat for each job.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

queryLog

Queries log entries by specifying the date, job name, or severity.

Request

```
<archiveLog>
   <queryLog>
      <jobName>
         <name/>
      </jobName>
      <severity>
         <normal/>
         <warning/>
         <error/>
      </severity>
      <direction>
         <older/>
         <newer/>
      </direction>
      <numOfEntries>
      <startHere>
         <oldest/>
         <newest/>
         <cookie/>
         <startDate/>
      </startHere>
   </queryLog>
</archiveLog>
```

Reply

```
<archiveLog>
   <queryLog>
      <logInfo>
         <date/>
         <jobName>
            <name/>
         </jobName>
         <severity>
            <normal/>
            <warning/>
            <error/>
         </severity>
         <message/>
      </logInfo>
      <startHere>
         <cookie/>
      </startHere>
      <result value=" ">
         <description/>
      </result>
   </queryLog>
</archiveLog>
```

Elements

jobName

(Optional) Specifies the name of the job. If this element does not exist, all jobs are returned.

name

Specifies the name of the job. Repeat for each selected job.

severity

(Required) Specifies one or more of the severity tags (normal, warning, and error).

normal

(Optional) Specifies a severity of normal.

warning

(Optional) Specifies a severity of warning.

error

(Optional) Specifies a severity of error.

direction

(Optional) Specifies the direction of log entries. If this element does not exist, an older direction is assumed.

older

(Optional) Specifies to return older log entries.

newer

(Optional) Specifies to return newer log entries.

numOfEntries

(Required) Specifies how many entries to return.

startHere

(Optional) Specifies oldest, newest, the cookie, or a startDate. If this element does not exist, newest is assumed.

oldest

(Optional) Specifies to return the oldest log entries.

newest

(Optional) Specifies to return the newest log entries.

cookie

(Optional) Specifies the cookie that was returned the last time queryLog was called.

startDate

(Optional) Specifies the date and time, in generalized time YYYYMMDDHHMMSS format, at which to start the log entries.

logInfo

Specifies the log information. Repeat for each log.

date

Specifies the date, in generalized time YYYYMMDDHHMMSS format.

message

Specifies the log message.

startHere

(Required) Specifies the cookie element.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

setInfo

Modifies ArkManager's overall information.

Request

```
<arkConfigInfo>
   <setInfo>
      <arkConfig>
         <basic/>
         <defaults jobStatus=" "/>
         <job jobStatus=" "/>
      </arkConfig>
   </setInfo>
</arkConfigInfo>
```

Reply

```
<arkConfigInfo>
   <setInfo>
</arkConfigInfo >
<result value=" ">
   <description/>
</result>
```

Elements

arkConfig

Specifies the archive configuration information to set.

basic

(Optional) Specifies to set all basic configuration information.

defaults

(Optional) Specifies how to change the configuration information. Whenever defaults is modified, all jobs are modified accordingly.

job

(Optional) Specifies the job information for each requested job.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

Attributes

jobStatus

(Optional) Specifies how to change the configuration information:

add
modify

delete

The default value is add. To add configuration information, no job with the same name should exist. To modify and delete configuration information, a job with the same name should exist.

startJob

Starts a job.

Request

```
<jobControl>
   <startJob>
      <name/>
      <now/>
      <copyAll/>
   </startJob>
</jobControl>
```

Reply

```
<jobControl>
   <startJob>
     <result value=" ">
         <description/>
      </result>
   </startJob>
</jobControl>
```

Elements

name

(Required) Specifies the job name.

now

(Optional) Specifies that the job starts now. If this element does not exist, the job starts at the next scheduled time.

copyAll

(Optional) Specifies that all of the files are copied. If this element does not exist, only modified files are copied.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

stopJob

Stops a job.

Request

```
<jobControl>
   <stopJob>
      <name/>
   </stopJob>
</jobControl>
```

Reply

```
<jobControl>
   <stopJob>
     <result value=" ">
         <description/>
      </result>
   </stopJob>
</jobControl>
```

Elements

name

(Required) Specifies the job name.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

testFilter

Tests whether a particular path passes a job filter's definition (and will be archived).

Request

```
<testFilter>
   <jobName/>
   <path/>
</testFilter>
```

Reply

```
<testFilter>
   <pass|fail/>
</testFilter>
```

Elements

jobName

Specifies the name of the job to be tested against.

path

Specifies the test path.

pass|fail

Specifies whether the path would pass the job's filter.

9.2 archive.cmd Definitions

Starting from NetWare 6.5 SP1, archive data is stored in a MySQL database. The following commands support multiple jobs on the same volume:

- “deleteFile” on page 427
- “getContentVersions” on page 429
- “getDirContents” on page 432
- “getVersions” on page 434
- “restoreFile” on page 437
- “shutdown” on page 439

Multiple commands can be combined inside of one archiveRequest element.

All paths are root based and separated by a forward slash (/) unless otherwise specified for a particular element.

deleteFile

Deletes a file and all of its archived content versions from the archive server.

Request

```
<archiveRequest version="2.0">
   <deleteFile>
      <serverIPAddress/>
      <volume/>
      <fileName/>
      <archiveInfo>
         <job>
            <jobName/>
            <fileKey/>
            <metaDataKey/>
         </job>
      </archiveInfo>
   </deleteFile>
</archiveRequest>
```

Reply

```
<archiveReply version="2.0">
   <deleteFile>
      <result value=" ">
         <description/>
      </result>
   </deleteFile>
</archiveReply>
```

Elements

serverIPAddress

Specifies the server where the original file came from.

volume

Specifies the volume where the original file came from.

fileName

(Optional) Specifies the full path of the name of the file. Separate the path with forward slashes (/), starting from the volume root. If archiveInfo isn't specified, the full path is evaluated. Component names with only a current status are used when parsing the path. If archiveInfo is specified, fileName can be omitted.

If fileName specifies a directory, all entries contained in the directory are deleted, except for subfiles and subdirectories renamed to entries. If fileName specifies a file and metaDataKey or archiveInfo isn't specified, all content versions are deleted. If fileName specifies a file and at least one metaDataKey is specified, only the content versions specified in metaDataKey are deleted.

archiveInfo

(Optional) Specifies the information returned from a previous query.

job

Specifies the job. Repeat for each job.

jobName

Specifies the name of the job as received from a directory content query or a version query.

fileKey

Specifies the file key as received from a directory content query or a version query.

metaDataKey

(Optional) Specifies the content version. Repeat for each target. If the target is a file, the specific content version specified by this value is deleted.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

getContentVersions

Returns archive file versions.

Request

```
<archiveRequest version="2.0">
   <getContentVersions maxReturnEntries=" ">
      <serverIPAddress/>
      <volume/>
      <fileName/>
      <archiveInfo>
         <job>
            <jobName/>
            <fileKey/>
         </job>
      </archiveInfo>
      <startDate/>
      <endDate/>
      <startHere/>
   </getContentVersions>
</archiveRequest>
```

Reply

```
<archiveReply version="2.0">
   <getContentVersions>
      <fileVersion>
         <name/>
         <date/>
         <size/>
         <modifyTime/>
         <modifier/>
         <archiveName>
            <actualName/>
         </archiveName>
         <archiveInfo>
            <server/>
            <volume/>
            <job>
               <jobName/>
               <fileKey/>
               <metaDataKey/>
            </job>
         </archiveInfo>
      </fileVersion>
      <startHere/>
      <ipAddress/>
   </getContentVersions>
   <result value=" ">
      <description/>
   </result>
</archiveReply>
```

Elements

serverIPAddress

Specifies the server where the original file came from.

volume

Specifies the volume where the original file came from.

fileName

(Optional if archiveInfo is specified) Specifies the full path of the file name to return versions for. The path should be separated by forward slashes (/), starting from the volume root. If archiveInfo isn't specified, the full path is evaluated. Only component names with a current status are used when parsing the path.

archiveInfo

(Optional) Specifies the archived information received from a previous query.

job

Specifies the job. Repeat for each job.

jobName

Specifies the name of the job as received from a directory content query or a version query.

fileKey

Specifies the file key as received from a directory content query or a version query.

startDate

(Optional) Specifies the most recent date, in generalized time and date YYYYMMDDHHMMSS format, to return.

endDate

(Optional) Specifies the oldest date, in generalized time and date YYYYMMDDHHMMSS format, to return.

startHere

(Optional) Specifies where to start as returned from a previous query. If this element does not exist, the search starts at the beginning.

fileVersion

Specifies the version of the file. Repeated for each file version entry.

date

Specifies the date, in generalized time and date YYYYMMDDHHMMSS format, the file was archived.

modifyTime

Specifies the modified time, in generalized time and date YYYYMMDDHHMMSS format, from the file's metadata.

actualName

Specifies an opaque name from which the file can be directly accessed.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

Attributes

maxReturnEntries

(Optional) Specifies the maximum number of entries to return. If this attribute is not specified, the request handler decides how many entries to return.

getDirContents

Returns archived directory contents.

Request

```
<archiveRequest version="2.0">
   <getDirContents maxReturnEntries=" ">
      <serverIPAddress/>
      <volume/>
      <dirName/>
      <archiveInfo>
         <job>
            <jobName/>
            <fileKey/>
         </job>
      </archiveInfo>
      <startHere/>
      <ipAddress/>
   </getDirContents>
</archiveRequest>
```

Reply

```
<archiveReply version="2.0">
   <getDirContents>
      <dirName>
         <name/>
         <archiveInfo>
            <job>
               <jobName/>
               <fileKey/>
            </job>
         </archiveInfo>
      </dirName>
      <startHere/>
   </getDirContents>
   <result value=" ">
      <description/>
   </result>
</archiveReply>
```

Elements

serverIPAddress

Specifies the server where the original file came from.

volume

Specifies the volume where the original file came from.

dirName

(Optional if archiveInfo is specified) Specifies the full path of the directory name to return versions for. The path should be separated by forward slashes (/), starting from the volume root. If archiveInfo isn't specified, the full path is evaluated. Only component names with a current status are used when parsing the path.

archiveInfo

(Optional) Specifies the archived information received from a previous query.

job

Specifies the job. Repeat for each job.

jobName

Specifies the name of the job as received from a directory content query or a version query.

fileKey

Specifies the file key as received from a directory content query or a version query.

startHere

(Optional) Specifies where to start as returned from a previous query. If this element does not exist, the search starts at the beginning.

dirName

Specifies the directory element. Repeat for each entry.

name

Specifies the entry name.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

Attributes

maxReturnEntries

(Optional) Specifies the maximum number of entries to return. If this attribute is not specified, the request handler decides how many entries to return.

type

Specifies the type of object: directory, file, or unknown. If type is unknown, call getVersions (page 434) to retrieve more information. The unknown designation is used only if the file status is current and there's no other duplicated name.

getVersions

Returns archived directory versions.

Request

```
<archiveRequest version="2.0">
   <getVersions maxReturnEntries=" ">
      <serverIPAddress/>
      <volume/>
      <dirName/>
      <archiveInfo>
         <job>
            <jobName/>
            <fileKey/>
         </job>
      </archiveInfo>
      <startHere/>
   </getVersions>
</archiveRequest>
```

Reply

```
<archiveReply version="2.0">
   <getVersions>
      <dirName>
         <name/>
         <status>
            <current/>
            <renamed/>
            <deleted/>
            <changeTime/>
         </status>
         <currentName/>
         <archiveInfo>
            <job>
               <jobName/>
               <fileKey/>
            </job>
         </archiveInfo>
      </dirName>
      <startHere/>
      <ipAddress/>
   </getVersions>
   <result value=" ">
      <description/>
   </result>
</archiveReply>
```

Elements

serverIPAddress

Specifies the server where the original file came from.

volume

Specifies the volume where the original file came from.

dirName

(Optional if archiveInfo is specified) Specifies the full path of the directory name to return versions for. The path should be separated by forward slashes (/), starting from the volume root. Components with only a current status are used when parsing the path. If both archiveInfo and

dirName are specified, the last component name in dirName is used to get versions under the parent directory that is specified in archiveInfo. If only archiveInfo is specified, the object specified by fileKey is used to get versions.

archiveInfo

(Optional) Specifies the archived information received from a previous query.

job

Specifies the job. Repeat for each job.

jobName

Specifies the name of the job as received from a directory content query or a version query.

fileKey

Specifies the file key as received from a directory content query or a version query.

startHere

(Optional) Specifies where to start as returned from a previous query. If this element does not exist, the search starts at the beginning.

dirName

Specifies the directory element. Repeat for each entry.

name

Specifies the entry name.

status

Specifies the status of the file or directory: current, renamed, or deleted.

current

Specifies that the file or directory is current.

renamed

Specifies that the file or directory is renamed.

deleted

Specifies that the file or directory is deleted.

changeTime

(Optional) Specifies the time, in YYYYMMDDHHMMSS format, when the status is changed. Returned if the status is renamed or deleted.

currentName

Specifies the current name for the file or directory specified by dirName. This element is returned if the status is renamed. The currentName is different from the name specified by dirName.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

Attributes

maxReturnEntries

(Optional) Specifies the maximum number of entries to return. If this attribute is not specified, the request handler decides how many entries to return.

type

Specifies the type of object: directory or file.

restoreFile

Restores a file from the archive server to another server.

Request

```
<archiveRequest version="2.0">
   <restoreFile>
      <source>
         <fileName>
            <actualName/>
         </fileName>
         <archiveInfo>
            <server/>
            <volume/>
            <job>
               <jobName/>
               <fileKey/>
               <metaDataKey/>
            </job>
         </archiveInfo>
      </source>
      <destination>
         <serverIPAddress/>
         <fileName/>
         <createDirs/>
         <overwrite/>
      </destination>
   </restoreFile>
</archiveRequest>
```

Reply

```
<archiveReply version="2.0">
   <restoreFile>
      <result value=" ">
         <description/>
      </result>
   </restoreFile>
</archiveReply>
```

Elements

actualName

Specifies the name, in the format sent from the archive server, of the file. It's the name received from getContentVersions (page 429).

archiveInfo

(Optional) Specifies the archived information received from a previous query.

job

Specifies the job. Repeat for each job.

jobName

Specifies the name of the job as received from a directory content query or a version query.

fileKey

Specifies the file key as received from a directory content query or a version query.

fileName

Specifies the name of the file, including the volume name. Use forward slashes as separators.

createDirs

(Optional) Specifies that the destination directories should be created if they don't yet exist.

overwrite

(Optional) Specifies that the destination file should be overwritten if it exists.

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

shutdown

Shuts down the ArkManager process and stops all jobs.

Request

```
<archiveRequest>
   <control>
      <shutdown/>
   </control>
</archiveRequest>
```

Reply

```
<archiveReply>
   <control>
      <result value=" ">
         <description/>
      </result>
   </control>
</archiveReply>
```

Elements

result

Specifies an error value or zero (for no error).

description

Specifies a text description of the returned result.

10 linux.cmd Definitions

Linux uses thefollowing commands that are defined in the /_admin/Manage_NSS/linux.cmd file:

- “poolIDToName” on page 442
- “uidToEquivalentGUIDs” on page 443
- “userIDToName” on page 444
- “volumeIDFileIDToPath” on page 445
- “volumeIDToName” on page 446
- “addPool” on page 447
- “addVolume” on page 448
- “poolUpdate” on page 449
- “getVolumeInfo” on page 450
- “resetIDs” on page 451

Every command is wrapped with either linxRequest or linuxReply elements, as shown in the following examples:

```
<linuxRequest>
   <storage>
      <volumeIDFileIDToPath>
         <volumeID>aaaa-bbbb-cccc-dd-ee-nnnnnn</volumeID>
         <fileID>
      </volumeIDFileIDToPath>
   </storage>
</linuxRequest>
```

```
<linuxReply>
   <storage>
      <volumeIDFileIDToPath>
         <volumeName>POOLNAME</volumeName>
         <path/>
         <result value="0">
            <description/>success</description>
         </result>
      </volumeIDFileIDToPath>
   </storage>
   <result value="0">
      <description/>zOK</description>
   </result>
</linuxReply>
```

poolIDToName

Returns a pool name from a pool GUID (for Linux only).

Request

```
<linuxRequest>
   <storage>
      <poolIDToName>
         <poolID/>
      </poolIDToName>
   </storage>
</linuxRequest>
```

Reply

```
<linuxReply>
   <storage>
      <poolIDToName>
         <poolName/>
         <result value=" ">
            <description/>
         </result>
      </poolIDToName>
   </storage>
   <result value=" ">
      <description/>
   </result>
</linuxReply>
```

Elements

poolID

Specifies the ID of the pool in the following format:

```
aaaaaaaa-bbbb-cccc-dd-ee-nnnnnnnn
```

poolName

Specifies the name of the pool.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

uidToEquivalentGUIDs

Returns the eDirectory GUIDs for the equivalent users and groups of a Linux UID.

Request

```
<linuxRequest>
   <storage>
      <uidToEquivalentGUIDs>
         <uid/>
      </uidToEquivalentGUIDs>
   </storage>
</linuxRequest>
```

Reply

```
<linuxReply>
   <storage>
      <uidToEquivalentGUIDs>
         <equivalentID/>
            <userID/>
            <userID/>
            . . .
         </equivalentID>
         <result value=" ">
            <description/>
         </result>
      </uidToEquivalentGUIDs>
   </storage>
   <result value=" ">
      <description/>
   </result>
</linuxReply>
```

Elements

uid

Specifies the UID that you want the equivalent eDirectory GUID for.

userID

Returns the equivalent GUID. For example

```
<userID>f0624d95-5578-4b71-7c-af-954d62f07855</userID>
<userID>f8855fe2-860e-4bbf-a2-96-e25f85f80e86</userID>
```

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

userIDToName

Returns a user name from a user GUID (for Linux only).

Request

```
<linuxRequest>
   <storage>
      <userIDToName>
         <userID/>
      </userIDToName>
   </storage>
</linuxRequest>
```

Reply

```
<linuxReply>
   <storage>
      <userIDToName>
         <userName/>
         <result value=" ">
            <description/>
         </result>
      </userIDToName>
   </storage>
   <result value=" ">
      <description/>
   </result>
</linuxReply>
```

Elements

userID

Specifies the ID of the user in the following format:

```
aaaaaaaa-bbbb-cccc-dd-ee-nnnnnnnn
```

userName

Specifies the name of the user.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

volumeIDFileIDToPath

Returns a path from a volume ID to a zID.

Request

```
<linuxRequest>
   <storage>
      <volumeIDFileIDToPath>
         <volumeID/>
         <fileID/>
      </volumeIDFileIDToPath>
   </storage>
</linuxRequest>
```

Reply

```
<linuxReply>
   <storage>
      <volumeIDFileIDToPath>
         <volumeName/>
         <path/>
         <result value="">
            <description/>
         </result>
      </volumeIDFileIDToPath>
   </storage>
   <result value="">
      <description/>
   </result>
</linuxReply>
```

Elements

volumeID

Specifies the ID of the volume in the following format:

```
aaaaaaaa-bbbb-cccc-dd-ee-nnnnnnnn
```

fileID

Specifies the ID (zID) of a file on the volume.

volumeName

Specifies the name of the volume.

path

Specifies a Linux namespace path.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

volumeIDToName

Returns a volume name from a volume GUID (for Linux only).

Request

```
<linuxRequest>
   <storage>
      <volumeIDToName>
         <volumeID/>
      </volumeIDToName>
   </storage>
</linuxRequest>
```

Reply

```
<linuxReply>
   <storage>
      <volumeIDToName>
         <volumeName/>
         <result value=" ">
            <description/>
         </result>
      </volumeIDToName>
   </storage>
   <result value=" ">
      <description/>
   </result>
</linuxReply>
```

Elements

volumeID

Specifies the ID of the volume in the following format:

```
aaaaaaaa-bbbb-cccc-dd-ee-nnnnnnnn
```

volumeName

Specifies the name of the volume.

result

Specifies an error value or 0 (for no error).

description

Specifies a text description of the result.

addPool

Creates a pool on the Linux device.

Request

```
  <linuxRequest>
     <storage>
        <addPool>
           <poolName/>
           <device/>
        </addPool>
     </storage>
  </linuxRequest>
```

Reply

```
<linuxReply>
   <storage>
     <addPool>
        <result value="0"><description>success</description></result>
     </addPool>
   </storage>
<result value="0"><description>zOK</description></result>
</linuxReply>
```

Elements

poolName

Specifies the name to be given to the new pool.

device

A pool is stored on this specified device.

result

Specifies an error value or 0 (for no error).

It does not create the NDS object, hence create the object using *Update NDS* provided by the NSS utilities.

addVolume

Adds a NSS volume to a pool.

Request

```
<linuxRequest>
     <storage>
        <addVolume>
           <volumeName/>
           <poolName/>
        </addVolume>
     </storage>
</linuxRequest>
```

Reply

```
<linuxReply>
     <storage>
        <addVolume>
        <result value="0"><description>success</description></result>
        </addVolume>
     </storage>
<result value="0"><description>zOK</description></result>
</linuxReply>
```

Elements

volumeName

Specifies the name of the volume to create.

poolName

Specifies the name of the NSS storage pool on which the volume is created.

result

Specifies an error value or 0 (for no error).

poolUpdate

Updates the pool information such as pool size and state of the pool (shared or non-shared).

Request

```
<linuxRequest>
      <storage>
         <poolUpdate>
            <poolName/>
            <size/>
            <shared/>
            <notshared/>
         </poolUpdate>
       </storage>
</linuxRequest>
```

Reply

```
<linuxReply>
      <storage>
         <poolUpdate>
           <result value="0"><description>success</description></result>
         </poolUpdate>
      </storage>
<result value="0"><description>zOK</description></result>
</linuxReply>
```

Elements

poolName

Specifies the name of the pool.

shared

Specifies the state of the pool (shared).

notshared

Specifies the state of the pool (non-shared).

result

Specifies an error value or 0 (for no error).

getVolumeInfo

Returns detailed information about an existing NSS volume.

Request

```
<linuxRequest>
     <storage>
        <getVolumeInfo>
           <volumeName/>
        </getVolumeInfo>
     </storage>
</linuxRequest>
```

Reply

```
<linuxReply>
     <storage>
        <nameSpace/>
        <getVolumeInfo>
          <result value="0"><description>success</description></result>
        </getVolumeInfo>
     </storage>
<result value="0"><description>zOK</description></result>
</linuxReply>
```

Elements

volumeName

Specifies the name of the volume to return information for.

result

Specifies an error value or 0 (for no error).

resetIDs

Resets the ID caches.

Entries

- Invalidate all entries in the GUID to ID cache
- Invalidate entries in the ID to GUID cache

Request

```
<linuxRequest>
     <storage>
        <resetIDs>
     </storage>
</linuxRequest>
```

Reply

```
<linuxReply>
     <storage>
     </storage>
<result value="0"><description>zOK</description></result>
</linuxReply>
```

Elements

result

Specifies an error value or 0 (for no error).

11 Advanced Concepts

In addition to the predefined commands, Virtual File Services (VFS) can also be used to implement much more complex interfaces by allowing write operations to be sent to user-defined functions and then returning the results of those functions to subsequent read operations. VFS allows any program that can handle file I/O to interact with functions in the file system, which allows you to implement your own interfaces using any tool that has file system access.

For example, scripting languages (like PERL) can be used to manage system functionality using standard file system functions. The current NSS ConsoleOne interface is implemented as commands written to the manage.cmd and nds.cmd command files in the admin volume.

This section covers the following concepts:

- Section 11.1, “Transformation Templates,” on page 453
- Section 11.2, “Virtual I/O Commands,” on page 456

11.1 Transformation Templates

Transformation Templates are the instructions that set up the behavior definition for a virtual file. They are contained in XML and describe how the information on the system (such as the contents of a memory location) is transformed for read and write file operations.

A transformation template can also identify multiple datastreams (see “Datastreams” on page 455) for the virtual file. If a datastream has no name, it is used to satisfy requests that are not directed to a specific datastream.

All virtual files require a transformation template to function correctly. No read or write operations on a file will work until a virtual I/O command has been written that defines the behavior of the file to be accessed.

To define a virtual I/O command, write the following to the file followed immediately by the transformation template:

```
<virtualIO>
   <define/>
</virtualIO>
```

The template identifies each of the datastreams and how it should be rendered.

The following is the DTD for the XML that is used to define the transformation template:

```
<!ELEMENT transform (datastream+)>
<!ELEMENT datastream (data | location | function)>
<!ATTLIST datastream
   name CDATA #IMPLIED>

<!ELEMENT data (#PCDATA)>

<!ELEMENT location (readloc?, writeloc?)>
<!ATTLIST location
   symname CDATA #REQUIRED
   offset CDATA #IMPLIED>
<!ELEMENT readloc (format, leadingtext?, trailingtext?)>
<!ELEMENT writeloc (format)>
<!ELEMENT format (byte | word | long | quad | raw)>
<!ELEMENT byte EMPTY>
<!ATTLIST byte
   signed (yes | no) #IMPLIED>
<!ELEMENT word EMPTY>
<!ATTLIST word
   signed (yes | no) #IMPLIED>
<!ELEMENT long EMPTY>
<!ATTLIST long
   signed (yes | no) #IMPLIED>
<!ELEMENT quad EMPTY>
<!ATTLIST quad
   signed (yes | no) #IMPLIED>
<!ELEMENT raw EMPTY>
<!ATTLIST raw
   length CDATA #REQUIRED>
<!ELEMENT leadingtext (#PCDATA)>
<!ELEMENT trailingtext (#PCDATA)>

<!ELEMENT function (readfunc, writefunc?)>
<!ELEMENT readfunc (#PCDATA)>
<!ATTLIST readfunc
   symname CDATA #REQUIRED
   cookie (yes | no) #IMPLIED>
<!ELEMENT writefunc (#PCDATA)>
<!ATTLIST writefunc
   symname CDATA #REQUIRED>
```

The transformation tags are interpreted as follows:

transform	Main transformation template element. It contains one or more datastream elements.
datastream	Identifies and defines a datastream. The only attribute, name, is optional. If the name attribute is not used, the datastream is used as the default datastream. If more than one datastream has no name or has the same name, the first one encountered in the template is used. The name must be less than or equal to VIRT_DATASTREAM_NAME_SIZE (currently 63 bytes). The datastream element must contain a data, location, or function element.
data	Indicates that the content of the datastream is contained in the data element itself. The data element holds the data that is to be returned for a read operation or changed for a write operation. This tag is useful for defining help text for the file.
location	Defines a memory location that will be used as the target of read and write operations. The tag has one required (symname) and one optional (offset) attribute. Symname defines a symbolic name that can be dynamically imported. Offset defines a numeric value that indicates how many bytes the memory location is from the specified symname. The location element must contain a readloc element and can optionally contain a writeloc element. If no writeloc element is defined, the location cannot be written to.

readloc	Defines the format of the result and tells VFS how to interpret the data at the given memory location. It also includes any text that will be used with the data from the memory location. The format element is required. The leadingtext and trailing text elements are optional and define the text to be used both in front of and behind the value retrieved from memory, respectively.
format	Contains one of the following five elements: byte, word, long, quad, or raw. The first four of these elements define the size of the memory location being examined and imply that the result generated will be an ASCII representation of the decimal value. The raw element does not perform any conversion.
byte	Indicates that a single byte will be converted to its ASCII decimal value. The optional attribute, signed, indicates whether the conversion should be to a signed value. This attribute should be assigned either a "yes" or "no" value. If the attribute is not specified or if its value is not "yes" or "no," it default to "yes."
word	Indicates that a single word will be converted to its ASCII decimal value. The optional attribute, signed, indicates whether the conversion should be to a signed value. This attribute should be assigned either a "yes" or "no" value. If the attribute is not specified or if its value is not "yes" or "no," it default to "yes."
long	Indicates that a four-byte memory location will be converted to its ASCII decimal value. The optional attribute, signed, indicates whether the conversion should be to a signed value. This attribute should be assigned either a "yes" or "no" value. If the attribute is not specified or if its value is not "yes" or "no," it default to "yes."
quad	Indicates that an eight-byte memory location will be converted to its ASCII decimal value. The optional attribute, signed, indicates whether the conversion should be to a signed value. This attribute should be assigned either a "yes" or "no" value. If the attribute is not specified or if its value is not "yes" or "no," it default to "yes."
raw	Indicates that the unconverted value at the previously specified memory location should be used as the virtual data. The attribute, length, is required and indicates how many bytes are retrieved for a read operation and how many bytes can be written for a write operation.
leadingtext	Contains any text that should be put in the result buffer in front of the data being rendered by the formal element.
trailingtext	Contains any text that should be put in the result buffer behind the data being rendered by the formal element.
writeloc	Contains only a format element.
function	Contains a readfunc and, optionally, a writefunc element. If no writefunc element is included, you cannot write to the datastream.
readfunc	Defines a function that will be used as the target of read operations. The tag has one required attribute, symname, which gives a symbolic name that can be dynamically imported.
writefunc	Defines a function that will be used as the target of write operations. The tag has one required attribute, symname, which gives a symbolic name that can be dynamically imported.

11.1.1 Datastreams

Each datastream has one of the following types that identifies how the data for that datastream is rendered:

Data in the template

One source of virtual data is the template itself, which is useful for descriptive text. For example, one datastream in a virtual file might return the contents of the memory location that contains the size of the file cache. Another datastream in the same file could be of the data type and contain help text that describes what the file cache size represents and how changing it may affect a system's performance.

Contents of a memory location

This datastream type renders its information from a memory location on a server. The template designates where the memory location is, how long the data is, and whether is should be converted to ASCII or returned as "raw" data.

Information rendered by a function

This type of datastream specifies two functions: one for accepting data written to the file and the other to generate data to be read from the file. This functionality gives great flexibility as to the type of virtual data that can be handled since the format of the data is determined by the function that is being called.

The functions to be used must be exported. VFS imports the symbol the first time it is encountered after the virtual file is created. It remains in an imported state until the file is deleted. Note that delete is an asynchronous operation and unimporting the symbol can occur after the return from a call to delete the file. In such cases, you should check the return code from the unexport function to ensure it successfully succeeded. The function can fail if the unimport has not yet occurred in VFS.

11.2 Virtual I/O Commands

A virtual file can contain a default datastream that is read or written if a normal read or write operation is posted against the file and no virtual I/O command has been received. All other read and write operations must be preceded by a virtual I/O command, which allows the requester to target either the transformation template of the file or a named datastream that has been defined for the file.

The following is the DTD for the XML used in virtual I/O commands:

```
<!ELEMENT virtualIO (datastream)|define|link)>
<!ELEMENT datastream EMPTY>
<!ATTLIST datastream
   name CDATA "default">
<!ELEMENT define EMPTY>
<>
```

The virtual I/O tags are interpreted as follows:

virtualIO	The main element. It must contain either a datastream or a define element.
define	Used when you wish to view or change the transformation template of a file through subsequent read and write operations.
datastream	Specifies that subsequent read and write operations will be on a virtual datastream. It has one optional attribute, name, which gives the name of the target datastream. The name must correspond to the name of one of the datastreams that is defined in the transformation template of the target file. If the name is not specified, the default datastream is used (the one in the transformation template that has no name), which is the same as if there is no virtual I/O command.

12 Values

This section lists the following common values associated with Virtual File Services:

- Section 12.1, “Device Types,” on page 457
- Section 12.2, “Enabled Attributes Bits,” on page 457
- Section 12.3, “Job Types,” on page 458
- Section 12.4, “Mirror Group Statuses,” on page 458
- Section 12.5, “NSS Volume States,” on page 458
- Section 12.6, “Pool States,” on page 459
- Section 12.7, “Pool Types,” on page 459
- Section 12.8, “State Values,” on page 459
- Section 12.9, “Traditional Volume States,” on page 460
- Section 12.10, “Volume States,” on page 461
- Section 12.11, “Volume Types,” on page 461

12.1 Device Types

The deviceType element can have the following values:

0 MM_DIRECT_ACCESS_DEVICE
1 MM_SEQUENTIAL_ACCESS_DEVICE
2 MM_PRINTER_DEVICE
3 MM_PROCESSOR_DEVICE
4 MM_WORM_DEVICE
5 MM_CD_ROM_DEVICE
6 MM_SCANNER_DEVICE
7 MM_MO_DEVICE
8 MM_MEDIA_CHANGER_DEVICE
9 MM_COMMUNICATION_DEVICE

12.2 Enabled Attributes Bits

The enabledAttributesBits element can have the following values:

zPOOL_FEATURE_PERSISTENT_FEATURES (0x01)

The pool's enabled features are stored persistently.

zPOOL_FEATURE_SHARED_CLUSTER (0x02)

The pool is part of a cluster.

zPOOL_FEATURE_READ_ONLY (0x04)

The pool is read only.

zPOOL_FEATURE_VERIFY (0x08)

The pool supports a verify operation.

zPOOL_FEATURE_REBUILD (0x10)

The pool supports a rebuild operation.

zPOOL_FEATURE_MUTLIPLE_VOLUMES (0x20)

The pool can support multiple logical volumes.

zPOOL_FEATURE_SNAPSHOT (0x40)

The pool is a snapshot of another pool.

12.3 Job Types

The jobType element can have the following values:

Move
Split
Copy
Unknown

12.4 Mirror Group Statuses

The mirrorGroupStatus element can have the following values:

MM_MIRROR_GROUP_IN_SYNC (0x00000001)

If this bit is set, the mirror group is fully synchronized.

MM_MIRROR_ALL_PRESENT (0x00000002)

If this bit is set, all partitions which belong to the mirror group are present.

MM_MIRROR_OPERATIONAL (0x00000004)

If this bit is set, the mirror group is operational.

MM_MIRROR_PARTIAL_SYNC (0x00000010)

If this bit is set, the mirror group is only partially synchronized.

MM_MIRROR_REMIRRORING (0x00000040)

If this bit is set, the mirror group is in the process of remirroring.

MM_MIRROR_OBJECT_ORPHANED (0x00000080)

If this bit is set, this mirror ID has been removed from the mirror group to which it once belonged, leaving it in an orphaned state.

12.5 NSS Volume States

For traditional volume states, see Section 12.9, “Traditional Volume States,” on page 460.

For NSS logical volumes, the volumeState element can have the following values:

Table 12-1 volumeState values for NSS logical volumes

Value	Description
mounted	Move this volume to a mounted state. If necessary, activate it as well.
dismounted	Move this volume to an active but not mounted state. If the volume is deactive, move it to an active state. The effect of this value is the same as the "active" value.
active	Move this volume to an active but not mounted state. If it is currently mounted, dismount the volume. If the volume is deactive, move it to an active state. The effect of this value is the same as the "dismounted" value.
deactive	Move this volume to a not active and not mounted state.

12.6 Pool States

The poolState element can have the following values:

0 Unknown: Pool is in an unknown state
2 Deactive: Pool is not activated
3 Maintenance: Pool is in maintenance mode
6 Active: Pool is activated

12.7 Pool Types

The type attribute of the getPoolInfo element can have the following values:

all Returns all available information
basic Returns only the basicInfo element
salvage Returns only the salvageInfo element
attributes Returns only the attributeInfo element
volumes Returns only the volumeInfo element
deletedVolumes Returns only the deletedVolumeInfo element

12.8 State Values

The state element can have the following values:

Invalid
Running
Completed
Scanning
Cancelled
Paused
Scheduled
Updating
NameSelect
Renaming
CreateJunc

Cleanup
ReplayingLog
RenameLogfile
NewLogfile
MoveTrustees
NewEFL
ReplayingEFL
Starting
Cancelling
Pausing
Suspending
RetryUpdating
RetryNameSelect
RetryRenaming
RetryCreateJunc
RetryCleanup
RetryReplay
RetryRenameLogfile
RetryNewLogfile
RetryMoveTrustees
RetryNewEFL
RetryReplayEFL
FilesSkipped
CleanupFailed
Failed
FailedFileRead
FailedFileRestore
FailedBeginBackup
FailedLogin
FailedTargetVersion
FailedNoManagementContext
FailedNotSameManagementContext
FailedLogFile
Unknown

12.9 Traditional Volume States

For NSS logical volume states, see "NSS Volume States" on page 458.

For traditional NetWare volumes, the volumeState element for the modifyState command can have the following values:

mounted Move this volume to a mounted state

dismounted Move this volume to a not mounted state

12.10 Volume States

The poolState element and volumeState element (for NSS logical volumes) can have the following values:

deactive The volume is not currently activated
active The volume is currently activated but not mounted
mounted The volume is currently activated and mounted
maintenance The volume is in need of repair or is being repaired
unknown The volume is in an unknown state

12.11 Volume Types

The type attribute of the getVolumeInfo element can have the following values:

"all" Returns all available information
"basic" Returns only the basicInfo element
"salvage" Returns only the salvageInfo element
"attributes" Returns only the attributeInfo element
"compression" Returns only the compressionInfo element
"deletedVolumes" Returns only the deletedVolumeInfo element

13 Functions

The actual file system functions used by VFS are standard create, delete, open, close, read, and write functions.

This section describes the following virtual I/O commands and transformation templates and the DTD and XML tag definitions for each command:

- “MGMT_FindFirstElement” on page 464
- “MGMT_MakeCommandVirtualFile” on page 465
- “MGMT_MakeCommandVirtualFileWithHelp” on page 466
- “MGMT_MakeFunctionVirtualFile” on page 467
- “VIRT_AddResultData” on page 468
- “VIRT_AddResultElement” on page 469
- “VIRT_AddResultTag” on page 470
- “VIRT_MakeResultsImportant” on page 471
- “VIRT_MakeResultsNormal” on page 472
- “VIRT_ResetResult” on page 473
- “XML_BackwardFindEndTag” on page 474
- “XML_findEndOfNonWhiteSpace” on page 475
- “XML_ForwardFindTag” on page 476
- “XML_GetNextTag” on page 477
- “XML_GetTagElement” on page 478

MGMT_FindFirstElement

Determines if the complete main XML element has been received in a write operation. This function builds up an internal buffer that holds the entire XML stream until the main element is closed.

Service: VFS

Syntax

```
#include <nssPubs.h>

STATUS MGMT_FindFirstElement (
   VirtInfo_s         *virtInfo,
   utf8_t             *tagName,
   NINT                bufferLength,
   BYTE               *buffer,
   NINT                offset,
   XML_ElementInfo_s  *element);
```

Parameters

virtInfo

Points to a structure that is unique for each open instance of the virtual file. It contains information about the state of the file, as well as the results buffer. Usually, you will not change the contents of this structure but simply pass it to other functions.

tagName

Points to the name of the main element of the XML that is to be parsed.

bufferLength

Specifies the length of the data that is being passed to the write function.

buffer

Points to the data that is being passed to the write function.

offset

Specifies the offset of the write.

element

Points to a structure that contains the results of the XML element parsing. It includes a pionter to the start of the element's data and to the end of the element's data.

Return Values

The return status should either be zOK or a valid NSS error code.

MGMT_MakeCommandVirtualFile

Defines a virtual file that has a write function for the default datastream. The write function generates a response that can be read from the results buffer.

Service: VFS

Syntax

```
#include <nssPubs.h>

STATUS MGMT_MakeCommandVirtualFile (
   Key_t    key,
   utf8_t  *writeRoutine,
   utf8_t  *writeParm);
```

Parameters

key

Specifies the key returned from zCreate or zOpen in File System Services (64-Bit) (http://developer.novell.com/wiki/index.php/File_System_Services_%2864-Bit%29).

writeRoutine

Points to the NULL-terminated name of the public symbol for the write function.

writeParm

Points to the string that is passed as the parm parameter to the write function.

Return Values

The return status should either be zOK or a valid NSS error code.

MGMT_MakeCommandVirtualFileWithHelp

Defines a virutal file (similar to MGMT_MakeCommandVirtualFile (page 465)) This function allows you to define a help string that is read from the file by default. The specified write function is put in a datastream named "command" and must be accessed using that datastream.

Service: VFS

Syntax

```
#include <nssPubs.h>

STATUS MGMT_MakeCommandVirtualFileWithHelp (
   Key_t    key,
   utf8_t   help,
   utf8_t  *writeRoutine,
   utf8_t  *writeParm);
```

Parameters

key

Specifies the key returned from zCreate or zOpen in File System Services (64-Bit) (http://developer.novell.com/wiki/index.php/File_System_Services_%2864-Bit%29).

help

Specifies a NULL-terminated string that is returned on the default datastream from a read operation.

writeRoutine

Points to the NULL-terminated name of the public symbol for the write function.

writeParm

Points to the string that is passed as the parm parameter to the write function.

Return Values

The return status should either be zOK or a valid NSS error code.

MGMT_MakeFunctionVirtualFile

Defines a virtual file that uses functions to satisfy read and write operations for the default datastream.

Service: VFS

Syntax

```
#include <nssPubs.h>

STATUS MGMT_MakeFunctionVirtualFile (
   Key_t    key,
   utf8_t  *readRoutine,
   utf8_t  *readParm,
   utf8_t  *writeRoutine,
   utf8_t  *writeParm,
   BOOL     withCookie);
```

Parameters

key

Specifies the key returned from zCreate or zOpen in File System Services (64-Bit) (http://developer.novell.com/wiki/index.php/File_System_Services_%2864-Bit%29).

readRoutine

Points to the NULL-terminated name of the public symbol for the read function.

readParm

Points to the string that is passed as the parm parameter to the read function.

writeRoutine

Points to the NULL-terminated name of the public symbol for the write function.

writeParm

Points to the string that is passed as the parm parameter to the write function.

withCookie

Specifies if you want this function to be a cookie type read function. If set to TRUE, the virtual file system assumes that each read operation calls the read function and is never satisfied from the results buffer.

Return Values

The return status should either be zOK or a valid NSS error code.

VIRT_AddResultData

Adds the specified string to the result buffer. (If needed, this function also extends the buffer.)

Service: VFS

Syntax

```
#include <nssPubs.h>

STATUS VIRT_AddResultData (
   VirtInfo_s  *virtInfo,
   utf8_t      *data);
```

Parameters

virtInfo

Points to the structure that is passed in and updated.

data

Points to a NULL-terminated string that will be added to the result buffer.

Return Values

The return status should either be zOK or a valid NSS error code.

VIRT_AddResultElement

Adds a complete element to the result buffer.

Service: VFS

Syntax

```
#include <nssPubs.h>

STATUS VIRT_AddResultElement (
   VirtInfo_s  *virtInfo,
   utf8_t      *tagName,
   utf8_t      *data,
   BOOL         newLine);
```

Parameters

virtInfo

Points to the structure that is passed in and updated.

tagName

Points to the NULL-terminated tag name that is to be used for both the beginning and ending tags of the element.

data

Points to a NULL-terminated string that contains the contents of the element.

newLine

Specifies if a new line character should be added to the result buffer:

TRUE Add a new line character
FALSE Do not add a new line character

Return Values

The return status should either be zOK or a valid NSS error code.

VIRT_AddResultTag

Adds the specified tag to the results buffer. (This function also adds the angle brackets, an optional end tag indicator, and new line characters).

Service: VFS

Syntax

```
#include <nssPubs.h>

STATUS VIRT_AddResultTag (
   VirtInfo_s  *virtInfo,
   utf8_t      *tagName,
   BOOL         endTag,
   BOOL         newLine);
```

Parameters

virtInfo

Points to the structure that is passed in and updated.

tagName

Points to the NULL-terminated tag name to be added.

endTag

Specifies if the end tag slash should be added to the tag:

TRUE Add the end tag slash
FALSE Do not add the end tag slash

newLine

Specifies if a new line character should be added to the result buffer:

TRUE Add a new line character
FALSE Do not add a new line character

Return Values

The return status should either be zOK or a valid NSS error code.

VIRT_MakeResultsImportant

Marks the result buffer as needing to be read, which assures that all reads coming back from the results buffer until the results are set back to normal.

Service: VFS

Syntax

```
#include <nssPubs.h>

void VIRT_MakeResultsImportant (
   VirtInfo_s  *virtInfo);
```

Parameters

virtInfo

Points to the structure that is passed in and then updated.

VIRT_MakeResultsNormal

Marks the result buffer as being normal. This function is called after calling VIRT_MakeResultsImportant (page 471) to cause the result buffer to be treated in a normal manner.

Service: VFS

Syntax

```
#include <nssPubs.h>

void VIRT_MakeResultsNormal (
   VirtInfo_s  *virtInfo);
```

Parameters

virtInfo

Points to the structure that is passed in and then updated.

VIRT_ResetResult

Resets the results buffer to have no content.

Service: VFS

Syntax

```
#include <nssPubs.h>

void VIRT_ResetResult (
   VirtInfo_s  *virtInfo);
```

Parameters

virtInfo

Points to the structure that is passed in and then updated.

XML_BackwardFindEndTag

Searches backwards from the cursor to the start of the buffer to find the specified tag as an end tag.

Service: VFS

Syntax

```
#include <xmlnss.h>

STATUS XML_BackwardFindEndTag (
   utf8_t  *tag,
   utf8_t  *cursor,
   utf8_t  *bufferStart,
   utf8_t  **startOfTag);
```

Parameters

tag

Points to the name of the tag (NULL-terminated string) to search for.

cursor

Points to the starting search position. The search works backwards from this position.

bufferStart

Points to the start of the buffer (the last position to be searched).

startOfTag

Points to the position of the first angle bracket in the end tag.

Return Values

Returns zOK if the tag is found. Otherwise, it returns zFAILURE.

XML_findEndOfNonWhiteSpace

Moves the cursor to the address of the last non-white space character.

Service: VFS

Syntax

```
#include <xmlnss.h>

void XML_findEndOfNonWhiteSpace (
   utf8_t  **ptr,
   utf8_t   *endptr);
```

Parameters

ptr

Points to the current cursor location on input. On output, it points to the position of the last non-white space character.

endptr

Points to the last valid character to be searched.

XML_ForwardFindTag

Searches forward from the given cursor position to the end of the buffer to find the specified tag. Note: This function currently assumes that the tag has no attributes.

Service: VFS

Syntax

```
#include <xmlnss.h>

STATUS XML_ForwardFindTag (
   utf8_t   *tag,
   NINT      tagLen,
   utf8_t   *cursor,
   utf8_t   *bufferEnd,
   utf8_t  **endOfTag);
```

Parameters

tag

Points to the name of the tag to be searched for.

tagLen

Specifies the length of the tag name.

cursor

Points to the starting position of the search.

bufferEnd

Points to the end of the buffer to search.

endOfTag

Points to the closing angle brack of the tag.

Return Values

The return status should either be zOK or a valid NSS error code.

XML_GetNextTag

Returns the next tag that is found in the specified buffer. The function also updates the element information for the found element.

Service: VFS

Syntax

```
#include <xmlnss.h>

STATUS XML_GetNextTag (
   utf8_t             *bufferStart,
   utf8_t             *bufferEnd,
   XML_ElementInfo_s  *elementInfo,
   utf8_t            **tagName,
   NINT               *tagLen);
```

Parameters

bufferStart

Points to the starting position for the search.

bufferEnd

Points to the ending position for the search.

elementInfo

Points to the structure that is updated to show where the element starts and ends.

tagName

Points to the beginning of the tag name in the buffer.

tagLen

Points to the length of the tag name.

Return Values

Returns zOK if the tag is found. Otherwise, it returns zFAILURE.

Remarks

.

XML_GetTagElement

Finds the entire element for a given tag. The function searches between the given start and end pointers and returns an updated element structure.

Service: VFS

Syntax

```
#include <xmlnss.h>

STATUS XML_GetTagElement (
   utf8_t             *tag,
   utf8_t             *bufferStart,
   utf8_t             *bufferEnd,
   XML_ElementInfo_s  *elementInfo);
```

Parameters

tag

Points to the NULL-terminated tag name for the element to be found.

bufferStart

Points to the starting position of the search.

bufferEnd

Points to the ending position of the search.

elementInfo

Points to the structure that is updated to show where the given element starts and ends.

Return Values

Returns zOK if the tag is found. Otherwise, it returns zFAILURE.

14 Examples

This section describes how to use VFS to access and control a hypothetical toaster object and contains the following sections:

- Section 14.1, “Creating a Virtual File,” on page 479
- Section 14.2, “Accessing a Virtual File with Perl,” on page 481

TIP: Note that since virtual files rely only on standard file system functions, almost any scripting language can be used—as long as it allows for direct control of the data being read and written so that extra formatting is not introduced.

14.1 Creating a Virtual File

The following are the necessary steps (as well as some example code) for creating a virtual file.

1. Create a file.
2. Write a virtual I/O command to the file, which tells the system that you want to access the actual contents of the file.
3. Write the transformation template to the file.

The following example contains some of the text to write to a transformation template for a virtual file. It contains the template for accessing portions of our hypothetical toaster object and begins with a virtual I/O command that tells the file system to work on the actual file contents. This template can be cut and pasted into a text file and copied to a virtual file, which will result in the virtual file containing a transformation template.

```
<virtualIO><define></virtualIO><?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE transform SYSTEM "virtualTemplate.dtd">
<transform>

<!-- The following datastream has no name and will be used as the default for read
and write requests that have no virtual I/O command preceding them. It takes a long
value from eight bytes past toasterObj and converts it to ASCII, then combines it
with leadingtext and trailingtext. When this datastream is read, it will return
"The current toast temperature is set to 125 degrees Fahrenheit," assuming that the
value is 125. A write operation will change the long to the value in the write
operation (and must be a number in ASCII).
-->
   <datastream>
      <location symname="toasterObj" offset="8">
         <readloc>
            <format><long signed="yes"></format>
            <leadingtext>The current toast temperature is set to
                  </leadingtext>
            <trailingtext>] degrees Fahrenheit.</trailingtext>
         </readloc>
         <writeloc>
            <format><long></format>
         </writeloc>
      </location>
   </datastream>

<!-- The following toastTemp datastream is similar to the default datastream except
```

```
that it has no leading and trailing text elements.
-->
   <datastream name="toastTemp">
      <location symname="toasterObj" offset="8">
         <readloc>
            <format><long signed="yes"></format>
         </readloc>
         <writeloc>
            <format><long></format>
         </writeloc>
      </location>
   </datastream>

<!-- The following toasterStructure datastream returns 24 bytes of raw data that
represents a structure of toaster data. A write operation will write up to 24 bytes
of binary data into memory, starting at toasterObj.
-->
   <datastream name="toasterStructure">
      <location symname="toasterObj">
         <readloc>
            <format><raw length="24"></format>
         </readloc>
         <writeloc>
            <format><raw length="24"></format>
         </writeloc>
      </location>
   </datastream>

<!-- The following toasterAccess datastream uses user-written functions
(ReadToasterVariable and WriterToasterVariable) to read and write the virtual
contents of the file. The value contained in the readfunc and writefunc tags
("temperature") is passed to the function as well as a buffer for the read and
write operations.
-->
   <datastream name="toasterAccess">
      <function>
         <readfunc symname="ReadToasterVariable">temperature</readfunc>
         <writefunc symname="WriteToasterVariable">temperature
            </writefunc>
      </function>
   </datastream>

<!-- The following toasterStatus datastream allows a command with a response to be
sent to the function ProcessToasterCommand. The write operations are taken to be
commands and the read operations read the response.
-->
   <datastream name="toasterStatus">
      <function>
         <writefunc symname="ProcessToasterCommand">full</writefunc>
      </function>
   </datastream>

<!-- These last two datastreams represent data taken directly from the template. In
this example, a read operation will result in a help message.
-->
  <datastream name="shorthelp">
      <data>Information about the toaster</data>
   </datastream>

   <datastream name="help">
```

```
      <data>This file contains information dealing with the toaster
        object.
            The datastreams are as follows:
          default:          The toast temperature with text.
          toastTemp:        The toast temperation (with no additional
                             text).
          toasterStructure: The raw data from the toasterStructure.
          toasterAccess:   A function that can allow access to any field
                              in the toaster object.
            toasterStatus:    Returns the contents of all fields in the
                              toaster object.
      </data>
   </datastream>
</transform>
```

14.2 Accessing a Virtual File with Perl

Now that we have created a virtual file with a transformation template, accessing it is as simple as using basic open, close, read, and write functions.

Using the Perl script examples in the following sections, you can accomplish the following tasks:

- Section 14.2.1, “Reading a Datastream,” on page 481
- Section 14.2.2, “Writing a Datastream,” on page 482
- Section 14.2.3, “Writing a Command,” on page 482

14.2.1 Reading a Datastream

The following is a Perl script that will read a datastream:

```
#
#   Perl script that reads a datastream from a virtual file
#
# This script takes the name of the virtual file as the first parameter # and an
optional datastream name as the second parameter. If the
# datastream is not specified, the actual contents of the file are
# read, which means that this script cannot be used to read the default #
datastream. However, a simple type command reads the default
# datastream with no problem. When using Perl, if you do not use the
# sysread and syswrite functions for reading and writing,
# the buffering can cause inconsistent results.

if (($#ARGV > 1) or ($#ARGV == -1))
{
   die "USAGE: readVirt.pl filename [datastream]\n";
}

open(FILE, "+<".$ARGV[0]) or die "Error opening $ARGV[0]";

if ($#ARGV == 0)
{
   $command = "<virtualIO><define></virtualIO>";
```

```
}
else
{
   $command = "<virtualIO><datastream name=\"".$ARGV[1]."\"></virtualIO>";
}
syswrite FILE, $command, length($command);

$len = 999;
while ( $len > 0)
{
   $len = sysread FILE, $buf, 1000;
   print("$buf");
}
close(FILE);
```

14.2.2 Writing a Datastream

Writing a datastream is similar to reading a datastream. The following is a Perl script that will read a datastream:

```
#
#   Perl script to write to a datastream in a virtual file
#
# This script takes as parameters the virtual file to be written to, a # file
ontaining the data to write, and an optional datastream. If the # datastream is not
given, the default datastream is used. Note the
# same restrictions as with Reading a Datastream on the use of sysread
# and syswrite.

if (($#ARGV > 2) or ($#ARGV < 1))
{
    die "USAGE: writeVirt.pl virtfile inputfile [datastream]\n";
}
open(VIRTFILE, "+<".$ARGV[0]) or die "Error opening $ARGV[0]";
open(INFILE, "<".$ARGV[1]) or die "Error opening $ARGV[1]";

if ($#ARGV == 2)
{
    $command = "<virtualIO><datastream name=\"".$ARGV[2]."\"></virtualIO>";
    syswrite VIRTFILE, $command, length($command);
}

$len = 999;
while ( $len > 0)
{
    $len = sysread INFILE, $buf, 1000;
    print("$buf");
    syswrite VIRTFILE, $buf, length($buf)
}

close(INFILE);
close(VIRTFILE);
```

14.2.3 Writing a Command

The following is a Perl script that writes a command to a datastream:

```
#
#   Perl script to write a command to datastream in a virtual file
#
# The difference between the command and the normal read operation is
# that the read operation returns a result generated by completing the
# write operation.

if (($#ARGV > 2) or ($#ARGV < 1))
{
   die "USAGE: cmdVirt.pl virtfile [datastream]\n";
}

open(VIRTFILE, "+<".$ARGV[0]) or die "Error opening $ARGV[0]";

if ($#ARGV == 1)
{
   $command = "<virtualIO><datastream name=\"".$ARGV[1]."\"></virtualIO>";
   syswrite VIRTFILE, $command, length($command);
}

print("Enter command: ");
$buf = <STDIN>;
chomp($buf);
syswrite VIRTFILE, $buf, length($buf);
sysread VIRTFILE, $result, 1000;
print("$result");

close(VIRTFILE);
```

A Revision History

This section outlines all the changes that have been made to the Virtual File Services documentation (in reverse chronological order).

December 2011	Modified the Chapter 4, “User Commands,” on page 323 by changing nssRequest to userRequest in all command categories, and added a new command category called listUserSpaceRestrictions (user) (page 330).
December 2008	Added the Statistics section in Chapter 1, “Basic Concepts,” on page 13.
	Added the ‘noAccessTime enabled’ to the modifyVolumeInfo in the Chapter 3, “manage.cmd Definitions,” on page 27.
	Added an ‘action’ variable in the listFileEvents section of the Chapter 6, “FileEvents.xml Definitions,” on page 351.
	Added addPool, addVolume, poolUpdate, getVolumeInfo, and resetIDs section in the Chapter 10, “linux.cmd Definitions,” on page 441.
February 28, 2008	Added the Linux pool snapshot commands to Pool section in the Chapter 3, “manage.cmd Definitions,” on page 27.
	Renamed the chapter Linux definitions to Chapter 10, “linux.cmd Definitions,” on page 441
	Added listEvmsVolumes function to Chapter 3, “manage.cmd Definitions,” on page 27
	Added <noSnapshot> tag to listPools (page 239) function.
October 17, 2007	Added links in the Preface to the “Archive Definitions” on page 407 and “linux.cmd Definitions” on page 441 sections.
	Added a tip to Chapter 1, “Basic Concepts,” on page 13 that the CDATA element can be used to pass strings containing special characters.
	Updated the description of the allocatedSize element of listPoolSnapshots (page 196).
June 27, 2007	Added the poolName and details elements to the Request of listPoolSnapshots (page 196).
	Changed the Request and Reply formats of setUserSpaceRestriction (user) (page 244).
February 28, 2007	Added the activatePoolSnapshot (page 163), addPoolSnapshot (page 171), deactivatePoolSnapshot (page 174), getPoolSnapshotInfo (page 189), listEvmsVolumes (page 301), listPoolSnapshots (page 196), removePoolSnapshot (page 215), and uidToEquivalentGUIDs (page 443) commands.
	Added an example to addUser (page 347) and to the context element.
	Changed the beginning and ending tags to ndsRequest and ndsReply for Chapter 5, “NDS.cmd Definitions,” on page 333.
	Marked getAdapterInfo (page 33), listAdapters (page 35), and getLSSInfo (page 132) and a few commands listed in Section 3.12, “Pool,” on page 162 as being implemented only on NetWare and not on Linux.

October 11, 2006	Added a note about the tree name needing to be included in the context element and an example to addTrustee (page 37). Also, added the values for the rights element. Added explanation of the type element to listPartitions (Server) (page 236). Added the values for the type element to listVolumes (page 300). Added an example to addTrustee (page 367), getFileInfo (page 373), modifyInheritedRightsFilter (page 379), and scanSalvageableFiles (page 388). Updated the links at the start of Chapter 9, “Archive Definitions,” on page 407.
June 21, 2006	Changed the listPartitions command to listPartitions (Server) (page 236). Also changed the opening tag from partition to partitionInfo.
March 1, 2006	Added navigational links.
October 5, 2005	Added information about what's returned when there's a missing segment in a mirror device to getDeviceInfo2 (page 85). Changed the sub-element names of the timeInfo element of setFileInfo (page 394). Transitioned to revised Novell documentation standards.
June 1, 2005	Added the id element to browseUserSpaceRestrictions (user) (page 240), getUserSpaceRestriction (user) (page 242), and setUserSpaceRestriction (user) (page 244). Added the mountPoint, mountPointRename, and nameSpace elements to modifyVolumeInfo (page 304). Made minor edits.
March 2, 2005	Added Chapter 10, “linux.cmd Definitions,” on page 441 and the following Linux commands: poolIDToName (page 442), userIDToName (page 444), volumeIDFileIDToPath (page 445), and volumeIDToName (page 446). Added addQuota (page 366) and scanSalvageableFiles (page 388). Obsoleted the former addQuota (obsolete) (page 126) function. Updated the elements and their descriptions for purgeDeletedFile (page 381), salvageDeletedFile (page 385), and setFileInfo (page 394). Added the volumeName element and its description to getFileInfo (page 373). Added the dstParentFullPath element and its description to salvageDeletedFile (page 385). Added the parentFullPath and volumeName elements and their descriptions to scanSalvageableFiles (page 388). Added segmentID, majorVersion, minorVersion, partitionType, mountPoint, hasSYS, bootable, restripeEnabled, remirrorEnabled, mirrorActive, and mirrorStatus to getDeviceInfo2 (page 85). Added the name and id elements to scanSalvageableFiles (page 388). Updated the directoryQuota element and added a description for the effectiveRights element to getFileInfo (page 373). Updated the examples in getUserSpaceRestriction (user) (page 242) and setUserSpaceRestriction (user) (page 244).

October 6, 2004	Made multiple changes for Linux users. Added the "Archive Definitions" on page 407 and Section 1.4.1, "eDirectory Name Formats," on page 15 sections. Added addPartition2 (page 145), listPartitions (page 152), and modifyPartition (page 156). Added the volumeReadAhead element to getVolumeInfo (page 292) and modifyVolumeInfo (page 304), added maintenance as a possible returned state to getState (page 192), and added the type attribute to listDevices (Server) (page 235). Changed listAdapter to listAdapters (page 35). Updated the reply information of listMultiPaths (page 103). Rewrote the Preface section to include where to find additional information about Virtual File Services.

June 9, 2004	Added the volumePassword element to addVolume (page 281) and the volumeEncrypted element to getVolumeInfo (page 292). Added the noDFSGUID to addTraditionalVolume (page 278) and addVolume (page 281). Also, added the following information on encrypted volumes: ◆ Section 1.7.1, “Encrypted Volumes,” on page 20 ◆ Section 1.7.2, “EVS Tests,” on page 22 ◆ Section 1.7.3, “Console Commands,” on page 22 Added Section 1.8, “Junctions,” on page 22 and the following related functions: createLink (page 113), deleteLink (page 115), modifyLink (page 119), and readLink (page 121). Added Section 3.18, “Volume MN Operations,” on page 313, changeJobState (page 314), createJob (page 315), getJobList (page 317), getJobStatus (page 318), and listSkippedFiles (page 320). Added lookup (page 264), added user and password elements to startRepair (page 272), and added lots of example responses to getVLDBInfo (page 255) in Section 3.16, “VLDB,” on page 248. Added getDeviceInfo2 (page 85) and listDevicePartitions (page 101) in Section 3.6, “Device,” on page 82, getPartitionInfo (page 148) to Section 3.11, “Partition,” on page 140, removeRAID2 (page 229) in Section 3.13, “RAID,” on page 220, removePool2 (page 214) and renamePoolSnapshot (page 219) in Section 3.12, “Pool,” on page 162, initDFSGUIDs (page 118) in Section 3.7, “DFS,” on page 112, and getServerConfiguration (AFP) (page 30) and setServerConfiguration (AFP) (page 31) in Section 3.1, “AFP,” on page 29. Added addContext (page 40), addDomainACL (page 41), addShare (page 42), createContextList (page 43), createDomain (page 44), deleteDomain (page 46), findContext (page 47), getCreateContextListStatus (page 48), getDomainConfiguration (page 49), getImportWindowsUsersStatus (page 51), getServerConfiguration (page 52), getShareProperties (page 55), importWindowsUsers (page 56), joinDomain (page 57), leaveDomain (page 59), listContexts (page 60), listDomainControllers (page 61), listImportedUsers (page 63), listShares (page 64), modifyContextList (page 66), modifyShare (page 67), removeContext (page 68), removeShare (page 69), setDomainConfiguration (page 70), and setServerConfiguration (page 72) in Section 3.4, “CIFS,” on page 39. Updated createNewService (page 251) to include new elements and added deleteService (page 253). Added the objectID and name elements to renameDevice (page 110). Added the immediatePurge element to the attributes section of the getFileInfo (page 373) response. Also, changed the name of the compressImmediatePurge element to compressImmediate.

February 18, 2004	In response to customer feedback, added the Section 1.6.1, “Freeze and Thaw Functionality,” on page 19 section and subsections on the Freeze and Thaw events. Also, added the Remarks section to poolFreeze (page 204).
	Added volume and server inventory information in Chapter 8, “Inventory.xml Definitions,” on page 397.
	Added browseUserSpaceRestrictions (user) (page 240), getUserSpaceRestriction (user) (page 242), setUserSpaceRestriction (user) (page 244), getAllEffectiveRights (page 369), and removeAllTrustees (page 382).
	Updated getFileInfo (page 373) to add the directoryQuota and getEffectiveRightsByUser information.
June 2003	Added documentation for salvageDeletedFile (page 385) and purgeDeletedFile (page 381).
	Added the removable element to the nssReply of listDevices (page 98) and added the updateVLDB element to the nssRequest of addVolume (page 281), addTraditionalVolume (page 278), removeVolume (page 310), renameVolume (page 312), and salvageVolume (page 79).
March 2003	Added EFL XML commands in Chapter 6, “FileEvents.xml Definitions,” on page 351.
	Added the following new commands: activatePoolSnapshot (page 163), addPool2 (page 169), addPoolSnapshot (page 171), addRAID2 (page 224), deactivatePoolSnapshot (page 174), expandPool2 (page 178), getAdapterInfo (page 33), getDeviceInfo (page 83), getLSSVolumeInfo (page 137), getNDSName (Volume) (page 286), getPathInfo (page 94), getPoolDevices (page 183), getPoolSnapshotInfo (page 189), getServerFreeSpace (page 233), listAdapters (page 35), listDevices (page 98), listDevicePools (page 102), listMultiPaths (page 103), listPoolSnapshots (page 196), listPools (page 239), removePoolSnapshot (page 215), renameDevice (page 110), and renameRAID (page 230).
	Added the following freeze/thaw commands: poolFreeze (page 204), poolFreezeStatus (page 206), and poolThaw (page 210).
	Updated the description of the ignoreShareState element in the Request for Pools' modifyState (page 202) and added a new pool type (segments) to getPoolInfo (page 184).
September 2002	Updated getPoolInfo (page 184) , listVolumes (page 300), and getTraditionalVolumeInfo (page 289).
May 2002	Added more complete element descriptions and command examples.
February 2002	Added commands for clustering operations, see addPool (page 166) and getDefaultClusterNames (page 179).
	Added example to Section 1.2, “VFS vs Traditional File System Access,” on page 14.
	Rewrote Chapter 3, “manage.cmd Definitions,” on page 27 to include closing XML tags and coordinating request and reply commands within the same section.
September 2001	Added as a new NDK component.