EMC® NetWorker®

Release 7.6 Service Pack 2

Command Reference Guide

P/N 300-011-695
REV A01

EMC Corporation
Corporate Headquarters:
Hopkinton, MA 01748-9103
1-508-435-1000
www.EMC.com

Published April, 2011

Preface

As part of an effort to improve and enhance the performance and capabilities of its product lines, EMC periodically releases revisions of its hardware and software. Therefore, some functions described in this document may not be supported by all versions of the software or hardware currently in use. For the most up-to-date information on product features, refer to your product release notes.

If a product does not function properly or does not function as described in this document, please contact your EMC representative.

Audience

This document is part of the NetWorker 7.6 Service Pack 2 documentation set, and is intended for use by system administrators during the execution of commands at the operating system command line.

Online help for commands

Help for commands is also available from the command line.

- The detailed command information that is available in this document is also available from the command line for any platform except Windows. To access this help from the command line, type **man** *command_name*, for example:

  ```
  man recover
  ```

  Note: The optional package, LGTOman must be installed to access help using the **man** command. The operating system path environment variable must also include the location of the man pages, otherwise, you must run the **man** command from the installed location of the man pages.

- To access *basic* help for any NetWorker command on any platform, type *command_name* **-help** from the command line, for example:

  ```
  recover -help
  ```

  Note: Basic help is limited to a list of the command's arguments, options, and parameters. More complete information is contained in this document.

Where to get help

EMC support, product, and licensing information can be obtained as follows.

Product information — For documentation, release notes, software updates, or for information about EMC products, licensing, and service, go to the EMC Powerlink website (registration required) at:

http://Powerlink.EMC.com

Technical support — For technical support, go to EMC Customer Service on Powerlink. To open a service request through Powerlink, you must have a valid support agreement. Please contact your EMC sales representative for details about obtaining a valid support agreement or to answer any questions about your account.

Your Comments

Your suggestions will help us continue to improve the accuracy, organization, and overall quality of the user publications. Please send your opinion of this document to:

BSGdocumentation@emc.com

If you have issues, comments, or questions about specific information or procedures, please include the title and, if available, the part number, the revision (for example, A01), the page numbers, and any other details that will help us locate the subject you are addressing.

NAME

nsrd − daemon providing the NetWorker service

SYNOPSIS

nsrd [−**k** *virtual-service-name*]

ansrd [**commentary**]

DESCRIPTION

The **nsrd** daemon provides an RPC-based save and recover service. This service allows users to save, query for, and recover their files across a network. The RPC program number provided by **nsrd** is 390103.

Normally **nsrd** is invoked from a startup shell script (for example *rc.local*, *rc.boot*) at boot-time, and should never need to be started directly by a user. After it is started, **nsrd** starts up the other daemons it needs to provide the NetWorker service.

The **nsrd** command must be run on a machine with appropriate resources. These resources include devices (for example, tape drives) which are under the control of the media multiplexor software (see **nsrmmd**(1m)), and sufficient disk space for the index daemons, (see **nsrindexd**(1m) and **nsrmmdbd**(1m)) to maintain the index of saved user files and volumes with corresponding files.

Each time a backup, recover, or another session begins, **nsrd** starts the program, **ansrd**, to process the requested session. The **ansrd** program is called an *agent*. The agent is in charge of monitoring that backup, recover, or another session, and automatically exits when a session completes. Using **ps**(1) or another process monitoring tool, you can inspect the subsequent parameters of **ansrd** to see what kind of session it is monitoring. If necessary, agents can be forcibly terminated to abort a backup or recover session. Agents cannot be run directly; they can only be started by **nsrd**.

When **nsrd** is started with the −**k** option, it checks to see whether it has been installed as a cluster service and that the virtual host which owns **/nsr/res** matches *virtual-service-name.* If either of these validation steps fails, **nsrd** exits immediately. (To check whether NetWorker has been installed as a cluster service, **nsrd** checks for a file called **NetWorker.clustersvr** in the directory containing the **nsrd** binary. To check that **/nsr/res** is owned by *virtual-service-name*, **nsrd** queries the cluster management software.)

If the −**k** option is not used when starting NetWorker in a cluster, the server assumes the identity of the virtual host which owns **/nsr/res.** If no virtual host owns **/nsr/res**, then **nsrd** will not start.

OPTIONS

−**k** *virtual-service-name*

Instructs **nsrd** to start up in cluster failover mode using *virtual-service-name* as its hostname/identity. This option is used by the NetWorker cluster control script which starts NetWorker.

FILES

/nsr/logs/daemon.raw

The file to which **nsrd** and other NetWorker daemons send information about various error conditions that cannot otherwise be logged using the NetWorker event mechanism.

/nsr/res/nsrdb

Information describing the NetWorker service and its resources (See **nsr_service**(5)).

NetWorker.clustersvr

If this file exists in the directory containing NetWorker's daemons, it indicates that the NetWorker server has been installed as a cluster service.

SEE ALSO **nsr**(1m), **nsr_service**(5), **nsr_render_log**(1m), **nsrmmd**(1m), **nsrmmdbd**(1m), **nsrindexd**(1m), **ps**(1), **rc**(1m)

NAME ascdcode − print error message for ASC/ASCQ error codes

SYNOPSIS **ascdcode** [**-o** *vendor id* [−**p** *product id*]] *ASC ASCQ*

DESCRIPTION The **ascdcode** program interprets Additional Sense Code (ASC) and Additional Sense Code Qualifier (ASCQ) data and returns an appropriate error message. The **ascdcode** program returns interpreted ASC/ASCQ data either for the named vendor and product IDs (with the **-o** and **-p** options), or for all libraries or devices as defined in the SCSI-3 Specifications (http://www.ncits.org) or individual vendors. For unimplemented ASC/ASCQ codes, the *ascdcode* program will return the message *Not implemented.*

OPTIONS −**o** *vendor id*

Checks to see if the vendor is an OEM and looks up the ASC/ASCQ error codes defined original vendor for the library or device. If the −**p** option is not specified, ascdcode will return vendor specific information for the *vendor id* specified with this option. The *vendor id* should be identical to the vendor string reported when you run the **inquire** program. For a complete list of vendor ID assignments, see the web page: http://www.t10.org/lists/vid-alph.htm. This option is only applicable to vendor specific ASC/ASCQ codes of value(s) greater than 0x7f.

−**p** *product id*

Use with −**o** to provide a library or device type with the OEM vendor. The *product id* should be identical to the library or device string reported when you run the **inquire** program.

The *ASC* argument should be the first of the pair of ASC/ASCQ error codes reported by the library or device. You may specify the value as a hexadecimal number by preceding the value with *0x* . The default value is assumed to be a decimal value.

The *ASCQ* argument should be the latter of the pair of ASC/ASCQ error codes reported by the library or device. You may specify the value as a hexadecimal number by preceding the value with *0x* . The default value is assumed to be a decimal value.

FILES **/INSTALL_PATH/lgtovendors**

The directory where the vendor specific files are installed.

/INSTALL_PATH/lgtovendors/OEM_MAP

The file that maps OEM vendors and their corresponding products to original vendors. See the file for a description on how to add entries for new OEM vendors and products.

SEE ALSO **libscsi**(1m)

LIMITATIONS The **ascdcode** program always uses the installed vendor specific files to look up vendor specific error messages. Formatting convention of ASC/ASCQ error codes and their corresponding error text in these files are of the form:

0x[ASC], 0x[ASCQ], [error message]

NAME cdi_block_limits − query block size limits on a tape device

SYNOPSIS **cdi_block_limits**
−**f** *device* [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_block_limits** program queries block size limits on a tape device. The **cdi_block_limits** program returns the block size limits for the named SCSI device (with the **-f** option). Note that a device's block size limits may be larger than the operating system's limits. This program specifically returns the device's block size limits.

OPERANDS −**f** *device*
Specifies the device to obtain block size information from.

OPTIONS −**t** Use the −**t** option to specify the method of tape functions to use to query block size limits. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for a complete list of access methods currently supported by the *cdi_block_limits* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_block_limits -f /dev/rmt/2cbn

Block size limits returned from   accessed through /dev/rmt/2cbn
                maximum block size allowed is 16776128
                minimum block size allowed is 61301
        cdi_info.drivestat is:
          status = 1, DRIVE_STATUS_NO_ERROR
          msg = Drive reports no error - but state is unknown
```

SEE ALSO **libcdi**(1m)

NAME cdi_bsf − issue a backward space file SCSI command to a tape device

SYNOPSIS **cdi_bsf** −**f** *device* −**n** *count* [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_bsf** program issues a backward space file (bsf) SCSI command to a tape device. The **cdi_bsf** program also returns the status of the named SCSI device (specified by the **-f** option).

OPERANDS

−**f** *device*
Specifies the device to send the bsf SCSI command to.

−**n** *count*
The file count for the bsf SCSI command. The "-n count" parameter is required.

OPTIONS

−**t** Use the −**t** option to specify the method of tape functions to use to issue the bsf SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for the complete list of access methods currently supported by the *cdi_bsf* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_bsf -f /dev/rmt/2cbn -n 2

CDI_BSF 2 successful.
    elapsed time for command was 0 seconds
    cdi_info.drivestat is:
        status = 0, DRIVE_STATUS_READY
        msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m)

NAME cdi_bsr − issue a backward space record command to a tape device

SYNOPSIS **cdi_bsr** −**f** *device* −**n** *count* [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_bsr** program issues a backward space record (bsr) SCSI command to a tape device. The **cdi_bsr** program also returns the status of the named SCSI device (specified by the **-f** option).

OPERANDS

−**f** *device*
Specifies the device to which to send the bsr SCSI command.

−**n** *count*
The record count for the bsr SCSI command.

OPTIONS

−**t** Use the −**t** option to specify the method of tape functions to use to issue the bsr SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for the complete list of access methods currently supported by the *cdi_bsr* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_bsr -f /dev/rmt/2cbn -n 2 -v

CDI_GET_VERSION returns 1
    CDI_BSR 2 successful.
        elapsed time for command was 0 seconds
        cdi_info.drivestat is:
            status = 0, DRIVE_STATUS_READY
            msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m)

NAME cdi_eod − send an end of data SCSI command to a tape device

SYNOPSIS **cdi_eod** −**f** *device* [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_eod** program sends an end of data (eod) SCSI command to a tape device. The **cdi_eod** program also returns the status of the named SCSI device (with the **-f** option).

OPERANDS −**f** *device*
Specifies the device to issue the eod SCSI command to. The −**f** option is a required option.

OPTIONS −**t** Use the −**t** option to specify the method of tape functions to use to issue the eod SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for a complete list of access methods currently supported by the *cdi_eod* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_eod -f /dev/rmt/2cbn

 CDI_EOD successful.
     elapsed time for command was 0 seconds
     cdi_info.drivestat is:
         status = 0, DRIVE_STATUS_READY
         msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m)

NAME cdi_filemark − issue a write filemark/setmark command to a tape device

SYNOPSIS **cdi_filemark** −**f** *device* [−**a**] [−**n** *count*] [−**s**] [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_filemark** program issues a write filemark/setmark SCSI command to a given device. The **cdi_filemark** program also returns the status of the named SCSI device (specified by the **-f** option). The default behavior is to write a single filemark to the specified device.

OPERANDS −**f** *device*
Specifies the device to send the write filemark SCSI command to.

OPTIONS −**a** Use asynchronous I/O for the operation. Rather than blocking till completion, the program will return immediately. The default is synchronous I/O.

−**n** *count*
The filemark count for the write filemark SCSI command. The default count is 1.

−**s** Write a setmark instead of a filemark for the operation.

−**t** Use the −**t** option to specify the method of tape functions to use for the write filemark SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for the complete list of access methods currently supported by the *cdi_filemark* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_filemark -f /dev/rmt/2cbn -n 2 -v

 CDI_GET_VERSION returns 1
    CDI_WRITE_FILEMARKS 2 successful.
        elapsed time for command was 0 seconds
        cdi_info.drivestat is:
            status = 0, DRIVE_STATUS_READY
            msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m)

NAME cdi_fsf − issue a forward space file SCSI command to a tape device

SYNOPSIS **cdi_fsf** −**f** *device* −**n** *count* [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_fsf** program issues a forward space file (fsf) SCSI command to a tape device. The **cdi_fsf** program also returns the status of the named SCSI device (specified by the **-f** option).

OPERANDS

−**f** *device*
Specifies the device to send the fsf SCSI command to. The −**f** option is a required option.

−**n** *count*
The file count for the fsf SCSI command.

OPTIONS

−**t** Use the −**t** option to specify the method of tape functions to use to issue the fsf SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for the complete list of access methods currently supported by the *cdi_fsf* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_fsf -f /dev/rmt/2cbn -n 2

 CDI_FSF 2 successful.
     elapsed time for command was 0 seconds
     cdi_info.drivestat is:
         status = 0, DRIVE_STATUS_READY
         msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m)

NAME cdi_fsr − issue a forward space record command to a tape device

SYNOPSIS **cdi_fsr** −**f** *device* −**n** *count* [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_fsr** program issues a forward space record (fsr) SCSI command to a tape device. The **cdi_fsr** program also returns the status of the named SCSI device (specified by the **-f** option).

OPERANDS

−**f** *device*
Specifies the device to send the fsr SCSI command to.

−**n** *count*
The record count for the fsr SCSI command.

OPTIONS

−**t** Use the −**t** option to specify the method of tape functions to use to issue the fsr SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for the complete list of access methods currently supported by the *cdi_fsr* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_fsr -f /dev/rmt/2cbn -n 2 -v

 CDI_GET_VERSION returns 1
    CDI_FSR 2 successful.
        elapsed time for command was 0 seconds
        cdi_info.drivestat is:
            status = 0, DRIVE_STATUS_READY
            msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m)

NAME cdi_get_config - get configuration information on a tape device

SYNOPSIS **cdi_get_config** −**f** *device* [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_get_config** program obtains configuration information on a tape device. The data output from this command is collected from the SCSI Mode Sense disconnect/reconnect, data compression and device configuration pages.

OPERANDS −**f** *device*
Specifies the device to obtain configuration information from.

OPTIONS −**t** Use the −**t** option to specify the method of tape functions to use to query the device for configuration information. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for a complete list of access methods currently supported by the *cdi_get_config* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_get_config -f /dev/rmt/2cbn -v

CDI_GET_VERSION returns 1
SCSI config info for   via /dev/rmt/2cbn:
    active_format               000d
    active_partition            00b8
    write_full_ratio            0000
    read_empty_ratio              007e
    write_delay_time              c950
    flags                    000000009cef2080
        Parameters Savable
        Change Active Format
        Data Buffer Recovery
        Block Identifiers Supported
        Report Setmarks
        Stop On Consecutive Filemarks:  Stop On 3 Consecutive Filemarks
        Recover Buffer Order
        Report Early Warning
        EOD defined:    Reserved (4)
        Enable EOD Generation
        Sync at Early Warning
        Soft Write Protection
    buffer_size_early_warning    00b80000
    data_compress_algorithm       0000
    discon_buffer_full           00ad
    discon_buffer_empty            00b8
    discon_bus_inactive          007e
    discon_time_limit            c950
    discon_connect_time_limit    ef75
    discon_max_burst_time        9eb8
    compression_algorithm        007ec950
    decompression_algorithm       ef759eb8
```

```
cdi_info.drivestat is:
    status = 1, DRIVE_STATUS_NO_ERROR
    msg = Drive reports no error - but state is unknown
```

SEE ALSO **libcdi**(1m)

NAME cdi_get_status − get status information from a tape device

SYNOPSIS **cdi_get_status** −**f** *device* [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_get_status** program obtains status information from a tape device. The data returned include tape density and block position.

OPERANDS −**f** *device*
Specifies the device to obtain status information from.

OPTIONS −**t** Use the −**t** option to specify the method of tape functions to use to query the device for status information. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for a complete list of access methods currently supported by the *cdi_get_status* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output from cdi_get_status:

```
% cdi_get_status -f /dev/rmt/2cbn

 CDI_GET_STATUS returns:
 DRIVE_STATUS_READY
 current density code = 00
 position is absolute block number 0
```

SEE ALSO **libcdi**(1m)

NAME cdi_inq − get inquiry information from a tape device

SYNOPSIS **cdi_inq** −**f** *device* [−**v**] [−**t**{ *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_inq** program obtains inquiry information from a tape device. The data returned include Vital Product Data (VPD) pages. Note that the **inquire (1m)** command can be used for a more comprehensive output of the serial number identifiers.

OPERANDS −**f** *device*
Specifies the device to obtain inquiry information from.

OPTIONS −**t** Use the −**t** option to specify the method of tape functions to use to query the device for inquiry information. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for a complete list of access methods currently supported by the *cdi_inq* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output from cdi_inq:

```
% cdi_inq -f /dev/rmt/2cbn

Standard Inquiry data:
    Vendor:
    Product:
    Rev:

VPD pages supported:
        Pages
    cdi_info.drivestat is:
        status = 1, DRIVE_STATUS_NO_ERROR
        msg = Drive reports no error - but state is unknown

% cdi_inq -f /dev/rmt/0cbn

Standard Inquiry data:
    Vendor:       QUANTUM
    Product:      DLT8000
    Rev:          0232

VPD pages supported:
        Pages 00 80 83 c0 c1
Serial number page (80):
        CX940P2410
Device ID page (83):
        IENN:00E09E600006A114

Non-standard pages displayed only with -v parameter
    cdi_info.drivestat is:
        status = 1, DRIVE_STATUS_NO_ERROR
        msg = Drive reports no error - but state is unknown
```

SEE ALSO **libcdi**(1m)

NAME cdi_load_unload − load or unload a tape device

SYNOPSIS **cdi_load_unload** −**f** *device* { −**l** | −**u** } [−**a**] [−**e**] [−**r**] [−**v**] [−**t**{ *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_load_unload** program loads or unloads medium into or from a tape device.

OPERANDS −**f** *device*
Specifies the device on which to perform the load/unload operation.

{ −**l** | −**u** }
Perform a load (-l) or unload (-u) medium operation.

OPTIONS

−**a** Use asynchronous I/O for the operation. Rather than blocking till completion, the program will return immediately. If this flag is set and CHECK CONDITION status is returned for the load/unload operation, the load or unload operation will not be performed. The default is synchronous I/O.

−**e** Position to end-of-medium before unloading the medium. If this flag is specified with the −**l** flag (i.e., load medium), the SCSI device will return CHECK CONDITION status and the sense key will be set to ILLEGAL REQUEST in the sense data.

−**r** Apply the correct tension to the medium. Not all devices have the capability to re-tension media. Please refer to the specific device manuals to confirm whether the re-tension function is available for the device.

−**t** Use the −**t** option to specify the method of tape functions to use to load/unload medium. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for a complete list of access methods currently supported by the *cdi_load_unload* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output from cdi_load_unload:

```
% cdi_load_unload -l -f /dev/rmt/2cbn

cdi load unload succeeds for   via /dev/rmt/2cbn:
    elapsed time for command was 0 seconds
    cdi_info.drivestat is:
        status = 0, DRIVE_STATUS_READY
        msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m)

NAME cdi_locate − position to a given block on a tape mounted on a tape device

SYNOPSIS **cdi_locate** −**f** *device* −**n** *block* [−**a**] [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_locate** program positions to a given block on a tape mounted on a tape device.

OPERANDS

−**f** *device*
Specifies the device to which to position.

−**n** *block*
Specify the block on a mounted tape to which to position.

OPTIONS

−**a** Use asynchronous I/O for the operation. Rather than blocking till completion, the program will return immediately. The default is synchronous I/O.

−**t** Use the −**t** option to specify the method of tape functions to use to position to a given block on tape. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for a complete list of access methods currently supported by the *cdi_locate* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output from cdi_locate:

```
% cdi_locate -f /dev/rmt/2cbn -n 300

CDI_GET_STATUS returns:
    locate successful:  position to block 300 took 0 seconds
        cdi_info.drivestat is:
            status = 0, DRIVE_STATUS_READY
            msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m)

NAME cdi_offline − issue an offline SCSI command to a tape device

SYNOPSIS **cdi_offline** −**f** *device* [−**v**] [−**t**{ *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_offline** program issues an offline SCSI command to a tape device. The **cdi_offline** program also returns the status of the named SCSI device (specified by the **-f** option). This operation is synonymous to issuing a load with no re-tension, and rewind to the beginning of tape SCSI command.

OPERANDS −**f** *device*

Specifies the device to send the offline request to. The −**f** option is a required option.

OPTIONS −**t** Use the −**t** option to specify the method of tape functions to use to issue the offline SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for the complete list of access methods currently supported by the *cdi_offline* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_offline -f /dev/rmt/2cbn

 CDI_OFFLINE successful.
     elapsed time for command was 11 seconds
     cdi_info.drivestat is:
         status = 0, DRIVE_STATUS_READY
         msg = null
```

SEE ALSO **libcdi**(1m)

NAME cdi_pr − issue SCSI persistent reservation commands to a tape device

SYNOPSIS

cdi_pr −**f** *device* [−**v**]

plus one of:

−**r** { *k* | *r* }

−**c**

plus one of:

r −**k** *key* [−**A**]

i −**k** *key* [−**A**]

c −**k** *key*

p −**k** *new* −**K** *old* −**t** { *e* | *E* | *A* | *w* | *W* | *a* }

a −**k** *key* −**K** *old* −**t** { *e* | *E* | *A* | *w* | *W* | *a* }

−**E** −**k** *key*

−**R** −**k** *key* −**t** { *e* | *E* | *A* | *w* | *W* | *a* }

−**Q**

DESCRIPTION The **cdi_pr** program issues various SCSI Persistent Reservation commands to a tape device. It is mainly intended as a tools for exploring the behavior of Persistent Reserve and should not normally be used for day-to-day operations.

You may also specify a persistent reservation key. This key is used to identify the host you are running on to the tape drive, and may be an 8 character text string (e.g. NetWorkr) or a text representation of a 64-bit hex number (e.g. 0x123456789abcdef0). The default reservation key is NetWorkr. This utility will always use the "exclusive access" type of persistent reservation.

The **cdi_pr** program also returns the status of the named SCSI device (specified by the **-f** option).

OPERANDS −**f** *device*

Specifies the device to send the reserve request to.

Subcommands:

−**r** {**r**|**k**}

Read a drive's current reservations (*r*) or keys (*k*) using Persistent Reserve In SCSI command.

−**c r** −**k key [**−**A]**

Send a Persistent Reserve Out register command, with option APTPL bit.

−**c i** −**k key [**−**A]**

Send a Persistent Reserve Out register command with ignore, with option APTPL bit.

−**c c** −**k key**

Send a Persistent Reserve Out clear key command.

−**c p** −**k key** −**K oldkey** −**t** { **e** | **E** | **a** | **w** | **W** | **A** }

Send a Persistent Reserve Out preempt command to preempt the reservation held by key *oldkey* and replace it with a reservation for key *key* of type specified by −**t**.

−a p −k key −K oldkey −t { **e** | **E** | **a** | **w** | **W** | **A** }
Send a Persistent Reserve Out preempt and abort command to preempt the reservation held by key *oldkey* and replace it with a reservation for key *key* of type specified by **−t** and abort any currently running tape command.

−c r −k key [−A]
Send a Persistent Reserve Out Register command for *key*, with optional APTPL bit.

−E −k key
Persistent Reserve Out Release command with specified key (confusing, isn't it?)

−R −k key −t { **e** | **E** | **A** | **w** | **W** | **a** }
Persistent Reserve Out Reserve command with specified key and reservation type

−Q Query the device's Persistent Reserve capabilities. (side effect is to clear any existing reservations and keys).

Parameters:

−k *persistent reserve key*
Specifies the key to use for a persistent reservation.

−K *persistent reserve key to preempt*
Specifies the key to preempt with this persistent reservation.

A persistent reservation key is a 64-bit value. This can hold 8 text characters or a 64-bit number. Specify either for this parameter. If the *key* entered starts with **0x** (zero x) then it is assumed to be a 64-bit number, otherwise it will be treated as an 8 character text string. The default value if you do not specify a key is **NetWorkr.**

−t *reservation type*
Specifies the type of reservation to be made. Allowed values are:

a	write exclusive - all registrants
A	exclusive access - all registrants
e	exclusive access - registrants only
E	exclusive access
w	write exclusive - registrants only
W	write exclusive

For information on those allowed values, consult a SCSI-3 specification such as ANSI NCITS 351-2001 (SPC-2) or SPC-3 working draft T10/1416-D.

Options

−v Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

Query different drives for their Persistent Reserve capabilities

cdi_pr -f /dev/rmt/1cbn -Q

```
Device /dev/rmt/1cbn (HP Ultrium 2-SCSI ):
supports Persistent Reserve but NOT Activate
Persist Through Power Loss bit
```

cdi_pr -f /dev/rmt/0cbn -Q

```
Device /dev/rmt/0cbn (HP Ultrium 1-SCSI ):
 does not seem to support Persistent Reserve at all
```

Register from this host with the key "Solaris"

cdi_pr -c r -k Solaris -f /dev/rmt/1cbn

```
CDI_PR command Register succeeds
Key "Solaris " was successfully registered
cdi_info.drivestat is:
    status = 0, DRIVE_STATUS_READY
    msg = The tape drive is ready for use
```

Read the keys from this drive

cdi_pr -f /dev/rmt/1cbn -r k

```
CDI_PR command Read Keys succeeds
Read keys returns:
    generation = 12
    data length = 16
    Keys:
        "Solaris "
        "Windows "
 cdi_info.drivestat is:
    status = 0, DRIVE_STATUS_READY
    msg = The tape drive is ready for use
```

Reserve this drive using the previously registered key of "Solaris" with reservation type of Exclusive

cdi_pr −f /dev/rmt/1cbn −R −k Solaris −t E

```
CDI_PR command Reserve succeeds
Reserve of type Exclusive Res only (3) with key "Solaris " was
successful cdi_info.drivestat is:
    status = 0, DRIVE_STATUS_READY
    msg = The tape drive is ready for use
```

Read the reservations from this drive

cdi_pr -f /dev/rmt/1cbn -r r

```
CDI_PR command Read Reservations succeeds
Read reservations returns:
    generation = 12
    data length = 16
    Reservations:
        Key: "Solaris ", type: Exclusive Res only (3),
            scope: LU, scope address: 0
cdi_info.drivestat is:
    status = 0, DRIVE_STATUS_READY
    msg = The tape drive is ready for use
```

Release the reservation of this drive that was made using the key "Solaris" of type Exclusive

cdi_pr −f /dev/rmt/1cbn −E −k Solaris −t E

```
CDI_PR command Release succeeds
Release with key "Solaris" was successful
cdi_info.drivestat is:
    status = 0, DRIVE_STATUS_READY
    msg = The tape drive is ready for use
```

If the drive is reserved by another host, you should see something like this:

cdi_pr -f /dev/rmt/1cbn -r r

```
CDI_PR command Read Reservations succeeds
Read reservations returns:
    generation = 12
    data length = 16
    Reservations:
        Key: "Windows ", type: Exclusive Res only (3),
            scope: LU, scope address: 0
cdi_info.drivestat is:
    status = 0, DRIVE_STATUS_READY
    msg = The tape drive is ready for use
```

cdi_pr −f /dev/rmt/1cbn −R −k Solaris −t E

```
CDI_PR command Reserve failed.
cdi_info.status = CDI_RESERVATION_ERROR (c)
cdi_info.drivestat is:
    status = 0, DRIVE_STATUS_READY
    msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m), **cdi_release**(1m), **cdi_reserve**(1m)

NAME cdi_release − issue a SCSI release command to a tape device

SYNOPSIS **cdi_release** −**f** *device* [−**T**{ *s* | *p* }] [−**k** *persistent reserve key*] [−**v**] [−**t**{ *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_release** program issues a SCSI release command to a tape device. This will either be a "simple" SCSI release (default or −**T s**) or a persistent reservation release if you specify −**T p.**

If you specify **Persistent** , you may also specify a persistent reservation key. This key is used to identify the host you are running on to the tape drive, and may be an 8 character text string (e.g. *NetWorkr*) or a text representation of a 64-bit hex number (e.g. *0x123456789abcdef0*). The default reservation key is *NetWorkr.* The key used (whether specified on the command line or the default) **must** match the key that was used to create the reservation. You can see any keys and persistent reservations for a particular drive using the **cdi_pr** utility. This utility will always use the "exclusive access" type of persistent reservation.

The **cdi_release** program also returns the status of the named SCSI device (specified by the **-f** option).

OPERANDS

−**f** *device*
Specifies the device to send the release request to.

−**T** *type*
Specifies the type of the release command that you wish to issue. Use 's' or 'S' for simple reserve, or 'p' or 'P' for persistent reservation release. The default is simple if you do not supply this operand.

−**k** *persistent reserve key*
Specifies the key to use for a persistent reservation release. A persistent reservation key is a 64-bit value. This can hold 8 text characters or a 64-bit number. You can specify either for this parameter. If the *key* entered starts with *0x* (zero x) then it is assumed to be a 64-bit number, otherwise it will be treated as an 8 character text string. The default value if no key is specified is *NetWorkr.*

OPTIONS

−**t** Use the −**t** option to specify the method of tape functions to use to issue the release SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Refer to the *libcdi (1m)* man page for the complete list of access methods currently supported by the *cdi_release* program.

−**v** Run the program in verbose mode. This option prints the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
CDI_RELEASE successful.
    cdi_info.drivestat is:
        status = 0, DRIVE_STATUS_READY
        msg = The tape drive is ready for use
```

If the drive is reserved by another host, you should see something like this:

```
CDI_RELEASE failed.
    cdi_info.status = CDI_RESERVATION_ERROR (c)
    cdi_info.drivestat is:
        status = 0, DEVICE_STATUS_READY
        msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m), **cdi_reserve**(1m), **cdi_pr**(1m)

NAME cdi_reserve - issue a SCSI reservation command to a tape device

SYNOPSIS **cdi_reserve** −**f** *device* [−**T** { *s* | *p* }] [−**k** *persistent reserve key*] [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_reserve** program issues a SCSI reservation command to a tape device. This will either be a "simple" SCSI reserve (default or −**T s**) or a persisitent reservation if you specify −**T p.**

If you specify **Persistent** , you may also specify a persistent reservation key. This key is used to identify the host you are running on to the tape drive, and may be an 8 character text string (e.g. NetWorkr) or a text representation of a 64-bit hex number (e.g. 0x123456789abcdef0). The default reservation key is NetWorkr. This utility will always use the "exclusive access" type of persistent reservation.

The **cdi_reserve** program also returns the status of the named SCSI device (specified by the **-f** option).

OPERANDS −**f** *device*
Specifies the device to send the reserve request to.

−**T** *type*
Specifies the type of the reservation command that you wish to issue. use 's' or 'S' for simple reserve, or 'p' or 'P' for persistent reserve. The default is simple if you do not supply this operand.

−**k** *persistent reserve key*
Specifies the key you wish to use for a persistent reservation. A persistent reservation command is a 64-bit value. This can hold 8 text characters or a 64-bit number. You can specify either for this parameter. If the *key* entered starts with **0x** (zero x) then it is assumed to be a 64-bit number, otherwise it will be treated as an 8 character text string. The default value if you do not specify a key is **NetWorkr.**

OPTIONS −**t** Use the −**t** option to specify the method of tape functions to use to issue the reserve SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for the complete list of access methods currently supported by the *cdi_reserve* program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
CDI_RESERVE successful.
    cdi_info.drivestat is:
        status = 0, DRIVE_STATUS_READY
        msg = The tape drive is ready for use
```

If the drive is reserved by another host, you should see something like this:

```
CDI_RESERVE failed.
    cdi_info.status = CDI_RESERVATION_ERROR (c)
    cdi_info.drivestat is:
```

status = 0, DEVICE_STATUS_READY
msg = The tape drive is ready for use

SEE ALSO **libcdi(1m), cdi_release(1m), cdi_pr(1m)**

NAME cdi_rewind - issue a rewind SCSI command to a tape device

SYNOPSIS **cdi_rewind** −**f** *device* [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_rewind** program issues a rewind SCSI command to a tape device. The **cdi_rewind** program also returns the status of the named SCSI device (specified by the **-f** option).

OPERANDS −**f** *device*
Specifies the device to which to send the rewind request.

OPTIONS −**t** Use the −**t** option to specify the method of tape functions to use to issue the rewind SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the **libcdi**(1m) page for the complete list of access methods currently supported by the **cdi_rewind**(1m) program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_rewind -f /dev/rmt/2cbn

CDI_REWIND successful.
    elapsed time for command was 2 seconds
    cdi_info.drivestat is:
        status = 0, DRIVE_STATUS_READY
        msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m)

NAME cdi_set_359x_eod - sets or clears IBM 3590, 3592 and TS1120 tape drive's 'Disable Crossing EOD' bit

SYNOPSIS **cdi_set_359x_eod** −**f** *device* [−**r**][−**t** {*s* | *t* | *g* | *n* | *m* | *i*}]

DESCRIPTION The **cdi_set_359x_eod** program uses mode sense and mode select commands to set a vendor-specific bit in IBM 3590, 3592 and TS1120 tape drives. These drives normally allow a program to read past the typical end of data marks on a tape which may allow a program to recover possibly overwritten data. However, this behavior confuses *NetWorker's* **scanner** utility, on some platforms causing a locked-up tape device which may require rebooting or power cycling to restore to normal operation.

If you are going to be using **scanner** on an IBM 3590, 3592 or TS1120 tape drive you should first run **cdi_set_359x_eod** to tell the tape drive that it should not allow reading past the end of data.

To restore the default state of the tape drives, you can run **cdi_set_359x_eod** with the −**r** flag which will reset (clear) the *'Disable Crossing EOD'* bit. Normal *NetWorker* operations do not seem to be affected by this bit either being set or cleared, so it is not really necessary to reset the bit after using scanner on a particular tape drive.

OPERANDS −**f** *device*

Specifies the device to perform the mode sense and mode select commands on. **cdi_set_359x_eod** will check the device's inquiry data to make sure that it is an IBM 3590, 3592 or TS1120 tape drive. If it is not, the command will exit with a message that includes the inquiry data retrieved from the specified device.

OPTIONS −**r** Tells cdi_set_359x_eod to reset the 'Disable Crossing EOD' bit back to the default (cleared) state.

−**t** Use the −**t** option to specify the method of tape functions to use to issue the SCSI mode sense and mode select commands. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the *libcdi (1m)* manpage for the complete list of access methods currently supported by the *cdi_mode_sense* program.

EXAMPLES

```
% cdi_set_359x_eod -f /dev/rmt/47cbn
359x Mode Page 0x25 Disable Crossing EOD bit succesfully SET

% cdi_set_359x_eod -f /dev/rmt/47cbn -r
359x Mode Page 0x25 Disable Crossing EOD bit successfully
    Reset to default value

% cdi_set_359x_eod -f /dev/rmt/11cbn
The drive you are working with (EXABYTE  Mammoth2) is not
    an IBM 3590, 3592 or TS1120.
```

SEE ALSO **libcdi**(1m), **cdi_mode_sense**(1m), **cdi_mode_select**(1m)

NAME cdi_set_compression - set or unset compression on a tape device

SYNOPSIS **cdi_set_compression**
−f *device* [**−v**] [**−c** {*Yes* | *y* | *1* | *No* | *n* | *0*}] [**−t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_set_compression** program sets or unsets compression on a tape device. This program is functional only on NT. On all other OS platforms, the program does nothing but returns SUCCESS.

OPERANDS **−f** *device*
Specifies the device to perform the SCSI command operation. The **−f** option is a required option.

OPTIONS **−c** *[Yes | y | 1 | No | n | 0]*
Specify whether to set or unset compression on the tape device. Use *Yes* , *y* , or *1* to set compression on the tape device. Use *No* , *n* , or *0* to unset compression on the tape device. The default is to unset compression on a tape device.

−t Use the **−t** option to specify the method of tape functions to use to set/unset device compression. If the **−t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the **libcdi**(1m) page for the complete list of access methods currently supported by the **cdi_set_compression** program.

−v Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output from the **cdi_set_compression** program:

```
% cdi_set_compression -f /dev/rmt/2cbn -c 1

   CDI_SET_COMPRESSION 0 successful.


% cdi_set_compression -f /dev/rmt/0cbn -c No

   CDI_SET_COMPRESSION 0 successful.
```

SEE ALSO **libcdi**(1m)

NAME cdi_space - provides a variety of tape positioning functions.

SYNOPSIS **cdi_space** −**f** *device* −**T** { *b* | *f* | *sf* | *eod* | *sm* | *ssm* } −**n** *count* [−**v**] [−**t** {*s* | *t* | *g* | *n* | *m* | *i*}]

DESCRIPTION The **cdi_space** program provides a variety of tape positioning operations to the user. The **cdi_space** program accepts block, filemark, setmark as valid tape positioning units. The **cdi_space** program also returns the status of the named SCSI device (specified by the **-f** option).

OPERANDS −**f** *device*
Specifies the device to perform the tape positioning operation on.

−**n** *count*
The unit count for the space SCSI command. If a count of 0 is specified, there will be no change in the tape position. If the count value is greater than 0, the tape positioning will be in the forward direction. A negative value for the count flag will cause the tape positioning to move backwards. This flag and its value are ignored if the tape positioning unit type is eod (end-of-data).

OPTIONS −**T** { *b* | *f* | *sf* | *eod* | *sm* | *ssm* }
Specify the type of space positioning unit to use. Valid types of units are:

SYMBOL	UNIT TYPE
b	block
f	filemark
sf	sequential filemark
eod	end-of-data
sm	setmark
ssm	sequential setmark

The default type is block.

−**t** Use the −**t** option to specify the method of tape functions to use to issue the space SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the **libcdi**(1m) page for the complete list of access methods currently supported by the **cdi_space** program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_space -f /dev/rmt/2cbn -T b -n 2

 CDI_SPACE 2 successful.
     elapsed time for command was 0 seconds
     cdi_info.drivestat is:
         status = 0, DRIVE_STATUS_READY
         msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m)

NAME cdi_ta - get TapeAlert information from or set TapeAlert on a tape device

SYNOPSIS **cdi_ta** −**f** *device* [−**d**] [−**i** *interval*] [−**l**] [−**m** *MRIE*] [−**n** *testflag*] [−**s**] [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_ta** program gets from or sets TapeAlert information on a tape device. The **cdi_ta** program also returns the status of the named SCSI device (specified by the **-f** option). Note that not all devices support the TapeAlert feature. If the device does not support the TapeAlert feature or the TapeAlert data returned by the device is invalid, cdi_ta will return the status CDI_IOCTL_ERROR (11). The set TapeAlert operation is currently not functional.

OPERANDS −**f** *device*
Specifies the device to send the ta SCSI command to.

OPTIONS −**d** Set the DExcept field to 1. If set to 1, disable all informational exception operations and ignore the MRIE field. The software must poll the TapeAlert log page. The default value for the DExcept field is 0.

−**i** *interval*
Set the interval timer for reporting exception conditions. If *interval* is set to 0, report informational exception conditions only once. The default value for the interval timer is 0.

−**l** Set what types of exception conditions are logged. If set to 0, which is the default, log only vendor specific exception conditions. This flag sets the log error condition to 1. The default is 0. Currently, only the values 0 and 1 are supported.

−**m** *MRIE*
Define the method used to report informational exception conditions. Values of 0x0 through 0x5 for the MRIE are defined as:

VALUE	*METHOD*
0x0	No reporting
0x1	Asynchronous event reporting
0x3	Conditionally generate recovered error
0x4	Unconditionally generate recovered error
0x5	Generate no sense
0x6	Only report informational exception conditions on request

Currently, only value 0x0 is supported. The default value for the MRIE field is 0.

−**n** *testflag*
Set or clear the TapeAlert test flag in the log page. If *testflag* is between the values 1 and 64, set the TapeAlert flag in the log page to the value of *testflag*. If *testflag* is between the values -1 and -64, clear the TapeAlert flag in the log page to the value of *testflag*. If *testflag* is equal to 32727 (0x7FFF), set all TapeAlert flags in the log page. The default value for *testflag* is 0.

−**s** Set TapeAlert data. If this flag is not specified, the program will get TapeAlert data.

−**t** Use the −**t** option to specify the method of tape functions to use for the get/set TapeAlert SCSI command. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the

libcdi(1m) page for the complete list of access methods currently supported by the **cdi_ta** program.

−v Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_ta -f /dev/rmt/2cbn

    CDI Get_TapeAlert returns (only flags that are SET will be shown):

    Tape Critical flags:

    Tape Warning flags:

    Tape Information flags:

    Changer Critical flags:

    Changer Warning flags:

    Changer Information flags:
        _info.drivestat is:
            status = 1, DRIVE_STATUS_NO_ERROR
            msg = Drive reports no error - but state is unknown
```

SEE ALSO **libcdi**(1m)

NAME cdi_tapesize - report tape capacity on a tape device

SYNOPSIS **cdi_tapesize** −**f** *device* [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_tapesize** program reports tape capacity on a tape device. The **cdi_tapesize** program also returns the status of the named SCSI device (specified by the **-f** option). Note that not all tape devices have the capability to report tape capacity. Please refer to the specific device manuals to confirm whether tape capacity reporting is available for the device.

OPERANDS −**f** *device*
Specifies the device to obtain tape capacity information from.

OPTIONS −**t** Use the −**t** option to specify the method of tape functions to use to to obtain tape capacity information. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the **libcdi**(1m) page for the complete list of access methods currently supported by the **cdi_tapesize** program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_tapesize -f /dev/rmt/2cbn
cdi_cmd failed - cdi_info.status = CDI_UNSUPPORTED_CMD (5)
errormsg = command is not supported by the selected target
    cdi_info.drivestat is:
        status = 1, DRIVE_STATUS_NO_ERROR
        msg = Drive reports no error - but state is unknown
```

SEE ALSO **libcdi**(1m)

NAME cdi_tur - send a test unit ready SCSI command to a tape device

SYNOPSIS **cdi_tur** −**f** *device* [−**v**] [−**t** { *s* | *t* | *g* | *n* | *m* | *i* }]

DESCRIPTION The **cdi_tur** program sends a test unit ready SCSI command to a tape device. The **cdi_tur** program also returns the status of the named SCSI device (specified by the **-f** option).

OPERANDS −**f** *device*
Specifies the device to send the SCSI command to.

OPTIONS −**t** Use the −**t** option to specify the method of tape functions to use to to perform the operation. If the −**t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions. Please refer to the **libcdi**(1m) page for the complete list of access methods currently supported by the **cdi_tur** program.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% cdi_tur -f /dev/rmt/2cbn
CDI_TUR successful.
    cdi_info.drivestat is:
        status = 0, DRIVE_STATUS_READY
        msg = The tape drive is ready for use
```

SEE ALSO **libcdi**(1m)

NAME changers − list SCSI autochangers attached to the system

SYNOPSIS **changers** [−**dpv**] [−**a** *b.t.l* | −**f** *filename*] [−**l**]

DESCRIPTION The **changers** program lists all of the SCSI autochangers (jukeboxes) connected to the current system.

OPTIONS

−**a** *b.t.l* Selects a specific ordinal SCSI address, where **b** is the logical SCSI bus, **t** is the SCSI target, and **l** is the SCSI logical unit number (LUN) on that target. See **libscsi**(1m).

−**f** *filename*
Specifies an explicit device file name for **changers** to use on platforms that support direct use of device file names for jukeboxes. At this time those platforms are Solaris 10+, AIX and Linux

−**d** Determines the names and addresses of the autochanger's media elements (for example, tape drives).

−**l** Performs a complete LUN search for all SCSI adapters in the system. This argument is accepted on all systems, but does not have any effect on HP-UX systems. Due to the method used to scan for available devices on HP-UX systems, all accessible devices are always shown, and the −**l** option has no additional effect. On all other platforms, the normal behavior is to start checking at LUN 0 for SCSI devices. The first empty LUN found will end the search for a given target ID. With the −**l** option, all LUN present on all target IDs for all SCSI busses in the system will be checked for devices. This can take a **very long time** and should therefore only be used when necessary. For example, a Fibre Channel adapter can support 126 target IDs, each of which may have 80 or more LUNs. Checking all LUNs on this single adapter may take over 10 minutes.

−**p** Tells changers to use persistent device names for jukeboxes on platforms where persistent names are supported. Currently only linux has such support.

−**v** Lists more detailed information about each autochanger. The details indicate how many media transports (MT, for example, robot arm), storage transports (ST, for example, slot), import/export elements (IE, for example, mail slot), and data transport (DT) elements the autochanger contains. The −**v** option also provides information about the element movement matrix supported by the autochanger.

EXAMPLE Sample output is shown below: hal$ changers -dv -a 0.2.0 *scsidev@0.2.0:Vendor <SPECTRA>, Product <4000>*

Data Transfer Element at address 80 is scsidev@0.5.0
Device:Vendor <HP>, Product <C1533A>
Type:Tape
System Name: /dev/rmt2.1
Data Transfer Element at address 81 is scsidev@0.6.0
Device:Vendor <HP>, Product <C1533A>
Type:Tape
System Name: /dev/rmt3.1

1 MT Element starting at address 79
60 ST Elements starting at address 1

```
1 IE Element  starting at address 0
2 DT Elements starting at address 80

                Element Movement Matrix

  ->DT,  ->IE,  ->ST,  ->MT
MT->DT,MT->IE,MT->ST,_____
ST->DT,ST->IE,ST->ST,ST->MT
IE->DT,_____,IE->ST,IE->MT
DT->DT,DT->IE,DT->ST,DT->MT
_____,_____,_____,_____
_____,_____,_____,_____
_____,_____,_____,_____
_____,_____,_____,_____
```

SEE ALSO **libscsi**(1m)

NAME dasadmin − ADIC/EMASS/Grau silo administrative utility
libstlemass − shared library for communication to
ADIC/EMASS/Grau silo

SYNOPSIS **dasadmin** command [options] [parameters]
dasadmin.exe command [options] [parameters] (NT only)
libstlemass.so (Solaris)
libstlemass.so.a (AIX)
libstlemass.sl (HPUX)
libstlemass.so.1 (SGI)
libstlemass.so (DECAXP)
libstlemass.dll (NT i386)

DESCRIPTION **dasadmin**

This is not a complete listing of all possible **dasadmin** commands, but does include those commands that are of use with NetWorker. For a complete discussion, see the DAS Installation and Administration guide provided by ADIC, EMASS or Grau.

mo[unt] [−**t** *type*] *volser* [*drive-name*]
Mounts the tape with the barcode label of *volser* into either the first available drive (if *drive-name* is not specified) or into the drive specified by *drive-name.* If the tape is not the type defined by **DAS_MEDIUM** or **ACI_MEDIA_TYPE,** you can use the −**t** *type* option to get the tape mounted. If the type of the tape and the defined type for the drive do not match, the silo will not load the tape. Note that the drive you are attempting to use must be allocated for your use before you can mount or dismount tapes. See **listd** and **allocd** below.

dism[ount] [−**t** *type*] *volser* | −**d** *drive-name*
Dismounts the tape that is either specified by *volser* or whatever is in the drive specified by *drive-name.* If the tape or drive are of a different type than your default, use the −**t** *type* parameter. As with **mount,** you must have the drive allocated to you to use this command.

ej[ect] [−**c**] [−**t** *type*] *volser-range area-name*
Ejects one or more tapes to the specified eject area. As with other commands, if the type of the tape you are ejecting is different from that defined by **DAS_MEDIUM** or **ACI_MEDIA_TYPE,** you will need the −**t** *type* option. The −**c** specifies a 'complete' ejection for the specified volsers. A complete ejection removes the entry for that volser from the silo controller's internal database. A NON-complete ejection will eject the tape, but the volser's entry in the database will remain, and the volser's state will be set to 'ejected'. This is useful if you anticipate replacing the tape in the silo soon.

in[sert] *area-name*
Moves all tapes that are currently in the specified insert *area-name* from the insert area to the normal storage locations for tapes.

inventory
Starts a full inventory of the silo. **USE WITH CAUTION!** An inventory of this sort can take a very long time! An inventory of a silo with 180 slots takes over 20 minutes.

view [−**t** *type*] *volser*
Displays the current status of *volser,* including the volser, type, attribute, and coordinate.

all[ocd] *drive-name* **UP|DOWN** *clientname*
The **allocd** command is used to allocate and deallocate drives for different

clients. Before you can use a tape drive, the drive must be **allocd'ed UP** for your system. If it is currently **allocd'ed UP** for a different client, it must first be **allocd'ed DOWN** for that client before being **allocd'ed UP** for your system. You cannot **allocd DOWN** a drive that has a tape in it. The tape must be dismounted first.

l[ist]d
listd or **ld** shows the current state of all the tape drives defined in the silo. The information presented will include the *drive-name,* the amu drive (the location in the silo), status (UP or DOWN), type, client the drive is allocated to, and the volser of any loaded tape.

show −**op** | −**ac** *client-name*
Shows the operational or access parameters for the specified *client-name.* You must include either −**ac** if you wish to see access parameters, or −**op** if you wish to see operational parameters for the *client-name.* Access parameters include volser ranges and drive ranges that the *client-name* is allowed to use. Operational parameters include whether the *client-name* has complete access, dismount privileges along with the IP address entered for *client-name.*

list *client-name*
Lists any outstanding requests that have been made by *client-name.* If there are any, they are shown, along with the request number and type.

can[cel] *request-id*
Allows you to cancel an outstanding request, assuming that you have the necessary privileges. Use the *request-id* that was shown by the **list** command.

qversion
Shows the version of the DAS server that you are connected to and the version of the ACI protocol you are using to talk to DAS.

qvolsrange *beginvolser endvolser count* [*clientname*]
qvolsrange is the way to obtain a list of the volsers that are available in the silo. *beginvolser* and *endvolser* are volsers of the form "123456". To use the first available or the last available, you can use "". *count* specifies the maximum number of volsers you wish to see.

ENVIRONMENT VARIABLES

These environment variables affect the operation of the silo, and since the processes that are using them include both the commands the user will enter and the processes that are spawned from **nsrd,** they need to be set in a location where they will be in place when **nsrd** is started. The three DAS_ variables are used by **libstlemass,** while **dasadmin** uses **ACI_MEDIA_TYPE** instead of **DAS_MEDIUM.**

For Solaris, the definitions should be placed in **/etc/rc.2/S95networker.**

For AIX, the definitions should be placed in **/etc/rc.nsr.**

For HPUX, the definitions should be placed in **/sbin/rc2.d/S900networker.**

DAS_SERVER
This is either the network name or the IP address of the system that is running DAS. For a single silo, this will usually be the silo controller system. In larger installations, there will probably be only one DAS server for the whole network. It is case-sensitive.

DAS_CLIENT
This is the network name of the system that NetWorker is running on. It is case-sensitive.

DAS_MEDIUM
This variable is used by **libstlemass.** It should be the same as ACI_MEDIA_TYPE. This is the type of tape drive you are connected to. If this is not specified, the default value of DLT will be used.

ACI_MEDIA_TYPE
This variable is used by **dasadmin.** It should be the same as **DAS_MEDIUM.** This is the type of tape drive you are connected to. If this is not specified, the default value of DLT will be used. Acceptable values are the same as those listed under **DAS_MEDIUM.**

EXAMPLES

NOTE on ranges:
The **dasadmin** utility will accept volser ranges for some commands. There are three acceptable variations for these ranges:
single volser: "000635"
multiple volsers: "000635, 000789, 098732"
true range: "000610 - 000745"

NOTE on *area-name* and *drive-name*:
area-names usually consist of a letter and 2 digits. The letter corresponds to whether you are referring to an insert area ("I") or an eject area ("E"). You will need to get the correct values from your silo administrator before using them.
drive-names are essentially free-form labels created by whomever installed the silo. They may or may not have any relevance to physical reality, so you will need to see the silo admin to get the correct names. If the silo admin is not available, you can get the same information using **dasadmin listd** along with **dasadmin show −op** *client-name* followed by **dasadmin show −ac** *client-name* commands.

To set up the environment variables necessary for silo operations:

```
setenv DAS_SERVER emask
setenv DAS_CLIENT aurora
setenv DAS_MEDIUM DLT
setenv ACI_MEDIA_TYPE DECDLT
```

To see a listing of all volsers available in the silo:

```
dasadmin qvolsrange "" "" 10000
```

To see the current status of the drives in the silo:

```
dasadmin listd
```

To change the allocation of a drive from client a4 to client aurora:

```
dasadmin allocd DLT1 DOWN a4
dasadmin allocd DLT1 UP aurora
```

SEE ALSO

nsrjb(1m), **jbconfig**(1m), **libstlstk**(1m), **mini_el**(1m), **ssi**(1m), **libstlibm**(1m)

DIAGNOSTICS

The only available diagnostic information is error messages that might be printed out by **dasadmin** and **libstlemass** in the course of normal operations.

NAME ddmgr − device detection manager that manages auto-detection on local and remote storage nodes

SYNOPSIS **ddmgr** [−**S**] [−**M**] [−**i**] [−**d**] [−**q**] [−**v**]

DESCRIPTION **ddmgr** is the main daemon for auto-detection that runs on the NetWorker server machine. It spawns child processes (of dvdetect) for each storage node on which devices are to be detected.

It starts the child processes with the help of the nsrmon(1m) process, and depends on nsrmon to report on the success or failure of the remote dvdetect process.

Once dvdetect on a storage node has finished its work of detecting devices, ddmgr takes up the task of creating resources for these detected devices, and in case of jukeboxes, tries to find out the device mapping (element id to device path) by spawning another process, dtbind. dtbind determines the device mapping by loading each drive in the jukebox that was detected and then trying to access it via various device paths till it finds the right one. This might take a long time depending on the type of the jukebox.

ddmgr is invoked by the nsrd process and is not to be invoked on the command-line.

OPTIONS

−**d** Tells **ddmgr** to detect and create device resources but not to enable them.

−**i** This option tells **ddmgr** to look for silos.

−**M** This option tells **ddmgr** that it has been invoked by the server and to direct messages to the daemon log.

−**q** This option tells ddmgr to run in the 'quiet' mode without printing any messages.

−**v** This option is used to run ddmgr in the verbose mode for more debug messages.

EXIT STATUS Exits with 0 on success and 1 on error. See error messages for more detail on errors.

SEE ALSO **nsrmon**(1m), **nsr_render_log**(1m)

DIAGNOSTICS Most, if not all, of ddmgr error reports is preceded by the phrase "Detection process for host X reports", followed by the actual error message. This error message is based on the error reported by the nsrmon process monitoring the dvdetect process, or in cases where nsrmon itself cannot be started, about the nsrmon process. The following are the error messages that ddmgr might produce along with their implications and possible solutions.

remote dvdetect exec failure. Errno 76
The remote storage node was unable to start the dvdetect process on the remote storage node. This could happen for various reasons, like the dvdetect binary not having execute permissions, or more commonly, the remote storage node not being configured to service requests from this server.

remote auto-detect feature not supported
Auto-detect was being performed on a host that does not support this feature. The client/storage-node should be 6.x or higher.

dvdetect process failed on signal
The remote dvdetect process was killed by a signal. This could happen even when the process encounters a memory fault. Check for core files in the nsr/cores directory.

dvdetect terminated due to timeout
The dvdetect process was terminated because of its inactivity for a certain period of time. The timeout is set by default to 15 minutes. This is not user configurable. **dvdetect process exited on signal** The local dvdetect process was killed by a signal. This could happen even when the process encounters a memory fault. Check for core files in the nsr/cores directory.

dvdetect exec failure
The ddmgr process was unable to start the dvdetect process on the server. Check for execute permissions on the dvdetect binary.

nsrmon exec failure
The ddmgr process was unable to start the nsrmon process on the server. Check for execute permissions on the nsrmon binary.

nsrmon process exited on signal
The nsrmon process exited on a signal. This could even happen when the process encounters a memory fault. Check the nsr/cores directory for a core file.

dvdetect failed with unknown error
ddmgr was unable to determine the cause of the failure of the dvdetect process.

nsrmon failed. No info in the resdb
The nsrmon process exited without loggin any information about either the remote dvdetect process or itself. Ddmgr is unable to verify status of both.

nsrmon failed. Invalid request or hostname
The nsrmon process was started with an invalid option or hostname. Check if the remote storage node is reachable from the server.

nsrmon failed. Authorization failure
The nsrmon process could not get authorization from the NetWorker server to talk to the remote storage node.

nsrmon exited on resdb access failure
The nsrmon process encountered errors in reading the NetWorker RAP database.

nsrmon exited on memory failure
The nsrmon process ran out of physical memory while processing. Add more memory.

nsrmon failed. Invalid request value
The nsrmon process was asked to perform a request it is not familiar with.

process exited with error
There was a problem with the detection process but ddmgr could not determine the exact cause of the failure.

RPC error Remote systems
The nsrmon process was unable to connect to the remote host. This could be because of network problems, or if the NetWorker processes were not installed on the remote system.

NAME dasadmin − ADIC/EMASS/Grau silo administrative utility
libstlemass − shared library for communication to
ADIC/EMASS/Grau silo

SYNOPSIS **dasadmin** command [options] [parameters]
dasadmin.exe command [options] [parameters] (NT only)
libstlemass.so (Solaris)
libstlemass.so.a (AIX)
libstlemass.sl (HPUX)
libstlemass.so.1 (SGI)
libstlemass.so (DECAXP)
libstlemass.dll (NT i386)

DESCRIPTION **dasadmin**

This is not a complete listing of all possible **dasadmin** commands, but does include those commands that are of use with NetWorker. For a complete discussion, see the DAS Installation and Administration guide provided by ADIC, EMASS or Grau.

mo[unt] [−**t** *type*] *volser* [*drive-name*]

Mounts the tape with the barcode label of *volser* into either the first available drive (if *drive-name* is not specified) or into the drive specified by *drive-name.* If the tape is not the type defined by **DAS_MEDIUM** or **ACI_MEDIA_TYPE,** you can use the −**t** *type* option to get the tape mounted. If the type of the tape and the defined type for the drive do not match, the silo will not load the tape. Note that the drive you are attempting to use must be allocated for your use before you can mount or dismount tapes. See **listd** and **allocd** below.

dism[ount] [−**t** *type*] *volser* | −**d** *drive-name*

Dismounts the tape that is either specified by *volser* or whatever is in the drive specified by *drive-name.* If the tape or drive are of a different type than your default, use the −**t** *type* parameter. As with **mount,** you must have the drive allocated to you to use this command.

ej[ect] [−**c**] [−**t** *type*] *volser-range area-name*

Ejects one or more tapes to the specified eject area. As with other commands, if the type of the tape you are ejecting is different from that defined by **DAS_MEDIUM** or **ACI_MEDIA_TYPE,** you will need the −**t** *type* option. The −**c** specifies a 'complete' ejection for the specified volsers. A complete ejection removes the entry for that volser from the silo controller's internal database. A NON-complete ejection will eject the tape, but the volser's entry in the database will remain, and the volser's state will be set to 'ejected'. This is useful if you anticipate replacing the tape in the silo soon.

in[sert] *area-name*

Moves all tapes that are currently in the specified insert *area-name* from the insert area to the normal storage locations for tapes.

inventory

Starts a full inventory of the silo. **USE WITH CAUTION!** An inventory of this sort can take a very long time! An inventory of a silo with 180 slots takes over 20 minutes.

view [−**t** *type*] *volser*

Displays the current status of *volser,* including the volser, type, attribute, and coordinate.

all[ocd] *drive-name* **UP**|**DOWN** *clientname*

The **allocd** command is used to allocate and deallocate drives for different

clients. Before you can use a tape drive, the drive must be **allocd'ed UP** for your system. If it is currently **allocd'ed UP** for a different client, it must first be **allocd'ed DOWN** for that client before being **allocd'ed UP** for your system. You cannot **allocd DOWN** a drive that has a tape in it. The tape must be dismounted first.

l[ist]d
listd or **ld** shows the current state of all the tape drives defined in the silo. The information presented will include the *drive-name,* the amu drive (the location in the silo), status (UP or DOWN), type, client the drive is allocated to, and the volser of any loaded tape.

show −**op** | −**ac** *client-name*
Shows the operational or access parameters for the specified *client-name.* You must include either −**ac** if you wish to see access parameters, or −**op** if you wish to see operational parameters for the *client-name.* Access parameters include volser ranges and drive ranges that the *client-name* is allowed to use. Operational parameters include whether the *client-name* has complete access, dismount privileges along with the IP address entered for *client-name.*

list *client-name*
Lists any outstanding requests that have been made by *client-name.* If there are any, they are shown, along with the request number and type.

can[cel] *request-id*
Allows you to cancel an outstanding request, assuming that you have the necessary privileges. Use the *request-id* that was shown by the **list** command.

qversion
Shows the version of the DAS server that you are connected to and the version of the ACI protocol you are using to talk to DAS.

qvolsrange *beginvolser endvolser count* [*clientname*]
qvolsrange is the way to obtain a list of the volsers that are available in the silo. *beginvolser* and *endvolser* are volsers of the form "123456". To use the first available or the last available, you can use "". *count* specifies the maximum number of volsers you wish to see.

ENVIRONMENT VARIABLES

These environment variables affect the operation of the silo, and since the processes that are using them include both the commands the user will enter and the processes that are spawned from **nsrd,** they need to be set in a location where they will be in place when **nsrd** is started. The three DAS_ variables are used by **libstlemass,** while **dasadmin** uses **ACI_MEDIA_TYPE** instead of **DAS_MEDIUM.**

For Solaris, the definitions should be placed in **/etc/rc.2/S95networker.**

For AIX, the definitions should be placed in **/etc/rc.nsr.**

For HPUX, the definitions should be placed in **/sbin/rc2.d/S900networker.**

DAS_SERVER
This is either the network name or the IP address of the system that is running DAS. For a single silo, this will usually be the silo controller system. In larger installations, there will probably be only one DAS server for the whole network. It is case-sensitive.

DAS_CLIENT
This is the network name of the system that NetWorker is running on. It is case-sensitive.

DAS_MEDIUM
This variable is used by **libstlemass.** It should be the same as ACI_MEDIA_TYPE. This is the type of tape drive you are connected to. If this is not specified, the default value of DLT will be used.

ACI_MEDIA_TYPE
This variable is used by **dasadmin.** It should be the same as **DAS_MEDIUM.** This is the type of tape drive you are connected to. If this is not specified, the default value of DLT will be used. Acceptable values are the same as those listed under **DAS_MEDIUM.**

EXAMPLES

NOTE on ranges:
The **dasadmin** utility will accept volser ranges for some commands. There are three acceptable variations for these ranges:
single volser: "000635"
multiple volsers: "000635, 000789, 098732"
true range: "000610 - 000745"

NOTE on *area-name* and *drive-name*:
area-names usually consist of a letter and 2 digits. The letter corresponds to whether you are referring to an insert area ("I") or an eject area ("E"). You will need to get the correct values from your silo administrator before using them.
drive-names are essentially free-form labels created by whomever installed the silo. They may or may not have any relevance to physical reality, so you will need to see the silo admin to get the correct names. If the silo admin is not available, you can get the same information using **dasadmin listd** along with **dasadmin show −op** *client-name* followed by **dasadmin show −ac** *client-name* commands.

To set up the environment variables necessary for silo operations:

```
setenv DAS_SERVER emask
setenv DAS_CLIENT aurora
setenv DAS_MEDIUM DLT
setenv ACI_MEDIA_TYPE DECDLT
```

To see a listing of all volsers available in the silo:

```
dasadmin qvolsrange "" "" 10000
```

To see the current status of the drives in the silo:

```
dasadmin listd
```

To change the allocation of a drive from client a4 to client aurora:

```
dasadmin allocd DLT1 DOWN a4
dasadmin allocd DLT1 UP aurora
```

SEE ALSO

nsrjb(1m), **jbconfig**(1m), **libstlstk**(1m), **mini_el**(1m), **ssi**(1m), **libstlibm**(1m)

DIAGNOSTICS

The only available diagnostic information is error messages that may be printed out by **dasadmin** and **libstlemass** in the course of normal operations.

NAME erase − erase a tape

SYNOPSIS **erase** [**-sr**] **-a** *b.t.l*

DESCRIPTION The **erase** program will send the SCSI ERASE command to the named device, using the LONG erase option unless the optional **-s** argument is specified.

OPTIONS

−**s** Uses the SHORT erase option, rather than the LONG option. LONG is used by default.

−**r** sends a REWIND command to the named device prior to issuing an erase command.

−**a** This is a required argument, and must be used to select a specific ordinal SCSI address (see **libscsi**(1m)) for the device that has the tape.

WARNINGS Be careful! This command destroys data! It does **not** prompt you to see whether you are sure you want to do this.

SEE ALSO **libscsi**(1m)

NAME generate_test_tape - perform generates a test tape for diagnostic purposes.

SYNOPSIS **generate_test_tape** −**f** *device* [−**z** *blocksize*] [−**s** *filesize*] [−**b** *maxblocks*] [−**m** *maxfiles*] [−**v**]

DESCRIPTION The **generate_test_tape** program generates a test tape mounted on a device for diagnostic purposes. 32 KB blocks are first written to the tape mounted on the device, with a filemark at every N number of blocks it has completed writing the total number of blocks specified by the user, or till it reaches end of tape.

OPERANDS −**f** *device*
Specifies the device to generate the test tape on.

OPTIONS −**b** *maxblocks*
Use the −**b** option to specify the maximum number of blocks to write to tape. The value of *maxblocks* must be greater than 0. If the −**b** and the −**m** options are not specified, the program will write to the end of tape or till a write error is encountered.

−**m** *maxfiles*
Use the −**m** option to specify the maximum number of files to write to tape. The value of *maxfiles* must be greater than 0. If the −**b** and the −**m** options are not specified, the program will write to the end of tape or till a write error is encountered.

−**s** *filesize*
Use the −**s** option to specify the file size (in number of blocks) to write to tape. The value of *filesize* must be greater than 0. The default file size is 1000 32KB blocks.

−**sz** *blocksize*
Use the −**s** option to specify the block size (in number of 1KB) to write to tape. The value of *blocksize* must be greater than 0. The default block size is 1000 32KB.

−**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% generate_test_tape -f /dev/rmt/3cbn -b 20 -s 2 -v
    ready to fill tape on QUANTUM DLT7000
            using device file /dev/rmt/2cbn
    each tape record will be 32768 bytes
    a filemark will be written every 2 records
    the process will end when 20 total records have been written to the tape
block = 1.  Buffer = 1 1 1 1
|FM|
block = 2.  Buffer = 2 2 2 2
block = 3.  Buffer = 3 3 3 3
|FM|
block = 4.  Buffer = 4 4 4 4
block = 5.  Buffer = 5 5 5 5
|FM|
block = 6.  Buffer = 6 6 6 6
block = 7.  Buffer = 7 7 7 7
```

```
|FM|
block = 8.  Buffer = 8 8 8 8
block = 9.  Buffer = 9 9 9 9
|FM|
block = 10.  Buffer = a a a a
block = 11.  Buffer = b b b b
|FM|
block = 12.  Buffer = c c c c
block = 13.  Buffer = d d d d
|FM|
block = 14.  Buffer = e e e e
block = 15.  Buffer = f f f f
|FM|
block = 16.  Buffer = 10 10 10 10
block = 17.  Buffer = 11 11 11 11
|FM|
block = 18.  Buffer = 12 12 12 12
block = 19.  Buffer = 13 13 13 13
|FM|
```

SEE ALSO **libcdi**(1m)

NAME gstclreport − NetWorker Management Console command-line reporting utility

SYNOPSIS **gstclreport**
−**r** *reportname* −**u** *username* [−**P** *password*] [−**a** *chartselector*] [−**c** *charttype*] [−**f** *filename*] [−**n** *fontfamily*] [−**o** *orientation*] [−**v** *viewtype*] [−**x** *exporttype*] [−**C** *parameter_name parameter_value*]

DESCRIPTION **gstclreport** provides a command-line interface for running reports. The required option -u must specify a valid NetWorker Management Console user name. The report will be run using this user's login credentials and is subject to any permission restrictions placed on this user. The optional -P option can be used to specify the password for this user. If the -P option is omitted, then **gstclreport** will prompt for the password at the command line.

The reports that ship with NetWorker Management Console are called canned reports. They cannot be deleted. Users may create their own versions of these reports using the Console UI. Reports created by users appear under the canned report they were created from and are called custom reports. Custom reports are privately owned by users, and can be shared with others via the share command in the UI. Use the -r option to specify which report will be ran.

The final result of running **gstclreport** will be the exported output of the report. Various command line options can be used to configure the output. If an argument to an option contains a white space character, then that argument must be enclosed in quotes. For example, the -c option can be used to set the type of chart in a chart report. In order to get a stacked bar chart the option must be specified -c "stacking bar".

When running a canned report, optional options that are omitted will assume a default value. For instance the default value for *viewtype* in the -v option is table. When a custom report is run, values will be pulled from the custom report instead.

OPTIONS

-r *reportname* This required option specifies the name of the report to run. This may be either a canned report, or a custom report created by a user. Drill down reports of either kind cannot be run from **gstclreport.** If this is a custom report, then the user specified in the -u option must have permission to view this report. Permission is granted either because that user owns the report, or because that report has been marked as shared.

The name of the report must be the full path name to the report from the report hierarchy in the UI. For example:

"/Reports/Users/User Audit/West Coast Admins".

Rather than specify the full path report name, the user may elect to specify just the report name. If there is more than one report of that name in the system, we will run the first report we find of that name. For example, "West Coast Admins".

-u *username* This required option specifies the name of a NetWorker Management Console user. The report will be run as this user and is subject to any permission restrictions placed on that user. Users may only run reports they have permission to see either because they own the report, or the report is marked as shared.

When the report is run it is also subject to any host permission

restrictions place upon the specified user. Thus two different users with differing permissions may run the same report and get different results because of restrictions.

-P *password* This option should contain the password of the user specified in the -u option. If the -P option is omitted, then **gstclreport** will prompt for the password at the command line in order to continue.

-a *chartselector*
This option specifies a comma-separated list of the Y axis to display when viewing a chart report. The values in the list should match the values found in the Chart Selector input of the chart report in the UI for the particular report being run. These values will differ depending upon which report.

-c *charttype* This option specifies the type of chart to display for chart reports. Valid values are bar, pie, plot, and "stacking bar". The default chart type is bar. If this option is specified and no -v option is specified we will default to a chart view.

-f *filename* This option specifies the filename of the exported report output. This can be a full path filename, or a filename relative to the current directory. If this option is omitted, then the filename will be generated by taking the name of the report from the -r option and replacing all white space characters with underscore. The filename will have the correct file type extension added if it is missing.

When exporting chart reports to html, besides the html file generated we also create a directory which contains the chart image files. This directory name is created by taking the name of the filename without the extension and appending "_images" to it.

-n *fontfamily* This option specifies the name of a font family to override the default font used in this report. This font name should match a name from the View->Font->Font Name selector in the UI.

-o *orientation* This option specifies the page orientation to use when exporting the report. Valid options are portrait and landscape. The default orientation is portrait.

-v *viewtype* This option specifies the type of view for this report. Valid options are table and chart. The default view type is table.

-x *exporttype* This option specifies the export format of the report. Valid options are pdf, postscript, html, csv, and print. The default export type is pdf.

-C *parameter_name parameter_value*
A report may have associated with it a set of configuration parameter options. These options match the Parameters found on the Configure tab of the report. Each report will have a different set of configuration parameters and thus a different set of configuration parameter options.

The -C option contains two parts. The *parameter_name* should contain the name of a configuration parameter from the report's configure tab. The *parameter_value* should contain the input to the parameter option. The format of the *parameter_value* will vary on the type of control used in the UI. You can get further clarification by running the -h option to see all available -C options and their input types.

There are 3 possible types of inputs, either a single value, a comma-

separated list of values, or a date range. The date range may contain either one or two dates. If only one date is specified, it is assumed to be the from date. If there are two dates the first is the from date, and the second is the to date. The first date can be the special string "epoch" which indicates that the from date should be left empty.

Date parsing is performed in the current locale. A best attempt will be made to parse dates. Dates contain both a date and time portion, the time portion is optional. In the US locale, the following date portions will always be supported:

Format	Example
MM/DD/YY	07/25/04
MMM D, YYYY	Jul 25, 2004
MMMM D, YYYY	July 25, 2004
EEEE, MMMM D, YYYY	Sunday, July 25, 2004

Each locale will have a different set of supported formats. A best attempt has been made to support some variation of the MM/DD/YY format for each locale. For this format, some locales may have the day field before the month field. Others may choose to use "-" as the date field separator.

The time portion will either be in 24 hour time or 12 hour time depending upon locale. For the US locale, the following time formats will be supported:

Format	Example
h:mm a	11:27 AM
h:mm:ss a	11:27:03 AM
h:mm:ss a Z	11:27:03 AM PST

Each locale will have a different set of supported formats.

Instead of an absolute date, the from or to dates may contain a relative date. A relative date consists of a number greater than or equal to 0 followed by one of the following strings: hours, days, weeks, months, or years. The actual date is then derived by taking the current date and subtracting the indicated number of relative time.

-h When this option is used no report output is generated. Instead a usage statement is output to the command line. If this option is used along with the -u and -r options and either the -P option or a valid password is entered at the command prompt, then it will also output the set of configuration parameter options available through the -C option for this report.

EXAMPLES The example below shows how **gstclreport** is used to output a full usage statement to the command line for a report including parameter options.

```
% gstclreport -u username -P password -r "Server Summary" -h
usage: gstclreport [-h] -r reportname -u username
  [-P password] [-a chartselector] [-c chartype]
  [-f filename] [-n fontfamily] [-o orientation]
  [-v viewtype] [-x exporttype] [-C "Backup Type" argument]
  [-C Level argument] [-C "Save Time" argument]
  [-C "Server Name" argument]
where:
  -h                  Print this help message
  -r  reportname      The full path of the report to run like
                      "/Reports/Users/User List"
  -u  username        Log into GST server with given name
  -P  password        Log into GST server with given password
  -a  chartselector   The set of Y axis to display in a chart
  -c  chartype        The chart type [bar | pie | plot | "stacking
                      bar"]
  -f  filename        The name of the export file
  -n  fontfamily      A font family to override the default
  -o  orientation     The orientation [portrait | landscape]
  -v  viewtype        The view type [table | chart]
  -x  exporttype      The type of export [ pdf | postscript | html
                      | csv | print]
  -C  "Backup Type" argument  Where argument is a comma-separated
                              list of Backup Types
  -C  Level argument          Where argument is a comma-separated
                              list of Levels
  -C  "Save Time" argument    Where argument is a From and To date
  -C  "Server Name" argument  Where argument is a comma-separated
                              list of Server Names
```

The next example shows how to use **gstclreport** to run a canned report. The Client Summary report will be displayed as a table. The report will be configured to be run over a set of groups and from a certain date. The name of the output file has been derived from the name of the report.

```
% gstclreport -u username -P password
  -r "/Reports/NetWorker Backup Statistics/Client Summary"
  -C "Group Name" "Default, Nightly, Marketing, Building A,
  Building B"
  -C "Save Time" "01/01/2003 01:00 AM"
Generated Report "/Reports/NetWorker Backup Statistics/Client
  Summary" as file Client_Summary.pdf
```

The next example shows how to use **gstclreport** to run a custom report. Also note that we are using the relative date format input to the Save Time option. This means that the report will be run over a date range starting from 1 day ago up until right now. The report output will be a pie chart exported to html. The html output will be an html file named DailyGroups.html, and a directory named DailyGroups_images which will contain the chart images.

```
% gstclreport -u username -P password
  -r "/Reports/NetWorker Backup Statistics/Group Summary/Daily
        Group Report"
  -v chart -c pie -x html -f "DailyGroups" -C "Save Time" "1 day"
Generated Report "/Reports/NetWorker Backup Statistics/Group
```

Summary/Daily Group Report" as file DailyGroups.html

Command line options are checked to ensure they have legal values. This includes values passed into the -C option. This example shows a -C "Group Name" option. The **gstclreport** program will check to ensure that the group names listed not only exist, but that the current user has access to the NetWorker server that they exist on. All improper values will be ignored in generating the report, and these values will be printed in an informational message. In this example, the user does not have permission to view the NetWorker server containing the Marketing group, and the group Blah does not exist.

```
% gstclreport -u username -P password -r "Group Summary"
  -C "Group Name" "Default, Nightly, Marketing, Blah"

These configuration values were ignored:
  Group Name : Marketing, Blah
Generated Report "Group Summary" as file Group_Summary.pdf
```

Custom Reports are subject to user restrictions on their configuration parameters. Each custom report can contain configuration parameters that come from the Configuration tab of the report when the report is saved. These parameters values are checked against the user permissions of the user running the current report. If some of these requested parameter values belong to a NetWorker server the current user does not have permission for, then the report will be generated, but these values will not be used in generating the report. In this situation an informational message will be printed to the command line as in the example below.

```
% gstclreport -u username -P password -r "Other Group Summary"
Some report results were not displayed due to user restrictions
Generated Report "Other Group Summary" as file
Other_Group_Summary.pdf
```

EXIT STATUS If a fatal error occurs, the exit status is non-zero. If no fatal errors occur, the exit status is zero.

NAME gstd − GST server daemon

SYNOPSIS **gstd** [−**m** *module_path*] [−**n**]

DESCRIPTION **gstd** is the Generic Services Toolkit (GST) server program. It provides an RPC-based messaging server for NetWorker Management Console and related applications. The RPC program number provided by **gstd** is 390402.

Normally **gstd** is invoked from a startup shell script (for example, **/etc/init.d/gst**) at boot time, and should never need to be started directly by a user.

gstd must be run on a machine with appropriate resources. In the context of NetWorker Management Console, the program will acquire connections to NetWorker servers on the network. The process of managing and collecting report data for any number of servers requires a proportionate level of network bandwidth, CPU time and disk space.

OPTIONS

−**m** *module_path* A list of semicolon (;)-separated directories containing GST loadable modules is specified in *module_path*.

−**n** Remain in the foreground connected to the controlling terminal. Without this option, the default behavior is to disconnect from the controlling terminal and to run in the background as a daemon process.

FILES

<**product install dir**>**/etc/gstd.conf**
The master GST configuration file.

EXIT STATUS

0 Successful completion.

>**0** An error occurred.

SEE ALSO **recoverpsm**(1m), **savepsm**(1m)

NAME gstmodconf − NetWorker Management Console command to add or delete managed nodes

SYNOPSIS

gstmodconf
−**i** *file* −**l** *login* [−**P** *password*] [−**f** *function*] [−**s** *server*] [−**k**] [−**p** *port*]

DESCRIPTION

gstmodconf provides a command-line interface for adding or deleting managed nodes from a list of hostnames in an input file. This file is specified with the -i option. Only one hostname may be listed on each line of the file. Hosts are added or deleted at the base level of the Enterprise Hierarchy, as NetWorker managed nodes with all features enabled (Managed Events and Reporting Data).

If a node already exists anywhere in the Enterprise Hierarchy, it will not be added by this command. For deletion, nodes at the base level will be deleted, regardless of existing copies. This means that copies of nodes cannot be added with this command, but copies can be deleted.

If one prefers to place a newly created node into a folder of the Enterprise Hierarchy, the node can be moved there from within the Enterprise task, after logging into NetWorker Management Console from a browser.

Within the input file, blank lines and lines starting with a pound sign (#) are treated as comments and are ignored. Hostnames within the file must be newline-separated. This means that non-comment lines that contain more than one space-separated or tab-separated hostname are interpreted as an error.

By default, **gstmodconf** stops after an error is encountered. If one prefers to continue processing the list of hostnames after an error, use the -k option.

OPTIONS

-i *file* This option is used to specify the file that contains a list of hostnames. It is a required option.

-l *login* Specify the name of the NetWorker Management Console user which should be used to log in to the server. This option is required.

-P *password* Specify the password for the login name specified in the **-l** option.

-f *function* This option specifies the type of function that is performed by the command. Valid values are "add" or "delete". If this option is not specified, the "add" function is assumed.

-s *server* This option specifies the NetWorker Management Console server that **gstmodconf** will connect to. If this option is not specified, it is assumed that the server is on the same host where the command runs.

-k Continue reading and processing hostnames from the input file, ignoring errors that may be encountered when attempting to define previous managed nodes in the file.

-p *port* This option may be used to specify an alternate port where a NetWorker Management Console server is listening. The default port is 9001. If this option is not specified, and a non-default port is being used by the server, this command will attempt to locate the correct port. If this attempt is not successful, this option must be used.

EXAMPLES The example below shows how **gstmodconf** is used to create managed nodes from a list of hosts in the file host_list. In this example, the NetWorker Management Console server name is gstserver and host_list contains:

```
host1
host2

% gstmodconf -s gstserver -i host_list -l administrator
Password:
Trying 137.69.1.111...  connected.
processing file 'host_list'

adding host 'host1'
successfully added host 'host1'

adding host 'host2'
successfully added host 'host2'

// Closing connection...
```

The next example shows how using **gstmodconf** for a host that is already defined as a managed node produces an error.

```
% gstmodconf -s gstserver -i host_list -l administrator
Password:
Trying 137.69.1.111...  connected.
processing file 'host_list'

adding host 'host1'
// Error!
{
        string object_type = "gterror";
        int severity = 16;
        int reason = 23;
        list msg = {
                int level = 1;
                string text = "Host name already exists";
        };
}failed to add host 'host1'

// Closing connection...
```

EXIT STATUS If a fatal error occurs, the exit status is non-zero. If no fatal errors occur, the exit status is zero. When the -k option has been specified, the exit status will reflect the last non-comment line that was processed.

NAME gst_ha.cluster − configure NetWorker Management Console as highly-available

SYNOPSIS **gst_ha.cluster** [−**r**]

DESCRIPTION The **gst_ha.cluster** is an interactive script for configuring the NetWorker Management Console Server as a highly available application in a cluster. The script should be run, after a proper installation of NetWorker Management Console, on all of the nodes of the cluster to activate its failover and cluster-aware capabilities.

The configuration creates a global Console database that is used by the Highly Available Console server. The global Console database is on a shared storage medium and follows the Highly Available (virtual) Console Server on a failover.

When a Console server is configured by this script, the user is asked for cluster platform specific information to prepare for registration of the Console Server with the cluster software. Refer to the *NetWorker Installation Guide* appropriate for your cluster platform for specific instructions.

If a mistake is made during the configuration, run **gst_ha.cluster** with the −**r** option to undo the changes.

OPTIONS −**r** Used to removed the cluster configuration of Console server.

SEE ALSO **gstd**(1)

NAME libstlibm − shared library for communication to IBM 3494 silos

SYNOPSIS **libstlibm.so** (Solaris)
libstlibm.so.a (AIX)

DESCRIPTION **libstlibm.xxx** is a shared library that handles the communication between **nsrjb** and the IBM silo driver (on AIX) or daemon (on Solaris). The IBM driver/daemon then handles the communication over the network to the silo. There are no options, parameters or environment variables that affect the operation of **libstlibm.** The correct path to this file should be entered when an IBM silo is configured using **jbconfig.** The default values specified by **jbconfig** match the default locations chosen for the installation program, and in most cases can be accepted.

For NetWorker to work with the 3494, you must have first installed IBM's Automated Tape Library support.

On AIX, you will need to install a driver called **atldd** (Automated Tape Library Device Driver). You may also require the **IBMtape** driver (Enhanced Tape and Medium Changer Device Driver) if you are using 3590 drives in your 3494.

On Solaris, you will need to install the **lmcpd** package, (IBM Automated Tape Library Daemon) to use the silo. Again, if you are using 3590 drives, you will also need to install the **IBMtape** driver. Note that when you are using **IBMtape,** there will be two sets of device files that will access a given tape drive. There will be the standard Solaris style **/dev/rmt/Xmbn** type, and there will be the **IBMtape** supported files of the type **/dev/rmt/Xstbn.** You should use the IBM supported device files for proper operation of your tape drives.

Note: EMC cannot supply these IBM drivers. They may be available on an IBM Device Driver ftp site (208.200.29.244), but this is not necessarily a long-term IBM committed site.

SEE ALSO **nsrjb**(1m), **jbconfig**(1m), **dasadmin**(1m), **libstlemass**(1m), **ssi**(1m), **mini_el**(1m), **libstlstk**(1m)

DIAGNOSTICS Errors in communication between the NetWorker server and the IBM 3494 silo are difficult to diagnose. The best method is to use the IBM supplied utility **mtlib** to verify that you have properly configured the 3494 to communicate with your host, and that the entire pathway from either the lmcp driver (on AIX) or the lmcpd daemon (on Solaris) is functioning properly. If **mtlib** does not work, then there is no chance that NetWorker will work.

If there are any questions about the connection between your host and the 3494, it is best to consult IBM, as they support the connection between the host and the silo. IBM supports both network and serial cable connections to the silo. Since the nature of the connection is hidden from NetWorker by the driver/daemon, there is no difference to NetWorker between the two. Customers have successfully used both.

NAME ielem − initialize element status

SYNOPSIS **ielem** [−**a** *b.t.l*] [−**r** *eladdr.nel*]

DESCRIPTION The **ielem** program sends an INITIALIZE ELEMENT STATUS command to the named device.

Some changers support the ability to initialize element status for a range of elements. The command used for this is the Vendor Unique EXABYTE changer command:

INITIALIZE ELEMENT STATUS (with range)

(command opcode 0xE7).

OPTIONS −**a** *b.t.l* Selects a specific ordinal SCSI address, where **b** is the logical SCSI bus, **t** is the SCSI target, and **l** is the SCSI logical unit number (LUN) on that target. See **libscsi**(1m). This is a required option.

−**r** *eladdr.nel*
Specifies the range of elements, where *eladdr* is the starting decimal address (in the autochanger's numbering) of the element to start from, and *nel* is the number of status elements to read. This option can be used if your autochanger supports the Vendor Unique EXABYTE autochanger INITIALIZE ELEMENT STATUS command.

SEE ALSO **libscsi**(1m)

NAME inquire - list devices available

SYNOPSIS **inquire** [**-a** *b.t.l*] [−**clp**] [−**N** NDMPhost] [-s] [-T [-t]]

DESCRIPTION The **inquire** program lists SCSI devices available. The **inquire** program returns INQUIRY data either for the named SCSI device (with the **-a** option), or for all SCSI devices attached to the system. In addition to the standard SCSI inquiry data, inquire now returns serial number information obtained from the Vital Product Data (VPD) pages supported by the devices that are being queried. There may be anywhere from zero to eight different identifiers for each device, depending on which of the VPD pages that particular device supports.

In NetWorker 7.2.1 and higher, the support of LUS was discontinued for Solaris 10 and higher. This means that after installation 'inquire' might not necessarily show any devices meant to be used by NetWorker. If this is the case, the Solaris Server might not be configured correctly.

A quick check is to run 'cfgadm -lav' to see what is listed. Looking at the 'cfgadm' output and if the devices are listed, make sure that the '/dev/rmt' path is used for the devices. This is the preferred NetWorker path and it is created automatically by the Solaris 'st' driver. If the devices are not listed; please refer to the NetWorker Administration Guide, SUN Administration Guide and Manufacturer's Manual.

Sample output including serial number information:

```
scsidev@0.0.0:SEAGATE ST34371W SUN4.2G7462|Disk, /dev/rdsk/c0t0d0s2
        S/N: JDY217500LUW5N
scsidev@0.1.0:QUANTUM ATLAS IV 36 SCA 0B0B|Disk, /dev/rdsk/c0t1d0s2
        S/N: 363009430963
        ATNN:QUANTUM 363009430963
scsidev@0.6.0:TOSHIBA XM5701TASUN12XCD2395|CD-ROM, /dev/rdsk/c0t6d0s2
scsidev@4.0.0:SONY    TSL-11000       L1  |Tape, /dev/rmt/0cbn
        S/N: 0001100158
        ATNN:SONY    TSL-11000       0001100158
scsidev@4.0.1:SONY    TSL-11000       L1  |Autochanger (Jukebox)
        S/N: 3761633968
        ATNN:SONY    TSL-11000       3761633968
scsidev@4.2.0:IBM     ULTRIUM-TD1     0CE0|Tape
        S/N: 6811004028
        ATNN:IBM     ULTRIUM-TD1     6811004028
scsidev@4.3.0:HP      Ultrium 1-SCSI  N16D|Tape, /dev/rmt/1cbn
        S/N: GB81A00316
        ATNN:HP      Ultrium 1-SCSI  GB81A00316
scsidev@4.4.0:IBM     ULTRIUM-TD1     0CE0|Tape
        S/N: 6811003960
        ATNN:IBM     ULTRIUM-TD1     6811003960
scsidev@4.5.0:EXABYTE Exabyte 221L    2.4 |Autochanger (Jukebox)
        S/N: 99999999
```

Lines starting with S/N: represent the device's serial number as returned by VPD page 80 hex.

Lines that start with a four character prefix plus a colon are those returned in SCSI-3 format on VPD page 83 hex. The four character prefix tells which of the various SCSI-3 Device Identifiers it represents.

ATNN: ASCII Text identifier of unspecified format
describing the device itself (usually Vendor, Product, Serial number)

ATPN: ASCII Text identifier of unspecified format
describing the port that you are connected to the device through (not commonly used)

VENN: An ASCII vendor specific identifier of unknown
uniqueness describing the device itself

VEPN: An ASCII vendor specific identifier of unknown
uniqueness describing the port you are connected through

VBNN: A binary vendor specific identifier of unknown
uniqueness describing the device itself

VBPN: A binary vendor specific identifier of unknown
uniqueness describing the port you are connected through

IENN: An IEEE 64-bit identifier (EUI-64) describing
the device itself (shown in hexadecimal format)

IEPN: An IEEE 64-bit identifier (EUI-64) describing
the port you are connected through (shown in hexadecimal format)

WWNN: A Fibrechannel identifier (World Wide Node Name)
describing the device itself (shown in hexadecimal format)

WWPN: A Fibrechannel identifier (World Wide Port Name)
describing the port you are connected through (shown in hexadecimal format)

PORT: The relative port number that you are connected
through. Port "A" would return a value of 1, Port "B" a value of 2...

RESV: The device returned a combination of Association
and Identifier Type bits that was reserved at the time this code was written.

UNKN: The device returned information that this program
was unable to decipher

OPTIONS

−**a** *b.t.l* Selects a specific ordinal SCSI address, where **b** is the logical SCSI bus, **t** is the SCSI target, and **l** is the SCSI logical unit number (LUN) on that target. The option is not compatible with -N. See **libscsi**(1m).

−**c** *(NOTE: USE WITH CAUTION)*
This flag directly sends the SCSI inquiry command to the device and may cause unforeseen errors when there is other activity on the bus.

−**l** Performs a complete LUN search for all SCSI adapters in the system. This

argument is accepted on all systems, but does not have any effect on HP-UX systems because the method used to scan for available devices on HP-UX systems always shows all accessible devices. For systems other than HP-UX, the normal behavior is to start checking at LUN 0 for SCSI devices. The first empty LUN found will end the search for a given target ID. With the **−l** option, all LUNs present on all target IDs for all SCSI busses in the system will be checked for devices. This can take a **very long time** and should therefore only be used when necessary. For example, a Fibre Channel adapter can support 126 target IDs, each of which may have 80 or more LUNs. Checking all LUNs on this single adapter may take over 10 minutes. This option has no affect when -N present.

−p Tells inquire to display persistent device names for devices on platforms where persistent names are supported. If persistent names do not exist for any particular device then the normal device name will be shown. Currently only linux has such support. Specifying **−p** on a platform that does not have NetWorker-recognized persistent names will have no effect.

−N *NDMPhost*

Performs a device discovery on the NDMP Tape Server *NDMPhost.* User will be prompted for the NDMP user name and password. NDMP protocol exports only Jukeboxes and Tape Devices. No other device types will be discovered. When NDMP Tape Server is running at version 3 or higher and supports NDMP_CONFIG_GET_SCSI_INFO and NDMP_CONFIG_GET_TAPE_INFO interfaces, inquire will display the INQUIRY data for all the available Jukeboxes and Tape Devices. In all other cases, inquire will prompt for Jukebox handle and get the INQUIRY data for that Jukebox. This option is not compatible with -a. See -T for more details.

Sample output with NDMP Tape Server running at V3 and supporting of SCSI and TAPE CONFIG interfaces:

```
# inquire -N server-2
Enter NDMP user name: ? ndmp
Enter 'ndmp' password on NDMP host 'server-2' (characters will not be echoed):

Communicating to devices on NDMP Server 'server-2', this may take a while...

scsidev@178.0.0:QUALSTARTLS-6110        2.09|Autochanger (Jukebox), c178t0l0
                                        S/N: 44B43014
scsidev@178.0.1:QUANTUM DLT8000         0119|Tape, c178t0l1
                                        S/N: CX938P2489
                                        IENN:0000000000000000
```

Sample output with NDMP Tape Server running at V2:

```
# inquire -N molokai
Enter NDMP user name: ? root
Enter 'root' password on NDMP host 'molokai' (characters will not be echoed):

Communicating to devices on NDMP Server 'molokai', this may take a while...

NDMP Tape Server 'molokai' does not support of auto-discovery of SCSI and
TAPE Devices.
Will perform the operation on a single Jukebox in which you are interested.
```

```
Enter NDMP Jukebox handle: ? mc1

scsidev@-1.2.0:EXABYTE Exabyte 215      2.3 |Autochanger (Jukebox)
                                        S/N: 71000073
```

-s Suppresses the collection of serial number information by inquire, so that inquire returns the same output that it did before the serial number information was added. This option is primarily added so that any scripts that rely on the previous output behavior of inquire can be used with only minor modification.

-T This option is only valid when -N is present else it is ignored. The option will display the NDMP Tape Devices in a non standard format. The Device Model and Device Handle(s) will be displayed. This option is useful on NDMP Tape Servers that do not support NDMP_SCSI_OPEN interface on Tape Devices (For example, NetApp).

Sample output with -T option on NetApp Filer.

```
# inquire -N  molokai -T
Enter NDMP user name: ? root
Enter 'root' password on NDMP host 'molokai' (characters will not be echoed):

Communicating to devices on NDMP Server 'molokai', this may take a while...

scsidev@0.2.0:EXABYTE Exabyte 215      2.3 |Autochanger (Jukebox), mc1
                                        S/N: 71000073
scsidev@0.3.0:QUANTUM Powerstor L200  0022|Autochanger (Jukebox), mc0
                                        S/N: JF83801878

Model                          Device Handle
---------                      -----------------

Quantum DLT7000                nrst0l
                               nrst0m
                               nrst0h
                               nrst0a

Exabyte Mammoth-2 8mm          nrst2l
                               nrst2m
                               nrst2h
                               nrst2a
```

-t This option is only valid when -T is present else it is ignored. The option will display the vendor specific NDMP Tape Devices Attributes for each tape device handles that are displayed with option -T.

Sample output with -t option on NetApp Filer.

```
# inquire -N rainbow -T -t
Enter NDMP user name: ? root
Enter 'root' password on NDMP host 'molokai' (characters will not be echoed):
```

```
Communicating to devices on NDMP Server 'molokai', this may take a while...

scsidev@0.3.0:QUANTUM Powerstor L200  0022|Autochanger (Jukebox), mc0
                                S/N: JF83801878

Model           Device Handle                 Attributes
--------        ----------------              --------------

Quantum DLT7000  nrst0l             DENSITY --  81633 bpi 40 GB (w/comp)
                                    ELECTRICAL_NAME   --  0b.4
                                    SERIAL_NUMBER      --  CX902S0678
                                    WORLD_WIDE_NAME --
                                    ALIAS 0   --  st0

                 nrst0m             DENSITY  --  85937 bpi 35 GB
                                    ELECTRICAL_NAME   --  0b.4
                                    SERIAL_NUMBER      --  CX902S0678
                                    WORLD_WIDE_NAME  --
                                    ALIAS 0    --  st0

                 nrst0h             DENSITY --  85937 bpi 50 GB (w/comp)
                                    ELECTRICAL_NAME   --  0b.4
                                    SERIAL_NUMBER      --  CX902S0678
                                    WORLD_WIDE_NAME  --
                                    ALIAS 0   --  st0

                 nrst0a             DENSITY  --  85937 bpi 70 GB (w/comp)
                                    ELECTRICAL_NAME    --  0b.4
                                    SERIAL_NUMBER      --  CX902S0678
                                    WORLD_WIDE_NAME  --
                                    ALIAS 0    --  st0
```

SEE ALSO **libscsi**(1m)

WARNINGS Use this command with caution. The inquire command sends the SCSI inquiry command to all devices it detects on all SCSI buses. Running inquire during normal device operations may cause unforeseen errors. Data loss may result.

LIMITATIONS The **inquire** program always uses the built-in system drivers to test SCSI devices. The device type or path name printed by the **inquire** program may be incorrect for devices that require special, third-party drivers.

You must be logged in as the superuser (root) on Unix systems when running inquire. If not, the output may be erroneous.

NAME jbconfig − jukebox resource configuration tool

SYNOPSIS **jbconfig** [−**s** *server*] [-lp]

DESCRIPTION The **jbconfig** program provides an interactive script for configuring a jukebox (Media Autochanger Device) for use with a NetWorker server. The script pauses periodically for you to enter a response to a prompt. If you want to accept the default choice displayed in braces, press [RETURN] or [ENTER].

Starting with NetWorker 7.2.1 and above; the support of LUS was discontinued for Solaris 10 and above. If **jbconfig** reports that it cannot find any autochangers after installation, run **inquire** to make sure it is able to see the devices. Please refer to the **inquire**(1m) man page for more information. Sometimes in an ill-configured server, an autochanger is seen but its drives are not mapped to the '/dev/rmt' path but only to the '/dev/scsi/sequential' path. **jbconfig** will configure the autochanger using this path. The problem here is that a drive using the '/dev/scsi/sequential' path is assumed to be standalone drive. This means the autochanger will not work correctly.

After the jukebox is configured, use the **nsrcap**(1m) command or the Registration window to enter the enabler code for your Autochanger Software Module. You must have a separate enabler code for each jukebox you want to use with NetWorker.

OPTIONS −**s** *server*

Specifies the controlling server, when **jbconfig** is being used from a storage node. To define a jukebox resident on a storage node, the **jbconfig** command must be run on the storage node. See **nsr_storage_node**(5) for additional information on storage nodes.

−**l** Performs a complete LUN search for all SCSI adapters in the system when performing Autodetection. This argument is accepted on all systems, but does not have any effect on HP-UX systems. Due to the method used to scan for available devices on HP-UX systems, all accessible devices are always shown, and the −**l** option has no additional effect. On all other systems, the normal behavior is to start checking at LUN 0 for SCSI devices. The first empty LUN found will end the search for a given target ID. With the −**l** option, all LUNS present on all target IDs for all SCSI busses in the system will be checked for jukeboxes. This can take a **very long time** and should therefore only be used when necessary. For example, a Fibre Channel adapter can support 126 target IDs, each of which may have 80 or more LUNs. Checking all LUNs on this single adapter may take over 10 minutes.

−**p** Use persistent names for all automatically detected devices where available. This will affect the control ports used for autodetected SCSI Jukeboxes and device file names for tape drives that jbconfig is able to automatically detect and configure for you. If a given device does not have a persistent device name then jbconfig will use the normal device name for that device. Currently only linux persistent device names are automatically found and used by NetWorker. Specifying this flag on other platforms will have no effect.

CONFIGURATION DIALOG The first question **jbconfig** will ask you, is to select a type of jukebox to install.

1) Configure an AlphaStor Library.
2) Configure an Autodetected SCSI Jukebox.
3) Configure an Autodetected NDMP SCSI Jukebox.
4) Configure a SJI Jukebox.
5) Configure a STL Silo.

What kind of Jukebox are you configuring? [1]

Enter the number corresponding to the jukebox type you are installing. The default selection is 1.

An AlphaStor Library is any jukebox that is controlled by EMC AlphaStor. It is configured in NetWorker as a logical jukebox, with the actual jukebox operations carried out by AlphaStor.

An Autodetected SCSI Jukebox is any SCSI (Small Computer System Interface) based jukebox connected to a system that NetWorker can automatically detect.

Autodetected NDMP SCSI Jukebox is any SCSI (Small Computer System Interface) based jukebox connected directly to a Network Data Management Protocol (NDMP) Server, that NetWorker will automatically detect, with the provided NDMP hostname, user-id, user-password, and jukebox handle. (See example).

An SJI Jukebox is a Standard Jukebox Interface compliant jukebox. This is a list of well known SCSI based jukeboxes, plus any additional third party jukebox devices that adhere to this protocol that the you may have added to the system.

If you select the second choice (Install an Autodetected SCSI Jukebox), **jbconfig** will print out a list of jukeboxes it detects on the system.

For example:

These are the SCSI Jukeboxes currently attached to your system:
1) scsidev@0.2.0: other, Vendor <AIWA>, Product <AL-17D>
2) scsidev@2.2.0: DLI Libra Series
3) scsidev@1.4.1: ARC-DiamondBack
Which one do you want to install?

When this message appears, enter the number corresponding to the jukebox that you wish to configure. Note that if **jbconfig** was able to detect only one SCSI jukebox on the system, it will go ahead and select that jukebox as the one to be configured without waiting for the user to make the selection. This also applies to situations where there are multiple SCSI jukeboxes on the system and all but one are already configured in NetWorker. Even in this case **jbconfig** goes ahead and automatically selects the one that has not yet been configured without waiting for the user to make a selection.

If you choose to install an SJI compliant jukebox, **jbconfig** will print a list of known SJI Jukeboxes and will prompt you for the appropriate type that you want to configure.

For example:

Enter the number corresponding to the type of jukebox you are installing:
1) ADIC-1200c/ADIC-1200d
2) ADIC-VLS
3) ARC-DiamondBack
4) Breece Hill
5) DLI Libra Series
6) Quantum DLT/Digital DLT
7) EXB-10e/EXB-10h
8) EXB-10i
9) EXB-60
10) EXB-120
11) EXB-210
12) EXB-218

```
13) EXB-400 Series
14) HP-C1553A/Surestore 12000e
15) Metrum (SCSI)
16) Qualstar
17) Spectralogic
18) STK-9704/Lago 340
19) STK-9708/Lago 380 (SCSI) Datawheel
20) IBM 7331/IBM 9427
21) ATL/Odetics SCSI
22) HP-Optical 630MB/1.3GB
23) other
Choice?
```

When this message appears, enter the number corresponding to the appropriate model, for example, if you are installing an HP optical jukebox select the number "22".

For all jukebox types, **jbconfig** prompts you for the *name* you want to call this jukebox. This is a convenient way for you to identify the jukebox for yourself and NetWorker, for example, 'Engineering Autochanger'. NetWorker will store this name as a NetWorker resource (see **nsr_resource**(5)). When defining a jukebox attached to a storage node, **jbconfig** prefixes the hostname of the storage node to the beginning of the names using the remote device syntax ("rd=hostname:"). See **nsr_storage_node**(5) for additional information on storage nodes.

For all jukebox types, **jbconfig** prompts you for a *description* of this jukebox. This is another convenient way for you to identify the jukebox for yourself, for example, 'Engineering 4 Drive DLT Autochanger on Rack #2'.

For SJI jukebox types, **jbconfig** prompts you for the name of the control port associated with the jukebox being configured. For silos, this could be the name of the host running the silo software (for ACSLS & DAS) or the name of the 3494, depending on the type of Silo. For Autodetected SCSI jukeboxes, **jbconfig** detects the correct name and goes ahead with the configuration. This name is in the form of *libscsi* devices (see **libscsi**(1m)). For SJI compliant jukeboxes, no such detection is done. The name you enter should either be the device name for the jukebox as described in any third party SJI compliant driver installed, or the format used for autodetected jukeboxes. A list of attached autochangers can be obtained by running the **changers**(1m) command.

Once a control port is entered, **jbconfig** will check to see if the model selected is a SCSI or SJI based jukebox. If the jukebox model is a SCSI or SJI based jukebox, **jbconfig** will attempt to query the jukebox about various internal parameters (for example, number of slots and drives). If this query fails, it is possible that there is a device driver installation problem or a hardware problem.

Next, if the jukebox contains tape devices, you are asked if automated cleaning of devices in the jukebox should be turned on. If automated cleaning is enabled, the jukebox and all devices in the jukebox are configured for automated cleaning. On successful installation, the information that pertains to device cleaning for the jukebox and all its devices are displayed. Note that with the introduction of the Common Device Interface (CDI), NetWorker now has two events that will cause an automatic cleaning to occur: schedule-based cleaning, with devices being cleaned after a certain (configurable) amount of time has elapsed, and on-demand cleaning, where cleaning is initiated by TapeAlert warnings issued by the devices. Schedule-based cleaning is always active when autocleaning is enabled. On-demand cleaning is used when the CDI attribute for a tape device is set to anything other than 'Not Used' in the device resource. If on-demand cleaning is being used, you should set the Cleaning Interval for the device itself to a large time, such as 6 months, so that NetWorker does not clean the device unnecessarily. See **nsr_device.5** for a more detailed explanation of CDI, TapeAlert and

Cleaning Interval.

At this point, the user has an option of either going ahead with automatic configuration of the jukebox, accepting all detected information and default choices as correct, or choosing to custom configure some or all aspects of the configuration, including configuring devices as NDMP or shared devices, configuring drives that were not detected by **jbconfig,** or changing the model type of any of the detected devices. The user can choose to go the custom configuration route by answering 'yes' to the following question:

Do you want to change the model(s) or configure them as shared or NDMP drives? (yes / no) [no]

If the user chooses the custom configuration option, the user is given a choice of configuring the drives as NDMP and/or shared drives. Answering 'yes' to either of the prompts will take the user to other relevant questions about NDMP and/or shared drive configuration.

If the user chose 'yes' to configuring NDMP devices, **jbconfig** proceeds to prompt the user for this information. NDMP devices require a user name and password to be entered for each device. The user name and password correspond to the entries set in the NDMP server.

If the user chose 'yes' to configuring shared drives, the user is prompted for multiple device paths for each physical drive in the jukebox. These device paths would typically be located on different storage nodes within a data zone, under the control of one NetWorker Server. Drives or device paths on remote nodes are to be entered in the "host:<device-path>" form. It's not necessary that all drives in a jukebox be shared drives; entering a null response to a prompt for additional device paths for a drive skips that drive and takes you to the next step in the configuration. A unique 'hardware-id' of the form '<jukebox name> - <drive no>' is automatically assigned to each shared instance of a drive. The 'hardware-id' is how NetWorker keeps track of shared devices. See **nsr_device**(5) for a description of the hardware-id attribute.

Next, **jbconfig** prompts the user for the model of the drives being configured. In case **jbconfig** has been able to detect the model type(s), it will display this information and ask for confirmation. If not, it lets the user configure the model for each drive.

If you selected Autodetected SCSI jukeboxes, NetWorker determines the name of each media device by sending inquiries for information to the jukebox. Not all jukeboxes support this capability, but many do (for example, the Exabyte 210). This inquiry does not take place when the owning host is different than where **jbconfig** is running.

If configuring devices on a remote storage node, **jbconfig** asks the user if s/he wants to configure the node on which the device is being configured as a Dedicated Storage Node (DSN). A DSN is a node which allows only data from the local host to be backed up to its devices. See **nsr_device(5)** for more details on DSN. The question is of the form:

A Dedicated Storage Node can backup only local data to its devices. Should helium be configured as a Dedicated Storage Node? (yes / no) [no]

Earlier versions of **jbconfig** used to prompt the user for information about bar code readers in jukeboxes and whether volume names should match bar code labels. With NetWorker 7.0 and later, **jbconfig** tries to set these attributes either by querying the jukebox for information or making intelligent guesses. For Silos, the 'bar code reader' and 'match bar code labels' attributes in the jukebox resource are set to 'yes' by default. If it is a jukebox, **jbconfig** queries the jukebox for this information. If both features are supported by the jukebox, it sets both fields to 'yes.' If both features are

not supported, it sets both fields to 'no.' However, if the jukebox reports that it can handle volume tags, but has no bar code reader, **jbconfig** still sets both fields to 'yes,' since some jukeboxes with bar code readers tend to report this way. At the end of the installation **jbconfig** prints out this information and the user can use NetWorker Management Console to edit the jukebox resource to set the fields to 'No' if he so desires.

If the above two fields are set, the label templates will not be used by the jukebox, and each media volume must have a readable bar code label. Note that on some small jukeboxes, like the HP 1557A or the SONY TSL_A500C, setting 'bar code reader' to 'yes' may cause problems with the labeling. The solution is to set the appropriate attributes to 'No' as described above.

If the jukebox has been configured successfully you will see the following message:

Jukebox has been added successfully

The following configuration options have been set:

followed by a list of options that have been set by default.

JBCONFIG FILE

The file */nsr/jbconfig* is the jukebox models configuration file. This file can be used to configure a non-standard list of jukebox models.
VECTOR-TYPE MODEL-NAME<**NEWLINE**>, where VECTOR-TYPE is either SJI (the Standard Jukebox Interface) or ATL (RS232-based devices speaking the IGM-ATL serial communications protocol). The MODEL-NAME can be any string.

EXAMPLES

(User entries are in italics).

Example 1)

```
# jbconfig
        1) Configure an AlphaStor Library.
        2) Configure an Autodetected SCSI Jukebox.
        3) Configure an Autodetected NDMP SCSI Jukebox.
        4) Configure a SJI Jukebox.
        5) Configure a STL Silo.

What kind of Jukebox are you configuring? [1] 2 <RETURN>
These are the SCSI Jukeboxes currently attached to your system:
  1) scsidev@0.6.0: EXB-210
  2) scsidev@3.0.0: ADIC
Which one do you want to install?  1<RETURN>
Installing an 'EXB-210' jukebox - scsidev@0.6.0
What name do you want to assign to this jukebox device?  Engineering<RETURN>
Turn NetWorker auto-cleaning on (yes/no) [yes]?  yes<RETURN>
The following drives have been detected in this auto-changer:
 1> 8mm @ 1.1.0 ==> \\.\Tape0
 2> 8mm @ 1.2.0 ==> \\.\Tape1 These are all the drives that this auto changer possesses.
Do you want to change the model(s) or configure them as shared or NDMP drives?
(yes / no) [no] yes <RETURN>
Is (any path of) any drive intended for NDMP use? (yes / no) [no] yes <RETURN>
Is any drive going to have more than one path defined? (yes / no) [no] yes
<RETURN>
```

You will be prompted for multiple paths for each drive. Pressing <Enter> on a null default advances to the next drive.

Please enter the device path information in one of the following formats:

\\.\Tape0 --for local path or
host:device-path --for remote node or
host:drive-letter:directory path --for Windows disk file

Drive 1, element 82, system name = \\.\Tape0,
local bus / target / lun value = 1/1/0,
model 8mm
Device path 1 ? [\\.\Tape0]
Enter NDMP user name for host 'happy'? [] user1 <*RETURN*>
Enter NDMP password (characters will not be echoed): <*RETURN*>
Device path 2 ? [] helium:/dev/rmt/1cbn
Enter NDMP user name for host 'helium'? [] user3 <*RETURN*>
Enter NDMP password (characters will not be echoed): <*RETURN*>
Device path 3 ? [] <*RETURN*>

Drive 2, element 83, system name = \\.\Tape1,
local bus / target / lun value = 1/2/0,
model 8mm
Device path 1 ? [\\.\Tape1]
Enter NDMP user name for host 'ableix.emc.com'? [] <*RETURN*>
Device path 2 ? [] <*RETURN*>

Only model 8mm drives have been detected.
Are all drives in this jukebox of the same model? (yes / no) [yes] yes <*RETURN*>

A Dedicated Storage Node can backup only local data to its devices.
Should helium be configured as a Dedicated Storage Node? (yes / no) [no] no
<*RETURN*>
Jukebox has been added successfully

The following configuration options have been set:

> Jukebox description to the control port and model.
> Autochanger control port to the port at which we found it.
> Networker managed tape autocleaning on.
> At least one drive was defined with multiple paths. All such drives are defined with a hardware identification as well as a path value to avoid confusion by uniquely identifying the drive. The hardware identification for all drives which have one is always 'autochanger_name - Drive #' where "autochanger_name" is the name you gave to the autochanger that was just defined, and the # symbol is the drive numer.
> Barcode reading to on.
> Volume labels that match the barcodes.
> Slot intended to hold cleaning cartridge to 1. Please insure that a cleaning cartridge is in that slot
> Number of times we will use a new cleaning cartridge to 12.
> Cleaning interval for the tape drives to 6 months.

You can review and change the characteristics of the autochanger and its associated devices using NetWorker Management Console.

Would you like to configure another jukebox? (yes/no) [no] no <*RETURN*>

Example 2)

Here is an example of an AlphaStor library configured with NDMP devices on a storage node.

jbconfig -s server
On a storage node, the hostname is a prefix to the jukebox name.
Enter the hostname to use as a prefix? [brown.emc.com] <*RETURN*>
using 'brown.emc.com' as the hostname prefix

1) Configure an AlphaStor Library.
2) Configure an Autodetected SCSI Jukebox.
3) Configure an Autodetected NDMP SCSI Jukebox.
4) Configure a SJI Jukebox.
5) Configure a STL Silo.

What kind of Jukebox are you configuring? [1] <*RETURN*>
Installing an AlphaStor jukebox.
What name would you like to assign to the AlphaStor library?
myautoloader<*RETURN*>

Name of AlphaStor server host machine? [brown.emc.com] <*RETURN*>
Port number of AlphaStor server? [44475] <*RETURN*>

How many devices are to be configured (1 to 64)? [4] *2*<*RETURN*>
Enter hostname that owns logical device 1: ? [brown.emc.com] <*RETURN*>
Enter name of logical device 1: ? *stk1*<*RETURN*>
Should the drive be configured as a NDMP device? (yes/no) *y*<*RETURN*>
Enter NDMP user name: ? *root*<*RETURN*>
Enter NDMP password (characters will not be echoed): *password*<*RETURN*>
Enter hostname that owns logical device 2: ? [brown.emc.com] <*RETURN*>
Enter name of logical device 2: ? *stk2*<*RETURN*>
Should the drive be configured as a NDMP device? (yes/no) *y*<*RETURN*>
Enter NDMP user name: ? *root*<*RETURN*>
Enter NDMP password (characters will not be echoed): *password*<*RETURN*>

Enter application name defined in AlphaStor/SmartMedia for NetWorker?
[NetWorker@server] <*RETURN*>
Enter application key defined in AlphaStor/SmartMedia for NetWorker? [<none>]
<*RETURN*>

The barcode reader is enabled and volume labels are set to match barcode labels.

Jukebox has been added successfully
Would you like to configure another jukebox? (yes/no) *no*<*RETURN*>

Example 3)

Here is an example of configuring a jukebox attached to NDMP Tape Server.

jbconfig

1) Configure an AlphaStor Library.
2) Configure an Autodetected SCSI Jukebox.
3) Configure an Autodetected NDMP SCSI Jukebox.
4) Configure a SJI Jukebox.
5) Configure a STL Silo.

What kind of Jukebox are you configuring? [1]
3<RETURN>
Enter NDMP Tape Server name: ?
molokai<RETURN>
Enter NDMP user name: ?
root<RETURN>
Enter NDMP password (characters will not be echoed):
password<RETURN>
Communicating to devices on NDMP Server 'molokai', this may take a while...

These are the SCSI Jukeboxes currently attached to your system:
1) scsidev@0.2.0: Exabyte Jukebox
2) scsidev@0.3.0: Standard SCSI Jukebox, QUANTUM / Powerstor L200

Which one do you want to install?
1<RETURN>

Installing an 'Exabyte Jukebox' jukebox - scsidev1027.2.0.

What name do you want to assign to this jukebox device?
netapp_jb<RETURN>
Turn NetWorker auto-cleaning on (yes/no) [yes]?
yes<RETURN>

The drives in this jukebox cannot be auto-configured with the available information. You will need to provide the path for the drives.

Is (any path of) any drive intended for NDMP use? (yes / no) [no]
yes<RETURN>
Is any drive going to have more than one path defined? (yes/no) [no]
no<RETURN>

Please enter the device path information in one of the following formats:

\.Tape0 --for local path or
host:device-path --for remote node or
host:drive-letter:directory path --for Windows disk file

After you have entered a device path, you will be prompted for an NDMP user name for that path's host. If this device path is not an NDMP device, press the enter key to advance to the next device path. For NDMP devices, you need to enter the user name and password the first time we encounter that NDMP host. Pressing the enter key for the NDMP user name for any subsequent device path on the same host will set the user name and password to those defined the first time. You will not be prompted for the password in such a case.

Drive 1, element 82
Drive path ?
molokai;nrst2l<RETURN>
Enter NDMP user name for host 'molokai'? []
root<RETURN>

```
Enter NDMP password (characters will not be echoed):
password<RETURN>


Please select the appropriate drive type number:
 1) 3480          18) 9840          34) optical
 2) 3570          19) 9840b         35) qic
 3) 3590          20) 9940          36) SD3
 4) 4890          21) adv_file      37) sdlt
 5) 4mm           22) dlt           38) sdlt320
 6) 4mm 12GB      23) dlt1          39) SLR
 7) 4mm 20GB      24) dlt7000       40) tkz90
 8) 4mm 4GB       25) dlt8000       41) travan10
 9) 4mm 8GB       26) dst (NT)      42) tz85
10) 8mm           27) dtf           43) tz86
11) 8mm 20GB      28) dtf2          44) tz87
12) 8mm 5GB       29) file          45) tz88
13) 8mm AIT       30) himt          46) tz89
14) 8mm AIT-2     31) logical       47) tz90
15) 8mm AIT-3     32) LTO Ultrium   48) tzs20
16) 8mm Mammoth-2 33) LTO Ultrium-2 49) VXA
17) 9490


Enter the drive type of drive 1?
16<RETURN>


Jukebox has been added successfully

The following configuration options have been set:
  > Jukebox description to the control port and model.
  > Autochanger control port to the port at which we found it.
  > Networker managed tape autocleaning on.
  > Barcode reading to on.  Your jukebox does not report that it has a bar
    code reader, but it does report that it can handle volume tags.  Some
    jukeboxes that have barcode readers report this way.
  > Volume labels that match the barcodes.
  > Slot intended to hold cleaning cartridge to 1.  Please insure that a
    cleaning cartridge is in that slot
  > Number of times we will use a new cleaning cartridge to 5.
  > Cleaning interval for the tape drives to 6 months.
You can review and change the characteristics of the autochanger and its
associated devices using NetWorker Management Console.


Would you like to configure another jukebox? (yes/no) [no]
no<RETURN>
```

SEE ALSO **jbexercise**(1m), **nsr_device**(5), **nsr_jukebox**(5), **nsr_storage_node**(5), **nsr**(5), **nsrcap**(1m)

DIAGNOSTICS *unknown model* **invalid choice for 'model' (35022)**
Problem: The NetWorker system does not recognize the model chosen. If you added a /nsr/jbconfig* file after starting the daemons, you will see this error. **Solution:** Restart NetWorker.

root **on computer** *host* **is not on type: NSR's administrator list**
Problem: The user 'root' on the storage node 'host' is not on the administrator list of the NetWorker server. **Solution:** Add such an entry to the NetWorker server's administrator list. Note that the entry can be removed after this command completes.

NAME

jbedit − add and delete device definitions to and from a NetWorker jukebox.

SYNOPSIS

jbedit
−**a** −**f** <*device path*> −**E** <*element addr*> [−**s** *server*] [−**j** <*jukebox name*>] [−**v**]

jbedit
−**a** −**f** <*device path*> −**S** <*silo device identifier*> [−**s** *server*] [−**j** <*jukebox name*>] [−**v**] [−**F**]

jbedit
−**a** −**f** <*device name*> −**l** [−**s** *server*] [−**j** <*jukebox name*>] [−**v**]

jbedit
−**d** −**f** <*device path*> [−**s** *server*] [−**j** <*jukebox name*>] [−**v**]

jbedit
−**h**

DESCRIPTION

jbedit allows a NetWorker user with "Configure NetWorker" privileges to add or delete **drive** and **device** definitions to or from an existing jukebox definition in the NetWorker database. A **drive** is defined in NetWorker as "the physical backup object, such as a tape drive, disk, or file" and a **device** as "the access path to the physical drive." Every **drive** therefore has at least one **device** associated with it, and every **device** has exactly one **drive** with which it can be associated. The addition of a device, even on an existing storage node, may require changes to the Dynamic Drive Sharing (DDS) licenses.

jbedit complements the **reconfigure library** feature available through the **Networker Management Console. jbedit** can be used as a fallback means of editing library configurations if, for some reason, the **reconfigure library** program is limited (for example, the library does not return the drive serial numbers).

jbedit supports all direct attached SCSI/SJI, SAN, NDMP and AlphaStor libraries as well as Silos.

If the **NSR_JUKEBOX** environment variable is set, **jbedit** uses its value as the name of the jukebox to edit. This behavior may be overridden by −**j** option. If neither of these options are present, **jbedit** attempts to retrieve the list of all available jukeboxes on the server, and prompts for a jukebox to be selected if more than one is available.

To edit the library configuration, the jukebox must be **enabled** and **ready** (see **nsr_jukebox**(5)). **jbedit** can be run from any storage node, however it requires "Configure NetWorker" privileges.

OPTIONS

The following options are supported:

−**a** Add a drive/device.

−**d** Delete a drive/device.

−**h** Display the jbedit options and their usage.

−**E** The drive **element address.**

This is the data element address of the **drive** with which the **device** is associated. When a new **drive** or **device** is being added to a jukebox, **jbedit** needs to know the data element address of the drive/device within the jukebox. The data element address is the "decimal number" that the jukebox assigns to each of its drives. Information about drives, data element addresses and other

information associated with a jukebox can be found in **relem**(1m), **changers**(1m), **sn**(1m) and **sjisn**(1m).

−S The drive's **silo device identifier.** This is the identifier used by the silo controller to identify the **drive** with which the **device** is associated. Typical **silo device identifiers** are:

ACSLS silos: 0,0,2,4
IBM 3494: 00004040
DAS silos: drive1, sdlt_3, etc.
(DAS administrator defined text up to a couple of hundred characters)

If you get an error like "missing new drive information" it is likely that the **silo device identifier** you entered does not match anything that the silo actually knows about. You should recheck the value with the silo administrator.

Note that jbedit does not verify that the **silo device identifier** that you entered actually corresponds to the *device path* that you entered - only that the **silo device identifier** that you entered exists on the silo in question.

−F The **FORCE** flag can be used when adding devices to a silo to force jbedit to create the **NetWorker** device for the specified *device path.*

Note that we prefer you to use the **Scan for devices** function rather than this option since the scan will collect all of the information needed for proper configuration of **Dynamic Drive Sharing.**

The **FORCE** flag will only work if the device you are adding is present on the system that you are running jbedit on.

−l The device being added is logical. Logical devices may be added to AlphaStor libraries only.

−j Name of the jukebox that is to be edited.

−f Device path to be added or deleted.

A new device/drive instance may be added to a library only if the device is auto-detected and available as an **unconfigured device** in the **NSR Storage Node** resource, it is a logical device (see **−l** option), or if it is available as a standalone **NSR device** device. See **ddmgr**(1m) and **nsr_storage_node_resource**(5) for more information on how to detect devices on a storage node.

See **EXAMPLES** on how to add a new device/drive to a jukebox. Also see specific format restrictions when adding an NDMP device path.

−s Name of the NetWorker server. If not specified, local storage node is assumed as the NetWorker server.

−v Run in verbose mode. Multiple **−v** options can be specified to increase the level of verbosity. The higher the level, the more verbose the output will be. Currently has a maximum of 5.

EXIT STATUS

jbedit exits with a 0 on success and with a non-zero value on failure.

EXAMPLES

Adding a drive/device:

To add a new device **/dev/rmt/0cbn** from this storage node:

jbedit -s server -j jbname -a -f /dev/rmt/0cbn -E 82

To add a new ACSLS silo device **/dev/rmt/13cbn** from this storage node:
jbedit -s server -j siloname -a -f /dev/rmt/13cbn -S 0,0,2,3

To add a new device **/dev/rmt/0cbn** from storage node "sn":
jbedit -s server -j jbname -a -f sn:/dev/rmt/0cbn -E 82

To add an NDMP device **nrst0l** from storage node "ndmpsn":
jbedit -s server -j jbname -a -f "ndmpsn:nrst0l (NDMP)" -E 82

Please make sure that the device path ends with " **(NDMP)".**

To add a logical device **ldev01** to an AlphaStor library from storage node "sn":
jbedit -s server -j jbname -a -f sn:ldev01 -l

Deleting a drive/device:

To delete device **/dev/rmt/0cbn** from this storage node:
jbedit -s server -j jbname -d -f /dev/rmt/0cbn

To delete silo device **/dev/rmt/13cbn** from this storage node:
jbedit -s server -j siloname -d -f /dev/rmt/13cbn

To delete device \\.**tape0** from storage node "sn":
jbedit -s server -j jbname -d -f sn:\\.\tape0

To delete an NDMP device **nrst0l** from storage node "ndmpsn":
jbedit -s server -j jbname -d -f "ndmpsn:nrst0l (NDMP)"

FILES

/nsr/res/nsrdb The NetWorker resource database.

SEE ALSO

ddmgr(1m), **changers**(1m), **jbconfig**(1m), **nsrjb**(1m), **relem**(1m), **sjisn**(1m), **sn**(1m), **nsr_device**(5), **nsr_jukebox**(5), **nsr_storage_node**(5), **nsr_storage_node_resource**(5), **EMASS_silo**(1m), **IBM_silo**(1m), **STK_silo**(1m)

DIAGNOSTICS

The following are some messages that jbedit might produce along with their implications and possible solutions.

cannot connect to NSR service on <server name>
jbedit was unable to connect to the NetWorker server on the specified host name. Check that the server is up and running on the server host and that it is reachable from the host on which jbedit is running and retry the operation.

User <user name> on <storage node> does not have Configure NetWorker privilege on <server name>
The user does not have privileges to add/remove devices.

No jukeboxes are currently usable.
There are no jukeboxes configured on the server or no configured jukeboxes are **enabled** and **ready.** Please check **nsr_jukebox**(5) for more details on the "enabled" and "ready" states.

Cannot find jukebox <jukebox name> for server <server name>
The jukebox name specified with **-j** option is invalid. Please check that the jukebox name is defined on the server.

Cannot find any jukebox with a device named <device name> for server <server name>
While deleting the device with **-d** option, either the jukebox name specified with **-j** option is invalid or the device name specified with **-f** option is invalid.

Couldn't retrieve NSR Storage Node resource information for <storage node>
The **nsr_storage_node_resource**(5) on which the device is going to be created does not exist on the server. Create a new storage node for <storage node> and run 'Scan for devices' before adding any new devices from the <storage node>.

The device <device name> is already part of jukebox <jukebox name>
The device name specified with **-f** option already belongs to <jukebox name>.

Cannot find device <device name> in 'unconfigured device names' resource
The device name specified with **-f** option is not available from **nsr_storage_node_resource**(5). Please run 'Scan for devices' on the storage node to discover the device. You may also use the **-F** flag to override this error and create a basic NetWorker device if you are using jbedit to add a device to a silo, although the 'Scan' method is preferred as it collects all the information necessary to properly configure Dynamic Drive Sharing.

NAME jbexercise − NetWorker jukebox exerciser

SYNOPSIS **jbexercise −m model −c control_port** [**−V** *vendor_type*] [**−CdsIv**] [**−D** *drive_number*] [**−S** *slot*]

DESCRIPTION The **jbexercise** command tests the functionality of a jukebox. Before the command can be run, all contents of the jukebox must be emptied except for media loaded in the first and last slots. These pieces of media will be moved around the jukebox as part of the various tests performed by **jbexercise.**

There are two major tests of functionality: drives and slots. Normally both the drive and slot tests are run. Individual component types can be tested by using the **−d** (for drives) and **−s** (for slots) options. In addition, specific components can be singled out in the **−D** and **−S** options. When these options are given, the only test run is on that particular component, that is, if a specific slot is named, the drives test is not run. For drives, the logical address of the component should be given. For slots, the physical address should be given.

Upon startup, the program queries the user for the non-rewinding pathnames of the drives, if any, found in the configuration of the jukebox. This query is not performed if the user is using a jukebox which does not require media to be ejected from a device, that is, the device has automatic ejection capabilities.

The first test moves the media from the first slot to each of the drives. No operator intervention is required.

The second test loads the media from various slots to the first drive. The default is to test the media in the first and last slots in the jukebox. If a specific slot is being tested, the operator must first load that slot with media.

OPTIONS

−C Makes **jbexercise** return the configuration of the jukebox. No tests are run.

−c Specifies the control port which is used to interface with the jukebox (for example, 1.5.0 from scsidev@1.5.0 which could be found by issuing the **inquire**(1m) command).

−d Tests only drives.

−D Tests only the drive with the specified *drive_number.* The logical *drive_number* starts from 0 for the physical drive 1 in the jukebox.

−I Returns only an inventory of the jukebox. No tests are run.

−m Specifies the *model* of the jukebox. Note that for most jukeboxes, the model 'Standard SCSI jukebox' should be the model used with the '-m' option. However, there are certain jukeboxes which require special handling. For these, the specific model name should be specified with the '-m' option. For a list of currently supported jukebox models, run this command without any arguments.

−s Tests only slots.

−S Tests only the specified *slot.*

−v Verbose mode. Prints more information.

−V Specifies a particular vendor id. This allows the vendor to use the same driver for a number of jukebox models.

SEE ALSO **nsrjb**(1m), **nsr_jukebox**(5)

DIAGNOSTICS Most diagnostic messages are specific to each type of jukebox. General messages include the following:

invalid <component> specified
An invalid id for the <*component*> was given. The id must be within the valid ranges of the jukebox configuration. <*component*> can be a drive, port or slot.

status incorrect, media present
There is media loaded in a component but the *component status* operation does not indicate this.

status incorrect, invalid slot location
The *component status* operation is giving the incorrect source slot of the loaded media.

no drives found!!!
No drives were listed in the configuration.

no slots found!!!
No slots were listed in the configuration.

NAME jbverify − check jukebox/device configurations in NetWorker.

SYNOPSIS **jbverify** [−**a**] [−**d** { −**i**|−**u** }] [−**D** *devicename*]...
[−**f** *filename*] [−**F**] [−**h**] [−**H** *hostname*]...
[−**I** *Invoker*] [−**j**] [−**J** *JB name*]... [−**l**] [−**M**] [−**n**]
[−**N**] [−**P** *port*] [−**q**] [−**Q**] [−**r** *no. of retries*]
[−**R**] [−**S** *slot*] [−**s** *server*] [−**t**] [−**U**] [−**v**]... [−**Z**]

DESCRIPTION **jbverify** verifies the devices defined in the NetWorker database, making sure that each one of them is configured properly by checking them for accessibility and usability. To do this, **jbverify** makes use of NetWorker processes and requires that the NetWorker server (nsrd) be running on the server machine, and the NetWorker client (nsrexecd) be running on the client machines.

By default, **jbverify** checks all devices in the NetWorker database, but can be instructed to check only jukeboxes, only stand-alone drives or only local devices using the -j, -d and -l options respectively. Individual jukeboxes and drives can also be checked for by using the -J and -D options. Devices belonging to specific hosts can be checked using the -H option.

For jukeboxes, **jbverify** ensures proper configuration by loading a tape into each drive and unloading them, without performing any write operations on the tape. The only exception to this is if the -t option is used, explained below. A slot to be used for the test can be specified by using the -S option. If no slot is specified, **jbverify** goes through all the slots defined as available to NetWorker and loads the first one available.

Apart from checking for accessibility and usability, **jbverify** can run a series of tests on tapes loaded into the drives being tested by calling on NetWorker's **tapeexer** program (see tapeexercise (1m)), when used with the -t option.

Running **tapeexercise** involves writing to the tape to determine the tape drive's usability, so when -t is specified, any volume which has a NetWorker label on it is immediately rejected as unusable and the next slot is tried. If there are no non-NetWorker tapes in any of the slots, **jbverify** exits without doing any tests.

jbverify can be run on any storage node, and can be used to test any device on that storage node provided the device has been configured in NetWorker. When run on the NetWorker server, it can be used to test any device on the network which has been configured in NetWorker. For a storage node which is not a NetWorker server to be able to test devices other than its own, the nsrexecd on the target machine will have to be started with the -s option with the invoking storage node as the argument, or have the invoking storage node listed in the target machine's 'servers' file.

For example, if the NetWorker server is node NS, and there are two storage nodes Sto1 and Sto2: for Sto1 to test devices on Sto2, the nsrexecd on Sto2 should be started as "nsrexecd -s NS -s Sto1." Or the servers/rservers file on Sto2 should have Sto1 listed as one of the valid servers.

SmartMedia devices are not tested by **jbverify.**

jbverify has extensive verbose messages built into it. In case of error in operation or inexplicable behaviour, it is always helpful to use the -v options to diagnose the behaviour.

OPTIONS

−**a** Tells **jbverify** to check all devices, even if they are disabled. By default, disabled devices are not tested. This option is not supported at present.

−**d** This option tells **jbverify** to check only stand-alone drives. No jukebox devices

are tested.

−D This option is used to test a specific drive. The drive name should exactly match the name specified in the NetWorker drive resource. Multiple drives can be specified by using the -D option multiple times. If a jukebox drive is specified using this option, it is treated as a stand-alone drive.

−f Used to redirect **jbverify's** output to a file. The argument is the file name to which the output is to be redirected.

−F Reserved. This option is used internally by **jbverify** to indicate that this is a remotely forked **jbverify.**

−h Show the help options.

−H Tests the devices on the hostname mentioned. Use this option multiple times to test multiple hosts. Any other option specified on the command-line along with -H will be propagated to the remote host being tested, except for the -D and -J options. When -H is used, only devices belonging to that host are tested and hence only those -D and -J options which specify devices belonging to that host are propagated forward.

−i Go into interactive mode. Used with -d for stand-alone devices. This option is useful when testing stand-alone devices on the local machine. If a particular stand-alone device does not have a tape loaded, the -i option prompts the user to load a tape or cancel the operation so that it can skip to the next drive. The -l option has to be specified with the -i option. Cannot be used with jukeboxes.

−I Reserved. Used internally by **jbverify** to specify the name of the invoking host machine to a remote **jbverify.**

−j Check jukebox devices only. **jbverify** checks only jukebox devices defined in the NetWorker database. All other devices are ignored.

−J This option is used to test a specific jukebox. The jukebox name should exactly match the name specified in the NetWorker jukebox resource. Multiple jukeboxes can be specified by using the −J option multiple times.

−l Check local devices only.

−M Reserved. Used internally by **jbverify** to indicate that it is being invoked by a NetWorker process. Messages are sent to the NetWorker server instead of being echoed to the stdout.

−n Perform tests in the no-op mode. **jbverify** runs through the motions of testing the devices after duly processing all given options but does not actually do the tests.

−N For a remote **jbverify,** put nsrexec into the same verbose mode as the **jbverify.** Usually redundant, but may be useful for debugging.

−P Reserved. Used internally by the **jbverify** process to indicate to a remote **jbverify** the port number on which the server is listening.

−q Run both the local and the remote **jbverify** in the quiet mode.

−Q Run only the remote **jbverify** in the quiet mode. The results of the remote **jbverify** operation can still be seen in the final status report printed out by the local **jbverify.** If -v is used on the command-line with -Q, the local **jbverify** will run in the verbose mode while the remote **jbverify** runs quietly. -q and -v are mutually exclusive. Specifying both will result in **jbverify** running in level 1 verbose mode.

−r Number of retries on error. Chiefly used on load and unload errors. **jbverify** will retry the number of times specified if there is an error in operation.

−S Slot to be used for jukebox devices. The given slot will be used for loading tapes into jukebox devices during the test. If multiple jukeboxes are to be tested, make sure that the same slot in each of those jukeboxes has a valid tape. If -t is specified, the tape in the slot has to be a non-NetWorker tape, else **jbverify** exits with an error.

−s Name of NetWorker server being tested.

−t Perform **tapeexercise** on tapes. See **tapeexercise**(1m) for details. If -t is specified, there has to be a non-NetWorker tape in one of the slots for the exercise to proceed. If -S is specified, the specified slot has to contain a non-NetWorker tape.

−u Run in unattended mode. Similar to the -i option and used for stand-alone devices only. If any device is not loaded with a tape, the -u option skips the device and goes on to the next one in the list. Either -u or -i has to be specified with the -d option.

−U Output a UTF-8 encoded file. When used with -f option, the output file will be UTF-8 encoded.

−v Run in verbose mode. Multiple -v options can be specified to increase the level of verbosity. Higher the level, more verbose the output. Currently has a maximum of 5.

−Z Reserved.

EXIT STATUS

The following are the error numbers with which **jbverify** could exit:

ENWTAPE (51) : Found NetWorker tape when trying to run tapeexercise.
ELOADDETECT (52) : Unable to detect loaded state of a device.
EMEMORY (53) : Out of memory.
ESRCEMPTY (54) : The source slot was empty.
EDSTFULL (55) : The destination drive was full.
EUNLOAD (56) : Error in unload.
EUNKNOWN (57) : Unexpected error.
ERDLABEL (58) : Error in read label operation.
ESPAWN (59) : Error in spawn operation.
EREAP (60) : Error in reaping tapeexercise program.
ELOADED (61) : Drive already loaded.
ECONNECT (62) : Error in connect operation.
ETAPE(40) : Error in tape device in tapeexercise.
EBASICTEST(41) : Error in basic test in tapeexercise.
EEOTTEST(42) : Error in EOT test in tapeexercise.
EFSFTEST(43) : Error in FSF test in tapeexercise.

EXAMPLES

Testing all devices without tapeexercise:

To test all stand-alone and jukebox devices, just run jbverify without any options:

```
jbverify
```

To test all devices with verbose messages, use the **-v** option the required number of times:

```
jbverify -v -v -v
```

Testing only stand-alone devices, in interactive mode:

To test only stand-alone devices, use the **-d** option. **-i** sets the interactive

mode:
jbverify -d -i -l

The -l option has to be specified when using the -i option since interactive mode is not supported for remote devices.

Testing only jukebox devices:
To test only jukebox devices, use the **-j** option:
jbverify -j -v -v

Redirecting output to a file:
To redirect the output of jbverify to a file, use the **-f** option:
jbverify -j -f output.jbv −v −v −v

Testing remote hosts
To test all the jukebox devices on hosts A and B, use the **-H** option:
jbverify −H A −H B −j −f outputfile

This tests only the jukebox devices on both hosts A and B and redirects the output to outputfile.

Running in quiet mode
To run jbverify in the quiet mode, use the −**q** option:
jbverify −q

This will result in only the final status report being printed. To run the local jbverify in the verbose mode, but all remote operations quietly, use the −**Q** option:
jbverify -v -v -v -Q

This will result in verbose output for all local operations but none for the remote ones. The status of the remote operations can be seen in the final status report.

Specifying no. of retries on load/unload operations:
To specify a certain no. of retries on errors, use the −**r** option:
jbverify −j −r 10 −S 12 −v

The above command makes jbverify use slot 12 of the jukebox for load and unload operations and makes it retry 10 times on errors.

Running tapeexercise on tapes:
To run tapeexercise on tapes loaded into devices, use the −**t** option:
jbverify −j −S 12 −t −v

FILES **/nsr/res/nsr.res** The NetWorker resource database.

SEE ALSO **jbconfig**(1m), **jbexercise**(1m), **nsrjb**(1m), **nsr_device**(1m), **nsr_jukebox**(5), **nsr_storage_node**(5), **tapeexercise**(1m)

DIAGNOSTICS The following are the error messages that jbverify might produce along with their implications and possible solutions.

Bad resource database file!
jbverify was unable to get the resource information about devices from the NetWorker RAP database. Check if the NetWorker Server is up and running and if it is reachable from the current host.

Basic Test in tape exercise failed!
The Basic Test in tapeexercise failed on the loaded tape. See tapeexercise (1m) for more details.

Can't specify both -i and -u at the same time!
The -i and the -u options are mutually exclusive. Choose one of them and retry

operation.

Cannot use slot for stand-alone devices! Ignoring...
The -S option is useful only for jukeboxes. This is just a warning that the option is being ignored.

Cannot run in interactive mode for remote devices
-- use -l!
The -i option is currently supported only for local devices. Specify -l to test only the local devices.

Could not connect to server! Quitting...
The remote jbverify could not connect to the main jbverify for some reason. Examine other error messages to establish cause.

Could not establish server socket! Quitting...
jbverify could not open a socket to receive requests from remote jbverifys. Examine previous error messages for exact cause of problem.

Could not extract control port info.
jbverify was unable to parse the jukebox resource information it obtained about a jukebox from the RAP database. This might indicate a corruption of the RAP database in NetWorker. Check if you can see the contents of the jukebox resource from NetWorker Management Console. Retry operation.

Couldn't find control port in JB definition!
jbverify was unable to parse the jukebox resource information it obtained about a jukebox from the RAP database. This might indicate a corruption of the RAP database in NetWorker. Check if you can see the contents of the jukebox resource from NetWorker Management Console. Retry operation.

Could not find enabled drive <name> in database!
A device was specified to be tested and jbverify could not find this device in the resource database of NetWorker. The most common reason would be incorrectly specifying a device name. The device name has to exactly match the name given in the NetWorker device resource, including the "rd=..." prefix, if any.

Could not find jukebox <name> in database!
A jukebox was specified to be tested and jbverify could not find this jukebox in the resource database of NetWorker. The most common reason would be incorrectly specifying a jukebox name. The name has to exactly match the name given in the NetWorker jukebox resource, including the "rd=..." prefix, if any.

Could not get block size for this tape!
jbverify could not find the defined blocksize for this tape. A default of 32k is usually assumed.

EOT Test in tape exercise failed!
The EOT Test in tapeexercise failed on the loaded tape. See tapeexercise (1m) for more details.

Error in checkmedia operation on host <name>!
The remote jbverify reported an error in checking the status of the device. See earlier error messages for more information.

Error! Directory <name> doesn't exist!
This message is printed when processing a disk file drive and the said directory does not exist.

Error in eject tape from drive <name>! Skipping...
There was a problem in ejecting the tape in the said drive. jbverify will skip

testing this device and continue on with the next in line.

Error in read label operation! Cannot proceed with test!
There was a problem while trying to read data off the loaded tape. Check previous error messages to find the cause.

Error in resdb_query in getting device info.
jbverify was unable to get the resource information about devices from the NetWorker RAP database. Check if the NetWorker Server is up and running and if it is reachable from the current host.

Error in resdb_query in getting JB info.
jbverify was unable to get the resource information about jukeboxes from the NetWorker RAP database. Check if the NetWorker Server is up and running and if it is reachable from the current host.

Error in unload. Drive <num> (<name>), slot <num>
There was an error in the unload operation of the said drive. Check previous error messages for possible cause and error no. Try operation again in a higher verbose mode.

Error in unloading jukebox drives: <name>
There was an error while trying to unload the said drive. Check other error messages for cause.

Error reported in eject tape from drive <name>! Device is offline.
There was an error reported by the NetWorker process during the eject operation, but the tape seems to have been ejected; jbverify will continue to unload the tape to its slot.

FSF Test in tape exercise failed!
The FSF Test in tapeexercise failed on the loaded tape. See tapeexercise (1m) for more details.

Failed to create xdr stream!
This usually denotes lack of enough physical memory in the system. Check earlier error messages for more information.

Failed to detect loaded volume on drive <name> even after <num> tries. Giving up...
jbverify failed to detect a loaded tape drive after putting a tape into the drive. Sometimes this might happen if the drive is slow and the delay is too little. Try the operation again with a high number as the argument to the -r option or increase the load sleep attribute in the jukebox resource.

Failed to get connection from remote jbverify! errno: <num>
jbverify started a remote jbverify and is waiting for it to connect to it but has timed out without getting a connection request. Examine other error messages to find the cause. One common cause is that the machine you are running jbverify on does not have the permission to request execution on the remote machine. To obtain permission, the nsrexecd on the remote machine has to be started with "-s <local machine name>." See example in the main section of this man page for more information.

Failed to get response for check media from remote host <name>
jbverify failed to get response from the remote jbverify on a request for checking the status of a device. This could be because the remote jbverify was killed or terminated abnormally, the remote machine went down, or just network

problems. Check if you can ping the machine and retry operation.

Failed to get stat packet!
jbverify failed to receive an expected status packet from the remote jbverify. This could be because the remote jbverify was killed or terminated abnormally, the remote machine went down, or just network problems. Check if you can ping the machine and retry operation.

Failed to read request packet from server!
A remote jbverify failed to receive a request packet from the main jbverify. This could be because the main jbverify was killed or terminated abnormally, the machine went down, or just network problems. Check if you can ping the machine and retry operation.

Failed to redirect output to <name>. Errno <num>
A system call failed. Run in verbose mode and contact support with error numbers and messages.

Failed to send FMEDIA on sock <num>! Errno <num>
jbverify failed to send a request to check device status to the remote jbverify. This could be because the remote jbverify was killed or terminated abnormally, the remote machine went down, or just network problems. Check if you can ping the machine and retry operation.

Failed to spawn tapeexer!
Failed to exec the NetWorker binary tapeexer. Check if the binary exists and if it has the adequate permissions. Check other error messages for causes.

Failed to start nsrexec! errno: <num>
jbverify failed to start the nsrexec process on the local machine. Examine previous error messages for exact cause. Some of the causes could be missing nsrexec binary, missing execute permissions, corrupt file etc.

Failed to start remote jbverify on <host>! errno <num>
jbverify was unable to start jbverify on a remote machine. Check earlier messages for more information. Some of the causes could be that nsrexecd is not running on the remote machine, nsrexecd is of a version prior to 6.1, you are running jbverify on a machine which is not the server and which is not allowed to request execution on the remote machine. The last can be rectified by running the remote nsrexecd with "-s <server> -s <machine>" option where <server> is the NetWorker Server machine and <machine> is the machine on which you are running jbverify. See explanation and example in the main section of this man-page for more details.

Have to specify -i or -u with -d option.
The -d option has to be specified with either the interactive (-i) or the unattended mode (-u). Choose one of them and retry operation.

Invalid option specified: <option>.
An invalid option was specified. Use the -h option to get a list of valid options.

Malloc error
System is out of physical memory. jbverify failed to allocate the required memory for an operation. Exit some applications and retry the operation or increase the amount of memory on the machine.

NetWorker tape (<label>) in the drive. Cannot proceed with test!
The drive has a tape with the said NetWorker label on it. If jbverify is run with -t, it needs a tape without a NetWorker label on it to successfully run the tapeexercise program on it. If there is no non-NetWorker tape in any of the

slots, place one into one of the slots and retry the operation.

No block size found for this device: <name>!
jbverify could not find a blocksize defined for this device in the NetWorker database. This usually means that a default of 32k is assumed for this device.

No enabled stand alone devices found.
The current configuration has no stand-alone device defined. This is just an informational message.

No enabled jukeboxes found in database.
The current configuration has no jukeboxes defined. This is just an informational message.

No tape in slot <num>. Quitting...
If -S was specified and there is no tape in the specified slot, jbverify posts this message and quits. Put a tape in the slot or specify another slot with a tape in it and retry operation.

Query resdb failed, err: <errmsg>.
A RAP query to the NetWorker database failed. Check if the NetWorker Server is up and running and if it is reachable from the current host.

Ran out of slots to choose from! Quitting...
While trying to find a slot to use to load a tape into a jukebox device, jbverify has run out of slots to try. If run with -t, jbverify needs to find a slot which has a tape without a NetWorker label on it as it will not overwrite NetWorker tapes even if they are no longer in the media database.

Received invalid request from server:type: %d
jbverify received an unexpected request from the main jbverify. This could happen if the two machines involved are running different versions of jbverify. Check and make sure that this is not so. This could also mean memory corruption. Retry operation at verbose level 5 and if error persists, send log to customer support at EMC.

Received unknown packet from remote host <name>!
jbverify received an unexpected packet from the remote jbverify. This could mean memory corruption. Retry operation at verbose level 5 and if error persists, send log to customer support at EMC.

SCO postion Test in tape exercise failed!
The SCO position Test in tapeexercise failed on the loaded tape. See tapeexercise (1m) for more details.

Skipping disabled drive <name>
jbverify, at present, does not test drives disabled in NetWorker. In the future, the -a option may be enabled to do so.

Skipping to next drive in list...
After a load/unload error, jbverify is stopping the test of a drive and moving on to the next one in its list.

Slot <slot num> has Networker tape.
The -t option was used and the slot from which the drive was loaded contained a NetWorker tape. If the -S option was used, this is a fatal error. If not, jbverify will try other slots to see if it can find a non-NetWorker tape.

Slot needs to be a valid number!
The slot specified with the -S option has to be a real number.

Source slot empty! <slot num>
The -S option was used but the specified slot did not contain a tape. Specify a slot which has a tape in it. If the -t option is also being used, specify a slot

with a non-NetWorker tape in it.

Tapeexer executable not found!
The tapeexer executable was not found. Check if it exists.

Tapeexer exited on signal <num>
The tapeexer process was killed by the given signal.

Tapeexer exited abnormally with exit code <num>
The tapeexer process exited abnormally with the given exit code.

Tapexercise on <name> exited without an exit status!
jbverify was unable to get the exit status of the tapeexer process. This is a very rare case and might never happen unless the OS has a bug.

Unable to authenticate remote process!
jbverify was unable to authenticate a connection request from the remote process.

Unable to get JB name! Skipping to next...
jbverify was unable to find any name specified for the jukebox in the jukebox resource. Check the jukebox resource for any corruption and restore the NetWorker resource directory if needed.

Unable to find any devices in jukebox!
jbverify was unable to find any devices configured for the jukebox. This is an error condition since it is not usually possible to have an enabled jukebox in NetWorker with no defined devices. Check NetWorker configuration and run jbverify again.

Unable to get device info for <name>
jbverify could not find any info for this device in the NetWorker database. Check that the name of the device matches exactly with the name defined in the NetWorker resource, including the "rd=..." prefix if it is a remote device/jukebox.

Unable to get JB name! Skipping to next...
jbverify was unable to parse the jukebox resource information it obtained about a jukebox from the RAP database. This might indicate a corruption of the RAP database in NetWorker. Check if you can see the contents of the jukebox resource from NetWorker Management Console. Retry operation.

Unable to load tape into drive <num> (<name>) as it seems to be loaded!
The said drive contains a tape even though jbverify must have unloaded it before trying the load. This might happen if the drives are not configured in the right order in the jukebox. Check if the order of the drives is correctly configured in NetWorker.

Unable to to malloc for connlst! errno: <num>
System is out of physical memory. jbverify failed to allocate the required memory for an operation. Exit some applications and retry the operation or increase the anount of memory on the machine.

Unable to open <name>. Errno: <num>
jbverify was unable to open the filename specified with the -f option. Check if you have permissions.

Unable to proceed to test drive <no>(<name>) in JB <name> as device is still loaded!
jbverify found the said drive to be loaded inspite of unloading it before accessing it. Check if any other application is using this jukebox. Also check if the previous unload operation by jbverify failed by looking at the error messages

or by running in higher verbose mode.

Unable to unload drive <num> (<name>)! May not be configured right!

jbverify was unable to unload the said drive. This might happen if the drives are not configured in the right order in the jukebox. Check if the order of the drives is correctly configured in NetWorker.

Unknown state. Quitting...

jbverify cannot determine the status of a load. This might happen with corrupted memory. Try operation again and contact support in case of failure.

NAME jobkill − NetWorker jobs termination program

SYNOPSIS

jobkill
[−**s** *server*] [−**c** <*client*>] [−**t** <*job type*>] [−**f** <*output_file*>] [−**T** <*timeout*>]

jobkill
[−**s** *server*] [−**f** <*output_file*>] [−**T** <*timeout*>] −**j** <*jobid*>

jobkill
[−**s** *server*] [−**c** <*client*>] [−**t** <*job type*>] [−**f** <*output_file*>] [−**T** <*timeout*>] −**i** <*input_file*>

DESCRIPTION

jobkill utility allows an administrator to kill individual jobs by specifying their jobid, or it will query jobs database for running jobs of a given type and/or on a given client and let an administrator kill them from the interactive prompt. Without arguments **jobkill** will query for all running jobs. If there are no running jobs fulfilling the criteria, **jobkill** exits silently.

In all modes, **jobkill** takes "-s <server>" option to specify the server on which **nsrjobd** is running, and "-f <file>" to specify the file to which the output should be directed. In the "-i <input file>" mode, '-' is supported for input to be read from stdin. "-w" with no arguments can be used to instruct **jobkill** to poll **nsrjobd** for status until all requests have been obeyed by remote jobs.

One must be root to execute **jobkill.** Operate NetWorker priviledge is required for terminating jobs.

jobkill in default behaviour reports success once the termination request is acknowledged by **nsrjobd.** Normally **jobkill** does not wait for the termination request to complete, due to the asynchronous nature of the termination handling, and lack of a parent-child relationship between **jobkill** and the job being killed. **-w** provides an option to poll jobs' status to detect the success or failure of actual termination operations, but one needs to remember that it may take several minutes before the job obeys the request. This option is intended to be used in non-interactive mode, especially with the list of jobids provided in an input file. In interactive mode, a user can easily verify that the job had successfully exited using the 'r' (refresh) command at the **jobkill's** prompt.

jobkill can terminate anything that listens on a channel it has with **nsrjobd.** That means either an entire **savegrp** or any worker job spawned by **nsrjobd.** It cannot kill manually started jobs (with exception of creator jobs such as **savegrp**). A job terminated via **jobkill** will have the attribute "Reason job was terminated:" filled in with "Kill request from jobkill utility"

OPTIONS

−**s** *server*
The name of the NetWorker server to contact. Appropriate permissions level (Operate NetWorker) will be enforced.

−**c** *client*
In interactive mode, limit the query to running jobs on specified client only. Can be combined with -t to further narrow down the result list.

−**j** *job id*
Single job id of the job to terminate.

−**t** *job type*
In interactive mode, limit the query to running jobs of the specified type only. Can be combined with -c to further narrow down the result list.

−**T** *timeout*
Timeout in seconds to wait before issuing a forceful shutdown signal (an equivalent of kill -9).

−**i** *input file*
Input file containing the list of job ids to terminate. '-' indicates stdin.

−**f** *output file*
File to direct the output to.

−**w** Wait for the jobs to terminate before exiting. Due to the asynchronus nature of interaction with **nsrjobd,** normally **jobkill** considers it a success when **nsrjobd** indicates that the signal was sent on the channel without errors. It may take much longer for the job to actually exit. In interactive mode, **r (refresh)** can be used to check whether the job remains active. Using **-w** will cause **jobkill** to poll nsrjobd for status of the jobs until all termination requests have been obeyed. This option is intended for non-interactive mode of operation.

Example usage:

1) to kill an individual job
jobkill -j jobid

2) to query nsrjobd and specify the jobid at the prompt
jobkill [-c <client>] [-t <type>]

3) to kill multiple jobs at once using an input file
jobkill -i <input_file>

NAME jobquery − NetWorker jobs database query program

SYNOPSIS

jobquery
[−**s** *server*] [−**i** *file*]

jobquery
[−**s** *server*] [*query*]

DESCRIPTION The **jobquery** command is a command-line based program used to query NetWorker server's jobs database. Its interface is similar to that of **nsradmin** program.

OPTIONS

−**i** *file* Takes input commands from *file* instead of from standard input. In this mode, the interactive prompt will not be printed.

−**s** *server*
Opens a connection to the named NetWorker server.

RECORDS Each jobs database entry is made up of a list of named attributes. Each attribute can have zero or more values. The attribute names and values are all represented by printable strings. Upper and lower case is not distinguished on comparisons, and spaces are ignored except inside the names and values.

The format for specifying attributes and attribute lists is:

attribute ::= *name* [**:** *value* [**,** *value*]*]
An attribute is a name optionally followed by a colon, followed by zero or more values, with values separated by commas. A comma at the end of a line continues the line.

attribute list ::= *attribute* [**;** *attribute*]*
An attribute list is one or more attributes separated by semicolons. A semicolon at the end of a line continues the line. The list is ended by a newline that is not preceded by a comma or semi-colon.

Here is an example of an attribute list:

```
name: "nyx.lss.emc.com:Probe";
type: savefs job;
jobid: 480435;
```

For more information on attributes and attribute lists see the **resource**(5), and **nsr_resource**(5), manual pages.

COMMANDS At each input prompt, **jobquery** expects a command name and some optional arguments. Command names can be shortened to the smallest unique string (for example, p for print). Command arguments are always specified in the form of an attribute list.

all
Displays all entries in the jobs database.

help Print usage of all available commands.

print [*query*]
Print the resources that match the current query. If a *query* is specified, it becomes the current query. If a name has been specified for the the current show list, only the attributes for the specified name in the show list will be displayed.

quit
Exits **jobquery**

show [*name***;** ...]

If a name list (really an attribute list with no values) is specified, add those names to the show list. Only these attributes will be displayed in subsequent **print** commands. If no name list is given the show list is cleared, resulting in all attributes being shown.

types Print a list of all known types.

. [*query*]
If a *query* is specified, this command will set the current query without printing the results of the query. Otherwise, it will display the current query, show list, server binding, and options.

EXAMPLES

print type:savefs job
Print all entries of type **savefs job** and make this the current query.

show type; name
Set the show list to display only the attributes **type** and **name**.

SEE ALSO

nsr_resource(5), **resource**(5), **nsradmin**(1m).

DIAGNOSTICS

The following exit status values are meaningful:

0 Program exited normally.

1 There was a usage or other non-query related error.

NAME lcmap − determine path-ownership in a cluster

SYNOPSIS **lcmap**

DESCRIPTION NetWorker software in a clustered environment needs to determine a mapping of a save path (filesystem or raw device) to the proper physical or virtual cluster client. This ensures that the data is saved to the correct NetWorker client name and facilitates recovery of a virtual NetWorker client irrespective of its current physical host. Path-ownership resolution also allows NetWorker server to identify with a virtual service in a cluster with its own configuration directory and become a highly available application when put under control of the cluster software.

A platform and cluster specific **lcmap** script is installed when the **networker.cluster**(1m) script is run in order to configure the NetWorker software as highly available. The **lcmap** script queries the cluster software and outputs to stdout the path-ownership information in the EMC Resource Administration Platform (RAP) format, which is cluster and platform independent. The **lcmap** script is called by NetWorker software whenever the path-ownership information needs to be determined or updated. The format of the output is NetWorker internal and can change between the releases. The script should not be modified manually.

EXAMPLES The following is an example output from lcmap script:

```
type: NSR_CLU_TYPE;
clu_type: NSR_LC_TYPE;
interface version: 1.0;

type: NSR_CLU_VIRTHOST;
hostname: chase;
owned paths: /global/chase/nsr;

type: NSR_CLU_VIRTHOST;
hostname: hunt;
owned paths: /global/hunt/data1, /global/hunt/data2;
```

SEE ALSO **save**(1m), **savegrp**(1m), **pathownerignore**(5), **networker.cluster**(1m)
The *NetWorker Administration Guide*

NAME ldunld − load or unload a tape device

SYNOPSIS **ldunld** { **-u** | **-l** } [**-a** *b.t.l*]

DESCRIPTION The **ldunld** program sends a **load** or **unload** command to a specified device.

OPTIONS

−**a** *b.t.l* Selects a specific ordinal SCSI address, where **b** is the logical SCSI bus, **t** is the SCSI target, and **l** is the SCSI logical unit number (LUN) on that target (see **libscsi**(1m)). This is a required option.

-u Unloads media from the specified device.

-l Loads media to the specified device.

SEE ALSO **libscsi**(1m)

NAME lgtolic − EMC license utility command

SYNOPSIS
lgtolic [−**s** *server*] −**c** *enabler_code*
lgtolic −**i** [−**m** *hostfile_dir*]
lgtolic [−**s** *server*] −**l**
lgtolic [−**s** *server*] −**u** *enabler_code* −**a** *authorization_code*
lgtolic [−**s** *server*] −**v** *enabler_code*

DESCRIPTION The **lgtolic** command is used to manipulate EMC licenses that are stored within a license resource database. The license resource database is administered by a EMC license daemon. For a description of the license daemon, see **lgtolmd(1m).**

OPTIONS

−**a** *authorization_code*
Authorizes a license with the specified authorization code, making the license permanent. Specify the license to be authorized by using the −**u** option and the −**a** option. To obtain authorization codes for this product via the World Wide Web, simply point your web browser to customernet.emc.com to enter the enabler code for each authorization code that you request. For more details on product licensing, including other methods to obtain authorization codes, refer to the product Installation and Administration Guide and the latest Release Supplement.

−**c** *enabler_code*
Creates the license indicated by the specified enabler code. Enabler codes are listed on enabler certificates provided to you when you purchased this product. An authorization code is required to make each license permanent.

−**i** Prints out the hostid of the machine on which this command is running.

−**l** Lists all of the EMC product licenses currently stored within the license resource database. This is the default.

−**m** *hostfile_dir*
Specifies the directory where the hostids file resides. If this option is specified, the program will use the list of hostid(s) in the hostids file that resides in this directory to generate a composite hostid. This option is useful if the licensing manager is installed on a cluster machine or to force the hostid to be IP address-based instead of machine security ID-based on an NT machine. For NetWorker, the typical directory for the hostids file is */nsr/res.* For a standalone licensing manager running on a machine that does not have the NetWorker server installed, the typical path is */nsr/lic/res.* The format for the list of hostids in a hostids file is: **hostid1:hostid2:hostid3** where hostid is a hexadecimal string. This option must be used to specify a hostid file.

−**s** *server*
Specifies the hostname, RPC program number, and version for the license daemon whose database you are targeting. License daemon information is displayed in the following format:
< *hostname* >:< *rpc_number* >:< *version* >
Note: If you do not specify the −**s** *server* option, **lgtolic** uses the default values that map to the daemon used by the product shipped. The current default for the hostname is localhost, the default for the RPC program number is 390115, and the default for the RPC version number is 2. You can also use environment variables to change these three defaults. **LMD_HOSTNAME** changes the default hostname of the machine where the license daemon is running. **LMD_PROGNUM** changes the default RPC program number for connecting to

the license daemon. You should never have to use this. **LMD_VERSION** changes the default RPC version number for connecting to the license daemon. The following example uses default hostname and RPC program number, but uses RPC version number 1 to list all licenses.
Example: lgtolic -s "::1" -l
To specify a license daemon located on an alternative machine, use the **−s** < *hostname* > option.

−u *enabler_code*
Updates an installed license with the authorization code specified with the **−a option** at the command line.

−v *enabler_code*
Deciphers the specified enabler code. The generated output includes information about the license name, type, serial number, and count.

NOTES The daemon information provided by the **−s** option can also be obtained by using the following environment variables:

LMD_HOSTNAME, The name of the host for the license daemon

LMD_PROGNUM, The program number for the license daemon

LMD_VERSION, The version number for the license daemon

DIAGNOSTICS Program not registered
lgtolmd is not running.

Unknown host
Either the default hostname is invalid or the hostname specified with the **−s** option is invalid.

SEE ALSO **lgtolmd**(1m)

NAME lgtolmd − Legato license daemon

SYNOPSIS **lgtolmd** −**p** *product* −**n** *version*

DESCRIPTION The **lgtolmd** daemon is an RPC-based licensing service. This service allows applications to store and manipulate license data. The RPC program number provided by **lgtolmd** is 390115. To support multiple instances of the protocol, the version number is unique to each application. The required parameters are determined by each product's installation script.

OPTIONS −**p** *product*

Specify the product that will be interfacing with the license daemon. The currently supported products are **gems** (for GEMS default install directory /gems) and **opt/SmartMedia** (for SmartMedia default install directory /opt/SmartMedia) on UNIX platforms. For NetWorker, **nsr/lic** (for NetWorker default install directory /nsr) on UNIX platforms.

−**n** *version*

Specify the version number. Some products use a unique version number. Currently, SmartMedia uses version 2 and GEMS Storage Reporter uses version 3. Both GEMS and NetWorker use version 1. The future plan is to have all EMC products use the same license manager, that is, version number 1.

FILES **/[product]/res/lgtolm.res**

Attributes describing the license daemon's license resources. This file should not be manually removed or modified in any way.

/[product]/res/lictype.res

For internal use only. This file should not be manually removed or modified in any way.

/[product]/logs/lgtolmd.log

Log file for diagnostic and informational messages on the license daemon. For example, if a license has expired, this information will be printed to this log as well as to the console.

SEE ALSO **lgtolic**(1m)

NAME libcdi - EMC Common Device Interface Library

SYNOPSIS **libcdi**

DESCRIPTION The **libcdi** library is the EMC Common Device Interface Library.

ACCESS METHODS The following access methods are currently supported by the *libcdi* library. For all CDI test commands, if the **−t** option is not specified, the default method is to use the OS tape driver SCSI passthrough functions.

OPTION ARGUMENT	ACCESS METHOD
g	Use old style MTIO platform generic functions
i	Use iSCSI functions
m	Use NDMP passthrough functions
n	Use NDMP tape functions
s	Use OS tape driver SCSI passthrough functions
t	Use OS tape driver IOCTL functions

NAME libscsi − SCSI device library

DESCRIPTION The SCSI device library is a private set of interfaces that NetWorker uses to communicate with SCSI devices.
Important: SCSI devices are named independently of the platform.

There are several functions in this library. The name of a SCSI device is identified as a combination of bus, target, and logical unit number (LUN) (*b*.*t*.*l*), where *b* is the logical scsi bus, *t* is the SCSI target, and *l* is the SCSI LUN on that target. Do *not* assume that a logical SCSI bus number is related to any specific platform or hardware bus number. Rather, a logical SCSI bus number is a dense positive integer address space that is consistent as long as the hardware configuration of the system remains the same. Target and LUN information is based upon the attached SCSI peripheral devices and their settings. Some platforms enable dynamic addition and removal of SCSI devices, but may require flushing of cached device information (see **lrescan**(1m)).

PERMISSIONS Typically, if a device has no system driver, system privileges are not required for users to send commands to this device. If a device has a system driver (for example, a tape drive), then system privileges are required in order to send a command to these devices.

SEE ALSO **lrescan**(1m)

NAME libsji − Standard Jukebox Interface (SJI) library

DESCRIPTION The Standard Jukebox Interface (SJI) is a public set of interfaces that NetWorker uses to communicate with jukeboxes. Generally the function of this library is to convert SJI commands, as formed by NetWorker, to the appropriate SCSI commands (since most autochangers are SCSI based). But the underlying attachment to the jukebox is irrelevant to the functioning of this interface.

There are three entry points into this library:

void * sji_open (char * devname)
This opens a channel to an SJI compliant jukebox, named *devname.* A channel token, of type *void* * is returned if the channel is opened successfully, otherwise a channel token of type *NULL is returned.* The device name *devame* can be a specific ordinal SCSI address, for example, *scsidev@b.t.l,* where *b* is the logical scsi bus, *t* is the SCSI target, and *l* is the SCSI logical unit number (LUN) on that target. For platforms that do not use EMC device drivers, the device name can also be a platform-specific style device name, for example, */dev/sjid1u1.*

int sji_cmd (void *token, int cmd, void *arg)
This sends an SJI command to the device opened by the **sji_open** command.

void sji_close (void *token)
This closes a channel to the device opened by the call made to the **sji_open** command.

FILES The location of the SJI library varies from platform to platform.

NAME dasadmin − ADIC/EMASS/Grau silo administrative utility
libstlemass − shared library for communication to
ADIC/EMASS/Grau silo

SYNOPSIS **dasadmin** command [options] [parameters]

dasadmin.exe command [options] [parameters] (NT only)

libstlemass.so (Solaris)
libstlemass.so.a (AIX)
libstlemass.sl (HPUX)
libstlemass.so.1 (SGI)
libstlemass.so (DECAXP)
libstlemass.dll (NT i386)

DESCRIPTION **For dasadmin:**

This is not a complete listing of all possible **dasadmin** commands, but does include those commands that are of use with NetWorker. For a complete discussion, see the DAS Installation and Administration guide provided by ADIC, EMASS or Grau.

mo[unt] [−**t** *type*] *volser* [*drive-name*]
Mounts the tape with the barcode label of *volser* into either the first available drive (if *drive-name* is not specified) or into the drive specified by *drive-name.* If the tape is not the type defined by **DAS_MEDIUM** or **ACI_MEDIA_TYPE,** you can use the −**t** *type* option to get the tape mounted. If the type of the tape and the defined type for the drive do not match, the silo will not load the tape. Note that the drive you are attempting to use must be allocated for your use before you can mount or dismount tapes. See **listd** and **allocd** below.

dism[ount] [−**t** *type*] *volser* | −**d** *drive-name*
Dismounts the tape that is either specified by *volser* or whatever is in the drive specified by *drive-name.* If the tape or drive are of a different type than your default, use the −**t** *type* parameter. As with **mount,** you must have the drive allocated to you to use this command.

ej[ect] [−**c**] [−**t** *type*] *volser-range area-name*
Ejects one or more tapes to the specified eject area. As with other commands, if the type of the tape you are ejecting is different from that defined by **DAS_MEDIUM** or **ACI_MEDIA_TYPE,** you will need the −**t** *type* option. The −**c** specifies a 'complete' ejection for the specified volsers. A complete ejection removes the entry for that volser from the silo controller's internal database. A NON-complete ejection will eject the tape, but the volser's entry in the database will remain, and the volser's state will be set to 'ejected'. This is useful if you anticipate replacing the tape in the silo soon.

in[sert] *area-name*
Moves all tapes that are currently in the specified insert *area-name* from the insert area to the normal storage locations for tapes.

inventory
Starts a full inventory of the silo. **USE WITH CAUTION!** An inventory of this sort can take a very long time! An inventory of a silo with 180 slots takes over 20 minutes.

view [−**t** *type*] *volser*
Displays the current status of *volser,* including the volser, type, attribute, and coordinate.

all[ocd] *drive-name* **UP|DOWN** *clientname*
The **allocd** command is used to allocate and deallocate drives for different

clients. Before you can use a tape drive, the drive must be **allocd'ed UP** for your system. If it is currently **allocd'ed UP** for a different client, it must first be **allocd'ed DOWN** for that client before being **allocd'ed UP** for your system. You cannot **allocd DOWN** a drive that has a tape in it. The tape must be dismounted first.

l[ist]d
listd or **ld** shows the current state of all the tape drives defined in the silo. The information presented will include the *drive-name,* the amu drive (the location in the silo), status (UP or DOWN), type, client the drive is allocated to, and the volser of any loaded tape.

show −op | −ac *client-name*
Shows the operational or access parameters for the specified *client-name.* You must include either −**ac** if you wish to see access parameters, or −**op** if you wish to see operational parameters for the *client-name.* Access parameters include volser ranges and drive ranges that the *client-name* is allowed to use. Operational parameters include whether the *client-name* has complete access, dismount privileges along with the IP address entered for *client-name.*

list *client-name*
Lists any outstanding requests that have been made by *client-name.* If there are any, they are shown, along with the request number and type.

can[cel] *request-id*
Allows you to cancel an outstanding request, assuming that you have the necessary privileges. Use the *request-id* that was shown by the **list** command.

qversion
Shows the version of the DAS server that you are connected to and the version of the ACI protocol you are using to talk to DAS.

qvolsrange *beginvolser endvolser count* [*clientname*]
qvolsrange is the way to obtain a list of the volsers that are available in the silo. *beginvolser* and *endvolser* are volsers of the form "123456". To use the first available or the last available, you can use "". *count* specifies the maximum number of volsers you wish to see.

ENVIRONMENT VARIABLES

These environment variables affect the operation of the silo, and since the processes that are using them include both the commands the user will enter and the processes that are spawned from **nsrd,** they need to be set in a location where they will be in place when **nsrd** is started. The three DAS_ variables are used by **libstlemass,** while **dasadmin** uses **ACI_MEDIA_TYPE** instead of **DAS_MEDIUM.**

For Solaris, the definitions should be placed in **/etc/rc.2/S95networker.**

For AIX, the definitions should be placed in **/etc/rc.nsr.**

For HPUX, the definitions should be placed in **/sbin/rc2.d/S900networker.**

DAS_SERVER
This is either the network name or the IP address of the system that is running DAS. For a single silo, this will usually be the silo controller system. In larger installations, there will probably be only one DAS server for the whole network. It is case-sensitive.

DAS_CLIENT
This is the network name of the system that NetWorker is running on. It is case-sensitive.

DAS_MEDIUM
This variable is used by **libstlemass.** It should be the same as ACI_MEDIA_TYPE.
This is the type of tape drive you are connected to. If this is not specified, the

default value of DLT will be used.

ACI_MEDIA_TYPE
This variable is used by **dasadmin.** It should be the same as **DAS_MEDIUM.** This is the type of tape drive you are connected to. If this is not specified, the default value of DLT will be used. Acceptable values are the same as those listed under **DAS_MEDIUM.**

EXAMPLES

NOTE on ranges:
The **dasadmin** utility will accept volser ranges for some commands. There are three acceptable variations for these ranges:
single volser: "000635"
multiple volsers: "000635, 000789, 098732"
true range: "000610 - 000745"

NOTE on *area-name* and *drive-name*:
area-names usually consist of a letter and 2 digits. The letter corresponds to whether you are referring to an insert area ("I") or an eject area ("E"). You will need to get the correct values from your silo administrator before using them.
drive-names are essentially free-form labels created by whomever installed the silo. They may or may not have any relevance to physical reality, so you will need to see the silo admin to get the correct names. If the silo admin is not available, you can get the same information using **dasadmin listd** along with **dasadmin show −op** *client-name* followed by **dasadmin show −ac** *client-name* commands.

To set up the environment variables necessary for silo operations:

```
setenv DAS_SERVER emask
setenv DAS_CLIENT aurora
setenv DAS_MEDIUM DLT
setenv ACI_MEDIA_TYPE DECDLT
```

To see a listing of all volsers available in the silo:

```
dasadmin qvolsrange "" "" 10000
```

To see the current status of the drives in the silo:

```
dasadmin listd
```

To change the allocation of a drive from client a4 to client aurora:

```
dasadmin allocd DLT1 DOWN a4
dasadmin allocd DLT1 UP aurora
```

SEE ALSO

nsrjb(1m), **jbconfig**(1m), **libstlstk**(1m), **mini_el**(1m), **ssi**(1m), **libstlibm**(1m)

DIAGNOSTICS

The only available diagnostic information is error messages that may be printed out by **dasadmin** and **libstlemass** in the course of normal operations.

NAME libstlibm − shared library for communication to IBM 3494 silos

SYNOPSIS **libstlibm.so** (Solaris)
libstlibm.so.a (AIX)

DESCRIPTION **libstlibm.xxx** is a shared library that handles the communication between **nsrjb** and the IBM silo driver (on AIX) or daemon (on Solaris). The IBM driver/daemon then handles the communication over the network to the silo. There are no options, parameters or environment variables that affect the operation of **libstlibm.** The correct path to this file should be entered when an IBM silo is configured using **jbconfig.** The default values specified by **jbconfig** match the default locations chosen for the installation program, and in most cases can be accepted.

For NetWorker to work with the 3494, you must have first installed IBM's Automated Tape Library support.

On AIX, you will need to install a driver called **atldd** (Automated Tape Library Device Driver). You may also require the **IBMtape** driver (Enhanced Tape and Medium Changer Device Driver) if you are using 3590 drives in your 3494.

On Solaris, you will need to install the **lmcpd** package, (IBM Automated Tape Library Daemon) to use the silo. Again, if you are using 3590 drives, you will also need to install the **IBMtape** driver. Note that when you are using **IBMtape,** there will be two sets of device files that will access a given tape drive. There will be the standard Solaris style **/dev/rmt/Xmbn** type, and there will be the **IBMtape** supported files of the type **/dev/rmt/Xstbn.** You should use the IBM supported device files for proper operation of your tape drives.

Note: EMC cannot supply these IBM drivers. They may be available on an IBM Device Driver ftp site (208.200.29.244), but this is not necessarily a long-term IBM committed site.

SEE ALSO **nsrjb**(1m), **jbconfig**(1m), **dasadmin**(1m), **libstlemass**(1m), **ssi**(1m), **mini_el**(1m), **libstlstk**(1m)

DIAGNOSTICS Errors in communication between the NetWorker server and the IBM 3494 silo are difficult to diagnose. The best method is to use the IBM supplied utility **mtlib** to verify that you have properly configured the 3494 to communicate with your host, and that the entire pathway from either the lmcp driver (on AIX) or the lmcpd daemon (on Solaris) is functioning properly. If **mtlib** does not work, then there is no chance that NetWorker will work.

If there are any questions about the connection between your host and the 3494, it is best to consult IBM, as they support the connection between the host and the silo. IBM supports both network and serial cable connections to the silo. Since the nature of the connection is hidden from NetWorker by the driver/daemon, there is no difference to NetWorker between the two. Customers have successfully used both.

NAME

ssi − StorageTek silo interface module (UNIX only)
mini_el − event logger for use with ssi (UNIX only)
libstlstk − shared library for communication to ssi

SYNOPSIS

ssi [−**A** *ACSLS server*] [−**a** *ACSLS port number*]
[−**S** *SSI port number*] [−**P** *port number*]
[−**r** *retry count*] &

mini_el [−**l** *logfile*] [−**d**] [−**h**] &

libstlstk.so (Solaris)
libstlstk.so.a (AIX)
libstlstk.sl (HPUX)
libstlstk.so.1 (SGI)
libstlstk.so.1 (DYNIX/ptx)
libstlstk.so (DECAXP)
libstlstk.dll (NT i386)

DESCRIPTION

NOTE: in this document, the term "ACSLS server" will be used to indicate the name of the system that is running any one of StorageTek's library manager programs: ACSLS on a Solaris or AIX host, Library Station on an MVS host, or Horizon Library Manager on a system running Windows NT or Windows 2000.

(UNIX only)

The **ssi** command is used indirectly by **nsrjb** to communicate with an ACSLS server. **nsrjb** loads **libstlstk,** which handles the TCP calls to and from **ssi. ssi** then handles all of communication to and from the ACSLS server. Starting with ACSLS version 5.3, it is possible to run NetWorker (either a server or a storage node) on the same host that ACSLS is running on.

ssi and **mini_el** must be running on the system on which jbconfig was run to create the jukebox resource. **ssi** and **mini_el** are almost always run as background processes, and are usually started automatically by the system.

In addition to **ssi** and **mini_el,** a shared library file (usually called **libstlstk.xxx** where xxx is an operating system-dependent extension) is also required. An appropriate version of this library is installed as part of NetWorker.

New in version 2.00 of **ssi:**

ssi now supports communication with the ACSLS server on a specified port number, using the −**a** command line option. This is part of the STK firewall enhancement. The ACSLS server must be running version 7.1 to use this functionality.

While you can still start **ssi** the same way as before - using the environment variable CSI_HOSTNAME to select the ACSLS server to connect to - you can also specify the ACSLS server hostname on the command line using the **-A** option. By using the −**a** option, you may specify the port number that the **ssi** process will use when connecting to the ACSLS server. The ACSLS server must be configured to listen on this port. Using the −**S** option, the **ssi** process can be configured to listen for response messages on a specific port. You may also specify the port number used for communication between NetWorker and that particular instance of **ssi** using the **-P** option. The allowed values for this port number are 50004 (for the first instance), 50011 to 50019, and 50021 to 50099. Note that if you specify a port number that is already being used by an instance of **ssi,** the specified port cannot be used, and the next available port in the allowed range will be selected. If the port number is not specified, each successive instance of **ssi** will take the next available port starting from 50004 and going upwards. If there are no available ports in the range, **ssi** will fail to load and should

display an error message. Note that specifying the port number is not necessary for normal operation. You do not need to insure that a given ACSLS server is always accessed over a given port. NetWorker and **ssi** use the name of the ACSLS server to establish a connection on the fly.

If the −**A** option is not used to specify a hostname on the **ssi** command line, the environment variable **CSI_HOSTNAME** must be set to the name of the library server, before the **ssi** process is started. If this variable is not found, **ssi** will exit with an error message.

mini_el is an event logger used by **ssi** to maintain a log of certain events. It should be started before **ssi.** Multiple instances of **ssi** will share a single instance of **mini_el.** A header consisting of the ACSLS server name and the local TCP port that **ssi** will be listening on is included at the start of any message placed into the log by any instance of **ssi**

(NT only)

On NT, the software equivalent to **ssi** and **mini_el** must be obtained from StorageTek as their product "Library Attach for NT". This package must be installed prior to configuring a Silo in NetWorker.

NOTE: Library Attach version 1.1 includes a portmapper function that will only install properly if the NetWorker services are not running. You should use Control Panel to stop the "NetWorker Backup and Recover Server" and the "NetWorker Remote Exec Service" before installing Library Attach. After Library Attach is installed, you should use Control Panel to start "NetWorker Remote Exec Service" and "NetWorker Backup and Recover Server".

NOTE: Since EMC does not supply "Library Attach for NT", we are unable to add the multiple ACSLS host functionality to our NT version of NetWorker.

NOTE: The firewall enhancements added to the **ssi** and **mini_el** processes are not available on systems running Windows.

(All platforms)

libstlstk.xxx is a shared library that handles the communication between **nsrjb** and **ssi** or Library Attach. **ssi** or Library Attach then handles the communication over the network to the library server (either ACSLS, Library Station or Horizon Library Manager). There are no options, parameters or environment variables that affect the operation of **libstlstk.** The correct path to this file should be entered when an STK silo is configured using **jbconfig.** The default values specified by **jbconfig** match the default locations chosen for the installation program, and in most cases can be accepted.

OPTIONS

mini_el:

−**l** *logfile*
: Specifies the filename of the logfile to be created by **mini_el.** The default value is **/nsr/logs/ssi_event.log.** If present, *logfile* must be the complete path to the logfile. If the file does not exist, it will be created. If the file does exist, it will be appended to. If there is not a −**l** parameter, the default logfile **/nsr/logs/ssi_event.log** will be used.

−**d**
: Sets the debug flag. **mini_el** will output debug information.

−**h**
: Displays usage information for **mini_el.**

ssi:

−**A** *ACSLS server* is required if the CSI_HOSTNAME environment variable has not been set to the name of the system running ACSLS, LibraryStation or Horizon.

−**a** *ACSLS port number* is only required if you need to specify the port number used for communication between the **ssi** process and the ACSLS server. If the ACSLS server is configured to listen on a specific port, this value should be set to that port number.

−**S** *SSI port number* will force the ssi process to listen on the specified port number. This port is used in communications with the ACSLS server.

−**P** *port number* is only required if you need to specify the port to be used for communication between NetWorker and the ssi process. The allowed values for this port number are 50004 (for the first instance), 50011 to 50019, and 50021 to 50099. Note that if you specify a port number that is already being used by an instance of **ssi,** the specified port cannot be used, and the next available port in the allowed range will be selected.

−**r** *retry count* is only required if you need to increase the retry count for communication between ssi and the ACSLS server due to network problems.

These parameters are not position sensitive. The command line option will be parsed accordingly in the **ssi** process.

ENVIRONMENT VARIABLES

ssi:

CSI_HOSTNAME (text, up to 256 chars, there is no default)
If an ACSLS server name is not found on the command line, **ssi** will use the hostname specified by this variable. It is limited to 256 characters, and should simply be the hostname running the library server program that you are trying to connect to. If neither the command line hostname nor this environment variable specify a hostname for **ssi** to use, **ssi** will exit with an error message.

SSI_HOSTNAME (text, up to 256 chars, there is no default)
This variable is intended for use on multi-homed systems. Normally, **ssi** uses the **gethostbyname** system function to determine the name to use for this side of the connection to the ACSLS server. On a system with several network interfaces, the name supplied by that function may not result in the use of the network interface needed to communicate with the ACSLS server. On these systems, you can explicitly specify the exact name of the network interface that **ssi** will use to connect to the ACSLS server. This variable needs to be set before **ssi** is started, and may be different for various instances of **ssi** In all cases, a message will be logged in the event log stating if this environment variable was found, and if not, that **ssi** will be using the hostname returned by **gethostbyname.** This is not an error message.

SSI_BASE_SOCKET (numeric, 0 < x < 64k, no default)
If you need to restrict the socket values that **ssi** communicates on, this variable specifies the starting number for **ssi** to use when it needs to open a socket to talk to the ACSLS server. It appears that **ssi** will only open two sockets if this variable is set. The first, at **SSI_BASE_SOCKET,** will be used to connect to any host. The second, at **SSI_BASE_SOCKET + 1,** will be used for direct communication to the ACSLS server. Note that there will still be the default sockets at 50001 and 50004 used to communicate between **mini_el** and **ssi,** but any communication between this host and the ACSLS server should occur using the two sockets starting at **SSI_BASE_SOCKET.**

NOTE: This environment variable will be ignored if the −**a** option is used with a valid port number.

TIME_FORMAT (time format string,
default = "%m-%d-%y %H:%M:%S")
If you wish to see time values printed in a format other than the default of Month-Day-Year Hour:Minute:Seconds, use this variable.
%m is replaced by the current month
%d is replaced by today's date
%y is replaced by the current year
%H is replaced by the current hour
%M is replaced by the current minute
%S is replaced by the current second

CSI_CONNECT_AGETIME (seconds, 0 < x < 31536000,
default = 600)
This will set the number of seconds for network connect aging purposes.

CSI_RETRY_TIMEOUT (seconds, 0 < x < 4,294,967,295, default = 4)
This will set how long **ssi** will wait before retrying a network request.

CSI_RETRY_TRIES (numeric, 0 < x < 100, default = 5)
This will set the number of times **ssi** will retry sending a network message before reporting an error.

CSI_TCP_RPCSERVICE (boolean, default is TRUE)
This sets whether **ssi** will use TCP sockets to connect with the library server.

CSI_UDP_RPCSERVICE (boolean, default is FALSE)
This sets whether **ssi** will use UDP sockets to connect with the library server. Setting **CSI_UDP_RPCSERVICE** to **TRUE** will allow **ssi** to communicate with a **csi** that is running on the same system.

EXAMPLES

Normal STK silo setup:

```
mini_el &
ssi acsls1 &
```

− or −

```
mini_el &
setenv CSI_HOSTNAME acsls1
ssi &
```

Connect to 3 different ACSLS servers:

```
mini_el &
ssi -A acsls1 &
ssi -A acsls2 &
ssi -A acsls3 &
```

− or −

```
mini_el &
setenv CSI_HOSTNAME acsls1
ssi &
setenv CSI_HOSTNAME acsls2
ssi &
setenv CSI_HOSTNAME acsls3
ssi &
```

Connect to 3 different ACSLS servers over 3 different network interfaces:

```
mini_el &
setenv SSI_HOSTNAME myhost_on_net1
```

```
ssi -A acsls1 &
setenv SSI_HOSTNAME myhost_on_net2
ssi -A acsls2 &
setenv SSI_HOSTNAME myhost_on_net3
ssi -A acsls3 &
```

Connect to ACSLS server configured to accept connections on port 30031

```
mini_el &
ssi -A acsls1 -a 30031 &
```

– or –

```
setenv CSI_HOSTNAME acsls1
mini_el &
ssi -a 30031 &
```

To have

mini_el use /nsr/logs/ssi.log.today as its log file

```
mini_el -l /nsr/logs/ssi.log.today &
ssi -A acsls1 &
```

FILES

/nsr/logs/ssi_event.log
default logfile created/appended to by **mini_el**

SEE ALSO

nsrjb(1m), **jbconfig**(1m), **dasadmin**(1m), **libstlemass**(1m), **libstlibm**(1m)

DIAGNOSTICS

Several startup and shutdown messages along with any errors in communication between the NetWorker server and the ACSLS server will be logged in the logfile /nsr/logs/ssi_event.log (or other logfile as specified on the **mini_el** command line). The messages from any one **ssi** instance will be preceded by the name of the ACSLS server that that instance will be communicating with plus the local TCP port number that will be used between NetWorker and **ssi.**

For example:

```
10-12-00 12:31:44 SSI[0]:
[devlab-acsls/50004] ONC RPC: csi_init(): Initiation Started
source csi_init.c; line 165

10-12-00 12:33:20 SSI[0]:
[acsls2/50011] ONC RPC: csi_init(): Initiation Completed
ONC RPC: csi_init(): ACSLS server acsls2 accessed through port 50011
```

NAME lrescan − rescan for devices

SYNOPSIS **lrescan**

DESCRIPTION The **lrescan** program tells the underlying SCSI library to discard any cached information and scan again for new devices (see **libscsi**(1m)).

SEE ALSO **libscsi**(1m)

NAME lreset − reset SCSI bus

SYNOPSIS **lreset** *busnumber*

DESCRIPTION The **lreset** program tells the underlying SCSI library (see **libscsi**(1m)) to reset the named logical scsi bus. You must have system privileges to execute this command.

WARNINGS This command can cause the destruction of vital data since it will cause a SCSI bus reset. This may also crash your system. You should only use it as an extreme last resort to abort a hung command.

SEE ALSO **libscsi**(1m)

NAME lusbinfo − print SCSI information

SYNOPSIS **lusbinfo** [**-v**]

DESCRIPTION The **lusbinfo** program prints a limited amount of information about the SCSI busses attached to the computer.

If you use the optional **-v** argument, additional information about the devices in the attached SCSI busses will also print.

NAME lusdebug − set library debugging level

SYNOPSIS **lusdebug** *debug-level*

DESCRIPTION The **lusdebug** command sets a debugging level for the underlying NetWorker SCSI device drivers.

Debugging level 0 (zero) turns off debugging. Larger numbers enable greater levels of debugging.

The lusdebug level can now be specified as a bitmask; bit X set will show messages that are set to show at debug level X+1. For example, bit 0 set will cause messages at debug level 1 to be displayed. The exact level for any given message is listed at the end of the message in parentheses, such as (1m) for a message displayed for debug level 8.

Using the bitmask allows you to display any or all levels of debugging information. The old method only allowed you to set the highest level you wished to see; all levels lower that the selected level were always displayed, whether you wanted to see them or not.

You may still specify debugging levels the old way using by using the values old1 through old9. The results will be displayed using the new bitmask format.

Values can be entered in decimal (0 to 65535), hex (0x0 - 0xffff), or binary (0b0 - 0b1111111111111111). Zeros after the 0x or 0b prefixes are not required for binary or hex values.

Values that correspond to previous debug levels are:

old	new decimal	new hex	new binary
1	1	0x0001	0x0000000000000001
2	3	0x0003	0x0000000000000011
3	7	0x0007	0x0000000000000111
4	15	0x000f	0x0000000000001111
5	31	0x001f	0x0000000000011111
6	63	0x003f	0x0000000000111111
7	127	0x007f	0x0000000001111111
8	255	0x00ff	0x0000000011111111
9	511	0x01ff	0x0000000111111111
10	1023	0x03ff	0x0000001111111111
11	2047	0x07ff	0x0000011111111111
12	4095	0x0fff	0x0000111111111111
13	8191	0x1fff	0x0001111111111111
14	16383	0x3fff	0x0011111111111111
15	32767	0x7fff	0x0111111111111111
16	65535	0xffff	0x1111111111111111

Values corresponding to individual debug level are:

old	new decimal	new hex	new binary
1	1	0x0001	0x0000000000000001
2	2	0x0002	0x0000000000000010
3	4	0x0004	0x0000000000000100

4	8	0x0008	0x0000000000001000
5	16	0x0010	0x0000000000010000
6	32	0x0020	0x0000000000100000
7	64	0x0040	0x0000000001000000
8	128	0x0080	0x0000000010000000
9	256	0x0100	0x0000000100000000
10	512	0x0200	0x0000001000000000
11	1024	0x0400	0x0000010000000000
12	2048	0x0800	0x0000100000000000
13	4096	0x1000	0x0001000000000000
14	8192	0x2000	0x0010000000000000
15	16384	0x4000	0x0100000000000000
16	32768	0x8000	0x1000000000000000

LIMITATIONS Invalid debug levels are treated the same as debug level zero.

Debug level values greater than 65535 (0xffff, binary 0x1111111111111111) will be treated as 65535 (0ffff, binary, and so forth).

NAME lus_add_fp_devs − Solaris-only script for updating the NetWorker USCSI driver (lus) configuration file

SYNOPSIS **lus_add_fp_devs** [−**v**]

DESCRIPTION The **lus_add_fp_devs** script is a Solaris-only script that updates the EMC lus driver configuration file (/usr/kernel/drv/lus.conf) with devices connected through Sun StorEdge(TM) Fibrechannel HBAs. If Sun StorEdge(TM) Fibrechannel HBA connected devices are found, **lus_add_fp_devs** will attempt to reload the lus driver. The reload attempt will fail if the lus driver is currently in use. In this case, the user will be asked to manually reload the lus driver for the updates to take effect.

The **lus_add_fp_devs** script is dependent on the existence of the Solaris **luxadm**(1M) administrative command.

OPTIONS −**v** Verbose mode.

EXAMPLE # lus_add_fp_devs

```
        Updating /usr/kernel/drv/lus.conf

No CONNECTED Sun StorEdge(TM) HBA ports found with luxadm -e port
 This script is only used to configure lus to allow access
 to devices connected through Sun StorEdge(TM) Fibrechannel HBAs
```

FILES **/usr/kernel/drv/lus.conf**
The EMC lus driver configuration file.

SEE ALSO **luxadm**(1M)

NAME

ssi − StorageTek silo interface module (UNIX only)
mini_el − event logger for use with ssi (UNIX only)
libstlstk − shared library for communication to ssi

SYNOPSIS

ssi [−**A** *ACSLS server*] [−**a** *ACSLS port number*]
[−**S** *SSI port number*] [−**P** *port number*]
[−**r** *retry count*] &

mini_el [−**l** *logfile*] [−**d**] [−**h**] &

libstlstk.so (Solaris)
libstlstk.so.a (AIX)
libstlstk.sl (HPUX)
libstlstk.so.1 (SGI)
libstlstk.so.1 (DYNIX/ptx)
libstlstk.so (DECAXP)
libstlstk.dll (NT i386)

DESCRIPTION

NOTE: in this document, the term "ACSLS server" will be used to indicate the name of the system that is running any one of StorageTek's library manager programs: ACSLS on a Solaris or AIX host, Library Station on an MVS host, or Horizon Library Manager on a system running Windows NT or Windows 2000.

(UNIX only)

The **ssi** command is used indirectly by **nsrjb** to communicate with an ACSLS server. **nsrjb** loads **libstlstk,** which handles the TCP calls to and from **ssi. ssi** then handles all of communication to and from the ACSLS server. Starting with ACSLS version 5.3, it is possible to run NetWorker (either a server or a storage node) on the same host that ACSLS is running on.

ssi and **mini_el** must be running on the system on which jbconfig was run to create the jukebox resource. **ssi** and **mini_el** are almost always run as background processes, and are usually started automatically by the system.

In addition to **ssi** and **mini_el,** a shared library file (usually called **libstlstk.xxx** where xxx is an operating system-dependent extension) is also required. An appropriate version of this library is installed as part of NetWorker.

New in version 2.00 of **ssi:**

ssi now supports communication with the ACSLS server on a specified port number, using the −**a** command line option. This is part of the STK firewall enhancement. The ACSLS server must be running version 7.1 to use this functionality.

While you can still start **ssi** the same way as before - using the environment variable CSI_HOSTNAME to select the ACSLS server to connect to - you can also specify the ACSLS server hostname on the command line using the **-A** option. By using the −**a** option, you may specify the port number that the **ssi** process will use when connecting to the ACSLS server. The ACSLS server must be configured to listen on this port. Using the −**S** option, the **ssi** process can be configured to listen for response messages on a specific port. You may also specify the port number used for communication between NetWorker and that particular instance of **ssi** using the **-P** option. The allowed values for this port number are 50004 (for the first instance), 50011 to 50019, and 50021 to 50099. Note that if you specify a port number that is already being used by an instance of **ssi,** the specified port cannot be used, and the next available port in the allowed range will be selected. If the port number is not specified, each successive instance of **ssi** will take the next available port starting from 50004 and going upwards. If there are no available ports in the range, **ssi** will fail to load and should

display an error message. Note that specifying the port number is not necessary for normal operation. You do not need to insure that a given ACSLS server is always accessed over a given port. NetWorker and **ssi** use the name of the ACSLS server to establish a connection on the fly.

If the −**A** option is not used to specify a hostname on the **ssi** command line, the environment variable **CSI_HOSTNAME** must be set to the name of the library server, before the **ssi** process is started. If this variable is not found, **ssi** will exit with an error message.

mini_el is an event logger used by **ssi** to maintain a log of certain events. It should be started before **ssi.** Multiple instances of **ssi** will share a single instance of **mini_el.** A header consisting of the ACSLS server name and the local TCP port that **ssi** will be listening on is included at the start of any message placed into the log by any instance of **ssi**

(NT only)

On NT, the software equivalent to **ssi** and **mini_el** must be obtained from StorageTek as their product "Library Attach for NT". This package must be installed prior to configuring a Silo in NetWorker.

NOTE: Library Attach version 1.1 includes a portmapper function that will only install properly if the NetWorker services are not running. You should use Control Panel to stop the "NetWorker Backup and Recover Server" and the "NetWorker Remote Exec Service" before installing Library Attach. After Library Attach is installed, you should use Control Panel to start "NetWorker Remote Exec Service" and "NetWorker Backup and Recover Server".

NOTE: Since EMC does not supply "Library Attach for NT", we are unable to add the multiple ACSLS host functionality to our NT version of NetWorker.

NOTE: The firewall enhancements added to the **ssi** and **mini_el** processes are not available on systems running Windows.

(All platforms)

libstlstk.xxx is a shared library that handles the communication between **nsrjb** and **ssi** or Library Attach. **ssi** or Library Attach then handles the communication over the network to the library server (either ACSLS, Library Station or Horizon Library Manager). There are no options, parameters or environment variables that affect the operation of **libstlstk.** The correct path to this file should be entered when an STK silo is configured using **jbconfig.** The default values specified by **jbconfig** match the default locations chosen for the installation program, and in most cases can be accepted.

OPTIONS

mini_el:

−**l** *logfile*
: Specifies the filename of the logfile to be created by **mini_el.** The default value is **/nsr/logs/ssi_event.log.** If present, *logfile* must be the complete path to the logfile. If the file does not exist, it will be created. If the file does exist, it will be appended to. If there is not a −**l***parameter, the default logfile* **/nsr/logs/ssi_event.log** will be used.

−**d** *Sets the debug flag***.**
: **mini_el** will output debug information.

−**h**
: Displays usage information for **mini_el.**

ssi:

−**A** *ACSLS server* is required if the CSI_HOSTNAME environment variable has not been set to the name of the system running ACSLS, LibraryStation or Horizon.

−**a** *ACSLS port number* is only required if you need to specify the port number used for communication between the **ssi** process and the ACSLS server. If the ACSLS server is configured to listen on a specific port, this value should be set to that port number.

−**S** *SSI port number* will force the ssi process to listen on the specified port number. This port is used in communications with the ACSLS server.

−**P** *port number* is only required if you need to specify the port to be used for communication between NetWorker and the ssi process. The allowed values for this port number are 50004 (for the first instance), 50011 to 50019, and 50021 to 50099. Note that if you specify a port number that is already being used by an instance of **ssi,** the specified port cannot be used, and the next available port in the allowed range will be selected.

−**r** *retry count* is only required if you need to increase the retry count for communication between ssi and the ACSLS server due to network problems.

These parameters are not position sensitive. The command line option will be parsed accordingly in the **ssi** process.

ENVIRONMENT VARIABLES

ssi:

CSI_HOSTNAME (text, up to 256 chars, there is no default)
If an ACSLS server name is not found on the command line, **ssi** will use the hostname specified by this variable. It is limited to 256 characters, and should simply be the hostname running the library server program that you are trying to connect to. If neither the command line hostname nor this environment variable specify a hostname for **ssi** to use, **ssi** will exit with an error message.

SSI_HOSTNAME (text, up to 256 chars, there is no default)
This variable is intended for use on multi-homed systems. Normally, **ssi** uses the **gethostbyname** system function to determine the name to use for this side of the connection to the ACSLS server. On a system with several network interfaces, the name supplied by that function may not result in the use of the network interface needed to communicate with the ACSLS server. On these systems, you can explicitly specify the exact name of the network interface that **ssi** will use to connect to the ACSLS server. This variable needs to be set before **ssi** is started, and may be different for various instances of **ssi** In all cases, a message will be logged in the event log stating if this environment variable was found, and if not, that **ssi** will be using the hostname returned by **gethostbyname.** This is not an error message.

SSI_BASE_SOCKET (numeric, 0 < x < 64k, no default)
If you need to restrict the socket values that **ssi** communicates on, this variable specifies the starting number for **ssi** to use when it needs to open a socket to talk to the ACSLS server. It appears that **ssi** will only open two sockets if this variable is set. The first, at **SSI_BASE_SOCKET,** will be used to connect to any host. The second, at **SSI_BASE_SOCKET + 1,** will be used for direct communication to the ACSLS server. Note that there will still be the default sockets at 50001 and 50004 used to communicate between **mini_el** and **ssi,** but any communication between this host and the ACSLS server should occur using the two sockets starting at **SSI_BASE_SOCKET.**

NOTE: This environment variable will be ignored if the −**a** option is used with a valid port number.

TIME_FORMAT (time format string,
default = "%m-%d-%y %H:%M:%S")
If you wish to see time values printed in a format other than the default of Month-Day-Year Hour:Minute:Seconds, use this variable.
%m is replaced by the current month
%d is replaced by today's date
%y is replaced by the current year
%H is replaced by the current hour
%M is replaced by the current minute
%S is replaced by the current second

CSI_CONNECT_AGETIME (seconds, 0 < x < 31536000,
default = 600)
This will set the number of seconds for network connect aging purposes.

CSI_RETRY_TIMEOUT (seconds, 0 < x < 4,294,967,295, default = 4)
This will set how long **ssi** will wait before retrying a network request.

CSI_RETRY_TRIES (numeric, 0 < x < 100, default = 5)
This will set the number of times **ssi** will retry sending a network message before reporting an error.

CSI_TCP_RPCSERVICE (boolean, default is TRUE)
This sets whether **ssi** will use TCP sockets to connect with the library server.

CSI_UDP_RPCSERVICE (boolean, default is FALSE)
This sets whether **ssi** will use UDP sockets to connect with the library server. Setting **CSI_UDP_RPCSERVICE** to **TRUE** will allow **ssi** to communicate with a **csi** that is running on the same system.

EXAMPLES

Normal STK silo setup:

```
mini_el &
ssi acsls1 &
```

− or −

```
mini_el &
setenv CSI_HOSTNAME acsls1
ssi &
```

Connect to 3 different ACSLS servers:

```
mini_el &
ssi -A acsls1 &
ssi -A acsls2 &
ssi -A acsls3 &
```

− or −

```
mini_el &
setenv CSI_HOSTNAME acsls1
ssi &
setenv CSI_HOSTNAME acsls2
ssi &
setenv CSI_HOSTNAME acsls3
ssi &
```

Connect to 3 different ACSLS servers over 3 different network interfaces:

```
mini_el &
setenv SSI_HOSTNAME myhost_on_net1
```

```
ssi -A acsls1 &
setenv SSI_HOSTNAME myhost_on_net2
ssi -A acsls2 &
setenv SSI_HOSTNAME myhost_on_net3
ssi -A acsls3 &
```

Connect to ACSLS server configured to accept connections on port 30031

```
mini_el &
ssi -A acsls1 -a 30031 &
```

– or –

```
setenv CSI_HOSTNAME acsls1
mini_el &
ssi -a 30031 &
```

To have

mini_el use /nsr/logs/ssi.log.today as its log file

```
mini_el -l /nsr/logs/ssi.log.today &
ssi -A acsls1 &
```

FILES

/nsr/logs/ssi_event.log
default logfile created/appended to by **mini_el**

SEE ALSO

nsrjb(1m), **jbconfig**(1m), **dasadmin**(1m), **libstlemass**(1m), **libstlibm**(1m)

DIAGNOSTICS

Several startup and shutdown messages along with any errors in communication between the NetWorker server and the ACSLS server will be logged in the logfile /nsr/logs/ssi_event.log (or other logfile as specified on the **mini_el** command line). The messages from any one **ssi** instance will be preceded by the name of the ACSLS server that that instance will be communicating with plus the local TCP port number that will be used between NetWorker and **ssi.**

For example:

```
10-12-00 12:31:44 SSI[0]:
[devlab-acsls/50004] ONC RPC: csi_init(): Initiation Started
source csi_init.c; line 165

10-12-00 12:33:20 SSI[0]:
[acsls2/50011] ONC RPC: csi_init(): Initiation Completed
ONC RPC: csi_init(): ACSLS server acsls2 accessed through port 50011
```

NAME mminfo − NetWorker media database reporting command

SYNOPSIS **mminfo** [−**avV**] [−**o** *order*] [−**s** *server*] [−**x** *exportspec*] [**report**] [**query**] [*volname...*]

< **report** >: [−**m** | −**p** | −**B** | −**S** | −**X** | −**r** *reportspec*]
< **query** >: [−**c** *client*] [−**l**] [−**N** *name*] [−**t** *time*] [−**q** *queryspec*]

DESCRIPTION The **mminfo** command reports information about NetWorker media and save sets. The **mminfo** command can produce several different reports depending on the flags specified. Several built-in reports can be specified using shorthand flags. Custom reports can also be specified. The default report, along with the built-in reports printed by the use of the −**v**, −**V**, −**m**, −**p**, −**S**, −**B**, and −**X** flags, are described first below. The custom query and report generators, using the −**q** *queryspec* and −**r** *reportspec* options, are described in the **CUSTOM QUERIES AND REPORTS** section. Other options are described in the **OPTIONS** section.

Without any options, **mminfo** displays information about the save sets that completed properly since the previous day's midnight, and are still contained in an online file index (browsable save sets). The following information is printed for each save set: the containing volume name, the client's name, the creation date, the size saved on that volume, the save set level, and the save set name. The size field is displayed in bytes (B), kilobytes (KB), megabytes (MB), gigabytes (GB), terabytes (TB), petabytes (PB), or exabytes (EB). The save set level will display 'full', 'incr', 'migration' or 1 through 9, for full, incremental, migration save sets, level 1 through 9, respectively. The level is only kept for scheduled saves and file migration; save sets generated by explicitly running the **save**(1m) command (called *ad hoc* saves) do not have an associated level.

Specifying the −**v** flag prints aborted, purged, incomplete and recoverable save sets in addition to the complete, browsable save sets printed by default. The −**v** flag also causes three additional fields to be displayed: the creation time, the internal save set identifier (ssid), and two flags. One character is used per flag.

The first flag indicates which part of the save set is on the volume. When the save is **c**ompletely contained on the volume, a **c** is displayed. An **h** is displayed when the save set spans volumes and the **h**ead is contained on this volume. The remaining sections will be on other volumes. An **m** is displayed when the save set spans volumes and a **m**iddle section is contained on this volume. The head and tail sections will be on different volumes. There may be more than one middle section. A **t** is displayed when the **t**ail section of a spanning save set is contained on this volume. Again, the other sections will be on other volumes.

The second flag indicates the status of the save set. A **b** indicates that the save set is in the online index and is **b**rowsable via the **recover**(1m) command. An **r** indicates that the save set is not in the online index and is **r**ecoverable via the **scanner**(1m) command. An **E** indicates that the save set has been marked eligible for recycling and may be over-written at any time. An **a** indicates that the save was aborted before completion. Aborted save sets are removed from the online file index by **nsrck**(1m). An **i** indicates that the save is still in progress.

An optional third flag indicates the type of save set. An **N** indicates an NDMP save set. An **R** indicates a raw partition backup, eg., Networker Modules like Oracle, Sybase and others that Networker supports. It does not denote the save set contains files utilizing the rawasm directive. A **P** indicates a snapshot save set. A **k** indicates a checkpoint enabled save set. A combination of **ak** denotes first and all intermediate partial save sets. A combination of **bk** denotes a complete or final partial save set.

An optional fourth flag **s** indicates whether an NDMP save set was backed up via **nsrdsa_save** to a NetWorker storage node.

The −**V** flag displays even more detail than the −**v** flag. This format also displays information such as, media file number and record number that can be used to speed the operation of the **scanner**(1m) command. The −**v** flag displays one line per save set per volume. The −**V** flag displays three lines for each section of a save set occurring within a file on a volume. A single save set will have multiple index entries if it starts in one file on a volume and ends in another. This report contains all of the information reported via the −**v** flag, but, because of the additional detail, some of this information is reordered. The first line will contain the volume name, the client's name, the size saved in that section, the save set level, and the save set name. The size field lists the number of bytes that are contained in the section, rather than the total amount of the save set on this volume. The second line contains the following fields: the internal save set identifier (ssid), the save time in seconds since 00:00:00 GMT, Jan 1, 1970, the creation data and time of day, the internal save set identifier (ssid), the save set browse time, and the clone instance retention time. The third line contains: the offset of the first and last bytes of the save set contained within section, the media file number, the first record within the media file containing data for this save set, the internal volume identifier (volid), the total size of the save set, and the flags, described in the −**v** paragraph above, indicating which part of the save set is contained in this media file (**c**, **h**, **m**, or **t**) and the save set's status (**b**, **r**, **a**, or **i**).

The −**p** flag causes **mminfo** to display a report on the browse and retention times for save sets. Each line of the report displays the save set creation date, and the stored browse and retention dates ('undef' is displayed when connecting to a downrev server), the save set identifier, the client's name, and the save set's name. The −**v** and −**V** options have no effect on the columns included in this report.

The −**m** flag causes **mminfo** to display the name of each volume in the media database, the number of bytes written to it, the percent of space used (or the word 'full' indicating that the volume is filled to capacity), the retention (expiration) time, the number of bytes read, the number of times the read-label operation has been performed on the volume (not the count of explicit mounts), and the volume's capacity. Volumes that are recyclable (see **nsrim**(1m)) are flagged by an **E** in the first column (meaning **E**ligible for recycling). If a volume has been marked as manually-recyclable, an **M** is displayed instead of the **E**. If a volume is both manually-recyclable and eligible for recycling, an **X** will be displayed. Archive and migration volumes are flagged by an **A**, also in the first column. If the volume is not an archive or migration volume, and is not recyclable, no flag appears.

Specifying the −**v** flag with the −**m** flag causes three additional fields to be displayed: the internal volume identifier (volid), the number of the next file to be written, and the type of media.

Using a −**V** flag with the −**m** adds a column of flags to the output. There are currently two possible flags. The **d** flag is set if the volume is currently being written (**d**irty). The **r** flag is set if the volume is marked as **r**ead-only. If neither condition is present, the flags column will be empty.

The −**S** flag displays a long, multiline save set report, which is used for debugging. The number of lines varies per save set. Due to the length, there are no column headers. Instead, each attribute of the save set is displayed in a 'name=value' manner, except the client and save set name, which are displayed as 'client:name', and the extended attributes, described below. The first line of each multiline group starts on the left margin and includes the save set identifier (ssid), save time as both a date/time string and seconds since 00:00:00 GMT, Jan 1, 1970, and the client and save set names. Subsequent lines for this save set are indented. If the save set is part of a save set

series (a ‘continued save set’) and is not the first in the series, the save set identifier of the previous save set in the series in shown on the second line by itself. The next line displays the level, the save set flags (in ‘ssflags’ format, as described in the table in the **CUSTOM QUERIES AND REPORTS** section), the save set size in bytes, the number of files in the save set, and the save set insertion date. The next line displays the save set’s create, completion, browse and retention (expiration) dates. The string ‘undef’ for any of the values on these two lines generally means an older server that does not store these values is being queried. If the client identifier is set, it is printed on the next line. If the save set has extended attributes (such as the group to which the save set was a part or the archive annotation), they are printed next, at most one attribute per line. The format of each extended attribute is "name: values;". The clones or instances of the save set are shown last (every save set has at least once instance). The first line of each clone shown the clone identifier, the date and time the instance was created, the clone retention date, and the per-clone flags (in ‘clflags’ format from the **CUSTOM QUERIES AND REPORTS** table). For each instance, each section of that instance is shown as a fragment line. The fragment line shows the offset of that fragment from the beginning of the save set, the volume identifier (volid) containing the fragment, the media file and record numbers of start of the fragment, an absolute positioning identifier (unused by existing servers), and the date of last access of the fragment. The −**v** and −**V** options have no effect on this report. The −**o** sort order options **o** and **m** are ignored when -S is specified.

The −**X** flag prepares a save set summary report instead of one or more lines per save set. Note that the entire media database must be examined to resolve this query, making it very slow and expensive. If used in conjunction with the **a** option, the query of all volumes is done to check for save sets. If used without the **a** option, only save set information in the last 24 hours, is considered. The summary lists the total number of save sets, and breaks the total down into several overlapping categories summarizing the save set types. The recent save set usage, if appropriate to the query, is also printed. The categories are the number of fulls, the number of incrementals, the number of other non-full, non-incremental saves, the number of ad hoc, archive, migration, empty and purged save sets, the number of index save sets, and finally, the number of incomplete save sets. For recent usage, the number of save sets per day is shown, up to a week ago, along with a summary of the week’s save sets and, if applicable, a summary of the month’s save sets. For each line, the number of files (saved in the time interval specified), number of save sets, total size, average size per save set, and average size per file are listed. The percentage of the amount saved for incrementals versus fulls and the percentage of browsable files are also printed, when appropriate. The −**v** and −**V** options have no effect on the summary report.

The −**B** flag performs a canned query to output, in a convenient format, the list of bootstraps generated in the previous five weeks. In this format, there is one line of output for each matched save set. Each line contains the save date and time, save level, save set identifier (ssid), starting file number, starting record number, and the volume. The equivalent query is described below in the **EXAMPLES** section. The −**v** and −**V** options have no effect on the bootstrap display.

OPTIONS

−**a** Causes queries to apply to all complete, browsable save sets, not just those in the last 24 hours. This option is implied by the −**c**, −**N**, −**q**, −**m**, and −**o** options, described below. When combined with a media-only report (−**m** or a custom report showing only media information), −**a** applies to all volumes, not just those with complete and browsable save sets.

−**c** *client*
Restricts the reported information to the media and/or save sets pertaining to the specified *client.* This is similar to specifying a client name using the

queryspec (see -q option) name. In both cases the names are matched using a case insensitive string comparison. If the *reportspec* (see -r option) includes *volume,* the reported information will include those pertaining to the *aliases* of the client. If information relating to the aliases of the client is not required in the output, when the *reportspec* includes *volume,* the -l option needs to be used in conjunction with -c *client.*

−**l** This option when used with -c *client* along with *reportspec* (see -r option) containing *volume,* the output will not include all the information pertaining to the *aliases* of the specific client.

−**m** Displays a media report instead of the default save set report (in other words, a report about the media containing save sets, not the save sets themselves).

−**N** *name*
Restricts the reported information to the media and/or save sets pertaining to the specified save set *name*.

−**o** *order*
Sorts the output in the specified order. Before displaying the save sets, they are sorted by various fields. Numeric fields are sorted least to greatest, other fields are sorted alphabetically. *order* may be any combination of the letters **celmnotR**, representing **c**lient, **e**xpiration date, **l**ength, **m**edia name, **n**ame of save set, **o**ffset on media (file and record number), **t**ime, and **R**everse, respectively. The default sorting order for save set reports is **mocntl**. The offset fields (file and record) are only considered when the −**V** option has been selected and for custom reports that show save set section (fragment) information. When applied to −**m** media-only reports, the length is the amount used on the volume, the time is the last time the media was accessed, and the other order flags are ignored.

−**p** Displays a report on the browse and retention times for save sets, described above.

−**q** *queryspec*
Adds the given query constraint to the list of constraints on the current query. Multiple −**q** options may be given. See the **CUSTOM QUERIES AND REPORTS** section below for the syntax of the *queryspec.*

−**r** *reportspec*
Appends the given report specification to the list of attributes to be displayed for the current query. Multiple −**r** options may be given. See the **CUSTOM QUERIES AND REPORTS** section below for the syntax of the *reportspec.*

−**s** *server*
Displays volume and save set information from the NetWorker system on *server*. See **nsr**(1m) for a description of server selection. The default is the current system.

−**t** *time* Restricts the reported information to the media and/or save sets pertaining to the save sets created on or after *time*. See **nsr_getdate**(3) for a description of the recognized time formats. The default is ‘yesterday’, except when using the following switches: **-a**, **-B**, **-c**, **-N**, **-m**, **-o** and **-q**. When using those switches, there is no default value for time. If you wish to see only the backups since yesterday, you will have to specify ‘-t yesterday’ explicitly.

−**v** Turns on the verbose display reports, described above.

−**x** *exportspec*
As an alternative to the default human-readable output format, *exportspec* provides for two styles of program-readable output formats. The *exportspec* ‘m’

displays XML output, while *exportspec* 'c<separator>' displays values separated by any single character or string. For example, **'mminfo −xc,'** will produce comma-separated values.

−B Runs the canned query to report bootstraps which have been generated in the past five weeks, as described above. This option is used by **savegrp**(1m) when saving the server's index and bootstrap.

−S Displays a long, multiline save set report, as described above.

−V Displays additional verbose report output, as described above.

−X Prepares a summary report, as described above.

CUSTOM QUERIES AND REPORTS

The custom query and report options of **mminfo** allow one to generate media and save set reports matching complex constraints without resorting to pipelines and scripts. This section describes the syntax of custom query and report specifications, and gives some simple examples. Further examples are shown in the **EXAMPLES** section, below.

The custom query option, **−q** *queryspec*, is an extension to the shorthand query options, such as **−c** *client*, which allow you to make queries based on almost any media or save set attribute in the database, and allow various comparisons in addition to the simple equality comparison provided by the shorthand options. The format of a *queryspec* is

[!] *name* [*comp value*] [, ...]

where *name* is the name of a database attribute, listed in the table below, *comp* is a valid comparator for the attribute, from the set '>', '>=', '=', '<=', '<', and *value* is the value being compared. Leading and trailing spaces can be used to separate the individual components of the specification. The comparator and value must be specified for all but flag attributes. Generally numeric attributes allow all five comparators, and character string attributes generally only allow equality. When comparing flags whose values are normally 'true' and 'false', one may alternatively use the '[!] *name*' syntax. The '*!name*' form is equivalent to '*name=false*', and '*name*' by itself is equivalent to '*name=true*'. The comparisons in the specification are separated by commas. If a time or a string contains commas, you must quote the value with single or double quotes. Quotes are escaped within a string by repeating them. The following is a valid string comparison:

name="Joe's daily, ""hot"" Save Set"

Note that command line shells also interpret quotes, so you will need to enclose the entire query within quotes, and quote the single value inside the query, possibly with a different kind of quote, depending on the shell. Except for multiple character-string values, explained below, all of the specified constraints must match a given save set and/or media volume before a line will be printed in the report. Multiple **−q** options may be specified, and may be combined with the shorthand query constraints **−c**, **−N** and **−t**. The order of the above query constraints is unimportant.

Numeric constraints, except for identifiers (volume, save set and clone identifiers), allow ranges to be specified, and all character string constraints allow multiple possible values to be specified. Note that times and levels are considered to be numeric values, not character strings. The upper and lower bounds of a numeric range are specified as two separate constraints. For example,

%used>20,%used<80

matches volumes that are between 20% and 80% used. All strings are also lists except 'attributes and volume attributes'. Each possible value of a given character-string attribute is specified as a separate equality constraint. For example,

client=pegasus,client=avalon

matches save sets from the client ‘pegasus’ or the client ‘avalon’.

Example, if ‘group’ string attribute is used multiple times, the ‘mminfo’ query would be

mminfo -av -q ‘group=Default, group=Test’

This would report save sets for both ‘Default’ and ‘Test’ groups.

The custom report option, −**r** *reportspec*, allows one to specify exactly which media and save set attributes should be shown in the report, the order of the columns, the column widths, and where line breaks should be placed. The format of a *reportspec* is

name [(*width*)] [, *name* [(*width*)] ...]

where *name* is the name of a database attribute, listed below, and the optional *width*, enclosed in parentheses, specifies how wide the column should be. Leading and trailing spaces are ignored. The default column width depends on the attribute; default widths are also shown in the table below.

Some of the column headings have a short and a long (or more descriptive) version of the text. When the column width is wide enough, the long column heading is displayed; otherwise, the short heading is displayed. The column heading may not align properly with the data if the default column width is too wide or too narrow for the heading in non-US locales. To align the column heading and data, specify a width along with the attribute name.

For example, for certain locales, the column heading and data may not be aligned properly with "mminfo -p -q group=default" command. To adjust the alignment, execute the command using a column width along with the attribute name. Depending on the size of the specified column width, this may cause the long heading to be displayed. For example:

mminfo -avot -q group=Default -r"savetime(17), ssbrowse(17), ssretent(17), ssid, client, name"

Multiple −**r** options may be specified. The order of the columns in the report will be left to right, and correspond to the order of the attribute names specified. Each line of output will contain all of the data requested (you can cause line breaks within a logical line by using the *newline* attribute name). If a value does not fit in the requested column width, subsequent values in the line will be shifted to the right (values are truncated at 256 characters).

The table below lists all of the recognized attribute names, their valid range of query values (or ‘NA’ for attributes that are only valid for report specifications), their default column width in characters (or ‘NA’ for flag attributes that are only valid for query specifications), and a short description.

Numeric attributes (shown as *number* in the valid range column of the table) can be specified using any of the comparators listed above, and can be used in range comparisons.

The =*id* attributes are used for various identifiers (volume identifier, save set identifier, and so on) and only allow equality comparisons. In most cases, if the column is narrow (less that 50 characters), only the short ID is shown, which corresponds to the ID used by downrev servers. If the column is wide enough, the full ID is shown. Client identifiers always display as full IDs, and clone identifiers always display as short IDs.

Flag attributes have the values ‘true’ or ‘false’, only apply as query constraints, and have corresponding flag summary strings for report specifications.

Time attributes are specified in **nsr_getdate**(3) format and are otherwise treated as numeric attributes (note that you will need to quote times that contain commas). The special time 'forever', when used as an expiration date, means a save set or volume will never expire. The special time 'undef' is displayed when the time is undefined. When output, times are displayed according to local settings, usually as *MM/DD/YY HH:MM:SS* for numeric month, day year (last two digits), hours, minutes, and seconds, respectively for English (United States) locale. If the column is very narrow (less that 17 characters), only the date is shown. Columns 22 characters wide will generally print the full date. This is dependent on the format reported by the operating system. If the returned date and time will not fit in the specified columns, only the date is shown.

For non-US locales, *time* attributes are displayed in locale date/time format, which usually require larger column width specification. If the column width is not big enough to display the entire locale date/time value for an attribute, <locale_date HH:MM> (24-hour time) format will be attempted. If the column width is still not big enough, the date/time column will only display <locale_date>.

For example, for certain locales, to display the locale date/time for savetime attribute, specify an appropriate width, such as:

```
mminfo -avot -r"volume, client, savetime(40), sumsize, level, ssid, name,
sumflags"
```

Size and *kbsize* attributes may have a scale factor appended to them: 'KB' for kilobytes, 'MB' for megabytes, 'GB' for gigabytes, 'TB' for terabytes, 'PB' for petabytes, or 'EB' for exabytes. The default scale (when no scale is explicitly specified) on query constraints for attributes is bytes; the default for *kbsize* attributes is kilobytes. The scale varies in reports, depending on the actual value.

String attributes may be any arbitrary character string, enclosed in quotes if necessary, as described above in the query syntax paragraph.

attribute name	value range	width	description
space	NA	1	White space before the next column.
newline	NA	1	Line break(s) within a logical line. *Width* is actually the number of newlines desired.
volume	string	15	The volume name.
volid	=id	11	The unique volume identifier.
barcode	string	15	The volume barcode, when set.
family	string	4	The media family (for example, tape, disk).
type	string	7	The media type (for example, 8mm, optical).
volflags	NA	5	Volume summary flags, **d** and **r**, for dirty (in use), scan and read-only.
state	NA	7	Volume state summary, **E**, **M**, **X** and **A**, meaning eligible for recycling, manually-recyclable, both, and archive or migration volumes, respectively.
full	flag	NA	Matches full volumes.
inuse	flag	NA	Matches in-use (dirty) volumes.
volrecycle	flag	NA	Matches recyclable volumes.
readonly	flag	NA	Matches read-only volumes.
manual	flag	NA	Matches manually-recyclable volumes.
scan	flag	NA	Matches volumes that need to be scanned in.
pool	string	15	The pool containing the volume.
location	string	15	The volume's location.

capacity	size	8	The volume's estimated capacity.
written	kbsize	7	Kbytes written to volume.
%used	number or 'full'	5	Estimated percentage used, or 'full' for volumes marked as full.
read	kbsize	8	Kbytes read (recovered) from the volume.
next	number	5	Next media file for writing.
nrec	number	5	Next media record for writing.
volaccess	time	9	Last time volume was accessed, for read or write, for save or recover type of operation. A mount operation will not necessarily cause the access time to be updated. Old servers do not provide this value reliably.
volretent	time	9	The date the last save set on this volume will expire.
olabel	time	9	The first time the volume was labeled.
labeled	time	9	The most recent time the media volume was (re)labeled.
mounts	number	6	Number of times the read-label operation is performed on the volume (not the count of explicit mounts).
recycled	number	4	Number of times the volume was relabeled.
avail	NA	3	Summary of volume availability, current valid values, **n** meaning nearline (that is, in a jukebox), and **ov** meaning the volume is being managed by SmartMedia.
near	flag	NA	Matches nearline volumes.
smartmedia	flag	NA	Matches volumes managed by SmartMedia.
metric	number	6	Volume speed and desirability metric (unused by existing servers).
savesets	NA	6	Number of save sets on a volume.
volattrs	NA	31	The extended volume attributes.
name	string	31	The save set name.
savetime	time	9	The save time (on the client).
nsavetime	NA	11	The save time, printed as seconds since 00:00:00 GMT, Jan 1, 1970.
sscreate	time	9	The creation time (on the server). If the client and server clocks are out of sync, this time may be different from the save time.
ssid	=id		10 The short format of ssid is the default. It can be ambiguous.
ssid	=id		53 The long ssid format is guaranteed to be unique for a particular saveset.
snap	flag	NA	Display snapshot backups only.
cover	flag	NA	Display cover save sets and ssflags will have 'K'.
level	0..9,	5	The backup level. Manual backups

	full, incr, migration or manual		are printed as 'manual' values in reports.
client	string	11	The client resource name associated with the host that was backed up in this save set.
attrs	NA	31	The extended save set attributes.
pssid	=id	11	When part of a save set series, the previous save set identifier in the series, zero for the first or only save set in a series.
ssflags	NA	7	The save set flags summary, one or more characters in the set **CvrENiRPKIFk**, for continued, valid, purged (recoverable), eligible for recycling, NDMP generated, incomplete, raw(not for savesets backed up using rawasm), snapshot, cover, in-progress and finished (ended), checkpoint restart enabled, respectively.
continued	flag	NA	Matches continued save sets.
recoverable	flag	NA	Matches recoverable (purged) save sets.
ssrecycle	flag	NA	Matches recyclable save sets.
incomplete	flag	NA	Matches incomplete save sets.
rolledin	flag	NA	Matches rolled-in save sets.
ndmp	flag	NA	Matches NDMP save sets.
checkpoint-restart	flag	NA	Match checkpoint restart enabled savesets.
dsa		flag	NA
raw	flag	NA	Matches raw save sets, containing partitions saved by NetWorker modules.
valid	flag	NA	Matches valid save sets. All save sets are marked 'valid' by current servers.
sumflags	NA	3	Per-volume save set summary flags, as described for the **−v** report.
fragflags	NA	3	Per-section save set summary flags, as described for the **−V** report.
totalsize	number	11	The total save set size.
nfiles	number	5	The number of the client's files in the save set.
ssbrowse	time	9	The save set's browse time. This is the time limit that the save set will remain browsable. 'undef' is displayed when connected to a downrev server.
ssretent	time	9	The save set's retention time (expiration time). This is the time limit that the save set will remain recoverable in the media database.
ssinsert	time	9	The save set's insertion time. This is the time the save set was most recently introduced into the database (for example, by a backup or by running **scanner**(1m)).
sscomp	time	9	The save set's completion time. This is

			the time the save set backup was completed.
clientid	=id	9	The globally unique client identifier for the host that was backed up in this save set.
copies	number	6	The number of copies (instances or clones) of the save set, all with the same save time and save set identifier.
validcopies	number	11	The number of successful copies (instances or clones) of the save set, all with the same save time and save set identifier.
cloneid	=id	11	The clone identifier of one copy.
clonetime	time	9	The time a copy was made.
clretent	time	9	The clone retention time is the time limit that the clone instance will remain recoverable in the media database.
clflags	NA	5	The clone flags summary, one or more characters from the set **aios** for aborted, incomplete, opaque (NDMP clone of regular NetWorker savesets), suspect (read error), respectively. This summary reflects the status an instance of a save set.
suspect	flag	NA	Matches suspect save set copies, copies that had errors during file recovery.
annotation	string	31	The (archive) save set's annotation. In a queryspec, the string is a regular expression in the form used by **grep**(1).
group	string	12	The group of this save set. This is the group that backed up this save set.
ssbundle	string	15	The save set bundle of this save set. This is used to stage several save sets together.
first	number	11	The offset of the first byte of the save set contained within the section.
last	NA	11	The calculated offset of the last byte of the save set contained within the current section.
fragsize	NA	7	The calculated size of the current section of the save set.
sumsize	NA	7	The calculated total size of all of the sections of the save set on this volume.
mediafile	number	5	The media file number containing the current section of the save set.
mediarec	number	5	The media record number where the first bytes of the save set are found within the current media file.
mediamark	number	5	The absolute positioning data for the current section (not used by existing servers).
checkpoint_id	string	10	Checkpoint id of the save set.
checkpoint_seq	string	10	Checkpoint sequence number of the save set.

EXAMPLES In the following examples, the equivalent shorthand and custom versions of the report are shown, when a shorthand option exists for a given report or query.

Display the information of savesets on a volume, with saveset-id in long ssid format(53 characters).

mminfo −av −r 'volume, name, savetime, ssflags, clflags, ssid(53)'

Display all bootstraps generated in the previous five weeks, as reported by **savegrp**(1m)**:**

mminfo −B
mminfo −N bootstrap −t '5 weeks ago' −avot
-r 'savetime(17),space,level(6),ssid'
-r 'mediafile(6),mediarec(7),space(3),volume'

Display information about all of the volumes:

mminfo −m
mminfo −a −r 'state,volume,written,%used,volretent,read,space'
-r 'mounts(5),space(2),capacity'

Display media information from volumes **mars.001** and **mars.002**:

mminfo −m mars.001 mars.002
mminfo −m -q 'volume=mars.001,volume=mars.002'

Display all browsable save sets named **/usr**:

mminfo −N /usr
mminfo −q name=/usr

Display browsable save sets named **/usr**, generated by client **venus**, in the past week:

mminfo −N /usr −c venus
mminfo −q 'name=/usr,client=venus'

Display browsable save sets named **/usr**, generated by client **venus**, on volume **mars.001**:

mminfo −N /usr −c venus mars.001
mminfo −q 'name=/usr,client=venus,volume=mars.001'

Display a media report of all volumes written on in the past week:

mminfo −m -t 'last week'
mminfo −m -q 'savetime>=last week'

Display a media report of all non-full volumes, showing the percent-used, pool and location of each volume:

mminfo −a −r 'volume,%used,pool,location' -q '!full'

Display a media report similar to the **−m** report but showing the barcode instead of the volume label:

mminfo −a −r 'state,barcode,written,%used,read,space'
-r 'mounts(5),space(2),capacity'

Display a verbose list of the instances of all save sets with more than one copy, sorted by save time and client name:

mminfo −otc −v −q 'copies>1'

Query that should be used to check whether all completed save sets on a volume (for example,nmsun118.001) have at least one successful clone on other volumes:

mminfo -q 'volume=nmsun118.001,validcopies>1'

Display all archive save sets with an annotation of "project data" for the past four months.

mminfo −q'annotation=project data'
-r"volume,client,savetime,sumsize,ssid,name,annotation"
-t'four months ago'

Display all snapshot save sets for the client **cyborg**.

mminfo −q'client=cyborg, snap'
-r"volume,client,savetime,sumsize,ssid,name,annotation"

-t'four months ago'

NOTE: This option is available with EMC PowerSnap Module only

Display all snapshot save sets with their snapshot handle, for the client **cyborg**. The snapshot handle is stored in the attribute ´*__snapid__´.

mminfo −a −S −q'client=cyborg, snap'
-t'four months ago'

NOTE: This option is available with EMC PowerSnap Module only

Display all checkpoint enabled savesets for the client **cyborg**.

mminfo −q'client=cyborg, checkpoint-restart'
-r"checkpoint_id,checkpoint_seq,volume,client,ssid,name"

Display the third partial saveset from the checkpoint restart sequence '1265299738'.

mminfo −q'checkpoint_id=1265299738,checkpoint_seq=3'
-r"checkpoint_id,checkpoint_seq,volume,client,ssid,name"

PRIVILEGE REQUIREMENTS

A User with "Recover Local Data" privilege is allowed to query the media database for save set information only for the client where mminfo command is invoked.

A User with "Remote Access" privilege is allowed to query the media database for save set information for any client.

A User with "Operate Devices and Jukeboxes" privilege is allowed to query the media database for detailed volume information. The user is still required to have either "Recover Local Data" or "Remote Access" privilege to be able to access save set information. The "Remote Access" privilege can be granted either through "the "Remote access all clients" privilege or through the "Remote access" attribute in client resource.

A user with "Monitor Networker" privilege can query the media database for volume and save set information for any client. This is equivalent to having both "Operate Devices and Jukeboxes" and "Remote Access" privileges.

FILES

/nsr/mm/mmvolume6
The save set and media volume databases (actually accessed by **nsrmmdbd**(1m)).

SEE ALSO

grep(1), **nsr_getdate**(3), **nsr_layout**(5), **nsradmin**(1m), **nsrmmdbd**(1m), **recover**(1m), **savegrp**(1m), **scanner**(1m).

DIAGNOSTICS

no matches found for the query
No save sets or volumes were found in the database that matched all of the constraints of the query.

invalid volume name '*volname*'
The volume name given is not in a valid format. Note that volume names may not begin with a dash. Queries that match no volumes will return the error 'no matches found for the query'.

only one of −m, −B, −S, −X or −r may be specified
Only one report can be generated at a time. Use separate runs of **mminfo** to obtain multiple reports.

invalid sorting order specifier, choose from '*celmnotR*'
Only letters from *celmnotR* may be used with the **−o** option.

only one −o allowed
Only one sorting order may be specified.

only one −s allowed
Only one server can be queried at one time. Use multiple runs of **mminfo** to obtain reports from multiple servers.

Out of Memory
The query exhausted available memory. Try issuing it again, using the sorting order −**om**, or make the query more restrictive (for example, list specific volumes, clients, and/or save set names).

invalid value specified for '*attribute***'**
The value specified is either out of range (for example, a negative number for a value that can only take positive numbers), the wrong type (an alphabetic string value specified for a numeric attribute), or just poorly formatted (for example, non-blank characters between a close quote and the next comma or a missing close quote).

value of '*attribute***' is too long**
The value specified for *attribute* is longer than the maximum accepted value. Query attributes must have values less than 65 characters long.

non-overlapping range specified for '*attribute***'**
The range specified for *attribute* is a non-overlapping numeric range, and cannot possibly match any save set or volume in the database.

unknown query constraint: *attribute*
The given query *attribute* is not valid. See the **CUSTOM QUERIES AND REPORTS** table for a list of all valid attribute names.

need a value for query constraint '*attribute***'**
The *attribute* is not a flag, and must be specified in the '*name comparator value*' format.

constraint '*attribute***' is only valid for reports**
The *attribute* specified for a query may only by used in report (−**r**) specifications. Calculated values, flag summaries, save set extended attributes, and formatting tools (**space** and **newline**) may not be used in queries.

invalid comparator for query constraint '*attribute***'**
The comparator used is not valid for the given *attribute*. See the **CUSTOM QUERIES AND REPORTS** section for a list of the valid comparators for *attribute*.

query constraint '*attribute***' specified more than once**
The given attribute was specified more than once with the same comparator, and is not a string attribute (string attributes can match one of several specific values).

unknown report constraint: *attribute*
The given report *attribute* is not valid; see the **CUSTOM QUERIES AND REPORTS** table for a list of all valid attribute names.

constraint '*attribute***' is only valid for queries**
The *attribute* specified for a report is a flag matching attribute and may only be used in query (−**q**) specifications. See the **CUSTOM QUERIES AND REPORTS** table for the appropriate flag summary attribute that one may use in reports of a given flag.

column width of '*attribute***' is invalid**
The width specified for *attribute* is out of range. Column widths must be positive numbers less than 256.

missing close parenthesis after report constraint '*attribute***'**
The width of *attribute* is missing a close parenthesis.

missing comma after report constraint '*attribute***'**
There are non-blank characters after the width specification for *attribute*

without any comma preceding them.

No data requested, no report generated
The given report specification contains only formatting, no data attribute names.

LIMITATIONS

You cannot specify save set extended attributes as query constraints.

You cannot list several possible equality matches for numbers, only for strings.

Some queries, namely those that are not highly selective (few query constraints) and use a sorting order where the volume name is not the primary sort key, still require **mminfo** to retrieve the entire database before printing any of it. Such queries use large amounts of memory in **mminfo**, but not, as was the case with older versions, in **nsrmmdbd**.

You cannot make a report that shows save set or media instances and a summary without running **mminfo** at least twice.

You cannot specify query constraints that compare database attributes with each other.

You cannot make a report that uses **-B** flag with **-c** flag.

NAME mmlocate − NetWorker media location reporting command

SYNOPSIS **mmlocate** [−**s** *server*] [−**l**
{ −**n** *volname* | −**i** *volid* | *location* }] [−**L**] [−**d** *location*]
[−**c** { −**n** *volname* | −**i** *volid* }] [−**u** { −**n** *volname*| −**i** *volid*| **location** }]

DESCRIPTION The **mmlocate** command is used to access and manage the volume location information contained in the media database. The information contained in a volume's location field is meant to give the user an idea of where the volume can physically be found. Other NetWorker commands will display the location along with the volume name (see the *versions* sub-command of **recover**(1m)). Any user can use this command with the **-l** (default) or **-L** options. The **-c**, **-d** and **-u** options are limited to NetWorker administrators (see **nsr**(1m)). -l is assumed by mmlocate if a -L, -c, -d or -u option is not specified.

Running **mmlocate** without any arguments lists all volumes and their locations for the specified server (if you do not specify a server, the current host is used).

Note that each time *nsrjb*(1m) moves a piece of media inside a jukebox, the location of a volume is set to the name of the jukebox. When using storage nodes, the name of the jukebox is used to indicate on which node the volume can be mounted. Hence, the first portion of this field containing the jukebox name should not be changed. When using volumes on a storage node that are not contained within a jukebox, this field can be used to indicate on which node a volume should be mounted, by giving it a value of any remote device on that node. See **nsr_storage_node**(5) for additional detail on storage nodes.

OPTIONS

−**c** Clears the location field for the specified volume.

−**d** *location*
Deletes all volumes associated with the specified location. A confirmation prompt appears prior to the deletion of each volume.

−**i** *volid*
Restricts the **mmlocate** operation to the specified volume ID (volid).

−**l** Lists entries. Performs a database query using the supplied *volume name*, *volume ID* or *location*.

If a volume name or volume id is given, then only the volume's location information is displayed. If a location is provided, then only volumes in that location are displayed. When the -l option is used without specifying any other options (volume name, volume id, or location), volumes without a set location are displayed.

−**L** Lists all *locations* found in the database.

−**n** *volname*
Restricts the operation to the volume name (*volname*) listed.

−**s** *server*
Accesses the *server*'s media database.

−**u** Updates the location for a volume. Locations are limited to a maximum length of 64 characters. The **-n** *volname* or **-i** *volid* and *location* options must be used in conjunction with the **-u** option.

EXAMPLES Update the media database to show that volume Offsite.011 is now at location 'Media Vault'

mmlocate −u −n Offsite.011 'Media Vault'

Delete volumes at location 'Media Shelf 6'
mmlocate −d 'Media Shelf 6'

Delete location information for volume NonFull.001
mmlocate −c −n NonFull.001

List the location of volume NonFull.001
mmlocate −n NonFull.001

List all volumes stored in the location 'Media Vault'
mmlocate 'Media Vault'

FILES **/nsr/mm/mmvolume** The media database.

SEE ALSO **nsrmm**(1m), **mminfo**(1m), **nsr**(1m), **nsrjb**(1m), **recover**(1m) **nsr_storage_node**(5)

DIAGNOSTICS *Server* **does not support remote update operations.**
If you are running **mmlocate** against an old server, you are not allowed to use the -u or -c options. You must login to that server and run the mmlocate program there.

NAME mmpool − NetWorker media pool reporting command

SYNOPSIS **mmpool** [−**s** *server*] [*volume...*]

[−**d** *pool*] [−**l** *pool*] [−**L**]

DESCRIPTION The **mmpool** command is used to access pool information stored in the NetWorker server's media database. This command can also be used to delete all the volumes in a particular pool. If you specify one or more volume names with the **mmpool** command, the report shows the pool that each named volume belongs to. By default, all volumes and their pools are displayed.

You cannot change the pool to which a volume belongs without relabeling the volume, which destroys all data stored on the volume. Pools are configured through a NetWorker administration tool, such as **NetWorker Management Console** or **nsradmin**(1m). These tools are used to create and modify unique *pool* (see **nsr_pool**(5)) resources.

OPTIONS −**d** *pool* Deletes all volumes for the given pool. The user will be prompted for deletion of each volume.

−**l** *pool* Lists all volumes and the pools to which they belong. If a pool is specified, **mmpool** only lists the volumes belonging to that pool.

−**L** Lists the names of all pool resources configured on the server.

−**s** *server*
Specifies the NetWorker server to act upon. See **nsr**(1m) for a description of server selection.

FILES **/nsr/mm/mmvolume (UNIX)**
The media database on the server.

SEE ALSO **nsr**(1m), **nsr_device**(5), **nsr_pool**(5), **nsradmin**(1m), **nsrjb**(1m), **nsrmm**(1m)

NAME mmrecov − recover a NetWorker media index

SYNOPSIS **mmrecov [−q | −v] [−N [−F]]**

DESCRIPTION The **mmrecov** command is used in recovering from the loss of a NetWorker server's critical files. **mmrecov** restores the media index and the server's resource files. Typical events causing such disasters are accidental removal of these files by a user or a disk crash on the NetWorker server itself. See **nsr_crash**(1m) for a discussion of general issues and procedures for NetWorker client and server crash recovery if you are running NetWorker for UNIX.

mmrecov is used to recover the NetWorker server's media database and resource files from the media (backup tapes or disks) when the media database or resource files have been lost or damaged. Note that this command *overwrites* the server's existing media index. The **mmrecov** command is not used to recover NetWorker clients' online indexes; you must use the **nsrck**(1m) command for this purpose.

The NetWorker system must be fully installed and correctly configured prior to using this command. If any of the NetWorker software is lost, re-install NetWorker from the distribution files before you run **mmrecov**. Use the same release of NetWorker, and install it in the same location as it was before the software was lost.

The **mmrecov** program extracts the contents of a *bootstrap* save set, which contains the media index and resource files. Once **mmrecov** is done running, you shut the NetWorker server down, move the recovered resource files into place, and restart the server. At this point, the file indexes for the server and client may be restored by using **nsrck.**

When **mmrecov** is started, it will ask for the device from which the bootstrap save set will be extracted. Next, it will ask for the bootstrap save set identifier. This number is found in the fourth column (labeled *ssid*) of the last line of the bootstrap information sheet printed by **savegrp** and **mminfo -B**, an example of which is shown below:

Jun 17 22:21 1992 mars's NetWorker bootstrap information Page 1

date	time	level	ssid	file	record	volume
6/14/92	23:46:13	full	17826163	48	0	mars.1
6/15/92	22:45:15	9	17836325	87	0	mars.2
6/16/92	22:50:34	9	17846505	134	0	mars.2 mars.3
6/17/92	22:20:25	9	17851237	52	0	mars.3

In the example above, the *ssid* of the most recent bootstrap save set is '**17851237**'. If you are cloning save sets, your bootstrap save set is also cloned, and you need to use the *second* to last save set. See the **RECOVERING FROM CLONE MEDIA** section for an example of boostrap information with cloned save sets.

Next, **mmrecov** prompts for the file and record location of the bootstrap save set. Both values may default to zero if they are not known. Note, however, that specifying the correct file and record numbers will allow NetWorker to more quickly locate the bootstrap save set. The file and record locations are the fifth and sixth columns of the bootstrap information sheet. In the example above, the values for the file and record locations are 52 and 0, respectively. Finally, **mmrecov** will prompt that the volume ('**mars.3**' in the example above) containing the selected bootstrap save set be inserted into the specified device. The ssid, file location, record location, and the physical volume must be determined by the user from the printed sheet, since **mmrecov** has no way of determining this information. On the other hand, if the volume containing the

bootstrap is not known, the **-B** option of **scanner**(1m) can be used to determine the file and record locations.

If the bootstrap save set spans more than one volume, multiple volume names are printed. The order printed is the order required by **mmrecov**. In the example above, the third save set produced on 6/16/92 begins on volume '**mars.2**' and spans to volume '**mars.3**'. If a bootstrap save set spans volumes, **mmrecov** will prompt for the name of the device where the next volume has been loaded when an end-of-volume occurs. The volume is then scanned, and the bootstrap save set extracted.

After the volume scan completes, **mmrecov** will complete. At this point, if your original server resource files were lost, you must shut down the NetWorker server, move the new resource files into place, and restart the NetWorker server. Now the indexes can be recovered.

At the end of a bootstrap recovery, the contents of the media database will be replaced by the data from the bootstrap saveset. If any volumes in the recovered database were written to after the bootstrap saveset was created, those savesets will not be in the media database and the volume records will have incorrect tape position. NetWorker will use the incorrect tape position to write new data, thereby overwriting data written since the bootstrap backup. In the case of file type or advanced file type devices, recover space operation will delete all the savesets which have been created since the time the bootstrap saveset was created.

Consider the following situation. The NetWorker databases needed to be recovered today and the latest bootstrap saveset available is one day old. After recovering the databases using this bootstrap saveset, the media database will contain volumes and saveset records as of yesterday. Any savesets created after the time the bootstrap saveset was created do not exist in the media database, but they do exist in the physical medium. NetWorker will use the stale position information from volume records for writing new data. This has the potential to overwrite the saveset records that are missing from the media database.

To prevent this from happening, mark all volumes with the flag "scan" using the option -N. This flag will instruct nsrmmd to find the real end of tape. This will prevent data loss. To recover those savesets created since the boostrap recovery, the administrator must scan in the volumes. For file type or advanced file type devices, the recover space operation will be suspended until the "scan" flag is turned off.

In order to recover the indexes for the server and client, you must run **nsrck -L7**. This command will reconstruct complete indexes from the save sets generated by the server's save schedule. Since the save sets may be spread across multiple volumes, **NetWorker Management Console** or **nsrwatch**(1m) should be run, and the volumes mounted as they are requested.

When **nsrck** completes, the message "**completed recovery of index for client '**<**client-name**>**'**" is displayed. Once a NetWorker client's index is recovered, that client can start recovering its files using **recover**. Note that it is not necessary for the server's index to be restored before the client indexes may be restored.

As stated earlier, the NetWorker resource files are saved as part of the bootstrap save set. If your resource files were also deleted, you may quickly replace them by copying or moving them from /nsr/res.R to /nsr/res. Before restoring them to /nsr/res, the daemons must be shut down (see **nsr_shutdown**(1m)).

Sometimes it is neccessary to recover the NetWorker server onto a new machine, for example, after a major hardware failure. When this occurs, the NetWorker Licensing software will detect the move. Once the NetWorker server has been moved to a new machine, it must be re-registered with Customer Support within 15 days of the move, or the server will disable itself. After disabling itself, you will only be able to recover

files; new backups cannot be performed until the server is re-registered. Notifications will be sent by the NSR Registration notification, warning of the need to re-register the product.

RECOVERING FROM CLONE MEDIA

If you are running **mmrecov** with clone media only, for example, at a remote site, you will need to perform the recovery using a slightly different method. When selecting the bootstrap identifier, make sure that you are using the information associated with the cloned save set: the last save set listed in the bootstrap output. Consider the following list of save sets:

Jun 17 22:21 1996 mars's NetWorker bootstrap information Page 1

date	time	level	ssid	file	record	volume
6/14/96	23:46:13	full	17826163	48	0	mars.1
6/14/96	23:46:13	full	17826163	12	0	mars_c.1
6/15/96	22:45:15	9	17836325	87	0	mars.2
6/15/96	22:45:15	9	17836325	24	0	mars_c.2
6/17/96	22:20:25	9	17851237	52	0	mars.3
6/17/96	22:20:25	9	17851237	6	0	mars_c.3

In the example above, the ssid of the most recent bootstrap save set is '**17851237**'. The cloned save set resides on *mars_c.3* and the values for the file and record locations are 6 and 0, respectively.

If you lost your resource files and need to use the ones restored from **mmrecov**, the NetWorker server needs to be shut down so that you can replace the installation resource files with your recovered ones.

Once the original resource files are in place, the NetWorker server should be restarted. After it is restarted, you may recover the indexes for the server and clients by issuing the **nsrck -L7** command. This command queries the media database for the index backups and restores the indexes for the server and each client. If all clone volumes needed are online when the index recovery proceeds, **nsrck** will complete on its own.

If some of the volumes are not online, then **nsrck** will attempt to recover the index from the original volume it was backed up to, and therefore request the original media. In the example bootstrap output above, *mars_c.1* and *mars_c.3* would both need to be online. If volume *mars_c.3* was the only volume online, then **nsrck** would also request *mars.1*. To finish recovering the server's index in this case, you need to perform the following steps:

1. Note what volumes are needed for recovery and delete them from the media database. **NetWorker Management Console** or **nsrwatch**(1m) lists the volumes needed for recovery in the *Pending* messages panel. Use **NetWorker Management Console** or **nsrmm**(1m) to delete the volumes from the media database.

   Given the scenario in the example above where only *mars_c.3* was mounted, we would have to delete *mars.1* from the media database, for example, **nsrmm** *-d mars.1*.

2. Restart the server to terminate the index recovery in progress.

   Use **nsr_shutdown**(1m) to bring the server down. Run **nsrd**(1m) to start the server again.

3. Recover the server's index by using **nsrck** *-L7 servername*.

When **nsrck** completes, the message "**The index is now fully recovered**" appears.

OPTIONS

−q Quiet. Displays only error messages.

−v Verbose. Generates debugging information.

−N Needs to scan in the volumes. After recovering media database, mark all volumes in the media database to indicate that they need to be scanned in.

−F Set "scan" flag to file and advanced file type volumes only. This option must be used with −N option. When specified, the "scan" flag will not set for volumes that are on tape media.

FILES

/nsr If this was a symbolic link when the bootstrap save set was created, it needs to be re-created manually prior to running **mmrecov**.

/nsr/res
This directory and its contents are saved as part of the bootstrap save set. **mmrecov** restores this directory, and then renames it to **/nsr/res.R**. The original directory is temporarily renamed to **/nsr/res.org** while the bootstrap save set is being recovered.

/nsr/mm/mmvolume
The NetWorker server's media index saved as part of the bootstrap save set, and unconditionally recovered by **mmrecov**.

DIAGNOSTICS

Failed to set 'scan' flag for the volume *volumename*
An error message to indicate that the "scan" flag was not set for the specified volume. The actual error message will follow this message.

BUGS

The name **mmrecov** is misleading; as a result, **mmrecov** is often used when it is not needed. A name like "recover_server_media_database_or_resource_files_when_missing" is more descriptive. Note that any part of the bootstrap save set contents are recoverable using normal recover procedures provided that the server's on-line index, resource files, and media index are intact.

To recover files that are not in the on-line file index (for example, files saved after the last run of **savegrp**), **scanner** must be used to rebuild the media and on-line file indexes from the contents of the volumes generated between the time of the last run of **savegrp** and the loss of the original index.

SEE ALSO

mminfo(1m), **nsr_crash**(1m), **nsr**(1m), **nsrck**(1m), **nsrd**(1m), **nsr_client**(5), **nsr_schedule**(5), **nsr_shutdown**(1m), **recover**(1m), **save**(1m), **savefs**(1m), **savegrp**(1m), **scanner**(1m), **nsrindexasm**(1m), **nsrmm**(1m), **nsrmmdbdasm**(1m), **nsrwatch**(1m), **nsr_getdate**(3)

NAME mm_data − NetWorker media multiplexor data (tape and disk) format

DESCRIPTION This documents the data format that the NetWorker media multiplexor daemon, **nsrmmd**(8), writes to long term storage media such as tapes and optical disks. See **nsr_device**(5) and **nsrmm**(8) for a discussion of supported device families and types. The format described here applies to any fixed record device, such as raw disks, or fixed record tape devices with file marks. NetWorker uses the *eXternal Data Representation* (XDR) standard to write media which can be interchanged among a wide variety of machines. Only the mechanism used to multiplex save set streams onto the storage media is described here; the formats of save set streams depend on the type of NetWorker client, and are described in **nsr_data**(5).

A *volume* is one physical piece of media such as a tape reel or disk cartridge. A tape volume is made up of multiple media *file*s, and each media file may contain several media *record*s. These media files and records should not be confused with a client's (for example UNIX or DOS) user files or records; the two do not necessarily correspond. For example, a given media file or even a single media record may contain many small client user files. On the other hand, a single large client file may be split across several media files, and even across several volumes. Media files do not span volume boundaries. Save sets may span media files and even volumes.

On most tapes, media files can be skipped very quickly by the device's hardware or associated device driver software, and the hardware can detect when an end of a file has been reached. On some tapes, records can also be quickly skipped forward. Otherwise, access to the media is sequential.

Media records are described by the **mrecord** structure. Label records are fixed in size, **MINMRECSIZE** bytes. Other records can potentially be a larger size that must be some constant for the rest of the volume. NetWorker always writes, reads and skips data in units of full-sized media records. Each **mrecord** contains zero or more **mchunks** These **mrecords** are used for storing one or more client save sessions or used by NetWorker for synchronization and labelling. The XDR format of a media file's **mrecord**s and **mchunk**s are as follows:

```
const MINMRECSIZE = 32768;      /* minimum media record size */
const MMAXCHK = 2048;           /* maximum number of chunks in record */
const MHNDLEN = 120;            /* private area length for handlers */

enum mrec_version {             /* mrecord version */
     MREC_VER5 = 0,             /* older format mrecord */
     MREC_VER6 = 6              /* current format mrecord */
};

/*
 * For media record format version 5, the data types lgui_t, lg_off64_t,
 * and lg_time64_t are defined as:
 */
typedef struct lgui_t unsigned long;
typedef struct lg_off64_t unsigned long;
typedef struct lg_time64_t unsigned long;

/*
 * For media record format version 6, the data types lgui_t, lg_off64_t,
 * and lg_time64_t are defined as:
 */
```

```
typedef struct lgui_t {             /* XDR encoded Unique Id. */
    char _bytes(20);
} lgui_t;
typedef struct lg_off64_t unsigned long long;
typedef struct lg_time64_t unsigned long long;


typedef lgui_t ssid_t;              /* save set id */
typedef lgui_t volid_t;             /* key for the volume database */

struct  mchunk {
    ssid_t mc_ssid;                 /* owning save set id */
    lg_off64_t mc_low;              /* 1st byte, relative to save stream */
    opaque mc_data<MINMRECSIZE>;/* chunk's data */
};

struct  mrecord {
    opaque mr_handler(MHNDLEN);/* private to media handler */
    mrec_version mr_version;    /* Media record version number */
    u_long mr_orec;             /* record size */
    volid_t mr_volid;           /* encompassing volume's id */
    u_long mr_fn;               /* encompassing file number */
    u_long mr_rn;               /* record number within the file */
    u_long mr_len;              /* record byte length */
    mchunk mr_chunk<MMAXCHK>;/* chunks of save streams */
};
```

The first field of an **mrecord**, **mr_handler**, is reserved for media-specific data (currently it is not used by any implementation). The **mr_version** field is the version number of the media record format. The size of the rest of the fields in the media record depends on the version number. The **mr_orec** field is the size of the current record. A media record's header fields, **mr_volid**, **mr_fn**, and **mr_rn**, are used to check the tape position and the data read from the record. The file numbers and record numbers start at zero and increment sequentially. The record number is reset each time the file number is incremented. On disks, file numbers are always zero. The **mr_len** field is the actual number of valid bytes in this record, as opposed to the size of the device's read or write request.

If file or record skipping is unreliable, NetWorker can still recover from isolated errors, at worst by rewinding and reading the tape from the start. If a volume can be physically unmounted or mounted without notice to the media management daemon, then the volume identifier in each record provides a quick way of verifying when this happens, without the need for a full rewind and reading of the label in most cases.

The **mchunk**s within an **mrecord** contain client data from one or more save sessions. The **mc_ssid** and **mc_low** values are used to reconstruct the save streams from the chunks within the records. The **mc_data** field holds the actual data of each chunk. For a given save set, **mc_low** plus the length of **mc_data** should equal the following chunk's value for **mc_low.** Save sets may by intermingled arbitrarily within media records.

The first chunk of the first record of the first media file on the volume encapsulates the **volume label** information; for some media, the second chunk contains additional volume information, for example, the media pool the volume belongs to: Subsequent data in the first file is reserved for future expansion. The label may be duplicated in a second file for redundancy, in case the first copy of the label gets accidentally

overwritten. The formats of the volume label and additional label information are described by the following XDR data structures:

```
const MVOLMAGIC = 0x070460;      /* volume magic number */
const NSR_LENGTH = 64;           /* length of several strings */
const RAP_MAXNAMELEN = 64;       /* maximum length of attribute name */

struct mvollabel {
      u_long  mvl_magic;              /* medium volume verification number */
      lg_time64_t mvl_createtime; /* time at which volume labeled */
      lg_time64_t mvl_expiretime; /* time for volume to expire */
      u_long  mvl_recsize;            /* expected size of mrecords */
      volid_t mvl_volid;              /* medium volume id */
      string  mvl_volname<NSR_LENGTH>;/* medium volume name */
};

struct vallist {
      vallist *next;
      string value<>;                 /* attribute value */
};

struct attrlist {
      attrlist *next;
      string name<RAP_MAXNAMELEN>;/* attribute name */
      vallist *values;                /* attribute values */
};

/*
 * Additional information may includes the following attributes
 * (listed by the name they are stored with):
 * "volume pool" : the media pool
 */
struct mvolinfo {
      struct attrlist *mvi_attributes;      /* any other information */
};
```

The **mvl_magic** field must be equal to **MVOLMAGIC** in order for the chunk to represent a valid volume label. If the volume label changes in the future, the new format will have another ''magic'' number, but the format described here must still be allowed. The **mvl_volid** is an internal identifier assigned and managed by the media manager. The **mvl_volname** is the volume name that is assigned when the media is first labeled. The time fields are in UST format − the number of seconds elapsed since 00:00 GMT, January 1, 1970. The **mvl_recsize** is the size of all subsequent media records found on the tape.

The **mvp_pool** is the pool name that is assigned when the media is first labeled. Different media pools allow administrators to segregate their data onto sets of volumes. Media cannot be reassigned from one media pool to another. Pool names are a maximum of **NSR_LENGTH** characters long.

Synchronization marks, called **schunks**, are also written periodically to the media for each save set. Synchronization chunks are used by **scanner**(8) when verifying or extracting directly from a volume. They are also used by **nsrmmd** when trying to recover from media errors during file recovery. The following XDR data structure describes a synchronization chunk:

```
typedef lgui_t clientid_t;

struct                                  ssclone_t {
        lg_time64_t ssc_cloneid;        /* unique UST stamp wrt ss_ssid */
        u_long ssc_flag;                /* lots of status buried here*/
        u_long ssc_frag;                /* not used, always 0 */
};

/*
 * Synchronization chunk of the newer MREC_VER6 media format.
 */
struct schunk {
        u_long ssi_gen;                 /* Not used. */
        ssid_t ssi_ssid;                /* save set identifier */
        ssid_t ssi_prev;                /* non-zero iff continuation */
        u_long ssi_level;               /* backup level*/
        lg_time64_t ssi_time;           /* save time on client */
        lg_time64_t ssi_create;         /* creation time on server */
        lg_time64_t ssi_insert;         /* insertion time on server */
        lg_time64_t ssi_complete;       /* completion time on server */
        clientid_t ssi_clientid;        /* client name identifier */
        u_long ssi_flags;               /* more details about this ss */
        string ssi_host<>;              /* client name - save set owner */
        string ssi_name<>;              /* symbolic name, for example "/usr" */
        uint64_t ssi_size;              /* actual number of bytes saved */
        uint64_t ssi_nfiles;            /* number of client files saved */
        u_long ssi_browse;              /* browse time offset */
        u_long ssi_recycle;             /* recycle time offset */
        struct attrlist *ssi_al;        /* generic RAP attribute list */
        ssclone_t ssi_clones<>;         /* information about this clone */
};

/*
 * Synchronization chunk of the older MREC_VER5 media format.
 */
struct old_schunk {
        opaque ssi_host[NSR_LENGTH];    /* save set host */
        opaque ssi_name[NSR_LENGTH];    /* symbolic name */
        u_long ssi_time;                /* save time */
        u_long ssi_expiry;              /* expiration date */
        u_long ssi_size;                /* actual size saved */
        u_long ssi_nfiles;              /* number of files */
        ssid_t ssi_ssid;                /* ssid for this save set */
        u_long ssi_flag;                /* various flags, see below */
        u_long ssi_info;                /* volid or ssid, see below */
};

#define SSI_START       1               /* start of a save set */
#define SSI_SYNC        2               /* synchronization point */
#define SSI_CONT        3               /* continued from another volume */
#define SSI_END         4               /* end of this save set */
#define SSI_SSMASK      0x0000000f      /* save set sync chunk type */
#define SSI_LBIAS       0x10000000      /* the level is included in the flags */
```

```
#define SSI_LMASK        0xff000000  /* mask to cover bits for level */
#define SSI_LSHIFT       24          /* shift amount for the level */
#define SSI_INCOMPLETE 0x00010000/* not finished (aborted) */
#define SSI_CONTINUED  0x00800000/* continued save set series */
```

The **ssi_ssid** is the save set identifier of this save set. The **ssi_time** field contains the create time of the save set in UST based on the client's clock. The **ssi_create** field contains the create time of the save set in UST based on the server's clock. The **ssi_insert** field contains the time the save set was inserted into the media database in UST based on the server's clock. The **ssi_complete** field contains the completion time of the save set in UST based on the server's clock. The **ssi_clientid** and **ssi_host** are the client identifier and name of the index which contains this save set. Traditionally this is the client identifier and name of the client where the save set originated. The **ssi_name** is the save set name to be presented to the user. These are both null-terminated strings, even though the fields are fixed length in the older version media records. The **ssi_size** and **ssi_nfiles** are the number of bytes and number of files saved so far for this save set. The **ssi_browse** is the time offset in seconds from the save set insertion time to the time this save set is no longer browsable. The **ssi_recycle** is the time offset in seconds from the save set insertion time to the time this save set becomes recyclable. The **ssi_al** is the generic save set attribute.

The **ssi_flag** indicates the type of this synchronization chunk and other information about the save set. In the older version synchronization chunk, this field also contains the level of this save set. There are four basic types of synchronization marks that can be found from examining **ssi_flag & SSI_SSMASK**. **SSI_START** is used to mark the beginning of a save set. **SSI_SYNC** marks a periodic synchronization point and is only written at an exact file boundary in the save set. **SSI_CONT** indicates that this is the continuation of a save set that started on a different volume. When **ssi_flag & SSI_SSMASK** is **SSI_CONT**, **ssi_prev** or **ssi_info** contains the volume identifier for the save set's preceding volume. These synchronization chunks are used when a save set spans a volume boundary. **SSI_END** marks the end of a save set.

On the new version of synchronization chunk, the **ssi_level** field contains the save set backup level. On the older version of synchronization chunk. Should the **SSI_LBIAS** bit be set then **ssi_flag** & **SSI_LMASK** shifted to the right by **the value of SSI_LSHIFT** specifies the level of the save set. The **SSI_INCOMPLETE** bit indicates that this save set did not finish properly. This could be caused by a user interrupting an in progress save.

The **SSI_CONTINUED** bit indicates that this save set is logically continued to or from another save set. These continued save sets are used to handle very large save sets. If the **SSI_CONTINUED** bit is set and **ssi_flag & SSI_SSMASK** is **SSI_START**, then **ssi_prev** or **ssi_info** gives the previous save set id that this save set was continued from. If the **SSI_CONTINUED** bit is set and **ssi_flag & SSI_SSMASK** is **SSI_END**, then **ssi_prev** or **ssi_info** gives the next save set id that this save set is continued to.

The **ssi_expiry** field is the expiration date, in UST, for this save set. This field is zero if an explicit save set expiration time was not specified when the save set was created. This field no longer exists in the new synchronization chunk.

SEE ALSO **nsr_device**(5), **nsr_data**(5), **nsrmm**(8), **nsrmmd**(8), **nsrmmdbd**(8), **nsr**(8), **scanner**(8)

RFC 1014 XDR: External Data Representation Specification

NAME msense − get mode sense data

SYNOPSIS **msense -a** *b.t.l* [**-p** *pagecode*]

DESCRIPTION The **msense** program will send a MODE SENSE command to the named device.

OPTIONS The required **-a** argument must be used to select a specific ordinal SCSI address (see **libscsi**(1m)).

The optional **-p** *pagecode* argument may be used to select a specific mode page, else all pages are fetched (code 0x3f). This argument must be specified in hexadecimal notation.

BUGS The output is not readable. It is intended as input to **pmode**(1m).

SEE ALSO **libscsi**(1m), **pmode**(1m)

NAME networker.cluster − configure NetWorker software as highly-available

SYNOPSIS **networker.cluster** [−**r** *nsr_bin*]

DESCRIPTION The **networker.cluster** is an interactive script for configuring the NetWorker Client or NetWorker Server as a highly available application in a cluster. It should be run after a proper installation of NetWorker on all the nodes of the cluster to activate the fail-over and cluster-aware capabilities of NetWorker.

The configuration separates the local NetWorker database area used by the cluster-aware NetWorker Client from the global NetWorker database area used by the Highly Available NetWorker Server. This is done via a set of symbolic links which are created to enable easy switching between the global and local NetWorker databases. The global NetWorker database is on a shared storage media and follows the Highly Available (virtual) NetWorker Server on a failover.

The **NetWorker.clustersvr** file is created in the NetWorker binaries directory to specify that NetWorker has been installed as a high-available service.

The script then creates the appropriate **lcmap**(1m) script for the cluster platform.

When NetWorker Server is configured, the user is asked for cluster platform specific information to prepare for registration of the NetWorker Server with the cluster software. However, the setup is finished manually. Only the NetWorker Client is started on the node by the **networker.cluster** script. Refer to the *NetWorker Installation Guide* appropriate for your cluster platform for specific instructions.

If a mistake is made during the configuration, run **networker.cluster** with the −**r** option to undo the changes.

OPTIONS −**r** *nsr-bin*

Used to remove the cluster configuration of NetWorker software. The optional *nsr-bin* parameter is used to specify the location of NetWorker binaries when NetWorker software is installed in non-default location.

SEE ALSO **nsrd**(1m), **pathownerignore**(5), **lcmap**(1m)
The *EMC NetWorker Installation Guide*

NAME nsr − introduction and overview of NetWorker

DESCRIPTION NetWorker facilitates the backup and recovery of files on a network of computer systems. Files and filesystems may be backed up on a scheduled basis. Recovery of entire filesystems and single files is simplified by use of an on-line index of saved files.

NetWorker uses a client-server model to provide the file backup and recover service. At least one machine on the network is designated as the NetWorker *server*, and the machines with disks to be backed up are NetWorker *clients*. Six daemons provide the NetWorker service, control access to the system, and provide index and media support. On the clients, there are special programs to access the file systems and communicate with the NetWorker server.

The NetWorker system has several parts. Commands and files are only briefly mentioned here; see the appropriate reference manual page for more detailed information. Each command has a manual page entry in section 8. The files and their formats are explained in section 5 manual pages.

The *NetWorker Administrator's Guide* provides information on configuring and administering a NetWorker system. It includes many examples and rationales for setting up and running a successful backup operation.

INSTALLATION How NetWorker is installed depends on the architecture of the machine upon which you are installing. For detailed installation instructions, see the *NetWorker Installation Guide* for your specific platform.

nsr_layout(5)
Describes where NetWorker programs, files, and manual pages are installed.

SERVER DAEMONS NetWorker uses a client-server model to provide a backup and recover service. The following daemons encompass the server side of NetWorker.

nsrd(1m) The main NetWorker daemon. **nsrd** handles initial communication with clients, and starts and stops the other NetWorker server daemons.

ansrd(1m) The *agent* nsrd process, spawned by **nsrd** in response to a recovery, clone, or other session. The **ansrd** daemon is invoked on an as-needed basis and is only present when there are sessions active to the NetWorker server. Modern versions of **save**(1m) do not require use of an **ansrd** daemon.

nsrindexd(1m)
This server daemon provides access to the NetWorker on-line index. The index holds records of saved files. The index allows clients to selectively browse and choose files to recover without having to access the backup media.

nsrmmdbd(1m)
The media management database daemon provides an index of save sets and media. The **nsrmmdbd** daemon provides a much coarser view of the saved files than does **nsrindexd**, and therefore the resultant index is usually much smaller.

nsrjobd(1m) The jobs daemon provides for the centralized monitoring and control of remote execution "jobs", typically save and directed recover. It manages the parallelism when spawning remote jobs and monitors the status for reporting and storing execution information. All scheduled backups are initiated from **nsrjobd.**

nsrmmd(1m) The media multiplexor daemon provides device support for NetWorker. When more than one client is saving files, the data from each client is multiplexed. During recovery operations, the data is demultiplexed and sent back to the requesting clients. When the multiple devices are enabled, several of these daemons may be active simultaneously.

ADMINISTRATION

NetWorker is administered via *resources* and *attributes*. Every resource has one or more attributes associated with it. For example, a device is a NetWorker resource type; an attribute of devices is the device *type*, for example, 4mm or 8mm. The NetWorker resource format is documented in **nsr_resource**(5). There is also a manual page for each NetWorker resource in section 5 of the manual.

Resource files are not normally edited by hand. Rather, a NetWorker tool (usually **NetWorker Management Console** or **nsradmin**(1m)) is used to modify resource files dynamically so that values can be checked and changes can be propagated automatically to the interested programs. The following are tools that are used to administer various aspects of NetWorker.

NetWorker Management Console
Monitors the activity of and administers NetWorker servers. **NetWorker Management Console** is a Java based application and is most users' primary interface to NetWorker.

nsradmin(1m)
A **curses**(3) based tool for the administration of NetWorker servers.

nsrwatch(1m)
A **curses**(3) based tool to monitor the activity of NetWorker servers.

nsrmm(1m) Media manager command. The **nsrmm** command is used to label, mount, unmount, delete and purge volumes. Mount requests are generated by **nsrmmd**, and displayed by **NetWorker Management Console** or **nsrwatch**. The size of the on-line user file indexes may be controlled by deleting and purging volumes.

nsrjb(1m) The NetWorker jukebox-controlling command. When dealing with a jukebox, **nsrjb**, rather than **nsrmm**, should be used to label, load, and unload the volumes contained within a jukebox.

nsrim(1m) Automatically manages the on-line index. It is usually run periodically by **savegrp**.

mminfo(1m)
Provides information about volumes and save sets.

nsrck(1m) Checks and repairs the NetWorker on-line index. It is run automatically when **nsrd** starts up if the databases were not closed cleanly due to a system crash.

nsr_render_log(1m)
Creates a human readable version of the Networker logs.

nsr_shutdown(1m)
A shell script used to safely shut down the local NetWorker server. The **nsr_shutdown** script can only be run by the super user.

SAVING FILES

NetWorker supports both scheduled and manual saving of files and filesystems. Each client may be scheduled to save all or part of its filesystems. Different clients may be scheduled to begin saving at different times.

save(1m) A command-line-based tool used to back up a specified file or group of files. The **save** command may be run manually by users and

administrators, or automatically by **savegrp**.

savegrp(1m) Used to initiate the backup of a group of client machines. Usually started automatically by the NetWorker server. The **savegrp** command also backs up the clients' on-line file indexes, which are stored on the server. When backing up the server itself, a *bootstrap* save set is also created.

nsrexec(1m) The *agent* savegrp process, spawned by **savegrp**. The **nsrexec** command monitors the progress of NetWorker commands.

nsrclone(1m)
The NetWorker save set/volume cloning command. Using **nsrclone**, *clones*, or exact replicas, of save sets or entire volumes can be made. Clone data is indistinguishable from the original data, except for the NetWorker media volumes upon which the data reside.

nsrexecd(1m)
NetWorker-specific remote execution service which runs on NetWorker clients. Used by **savegrp** to start **save** and **savefs** on client machines.

savefs(1m) Used by **savegrp** to determine characteristics of a client, and to map the save set *All* to the current list of all save sets on a client.

RECOVERING FILES

NetWorker maintains an on-line index of user files that have been saved. Users may browse the index and select files for recovery. This information is used to build a representation of the file heirarchy as of any time in the past. NetWorker then locates the correct volume and recovers the requested files.

recover(1m) Browses the on-line user file index and selects files and filesystems to recover.

nwrecover(1m)
A Motif-based tool for recovering files. The **nwrecover** command is the graphical equivalent of **recover**.

mmrecov(1m) Used only for disaster recovery. Recovers the special *bootstrap* index and the server's on-line file index. The **recover** or **nwrecover** commands are used to recover other on-line file indexes.

scanner(1m) Verifies correctness and integrity of NetWorker volumes. Can also recover complete save sets and rebuild the on-line file and media indexes.

nsr_crash(1m) A man page describing crash recovery techniques.

nsrinfo(1m) Used to generate reports about the contents of a client's file index.

APPLICATION SPECIFIC MODULES

In order to process user files in an optimal manner, NetWorker provides the ASM mechanism. Pattern matching is used to select files for processing by the different ASMs. The patterns and associated ASMs are described in **nsr**(5). The **save** command keeps track of which ASMs were used to process a file so that **recover** may use the same ASMs to recover the file.

uasm(1m) UNIX filesystem specific save/recover module. The **uasm** man page documents the general rules for all ASMs. The **uasm** command and its man page actually comprise several additional ASMs, including **compressasm**, **mailasm**, and **xlateasm**, to name a few.

nsrindexasm(1m)
Processes the on-line user file indexes.

nsrmmdbasm(1m)
Processes the on on-line media database.

SERVER LOCATION

On large networks there may be several NetWorker servers installed. Each NetWorker client command must select a server to use.

For server selection, the client commands are classified into two groups: *administration* and *operation*. The administration commands include **NetWorker Management Console, nsrwatch**, and **mminfo**. The operation commands include **save**, **savefs**, and **recover**. Both groups of commands accept a −**s** *server* option to explicitly specify a NetWorker server.

When a server is not explicitly specified, the operation commands use the following steps to locate one. The first server found is used.

1) The local machine is examined to see if it is a NetWorker server. If it is, then it is used.
2) The machine where the current directory is actually located is examined to see if it is a NetWorker server. If it is, then it is used.
3) The machine specified with the −**c** option is examined to see if it is a NetWorker server. If it is, then it is used.
4) The list of trusted NetWorker servers is obtained from the local machine's **nsrexecd**(1m). Each machine on the list is examined to see if it is a NetWorker server. The first machine determined to be a NetWorker server is used.
5) A broadcast request is issued. The first NetWorker server to respond to the request is used.
6) If a NetWorker server still has not been found, then the local machine is used.

The administrative commands only use step 1.

SECURITY

Before a save is allowed, there must be an **NSR client** resource created for the given client. Before a recovery is allowed, the server validates client access by checking the **remote access** attribute in the **NSR client** resource (see **nsr_client**(5)).

The **savegrp**(1m) command initiates the **save**(1m) command on each client machine in an NSR group by using the **nsrexecd**(1m) remote save execution service. See the **nsrexecd**(1m) man page for details. For backward compatibility with older versions of NetWorker, **savegrp**(1m) will fall back on using the **rsh**(1) protocol for remote execution if **nsrexecd** is not running on a particular client.

Access to the NSR resources through the **nsradmin**(1m) command or **NetWorker Management Console** is controlled by the **administrator** attribute on the NSR server resource (see **nsr_service**(5)). This attribute has a list of names of the users who have permission to administer that resources. Names that begin with an ampersand (&) denote netgroups (see **netgroup**(5)). Also names can be of the form *user@host* or **user=***user***,host=***host* to authorize a specific user on a specific host.

ROOT PRIVILEGES

The system administrator can grant root privileges to specific groups of users by changing the mode of a NetWorker program to setuid-root and setgid-group. (See **chgrp**(1) and **chmod**(1) for more details.)

When a user invokes a program that is both setuid-root and setgid-group, he may retain root privileges if one of the following is true:

1. The user's name and the program's group name are identical.
2. One of the process's supplementary group id names is identical to the program's group name. (See **getgroups**(2) for more details.)
3. The user's name is an element of the netgroup whose name is identical to the program's group name. (See **getgrnam**(3) for more details.)

For example, the mode and group owner of the **recover** command can be changed such that the **ls** output looks like:

```
-rws--s--x  1 root  staff  548808 Apr 18 16:04 recover
```

A user invoking this command will retain root privileges if (1) his name is ''staff'', or (2) he is a member of the group ''staff'', or (3) his name appears as an element of the netgroup ''staff''.

Granting root privileges may be applied to the following NetWorker programs: **nsrexec(1m), nsrports(1m), recover(1m), nwretrieve(1m), nwrecover(1m), nsrclone(1m), nsrssc(1m), nsrmm(1m), mmpool(1m), mmlocate(1m), nsrjb(1m), nsrinfo(1m), nsrstage(1m), nsrcap(1m), save(1m), nsrpmig(1m), nsrck(1m), nsrim(1m), jbconfig(1m), nsrcnct(1m),** and **scanner**(1m).

NAMING AND AUTHENTICATION

As described above, the NSR server only accepts connections initiated from the machines listed as clients or listed in the **remote access** list (for recovering). Since machines may be connected to more than one physical network and since each physical network connection may have numerous aliases, the policies below are used as a compromise between security and ease of use. For further information about naming in the UNIX environment, refer to **gethostent**(3) or other documentation on name services.

A client determines its own name as follows. First the client's UNIX system name is acquired via the **gethostname**(2) system call. The UNIX system name is used as a parameter to the **gethostbyname**(3) library routine. The client declares its name to be the official (or ''primary'') name returned by **gethostbyname**. This name is passed to the NetWorker server during connection establishment.

A server authenticates a client connection by reconciling the connection's remote address with client's stated name. The address is mapped to a list of host names via the **gethostbyaddr**(3) library function. Next, the client's stated name is used as a parameter to **gethostbyname** to acquire another list of host names. The client is successfully authenticated only if a common name between the two lists exists.

The NetWorker server maps a client's name to an on-line index database name by resolving the client's name to the official name returned by **gethostbyname**. This mapping takes place both at client creation time and at connection establishment time.

To ensure safe and effective naming, the following rules should be employed:

1) The NetWorker clients and servers should access consistent host name databases. NIS (YP) and the Domain Name System (DNS) are naming subsystems that aid in host name consistency.
2) All hosts entries for a single machine should have at least one common alias among them.
3) When creating a new client, use a name or alias that will map back to the same official name that the client machine produces by backward mapping its UNIX system name.

SEE ALSO

rsh(1), **gethostname**(2), **gethostent**(3), **netgroup**(5), **nsr**(5), **nsr_layout**(5), **nsr_resource**(5), **ypfiles**(5), **ypmake**(5), **mminfo**(1m), **nsr_crash**(1m), **nsr_service**(5), **nsr_render_log**(1m), **nsr_shutdown**(1m), **nsradmin**(1m), **nsrck**(1m), **nsrclone**(1m), **nsrd**(1m), **nsrexecd**(1m), **nsrim**(1m), **nsrindexasm**(1m), **nsrindexd**(1m), **nsrinfo**(1m), **nsrjb**(1m), **nsrls**(1m), **nsrmm**(1m), **nsrmmd**(1m), **nsrmmdbasm**(1m), **nsrmmdbd**(1m), **nsrwatch**(1m), **nwrecover**(1m), **recover**(1m), **mmrecov**(1m), **save**(1m), **savefs**(1m), **savegrp**(1m), **scanner**(1m), **uasm**(1m),**and**
the *NetWorker Administration Guide*

NAME nsr − NetWorker directive file format

DESCRIPTION This man page describes the format of **.nsr** directive files. These files are interpreted by **save**(8) and Application Specific Module (ASM) programs, during NetWorker backup processes. This format is also used in the *directive* attribute of the **nsr_directive**(5) resource.

Directives control how particular files are to be backed-up, how descendent directories are searched, and how subsequent directives are processed. For each file backed-up, any ASM information required to recover that file is also backed-up. This enables **recover**(8), or any ASM directly invoked, to recover a file correctly, even if the current directives have changed since the file was backed-up. See **uasm**(8) for a general description of the various ASMs.

The **.nsr** directive file in each directory is parsed before anything in that directory is backed up, unless NetWorker is being run in *ignore* mode. Each line of a **.nsr** directive file, and each line of the **directive** attribute, contains one directive. Any text after a "#" character until the end of the line is treated as a comment and discarded. Directives appear in one of three distinct forms:

[+] *ASM* [*args ...*] **:** *pattern ...*
save environment
<< dir >>

The three forms are referred to as ASM specifications, *save environment* directives, and *<< dir >>* directives, respectively.

Use ASM specifications (name and any arguments) to specify how files or directories with a matching *pattern* are backed-up. When a *pattern* matches a directory, the specified ASM is responsible for handling the directory and its contents. Any *pattern* or ASM arguments requiring special control or white space characters should be quoted using double quotes (").

A colon (**:**) is used as the separator between the ASM specification (and any arguments) and the *pattern* specification list. The *pattern* list for each ASM specification consists of simple file names or patterns. The *pattern* cannot be "**..**" and must not contain any "/" characters (all names must be within the current directory). The string "**.**" can be used to match the current directory. Standard **sh**(1) file pattern matching (*, [...], [!...], [x-y], ?) can be used to match file names. If a "+" precedes the ASM name, then the directive is propagated to subdirectories. When a directory is first visited, it is searched for a **.nsr** file. If one is found, it is then read. Each **.nsr** file is only read once. When starting a save at a directory below **/**, any **.nsr** files on the normalized path of the current working directory are read before any files are saved to catalog any propagated directives.

The following algorithm is used to match files to the appropriate ASM specification. First the **.nsr** file in the current directory (if any) is scanned from top to bottom for an ASM specification without a leading "+" whose *pattern* matches the file name. If no match is found, then the **.nsr** in the current directory is re-scanned for an ASM specification with a leading "+" whose *pattern* matches the file name (for clarity, we recommend placing all propagating ("+") directives after all the non-propagating directives in a **.nsr** file). If no match is found, then the **.nsr** file found in the "**..**" directory (if any) is scanned from top to bottom looking for a match with an ASM specification that has a leading +. This process continues until the **.nsr** file in the "/" directory (if any) is scanned. If no match is found (or a match is found with an ASM specification whose name is the same as the currently running ASM), then the currently running ASM will handle the save of the file.

Use *save environment* directives to change how ASM specifications and future **.nsr** files are used. The *save environment* directives do not take any file patterns. They affect the currently running ASM and subsequent ASMs invoked below this directory. There are three different possible *save environment* directives that can be used:

forget
Forget all inherited directives (those starting with a "+" in parent directories).

ignore
Ignore subsequent **.nsr** files found in descendent directories.

allow Allow **.nsr** file interpretation in descendent directories.

The << *dir* >> directive can be used to specify a directory where subsequent ASM specifications from the current **.nsr** file should be applied. This directive is intended to be used to consolidate the contents of several **.nsr** files to a single location or directory. The *dir* portion of this directive must resolve to a valid directory at or below the directory containing this directive or subsequent ASM specifications will be ignored. Relative path names should be used for file names to ensure the interpretation of subsequent ASM directives is consistent, even if a directory is mounted in a different absolute part of the filesystem.

There **must** be a << *dir* >> as the first directive in a directive file used in conjunction with the **−f option to save**(8), **savefs**(8) or with an ASM program. Also, when << *dir* >> directives are used in this manner, whether first or later in the file, absolute path names should be used to ensure appropriate interpretation. Absolute path names should also be used for each directory specified within the **directive** attribute of the **NSR directive** resource (see **nsr_directive**(5)).

When a << *dir* >> directive is used, subsequent directives are parsed and logged for later use. When a directory specified by *dir* is opened, any *save environment* directives specified for that directory (for example, **allow**, **ignore**, and **forget**) are processed first. If the ASM is not currently ignoring **.nsr** files and a local **.nsr** file exists, the file is read and processed. Finally, any of the non *save environment* directives specified for that directory are handled as if they where appended to the end of a **.nsr** file in that directory. If multiple << *dir* >> specifications resolve to the same directory, then the corresponding save directives are handled logically in "last seen first" order.

EXAMPLES

Having a */usr/src/.nsr* file containing:

```
+skip: errs *.o
+compressasm: .
```

will cause all files (or directories) located in the */usr/src* directory named **errs** or ***.o** (and anything contained within them) to be skipped. In addition, all other files contained in the */usr/src* directory will be compressed during save and will be set up for automatic decompression on recover.

Having a */var/.nsr* file containing:

```
compressasm: adm .nsr
null: * .?*
```

causes all files (or directories) and their contents located within the */var* directory and anything contained within them (except for those files located in the */var/adm* directory and the **.nsr** file itself) to be skipped, although all the names in the directory would be backed-up. In addition, since **compressasm** is a *searching* directive (see **uasm**(8)), the files contained within the */var/adm* directory will be compressed during backup and will be set up for automatic decompression on recover.

The following is an example of using the **/.nsr** file as a master save directive file for the entire filesystem by using << *dir* >> directives to consolidate the various ASM save directives to a single location:

```
# Master NetWorker directive file for this machine
<< ./ >>
# /mnt and /a are used for temporary fs mounting
# and need not be saved
        skip: mnt a
        +skip: core errs dead.letter *% *~
# Don't bother saving anything within /tmp
<< ./tmp >>
        skip: .?* *
<< ./export/swap >>
        swapasm: *
# Translate all mailboxes. Also, use mailasm to save each
# mail file to maintain mail file locking conventions and
# to preserve the last file access time.
<< ./usr/spool/mail >>
        xlateasm: .
        mailasm: *
# Allow .nsr files to be interpreted in /nsr, even if we
# are currently ignoring .nsr files. NetWorker
# applications (such as nsrindexd) set up their own private
# .nsr files which save index files more intelligently.
<< ./nsr >>
        allow
# We can rebuild any .o files in /usr/src
# from sources except those in /usr/src/sys.
<< ./usr/src >>
        +skip: *.o
<< ./usr/src/sys >>
        forget
```

FILES **.nsr** save directive file in each directory

SEE ALSO **sh**(1), **nsr_directive**(5), **nsrindexasm**(8), **nsrmmdbasm**(8), **recover**(8), **save**(8), **savefs**(8), **uasm**(8)

NAME nsraddadmin − add an entry to the administrator attribute

SYNOPSIS **nsraddadmin** −**u** *user-entry* [−**s** *server*]

DESCRIPTION The **nsraddadmin** program is used to add a user entry to a NetWorker server's administrator attribute. The program updates the server on the same host where the command runs. The addition of a user entry gives that user full administrator privileges on the NetWorker server. Please see **nsr_service**(5) for additional information about this attribute and for valid formats of the user entries.

OPTIONS −**u** Causes **nsraddadmin** to add a user entry to NetWorker's administrator attribute. Only one user entry at a time can be added with this command.

−**s** *server*
Opens a connection to the named NetWorker server instead of the local one.

SEE ALSO **nsr_service**(5), **nsrd**(1m)

DIAGNOSTICS **get resdb handle failed, err:** *error info*
An error occurred while connecting to the NetWorker server. Check to ensure the server is running, and retry the command.

query resdb failed, err: *error info*
An error occurred while querying the NetWorker server. Check to ensure the server is running, and retry the command.

RAP error: Permission denied, user '*user***' on '***hostname***' does not have 'Change security settings' privilege**
The user running this program is not listed in the administrator's list for the server. You need to be a valid administrator to run **nsraddadmin.**

user '*user-entry***' is already on the administrator list**
The user entry that was given on the command is already contained on the server's administrator list.

added user '*user-entry***' to the administrator list**
The user entry has been added to the server's administrator list.

NAME nsradmin − NetWorker administrative program

SYNOPSIS

nsradmin
[−**c**] [−**i** *file*] [−**s** *server*] [−**p** {*prognum* | *progname*}] [−**v** *version*] [*query*]

nsradmin
[−**c**] [−**i** *file*] [−**d** *resdir* ...] [−**t** *typefile*] [*query*]

nsradmin
[−**c**] [−**i** *file*] [−**f** *resfile* ...] [−**t** *typefile*] [*query*]

DESCRIPTION The **nsradmin** command is a command-line based administrative program for the NetWorker system. Normally **nsradmin** monitors and modifies NetWorker resources over the network. Commands are entered on standard input, and output is produced on standard output.

If **nsradmin** is started without a query argument, it uses a default query. If the daemon being administered is nsrd, then *all* resources will be selected by default; for all other daemons, no resources will be selected by default.

OPTIONS

−**c** Uses the **termcap**(5) and **curses**(3) packages to implement a full-screen display mode, just like the **visual** command described below. (UNIX Only)

−**d** *resdir*
Uses the NetWorker resource database *resdir* instead of opening a network connection. The database *resdir* must be in directory format. This should be used sparingly, and only when the NetWorker server is not running. Multiple −**d** and *resdir* arguments can be used to start **nsradmin** with access to more than one database at a time.

−**f** *resfile*
Similar to the −**d** *resdir* option except that it opens an existing resource file, rather than a resource directory. Some configuration databases are stored in file format, others in directory format.

−**i** *file* Takes input commands from *file* instead of from standard input. In this mode, the interactive prompt will not be printed.

−**s** *server*
Opens a connection to the named NetWorker server instead of allowing administration of all servers. Useful to limit the number of resources if there are many servers, or to administer when the RAP location service is not working.

−**p** {*prognum* | *progname*}
Use the given RPC program number or name instead of the default program number of 390103 - which refers to *nsrd.* Other suitable program arguments include, but are not limited to:

NetWorker Remote Execution Daemon:

390113 or *nsrexecd*

Host Agent Daemon:

390427 or *hagentd*

−**t** *typefile*
Uses the alternate file *typefile* to define RAP types.

−**v** *version*
Binds to the NetWorker RAP service with the given version number. The default is 2. This option is generally used only for debugging.

query If a query is specified (in the form of an attribute list), the edit operation is performed on the results of the query. See COMMANDS for more information on how the edit command works.

RESOURCES

Each NetWorker resource is made up of a list of named attributes. Each attribute can have zero or more values. The attribute names and values are all represented by printable strings. Upper and lower case is not distinguished on comparisons, and spaces are ignored except inside the names and values.

The format for specifying attributes and attribute lists is:

attribute ::= *name* [**:** *value* [**,** *value*]∗]
An attribute is a name optionally followed by a colon, followed by zero or more values, with values separated by commas. A comma at the end of a line continues the line.

attribute list ::= *attribute* [**;** *attribute*]∗
An attribute list is one or more attributes separated by semicolons. A semicolon at the end of a line continues the line. The list is ended by a newline that is not preceded by a comma or semi-colon.

Here is an example of an attribute list:

```
name: mars;
type: NSR client;
remote access: mars, venus, jupiter;
```

For more information on attributes, attribute lists and the NetWorker resource types, see the **resource**(5), and **nsr_resource**(5), manual pages.

COMMANDS

At each input prompt, **nsradmin** expects a command name and some optional arguments. Command names can be shortened to the smallest unique string (for example, p for print). Command arguments are always specified in the form of an attribute list. Most commands operate on a set of resources returned by a *query*. The query is specified as an attribute list which is used to match resources with the following rules:

1) The resource must match all the given attributes.

2) If more than one value is specified the resource can match any one of the values.

3) If an attribute is specified with no value the resource must contain an attribute of that name.

Thus, a query:

```
type:NSR device;
name:mars, venus;
test
```

will match all resources that have a **type** attribute with the value **NSR device** and a **name** attribute with a value of either **mars** or **venus**, and an attribute **test** with any value.

If the query has only one name and no values (for example, if there is no semi-colon or colon in it), then the program tries to guess a more reasonable query. If the name is a host name, then the query will select all the resources on the given host. Otherwise, the name will be interpreted as a type name, and all resources of that given type will

be selected.

bind [*query*]
Bind to the service that owns the resource described by *query*. If no query is specified, queries are sent to the RAP Resource Directory, and update, create, and delete commands to the service that owns the resource being changed. On failure, the previous service will continue to be used.

create *attribute list*
Create a resource with the given attributes. One of the attributes must be **type** to specify a NetWorker type that can be created. The **types** command can be used to find out which NetWorker types a server supports. Note that the RAP types are case sensitive and must be used exactly as shown by the types command. For example, NSR group is a valid type, whereas nsr group is not.

delete [*query*]
Delete the resources that match the current query. If a *query* is specified, it becomes the current query.

edit [*query*]
Edit the resources that match the current query. If a *query* is specified, it becomes the current query. If the environment variable EDITOR is set, then that editor will be invoked, otherwise **vi**(1) will be started. When the editor exits, **nsradmin** applies update, delete and create operations based on the changes to the resources. Be careful to not edit the resource identifier attribute, and to write the file out before exiting the editor. (UNIX Only)

help [*command*]
Print a message describing a command. If no command name is given a synopsis of all of the commands is printed.

option [*list*]
This command enables some options to change the display of resources. With no arguments it displays the current options; with a list of options it turns the specified ones on. The options are: **Dynamic** displays all dynamic attributes, even the normally hidden ones. **Hidden** displays all attributes, even the normally hidden ones. **Raw I18N** suppresses rendering of I18N text to the current locale, and displays the I18N data as raw structured text. **Resource ID** displays the resource identifier on each resource, a number that is used internally to provide sequencing and uniqueness. **Regexp**, when enabled, supports regular expression search for the resources.

print [*query*]
Print the resources that match the current query. If a *query* is specified, it becomes the current query. If a name has been specified for the the current show list, only the attributes for the specified name in the show list will be displayed.

quit
Exits **nsradmin.**

server [*servername*]
Bind to the given NetWorker server name. If no server is specified, the RAP location service will be used. On failure, the previous server will continue to be used.

show [*name***;** ...]
If a name list (really an attribute list with no values) is specified, add those names to the show list. Only these attributes will be displayed in subsequent **print** commands. If no name list is given the show list is cleared, resulting in all attributes being shown.

types Print a list of all known types.

unset [*list*]
This command turns off the specified option.

update *attributes*
Update the resources given by the current query to match *attributes*.

visual [*query*]
Enter a full-screen mode using the **curses**(3) package to step through commands in a perhaps more user-friendly manner than the command line interface. You can get this mode directly using the **−c** command line argument. (UNIX Only)

. [*query*]
If a *query* is specified, this command will set the current query without printing the results of the query. Otherwise, it will display the current query, show list, server binding, and options.

? [*command*]
Same as the **help** command above.

EXAMPLES

print type:NSR device
Print all resources of type **NSR device** and make this the current query.

show type; name
Set the show list to display only the attributes **type** and **name**.

delete
Delete all resources that match the current query.

delete type:NSR device; hostname: mars
Delete the resource with attributes: **type: NSR device** and **hostname: mars**.

edit type:NSR notification
Edit all resources of type **NSR notification**.

SEE ALSO

ed(1), **vi**(1), **curses**(3), **nsr_resource**(5), **termcap**(5), **nsr**(1m)

NOTES

If the backslash ("\") character is contained in a value that is entered for an attribute value when you create or update a RAP resource, it is treated as a marker that indicates that it may be combined with the following character to produce a special character. (This is similar behavior to that seen in various UNIX shells.)

If you wish your attribute value to contain an actual backslash character, then you should enter two backslashes in succession - e.g. C:\\dir_one\\dir_two

DIAGNOSTICS

The following exit status values are meaningful:

0 Interactive mode exited normally.

1 There was a usage or other non-query related error.

2 When reading input from a file (**−i** *file*), one or more RAP operations failed. This status is never returned interactively.

NAME nsralist − NetWorker archive request executor

SYNOPSIS **nsralist** −**R** *archive request name*

DESCRIPTION The **nsralist** command is used to execute an archive request (see **nsr_archive_request**(5)). The **nsralist** command is run automatically by **nsrd**(1m), as specified by each archive request resource.

The **nsralist** command will set up an RPC connection to **nsrexecd**(1m) to run **nsrarchive**(1m) on the specified client. If **nsrexecd** is unavailable, **nsralist** will fall back on using the **rcmd**(3) protocol and the client-side **rshd**(1m).

The **nsralist** monitors the execution of the archive command and stores any output in the log of the archive request. The **nsrarchive** command running on the client updates the server with its progress, including whether or not optional verification and cloning operations have completed successfully. See **nsrclone**(1m) for more information on cloning.

OPTIONS −**R** *archive request name*
This option specifies which archive request is supposed to be run.

FILES **/nsr/tmp/al.***request_name*
A lock file to keep multiple runs of the same archive list from running simultaneously.

SEE ALSO **nsrarchive**(1m), **nsrclone**(1m), **nsrexecd**(1m), **nsr_archive_request**(5).

NAME nsrarchive − archive files to long term storage with NetWorker

SYNOPSIS **nsrarchive** [−**BiInpqvxVy**] [−**b** *pool*] [−**C** *clone pool*] [−**f** *directive filename*] [−**G remove**] [−**I** *input_file*] [−**N** *name*] [−**R** *name*] [−**s** *server*] [−**T** *annotation*] [−**o** *save_operations*] [−**W** *width*] [*path* ...]

DESCRIPTION **nsrarchive** archives files, including directories or entire filesystems, to the NetWorker server (see **nsr**(1m)). The progress of an archive can be monitored using the Java based **NetWorker Management Console** or the **curses**(3X) based **nsrwatch**(1m) program for other terminal types. Use of **nsrarchive** is restricted to users in NetWorker administrator list or archive users lists with "Backup local data" privilege.

If no *path* arguments are specified, the current directory is archived. **nsrarchive** archives a directory by archiving all the files and subdirectories it contains, but it does not cross mount points or follow symbolic links.

The directive files (see **nsr**(5)) encountered in each directory is read by default, The files contain special instructions directing how particular files are to be archived (that is, compressed, skipped, etc.). These files are named .nsr for UNIX platforms and **nsr.dir** for Windows platforms.

Each file in the subdirectory structures specified by the *path* arguments is encapsulated in a NetWorker archive stream. This stream of data is sent to a receiving process (see **nsrd**(1m)) on the NetWorker server. Entries are added to the media database for the archive save set. The data eventually resides on a long term storage medium (see **nsrmmd**(1m)).

Details about handling media are discussed in **nsrmm**(1m) and **nsr_device**(5).

If the grooming option, −**G remove,** is requested, you can selectively remove files and directories that have been archived. If verification is requested, the files will not be removed if the verification failed. Likewise, the files will not be removed if a requested cloning operation fails. The user is prompted for confirmation before the files and directories are removed unless the −**y** option is supplied.

If the user does not supply a −**T** option on the command line, the user is prompted to enter an annotation for the archive.

OPTIONS

−**b** *pool* Specify a destination pool for the archive save set. If this option is not used, the Indexed Archive pool is used.

−**B** Force archive of all connecting directory information from root (''/'') down to the point of invocation. The connecting directory information is always archived even without this option if client file index is generated.

−**C** *clone pool*
Generate a clone of this archive save set to the specified *clone pool*.

−**E** Estimate the amount of data which will be generated by the archive, then perform the actual archive. The estimate is generated from the inode information, and thus the data is only read once.

−**f** *filename*
The file from which to read default directives (see **nsr**(5)). A *filename* of **-** causes the default directives to be read from standard input.

−**i** Ignore directive files as they are encountered in the subdirectory structures being archived.

−**I** *input_file*
In addition to taking the paths for nsrarchive from the command line, a list of paths in the named input_file will be archived. The paths must be listed one

per line. If no paths are specified on the command line, then only those paths specified in the input_file will be archived.

−G remove

Groom the files after they have been archived. If cloning or verification is requested, no grooming is performed until those operations have completed successfully.

Be aware that the groom option must be entered as **−G remove.** Entering −G alone will not invoke the groom option.

The user is prompted for removal of files and directories unless the **−y** option is supplied as one of the **nsrarchive** options. The valid remove responses and their meanings are:

n Keep the current file or directory.

y Remove the current file or directory.

N Keep all remaining files and directories.

Y Remove all remaining files and directories.

The default response, "**n**", is displayed within square brackets and can be selected by pressing [**Retum**]. When either **Y** or **N** is specified, there will be no further prompting and each subsequent removal decision is made as if the corresponding lower case letter has been selected.

nsrarchive creates a temporary file which contains a list of all files and directories to be removed. The temporary file is placed in */tmp* unless the environment variable **TMPDIR** is set.

−n No archive. Estimate the amount of data which will be generated by the archive, but do not perform the actual archive.

−N *name*

The symbolic name of this archive save set. By default, the first *path* argument is used as the name.

−v Verbose. Cause the **nsrarchive** program to tell you in great detail what it is doing as it proceeds.

−p Exit with status 0. Used by server to determine if client installed properly.

−q Quiet. Display only summary information or error messages.

−R *name*

This option should only be used by the **nsralist** program, which handles executing archive requests. Updates to the named archive request resource occur when this option is specified.

−s *server*

Specify which machine to use as the NetWorker server.

−T *annotation*

Archive save sets can be annotated with arbitrary text. This option specifies an annotation for the archive save set being generated.

−V Verify the archive save set after it completes.

−o *save_operations*

Save Operations of the form KEYWORD:TOKEN=STATE. It is used to configure VSS saves on Windows 2003. Examples:

"vss:*=off" Turn off VSS.

"vss:Microsoft Exchange Writer=off" Disable a writer.

"vss:C:=off" Disable VSS for a drive.

Please see the Admin Guide for more details.

−**W** *width*
The width used when formatting summary information output.

−**x** Cross mount points.

−**y** Answer yes to any questions.

SEE ALSO **curses**(3X), **nsr_getdate**(3), **nsr**(5), **nsr**(1m), **nsr_service**(5), **nsr_device**(5), **nsrmm**(1m), **nsrmmd**(1m), **nsrd**(1m), **nsrwatch**(1m), **nsrretrieve**(1m)

DIAGNOSTICS Exit Codes:
0 Normal exit.
Non-zero
Abnormal exit.

NAME nsrcap − update the capabilities of a NetWorker installation

SYNOPSIS **nsrcap** [−**vn**] { −**c** | −**d** | −**u** } **enabler-code** [−**a authorization-code**]

DESCRIPTION The **nsrcap** program is primarily used to enable new features on NetWorker. You can also use **nsrcap** to upgrade or downgrade NetWorker software features that are currently being used. (Upgrades and downgrades should be performed carefully. Read the options descriptions below). Enablers are separate from the NetWorker software, and are specified by an 18-digit enabler-code, usually displayed as 3 groups of 6 digits each. The authorization code is an 8 digit hexadecimal number. To enable a new feature, the **nsrd**(1m) program must be running on the system where the NetWorker server software is installed. To enable a new feature you must be logged in to the NetWorker server as administrator or root. The **nsrcap** program is run once for each feature you want to enable by specifying the 18-digit enabler-code. You may authorize a new enabler as you enable the feature or authorize an already existing enabler. If no errors occur, the following message gets displayed on the screen: "License enabler loaded. Please register all enablers immediately." You can inspect the enablers currently loaded by viewing the NSR license resources using **nsradmin**(1m).

OPTIONS

−**c** Causes **nsrcap** to enable a feature that is not currently installed, using the specified enabler code. Enabler codes are listed on enabler certificates provided when you purchase NetWorker. An authorization code is required to make each license permanent. To obtain authorization codes for your NetWorker product via the World Wide Web, simply point your web browser to URL: http://customernet.emc.com. You will need to enter the enabler code for each authorization code that you request. For more details on NetWorker licensing, including other methods to obtain authorization codes, refer to the NetWorker Installation and Administration Guide and the latest NetWorker Release Supplement. You can only load a feature once; an error is returned if you attempt to load the enabler more than once. You can only specify one of the −**c**, −**d**, or −**u** options.

−**d** Causes **nsrcap** to downgrade an existing Base or Jukebox enabler. After you downgrade the enabler, you cannot return to the previous level enabled on the system. *Do not* use the **-u** option unless instructed to do so by EMC Technical Support. You must specify one of the −**c**, −**d**, or −**u** options.

−**u** Causes **nsrcap** to enter an enabler that upgrades an existing Base or Jukebox enabler. After you upgrade the enabler, you cannot return to the previous level enabled on the system. If you use the Base enabler, it will put the new license in the grace mode. The nsrcap program utilizes the grace mode to ensure that the new enabler will not immediately time out. *Do not* use the **-u** option unless instructed to do so by EMC Technical Support.

−**v** Causes **nsrcap** to display more verbose information, describing the enabler being loaded. You must specify one of the −**c**, −**d**, or −**u** options.

−**n** No load. Causes **nsrcap** to inspect the enabler code for validity. When you specify the **-n** option, the enabler code you enter on the command line is inspected and verified, but is not entered into the NetWorker server's **nsr_license** resource. You must specify one of the −**c**, −**d**, or −**u** options.

−**a** Authorizes a license with the specified authorization code, making the license permanent. Specify the license to be authorized by using the −**c** option followed by the enabler-code and the −**a** option followed by the authorization-code. To obtain authorization codes for this product via the World Wide Web,

simply point your web browser to customernet.emc.com to enter the enabler code for each authorization code that you request. For more details on product licensing, including other methods to obtain authorization codes, refer to the product Installation and Administration Guide and the latest Release Supplement.

SEE ALSO

jbconfig(1m), **nsradmin**(1m), **nsrd**(1m), **lgtolic**(1m).

DIAGNOSTICS

enabler-code is too long
Enabler codes must be 18 digits in length. The code entered is longer than 18 digits and is invalid. Note that 24-digit enabler codes are intended for the NetWorker License Manager.

authorization-code is too long
Authorization codes must be 8 digits in length. The code entered is longer than 8 digits and is invalid.

authorization-code is too short
Authorization codes must be 8 digits in length. The code entered is shorter than 8 digits and is invalid.

invalid enabler code: xxxxxx-xxxxxx-xxxxxx
The 18-digit enabler code entered on the command line is invalid. Re-check the enabler-code on your enabler sheet.

License authorization fails
Either the 8-digit authorization code entered on the command line is invalid or this license has already been authorized and hence cannot overwrite existing authorization code. Re-check the enabler-code and the corresponding authorization-code.

cannot find a jukebox resource to enable
The code word entered is a jukebox license enabler, but there are no jukebox resources to enable. You need to run **jbconfig**(1m) to complete the jukebox installation before running **nsrcap.**

found a jukebox, but it had more than N slots.
Jukebox enablers can only enable jukeboxes with at most **N** physical slots, where **N** is the type of jukebox enabler. Either the jukebox was installed incorrectly, or you need to obtain a larger jukebox enabler.

this *enabler-code* **is already assigned**
The enabler-code entered is already loaded onto the system and cannot be used again for an upgrade.

no appropriate jukeboxes to upgrade
An upgrade was attempted, but no jukebox resources were found. Only use the **−u** option for jukeboxes when upgrading from one jukebox level to another, not on the initial installation. You also need to run **jbconfig**(1m) before running **nsrcap.**

this *enabler-code* **previously loaded**
The enabler-code entered has been loaded onto the system previously and cannot be used again. You need to purchase a new enabler-code for the upgrade.

don't know how to upgrade this enabler
don't know how to downgrade this enabler
The enabler-code entered is not for a base or jukebox enabler. These are the only types of enablers that you can currently upgrade or downgrade.

base enabler must be loaded before upgrading
base enabler must be loaded before downgrading

You cannot perform an upgrade or downgrade until a base product has been installed. Install a base enabler, and then perform the upgrade or downgrade.

cannot find the enabler to upgrade

A jukebox upgrade was attempted, but the license enabler for the jukebox is not currently loaded. You must use the **−c** option for the initial installation of a jukebox enabler, not the **−u** option.

RPC error: Program not registered

The **nsrcap** program requires that the NetWorker daemons be running. Start your NetWorker daemons (cd /; nsrd) and re-run the **nsrcap** program. If **nsrd** is already running, you have probably reached a resource limit on the server (for example, not enough memory, or no more processes).

RAP error: user *login name* **needs to be of the type: NSR administrator list.**

Your login name is not listed in the administrator's list for the server. You need to be a valid administrator to run **nsrcap.**

RAP error: ...

Various other errors can be returned from **nsrd** if the enabler is invalid. For example, if you try to load a base enabler onto a system that already has a base enabler loaded, or if you attempt to load a jukebox enabler before the jukebox has been completely installed. The specific problem will follow the **RAP error** prefix.

NAME nsrcat − NetWorker notification redirector for tty devices

SYNOPSIS **nsrcat [-n]**

DESCRIPTION The **nsrcat** command appends a carriage return to all newlines. This allows NetWorker notification messages can be redirected to the */dev/console* or */dev/tty* directory on systems with tty drivers that do not append a carriage return to output lines. This command reads text messages from standard input, appends a carriage return to the newline character, and writes the message to standard out.

OPTIONS **−n** Indicates that the codeset is to be converted from UTF-8 to the user's native character encoding.

EXAMPLES

```
type: NSR notification;
name: Log default;
action: nsrcat > /dev/console;
```

SEE ALSO **console**(4), **tty**(4), **nsr_notification**(5), **nsr**(5)

NAME nsrck − NetWorker index consistency check, repair, and recovery program

SYNOPSIS **nsrck** [−**qMv**] | [−**R** [−**Y**]] [−**L** *check-level* [−**t** *date*] | −**X** [−**x** *percent*] | −**C** | −**F** | −**m** | −**n** | −**c**] [−**T** *tempdir*] [*clientname* ...]

DESCRIPTION **nsrck** is used to check the consistency of the NetWorker online index of clients' save records. Normally, **nsrck** is started automatically and synchronously by the **nsrindexd**(1m) program when **nsrindexd** starts. You can modify the **nsrck** modes to allow normal users to run **nsrck** and retain root privileges (see **nsr**(1m) for more details).

When **nsrindexd** starts up, it determines whether any further checking of a client's index is necessary. This phase checks certain internal state of the index database, and if that state is consistent, avoids further passes. This phase also reports any suspicious-looking index names (that is indexes whose names cannot be mapped into network addresses). These online file indexes are then checked more rigorously.

nsrck detects whether any client indexes need to be converted and does the proper conversion. Converting the indices takes free space on the volume that contains the indices; if there is not sufficient free space, you may use the −**T** **tempdir** flag to specify a different directory which the conversion will use as its work space. You may also manually convert client indices by issuing the **nsrck** command manually.

There are seven different checking levels supported by **nsrck.** If client names are supplied, the check is performed on the given client names. If no names are given, the checks are performed for all client indexes. The check levels work as follows for each client checked:

Level 1 validates the online file index header, merging a journal of changes with the existing header. In addition, all save set record files and the corresponding key files are moved to the appropriate subdirectories under db6.

Level 2 does a level 1 check and checks the online file index for new and cancelled saves. New saves are added to the online file index, and cancelled saves are removed.

Level 3 does a level 2 check and reconciles the online file index with the online media index. Records that have no corresponding media save sets are discarded. Also all empty subdirectories under db6 directory are deleted.

Level 4 does a level 3 check and checks the validity of the online file index's internal key files. If any of these key files are invalid, they are rebuilt.

Level 5 does a level 4 check and verifies the digest of individual save times against their key files.

Level 6 does a level 5 check and extracts each record from each save time, verifying that each record can be extracted from the database. The digest of each save time is re-computed and compared against the stored digest, and the internal key files are rebuilt.

Level 7 does not do a level 6 check, but merges to the online file index, the index data recovered from backup media, rebuilds the internal key files, and rebuilds the index header. Note that it will not overwrite existing files in the client file index. So, if online client file index data already exists for a save set for a particular save time, it must be removed before Level 7 can be used to restore it from the backup media. The −**t date** option may be used to recover the index as of a specific time. Note that recovering the index to a specific time adds the entire contents of the index as of that time to the current index contents. This option allows browsing of save sets that have passed their browse policy and are still recoverable. The save sets referred to by the recovered index will be marked as browsable. They will remain browsable for the

length of time they were originally browsable.

For example, if a .rec file in the file index is corrupted, if a nsrck -L5 is not performed to purge the corrupted save set first before doing a nsrck -L7, then the recover will not overwrite the corrupted .rec file and the file index will remain corrupted.

Checks of a higher level generally take longer than checks at a lower level. Checks at a higher level provide a more thorough checking of the online file index. Level 7 is used when the online file index on disk needs to merge in file index data recovered from backup media. The **nsrck** program is restartable at any time during its execution. Therefore, it can survive system crashes or exhaustion of resources without losing data.

Each time the NetWorker server starts, it runs **nsrck -L 1** to perform a fast and efficient check for each of the configured client file indexes. Only the consistency of the index header and journal files are checked. It is generally not necessary (and very time consuming) to check every record and key file in the client file index at startup. The program **nsrim** will automatically invoke **nsrck -L 3** after updating the save set's browse and retention times in the media database to remove client file indexes that have exceeded the retention policy. If a problem is detected, a more thorough check will be automatically performed on client file index in question.

If you believe an index may be corrupt, you can manually run a higher level check on the index, for example:

nsrck -L 6

OPTIONS

−**C** This option validates the client's online file index header. It is identical to specifying the −**L 1** option.

−**c** This option is the same as using −**L 2.**

−**F** This option is the same as using −**L 2.**

−**t** *date* Recover the index as of the specified *date* (in **nsr_getdate(3)** format). This option is only valid with the −**L 7** option.

−**T** *tempdir*
Specifies a different directory to use for conversion. This is useful if your client indexes are on file systems that are nearly full. It will enable the conversion to use the tempdir specified as a work space for converting indexes. It is not recommended to use /tmp, since its contents are lost if your machine is rebooted.

−**L** *level*
Specifies the level of checking to use. The valid levels are 1-7.

−**M** Master mode (not advised for manual operation). This advises **nsrck** that it is being run by **nsrd**(1m) or another NetWorker daemon and should log messages with timestamps, and perform any other behavior expected by **nsrd**.

−**m** Invokes consistency checks to detect and remove inconsistent records from the media database. If inconsistent records are detected, the occurences will be recorded in the daemon log. If inconsistent save set records are detected and removed, then **nsrck -X** should be run to remove the associated index records from the client's online file index.

This option must only be run when the NetWorker server is idle, as the media database will be unresponsive while performing the consistency checks. This option performs the same operations that are invoked at startup after an improper media database shutdown is detected, namely:

1) A checksum verification is performed on every record in the media database

to verify record corruption has not occurred.

2) All records from previous media database versions will be upgraded to the current media database record format.

3) The client id map records are checked for unique identifiers and names.

4) Each client resource is then checked to verify a client id map record exists in the media database for the client resource.

5) Each save set record is checked for a valid client entry.

6) The save set records are then checked for valid and unique record identifier fields.

7) The volume records are then checked for unique record identifier and name fields.

8) Save sets records are checked to ensure each (continuation) save set reference exists in the media database.

9) Save sets records are checked to ensure that each volume reference exists in the media database.

10) The volume records are then checked to ensure all the save set references exist in the media database.

−n This option should only be used with the **-m** option. It is used to only report consistency errors in the media database, without repairing or removing the inconsistent entries.

−q Quiet mode. All advisory messages are suppressed.

−v Verbose mode. Advisory messages are emitted.

−R Removes the index for the client. This is valid only when the **−Y** option is also specified. If the **nsrck** is not in master mode, the user will be prompted with a warning indicating which online file indexes will be completely removed and given an opportunity to kill the command if this was not what the user intended.

−X This is the same as using **−L 3**

−x *percent*
This is the same as using **−L 1.** The "percent" value is ignored, but permitted. This allows customer scripts using this option to continue working.

−Y Used in conjunction with **−R** to remove online file indexes. Using this flag means that you really do wish to remove the online file index(es). If you fail to use this flag with the **−R** option, you will be warned that you need to add the **−Y** flag to the **nsrck** command.

FILES **/nsr/index/ clientname /db6/nsrck.lck**
nsrck locks this file thereby insuring that only one copy of **nsrck** is checking a client's index.

/nsr/index/*clientname*

/nsr/index/*clientname***/db6**

SEE ALSO **nsr_layout**(5), **nsr_policy**(5), **nsr_render_log**(1m), **hosts**(5), **nsr**(1m), **nsrd**(1m), **nsrindexd**(1m), **nsrmmdbd**(1m), **nsrim**(1m), **savegrp**(1m)

DIAGNOSTICS

checking index for *clientname*
Informative message that the files associated with the named client are being inspected.

WARNING no valid savetimes - cross-check not performed for *clientname*
During a cross-check, no save sets were found for this client. Since this situation can occur during disaster recovery, **nsrck** avoids deleting the entire contents client index.

cross-checking index for *clientname*
Displayed when the **-L 3** option is in effect.

completed checking *count* **clients**
Displayed as the program finishes, provided some form of checking was accomplished.

NAME nsrclone − NetWorker save set cloning command

SYNOPSIS

nsrclone [−**v**] [−**n**] [−**F**] [−**s** *server*] [−**J** *recover storage node*] [−**d** *save storage node*] [−**b** *pool*] [−**y** *retention*] [−**w** *browse*] [−**R**] [−**m**] { −**f** *file* | *volname...* }

nsrclone [−**v**] [−**n**] [−**F**] [−**s** *server*] [−**J** *recover storage node*] [−**d** *save storage node*] [−**b** *pool*] [−**y** *retention*] [−**w** *browse*] [−**R**] [−**m**] −**S** { −**f** *file* | *ssid...* }

nsrclone [−**v**] [−**n**] [−**F**] [−**s** *server*] [−**J** *storage node*] [−**d** *save storage node*] [−**b** *pool*] [−**C** *less than copies in pool*] [−**y** *retention*] [−**w** *browse*] [−**m**] −**S** −**t** *start time* [−**e** *end time*] [−**B** *pool name*]... [−**N** *saveset name*]... [−**l** *level or range*]... [−**c** *client name*]... [−**g** *group name*]...

nsrclone [−**v**] [−**n**] [−**F**] [−**s** *server*] [−**J** *recover storage node*] [−**d** *save storage node*] [−**b** *pool*] [−**C** *less than copies in pool*] [−**y** *retention*] [−**w** *browse*] [−**m**] −**S** −**e** *end time* [−**t** *start time*] [−**B** *pool name*]... [−**N** *saveset name*]... [−**l** *level or range*]... [−**c** *client name*]... [−**g** *group name*]...

nsrclone [−**v**] [−**n**] [−**F**] [−**s** *server*] [−**J** *recover storage node*] [−**d** *save storage node*] [−**b** *pool*] [−**y** *retention*] [−**w** *browse*] [−**R**] [−**m**] −**V** { −**f** *file* | *volid...* }

nsrclone [−**v**] [−**s** *server*] [−**J** *storage-node*] [−**y** *retention*] −**P** −**W** **volname**

DESCRIPTION The **nsrclone** program makes new copies of existing save sets. These copies are indistinguishable from the original, except for the volume(s) storing the copies. The copies are placed on different media volumes, allowing for higher reliability than a single copy provides. The copies may be made onto any kind of media (for example, save sets on an 8mm tape may be copied to a set of optical disks). However, all media used as the destination of an **nsrclone** operation must be in a *clone pool*. See **nsr_pool**(1m) for a description of the various pool types.

Although the command line parameters allow you to specify volume names or volume identifiers, **nsrclone** always copies complete save sets. Save sets that begin on a specified volume will be completely copied, so volumes may be requested during the cloning operation in addition to those specified on the command line. Conversely, save sets residing on the specified volumes that begin elsewhere are not cloned.

Note that **nsrclone** does not perform simple *volume duplication*, but rather, copies full save sets to a set of destination volumes in a given pool. If the first destination volume chosen cannot hold all of the save sets to be copied, another volume will be chosen. This allows you to use different kinds of media for each copy, allowing for variable sized volumes, such as tapes.

The **nsrclone** program, in conjunction with **nsrmmd**(1m), guarantees that each save set will have at most one clone on a given volume. When you specify a volume name or identifier, the copy of the save sets on that volume are used as the source. When save sets are specified explicitly, those with existing multiple copies are automatically chosen (copies of save sets that exist on volumes in a jukebox are chosen over those that require operator intervention). You can also specify which copy (clone) of a save set to use as the source, (see the −**S** option,in the options section).

Cloning between storage nodes is accomplished by an **nsrmmd**(1m) on the source node reading from a volume, and another **nsrmmd**(1m) on the target node writing to a volume. The source node is determined by the location of a source volume, which is given by where the volume is currently mounted or by its "location" field if unmounted (see **mmlocate**(1m)). The target storage node of a clone is determined by the "clone storage nodes" attribute of the source storage node's client resource, the "clone storage nodes", or the "storage nodes" attribute of the server's client resource in descending priority. Please note that nsrclone never looks at the clone storage node

affinity of the clients whose savesets are being cloned. See **nsr_storage_node**(5) and **nsr_client**(5) for additional detail on how these attributes are used and on other storage node information.

The -c, -N, -l & -g criteria options select candidate savesets for cloning and are intended to behave like the equivalent mechanisms in **mminfo**(1m).

The **nsrclone** program can also be used to clone NDMP (Network Data Management Protocol) save sets. If the save set to be cloned was backed up by nsrndmp_save via *nsrdsa_save* (Data Server Agent program where the save set's flags have 'N' and 's'), then the save set can be cloned to any NetWorker storage device other than an NDMP tape device. See **mminfo**(1m) for more details on the 'N' and 's' save set flags. Refer to **nsrndmp_save**(1m) for more information regarding NDMP backup. Non-DSA NDMP save sets can only be cloned to NDMP tape devices. Cloning from a non-NDMP tape device to an NDMP tape device, and vice-versa, is not supported.

When -m option is specified, **nsrclone** program is used to migrate (or stage) existing save sets on a manual basis. Migration is the process of moving one or more save sets between storage volumes. The process begins by making a clone of the specific save sets to the new volume specified, and then deleting the cloned save set entries from the media database (see the **-S** description). Finally, the save sets will be removed from the original source volumes. The second and the third operations are triggered by the successful completion of the previous operation. The data is moved to new media volumes, making room for new data on the original volumes.

Migration can be onto any media type (for example: save sets on a disk family volume can be migrated to a tape or disk volume). The **nsrclone** program supports save set and volume migration from disk family volumes.

For migration operation, if the **nsrclone** program encounters an error after successfully cloning some of the specified save sets, then it will delete only those successful save sets from the source volume before it gets aborted. Concurrent migration operation from RW and RO volumes of Data Domain and adv_file are not supported.

OPTIONS

−**b** *pool* Specifies the media pool to which the destination clones should be sent. The pool may be any pool currently registered with **nsrd**(1m) that has its status set to *clone*. The possible values can be viewed in NetWorker Management Console by clicking Media from the Administrator window, then selecting Media Pools from the left pane. If you omit this option, the cloned save sets are automatically sent to the *Default Clone* pool.

−**B** *pool name*
If a pool name is specified, only the save sets belonging to that pool will be selected. More than one pool name can be specified by using multiple -B options. This option can only be used with the -t or -e options.

−**C** *less than copies in pool*
Specifies the upper non-inclusive integer bound such that only savsets with a lesser number of clone copies in the target clone pool will be considered for cloning. Note that since the target is a clone pool, each saveset's original copy is never considered when tallying the saveset's number of copies. Likewise, any AFTD read-only mirror clone is not considered, as its read/write master clone will be already counted and there is only one physical clone copy between the related clone pair. This option can only be used with the -t or -e option.

−**f** *file* Instructs **nsrclone** to read the volume names, volume identifiers or save set identifiers from the file specified, instead of listing them on the command line. The values must be listed one per line in the input file. This option cannot be

specified with -t and -e options. The *file* may be "-", in which case the values are read from standard input. The cloning operation begins only when all the entries in the file are correctly specified; even if one of the entries is invalid, the operation will not continue and the corresponding error is reported.

−**R** Removes the input file that specifies the volume names, save set or volume identifiers to be cloned/staged. This option can only be specified with a **-f** option.

−**F** If specified will force nsrclone to skip all invalid savesets/volumes and continue cloning.

−**s** *server*
Specifies a NetWorker server to migrate save sets from. See **nsr**(1m) for a description of server selection. The default is the current system.

−**J** *recover storage node*
Specifies which host to use as the storage node for the recovery part of the cloning process (see nsr_storage_node(5)). The host specified must be included in "recover storage nodes" or "storage nodes" attribute of server's client resource. See **nsr_client**(5) for more information.

−**d** *save storage node*
Specifies which host to use as the storage node for the save part of the cloning process (see nsr_storage_node(5)).

−**v** Enable verbose operation. In this mode, additional messages are displayed about the operation of **nsrclone**, such as save sets that cross volumes, names of cloned volumes, or save set series expansions. If concurrent **nsrclone** operations are performed on the same save sets, it is possible for the volume names to be inaccurate. In that case **nsrclone** will issue a warning. Please see **DIAGNOSTICS** for the exact warning message.

−**y** *retention*
Sets the date (in **nsr_getdate**(3) format) when the cloned data will become recyclable. The special value **forever** is used to indicate that a volume that never expires (i.e. an archive volume) must be used. By default, the server determines this date for the save set based on the retention policies in effect. This option allows overriding the existing policies. See also **nsrmm**(1m) and its -S & -e options.

−**w** *browse*
Sets the date (in **nsr_getdate**(3) format) when the cloned saveset will become non-browsable. However, a saveset's existing browse policy is left unchanged if it is later than the intended time or if it has already passed, i.e. the saveset has become non-browsable and is in the purged-from-index state. This option requires the -y retention option and must not be greater than the retention time. See also **nsrmm**(1m) and its -S & -w options.

−**S** Causes **nsrclone** to treat subsequent command line parameters as save set identifiers, not volume names. Save set identifiers are unsigned numbers. You can find out the save set identifier of a save set using the **mminfo -v** command (see **mminfo**(1m)). The −**S** option is useful when you want to copy individual save sets from a volume or all save sets matching an **mminfo** query (see the examples below). The save set identifiers may also specify exactly which copy of a save set with multiple copies to use as the source. To specify exact copies, use the *ssid/cloneid* format for each save set identifier. In this case, the *ssid* and the *cloneid* are unsigned numbers, separated by a single slash (/). You can find out the *cloneid* for a particular copy by using the **mminfo -S** report, or a custom report.

−**V** Causes **nsrclone** to treat subsequent command line parameters as volume identifiers (volid), not volume names. Volume identifiers can be found using the **mminfo -mv** report, for example. This option can not be used in conjunction with the −**S** option.

Only one -V needs to be specified with multiple volids. Multiple -V's may cause this command to fail, see the CAVEATS section.

−**n** Do **not** execute. This option causes nsrclone to generate and print out the list of savesets to be cloned but not actually perform the operation. The list is newline terminated and the ids are of the form: ssid/cloneid.

−**N** *saveset name*
Specifies the saveset name for savesets that will be considered for cloning.

−**l** *level or range*
Specifies the level or n1-n2 integer range from 0 to 9 for savesets that will be considered for cloning. Use "manual" for ad-hoc (client-initiated) savesets, "full" for level full savesets, "incr" for level incremental savesets, integers 0 through 9 (where 0 also means full), etc. More than one level can be specified by using multiple -l options and/or the -l n1-n2 range format. This option can only be used with the -t or -e option.

−**e** *end time*
Specify the end time (in **nsr_getdate**(3) format) for selecting save set IDs for cloning. This option can only be used with the -S option. If not specified, end time is set as current time. Please note that, -e 0 is same as -e today.

−**t** *start time*
Specify the start time (in **nsr_getdate**(3) format) for selecting save set IDs for cloning. This option can only be used with the -S option. If not specified, start time is set as end time - 24 hours. Please note that, -t 0 is same as -t today. When specifying a time range, at least -t or -e option must be specified.

−**c** *client name*
If client name is specified, only the save sets belonging to that client will be selected. More than one client name can be specified by using multiple -c options. This option can only be used with the -t or -e option.

−**g** *group name*
If a group name is specified, only the save sets belonging to that group will be selected. More than one group name can be specified by using multiple -g options. It can be used with -c option. This option can only be used with the -t or -e option.

−**m** Performs the actual migration (or stage) operation.

−**P** Instructs **nsrclone** to perform cleaning operation for one volume. Scans a volume for save sets with no entries in the media data base and recovers their space. Space for recyclable and aborted save sets are also recovered from the volume with the save set entries removed from the media data base. You can perform this operation on disk family volumes. This option must be specified with a **-W** option.

−**W** *volname*
Specifies the name of the volume to be cleaned. This option cannot be used with −**S** or −**m** options.

CAVEATS On Linux, this command will fail when volids are specified with multiple -V's. This behavior can be changed by setting POSIXLY_CORRECT environment variable.

EXAMPLES

Copy all save sets that begin on the volume **mars.001** to a volume in the **Offsite Clone** pool:

nsrclone −b 'Offsite Clone' mars.001

Copy all complete save sets created during the weekend. If no time of day is specified with the date, **nsr_getdate (3)** uses midnight as the start time for copying all the complete save sets. Only complete save sets can be copied by **nsrclone**(1m):

**nsrclone -S 'mminfo −r ssid **
-q '!incomplete,savetime>last saturday,savetime<last monday'‘

Copy a specific clone of a specific save set:

nsrclone -S 1538800517/770700786

Copy all save sets created between time 01/21/05 14:50:03 and 01/24/05 14:50:03 for the group Default

nsrclone -S -t '01/21/05 14:50:03' -e '01/24/05 14:50:03' -g Default

Copy all save sets created in the last 24 hours for clients "rose" and "seam":

nsrclone -S -e now -c rose -c seam

Copy all save sets created in the last 24 hours for clients "rose" and "seam" with saveset names "/data1" and "/data2" for backup level "full" only:

nsrclone -S -e now -c rose -c seam -N /data1 -N /data2 -l full

Copy all save sets that were not copied to the default clone pool in a prior partially aborted nsrclone session, assuming no copies existed prior to that aborted session:

nsrclone -S -e now -C 1

As in the preceding but with extended retention and browse periods:

nsrclone -S -e now -C 1 -y 12/12/2010 -w 12/12/2010

Migrate all save sets from the volume **mars.101** and **jupiter.101** to a volume in the **Offsite Clone** pool:

nsrclone -m −b 'Offsite Clone' mars.101 jupiter.101

Migrate save sets **1234 and 4568** to a volume in the **Offsite Clone** pool:

nsrclone −b 'Offsite Clone' -m -S 1234 4567

Migrate clone instance **12345678 of save set 1234** to a volume in the **Default Clone** pool:

nsrclone −m −S 1234/12345678

Migrate all save sets created since last Saturday to a volume in the **Default Clone** pool:

**nsrclone −m −S 'mminfo −r ssid **
-q 'savetime>last saturday'‘

Recover space from volume **jupiter.013**:

nsrclone −P −W jupiter.013

Only complete save sets can be migrated by **nsrclone**(1m).

SEE ALSO

nsr_getdate(3), **nsr_client**(5), **nsr_device**(5), **nsr_pool**(5), **nsr_stage**(5), **nsr_storage_node**(5), **mminfo**(1m), **nsr**(1m), **nsrd**(1m), **nsrmmd**(1m), **nsrndmp_save**(1m)

DIAGNOSTICS

The exit status is zero if all of the requested save sets were cloned or migrated successfully, non-zero otherwise.

Several messages are printed signaling that **nsrd**(1m) is unavailable for cloning data; these are self-explanatory. You may also see a message from the following list.

adding save set series which includes parent *ssid*

If running in verbose mode, this message is printed when **nsrclone** notices that a requested save set is continued, requiring the entire series to be cloned (even if only part of the series was specified in the command line parameters).

adding save set series which includes descendent *ssid*
If running in verbose mode, this message is printed when **nsrclone** notices that a requested save set is a continuation, requiring the entire series to be cloned.

Cannot contact media database
The media database (and most likely other NetWorker services as well) on the named server is not answering queries. The server may need to be started, or if it was just started, it needs to finish its startup checks before answering queries.

cannot clone save set *number*, **series is corrupt**
The given save set is part of a save set series (used for saving very large files or filesystems), but not all of the save sets in the series were found in the media database. This can happen if, for example, you relabel a tape that contains part of a save set series.

cannot clone backup and archive data together
Archive and backup data is fundamentally different and cannot be cloned to the same pool. You need to run **nsrclone** twice, once to clone the backup save sets and once more for the archive save sets.

cannot open nsrclone session with *server*
This message is printed when the server does not accept clone sessions.

cloning not supported; upgrade required
Another enabler is required to use this feature.

cloning requires at least 2 devices
Cloning requires at least one read/write device and one read-only or read/write device, since data is copied from one volume directly to another.

server **does not support cloning**
The named server is not capable of cloning.

each clone host needs at least two enabled devices
When cloning between two storage nodes that share the same physical drive, each node must have at least two enabled devices.

error, no valid clones of ssid *number*
The listed save set exists, but cannot be cloned because there are no complete copies of the save set. The save set was either aborted or is in progress. Only complete save sets can be copied.

error, user *username* **needs to be on administrator list**
Only NetWorker administrators are allowed to make clones of backup save sets. NetWorker administrators are listed in the NSR server resource, see **nsr_service**(5) for more information. For servers with archive capability, users listed in the NSR archive client's user list are allowed to clone archive save sets, as long as they have the "Monitor NetWorker" privilege; users listed in the NetWorker administrator list will also be able to clone archive save sets.

no complete save sets to clone
Only complete save sets can be copied, and no complete save sets were found matching the requested command line parameters.

number **is not a valid save set**
The given save set identifier is not valid. Two forms are understood: simple save set identifiers and those with a cloneid specified. Simple save sets are unsigned numbers. The save set with the cloneid form is specified as two unsigned numbers separated by a single slash (/).

pool is not a cloning pool
The pool specified with the −**b** *pool* option is not a clone pool. You must

always use a pool with a type of "Backup Clone" or "Archive Clone" for the −**b** option.

save set *number* **does not exist**
The given save set (from a −**S** save set list) does not exist. Verify your save set identifiers using **mminfo**(1m).

save set *number* **crosses volumes; requesting additional volumes**
This message is printed in verbose mode when volume names or IDs were specified, but the given save set is only partially resident on the listed volumes. Since only complete save sets can be cloned, **nsrclone** automatically requests additional volumes.

save set clone *number/cloneid* **does not exist**
A specific clone of a save set was specified, but that save set has no clones with that clone identifier. Verify your save set identifiers using **mminfo**(1m).

volume *name-or-number* **does not exist**
The given volume (either a volume name or a volume id specified in the −**V** option) does not exist in the media database.

Cannot find volume *name* **in media data base**
The volume name specified in the −**W** option does not exist in the media database.

waiting 30 seconds then retrying
A temporary error occurred and **nsrclone** will automatically retry the request until the condition is cleared. For example, an error will occur if all devices are busy saving or recovering and **nsrclone** must wait for these devices become available.

WARNING: Multiple concurrent cloning operations on the same savesets have been detected. The list of volumes reported below may not be accurate.
nsrclone prints this message when it detects more clone instances than it expected. This happens when more than one nsrclone commands are run on same save set concurrently. Verify the clone volumes using **mminfo**(1m). Please note that the result of the clone operation is not affected by this warning.

Space can only be recovered from disk family
devices.
The given volume (if you specified the −**W** option) is not a disk family volume. This message is also printed after a successful migration of data from volumes of type other than a disk family device.

NAME nsrcpd − daemon providing remote client software installation services

SYNOPSIS **nsrcpd** [−**d** *debug-level*]

DESCRIPTION The **nsrcpd** daemon provides an RPC-based remote client software installation service. This service allows users to distribute and upgrade client software from a centralized software repository across a network. In addition, users can manage the centralized software repository and inventory existing NetWorker clients for currently installed software. The RPC program number provided by **nsrcpd** is 390437.

Normally, **nsrcpd** is invoked by nsrd upon receiving a request to start up the remote client software installation service and does not need to be started directly by a user.

The main thread for **nsrcpd** handles all RPC messages for the service. Each time a remote client installation operation begins, **nsrcpd** creates a new session and spawns a new thread to process that operation. The operation thread receives all data for the session from the main RPC thread and handles any user dialogs and processing for the operation. The operation thread automatically exits when a session completes. **Nsrcpd** automatically terminates after it has been idle for a pre-determined period.

Nsrcpd maintains a set of resources reflecting the current set of software products located in the centralized software repository, as well as the current set of products and packages installed on NetWorker clients in the datazone where it is running. These resources are managed during repository, inventory, and upgrade operations.

When **nsrcpd** is started with the −**d** <**debug level**> option, it runs with the requested debug level.

OPTIONS −**d** *debug-level*
Instructs **nsrcpd** to start up in debug mode with the requested debug level.

FILES **/nsr/logs/daemon.raw**
The file to which **nsrcpd** and other NetWorker daemons send information about various error conditions that cannot otherwise be logged using the NetWorker event mechanism.

/nsr/res/cpdb
Information describing the remote client installation service and its resources.

SEE ALSO **nsr**(1m), **nsrpush**(1m), **nsr_render_log**(1m)

NAME

nsrd − daemon providing the NetWorker service

SYNOPSIS

nsrd [−**k** *virtual-service-name*]

ansrd [**commentary**]

DESCRIPTION

The **nsrd** daemon provides an RPC-based save and recover service. This service allows users to save, query for, and recover their files across a network. The RPC program number provided by **nsrd** is 390103.

Normally **nsrd** is invoked from a startup shell script (for example *rc.local*, *rc.boot*) at boot-time, and should never need to be started directly by a user. After it is started, **nsrd** starts up the other daemons it needs to provide the NetWorker service.

The **nsrd** command must be run on a machine with appropriate resources. These resources include devices (for example, tape drives) which are under the control of the media multiplexor software (see **nsrmmd**(1m)), and sufficient disk space for the index daemons, (see **nsrindexd**(1m) and **nsrmmdbd**(1m)) to maintain the index of saved user files and volumes with corresponding files.

Each time a backup, recover, or another session begins, **nsrd** starts the program, **ansrd**, to process the requested session. The **ansrd** program is called an *agent*. The agent is in charge of monitoring that backup, recover, or another session, and automatically exits when a session completes. Using **ps**(1) or another process monitoring tool, you can inspect the subsequent parameters of **ansrd** to see what kind of session it is monitoring. If necessary, agents can be forcibly terminated to abort a backup or recover session. Agents cannot be run directly; they can only be started by **nsrd**.

When **nsrd** is started with the −**k** option, it checks to see whether it has been installed as a cluster service and that the virtual host which owns **/nsr/res** matches *virtual-service-name.* If either of these validation steps fails, **nsrd** exits immediately. (To check whether NetWorker has been installed as a cluster service, **nsrd** checks for a file called **NetWorker.clustersvr** in the directory containing the **nsrd** binary. To check that **/nsr/res** is owned by *virtual-service-name*, **nsrd** queries the cluster management software.)

If the −**k** option is not used when starting NetWorker in a cluster, the server assumes the identity of the virtual host which owns **/nsr/res.** If no virtual host owns **/nsr/res**, then **nsrd** will not start.

OPTIONS

−**k** *virtual-service-name*

Instructs **nsrd** to start up in cluster failover mode using *virtual-service-name* as its hostname/identity. This option is used by the NetWorker cluster control script which starts NetWorker.

FILES

/nsr/logs/daemon.raw

The file to which **nsrd** and other NetWorker daemons send information about various error conditions that cannot otherwise be logged using the NetWorker event mechanism.

/nsr/res/nsrdb

Information describing the NetWorker service and its resources (See **nsr_service**(5)).

NetWorker.clustersvr

If this file exists in the directory containing NetWorker's daemons, it indicates that the NetWorker server has been installed as a cluster service.

SEE ALSO **nsr**(1m), **nsr_service**(5), **nsr_render_log**(1m), **nsrmmd**(1m), **nsrmmdbd**(1m), **nsrindexd**(1m), **ps**(1), **rc**(1m)

NAME nsrexec − remotely execute NetWorker commands on NetWorker clients

SYNOPSIS **nsrexec**

DESCRIPTION The **nsrexec** command is run only by other NetWorker commands. It is used to remotely execute commands on NetWorker clients running **nsrexecd** and monitor the progress of those commands.

SEE ALSO **nsr**(5), **nsr**(1m), **nsrexecd**(1m), **savegrp**(1m)

NAME nsrexecd − NetWorker client execution service

SYNOPSIS **nsrexecd** [−**s** *server* [−**s** *server* ...]] [−**f** *serverfile*] [−**p** *savepath*] [−**i**] [−**r**]

DESCRIPTION **nsrexecd** is used by NetWorker servers to perform automatic operations on NetWorker clients. It is currently used by **savegrp**(1m) to start saves and storage node functions on NetWorker client machines. When storage node functions are in use, **nsrexecd** starts **nsrmmd**(1m) daemons and **nsrjb**(1m) commands on the host, and responds to polling requests from the server. See **nsr_storage_node**(5) for additional detail on storage nodes. The **nsrexecd** service is normally started at boot time on each NetWorker client machine. Since NetWorker servers are usually expected to be clients of themselves, **nsrexecd** runs on all NetWorker servers as well.

The **nsrexecd** service exports an RPC-based service to remotely execute NetWorker operations. All requests must be authenticated, and can optionally be restricted to specific NetWorker servers. Only **save** requests (for example, **save**(1m) or **savefs**(1m)) and storage node requests are allowed.

When command execution is requested, **nsrexecd** first verifies that the request is authenticated, and that it comes from a valid NetWorker server; the NetWorker server running on the localhost is always considered valid, independent from the options supplied to **nsrexecd**. Next, **nsrexecd** verifies that the command is a save command (for example, **save**(1m)). It then executes the specified command from the NetWorker binary directory. This directory is normally determined by the location of the **nsrexecd** executable, but can be specified on the command line.

OPTIONS

−**i** As part of the NetWorker server authentication, the server's network address is mapped to a name. The name is then reverse-mapped to a network address. The server is authenticated if and only if the original network address matches the reverse-mapped address. The −**i** flag skips the address comparison thereby allowing workarounds to misconfigured or misfeatured naming systems. This option should be used with care since it may allow the NetWorker client to send its data to an unauthorized machine.

−**f** *serverfile*
Specifies a file containing a list of NetWorker servers which can initiate saves. This file should list one server name per line. If no −**f** or −**s** options are specified, **nsrexecd** looks for a default file in this same format (or Mac preferences on the Mac client). The location of this default file is listed in the **FILES** section of this man page.

−**p** *savepath*
Tells **nsrexecd** to look for save commands in the *savepath* directory, rather than the default (the directory in which **nsrexecd** exists).

−**s** *server*
Allows save requests to be initiated only by the given NetWorker server. Multiple −**s** options may be given to allow access by several NetWorker servers. If a NetWorker server has multiple network interfaces, it is often best to list the hostname corresponding to each network interface, to avoid failed saves.

−**r** This option should be used with EMC PowerSnap module only. This option, would start another instance of nsrexecd for PowerSnap administration Purposes.

FILES

/nsr/res/nsrladb	The resource directory with attributes describing the NetWorker **nsrexecd** service and its resources (see **nsrla**(5)).
/nsr/res/servers	The file containing the default list of servers that can back up the NetWorker client.

SEE ALSO **nsrla**(5), **nsr_storage_node**(5), **nsrports**(1m), **save**(1m), **savefs**(1m), **savegrp**(1m)

NAME nsrim − NetWorker index management program

SYNOPSIS **nsrim** [−**c** *client*] [−**N** *saveset*] [−**V** *volume*] [−**lnqvMXC**]

DESCRIPTION The **nsrim** program is used to manage the NetWorker online file and media indexes. Normally, **nsrim** is invoked by **savegrp**(1m) command on completion, and by **nsrd**(1m) when *Remove oldest cycle* is selected from the NetWorker Administrator program. **nsrim** is not normally run manually. However, the command's modes can be modified such that normal users may run the command while retaining root privileges; see **nsr**(1m) for more details.

When the **savegrp**(1m) command launches **nsrim** at the end of its task, it checks the timestamp of the file /nsr/mm/nsrim.prv. If the timestamp of this file is greater than or equal to 23 hours, nsrim marks all save sets that are passed their browse and retention policy as *recyclable.* If save sets need to be monitored for their browse and retention policy more frequently (for example, if **savegrp**(1m) is run more frequently than every 23 hours), **nsrim -X** should be set up as a **cron**(1m) entry, or should be run manually.

nsrim uses *policies* to determine how to manage online entries. (See **nsr_policy**(5), **nsr_client**(5), and the *NetWorker Administrator's Guide* for an explanation of index policies). Entries that have been in an online file index longer than the period specified by the respective client's browse policy are removed. Save sets that have existed longer than the period specified by a client's retention policy are marked as *recyclable* in the media index. When all of the save sets on a volume have been marked recyclable, then the volume is considered recyclable. Recyclable volumes may be selected by NetWorker (and automatically relabeled by a jukebox) when a writable volume is needed to hold new backups. When a recyclable volume is reused, the old data is erased and is no longer recoverable. Space for recyclable and aborted save sets of a disk family volume (see **nsr_device**(5)) is removed from the volume (on disk) with the save set entries deleted from the media index and the data in these save sets will no longer be recoverable.

Unless the −**q** option is used, **nsrim** prints header and trailer information for each group of save sets. The header lists the save set type, the client name, the save set name, and the applicable browse and retention policies that apply to the save set. (See the example in this man page). There are four types of save sets:

Normal All save sets backed up automatically using **savegrp** that are associated with a schedule, a browse policy, and a retention policy.

Ad hocs User-initiated save sets are designated by appending *ad hocs* to the header line.

Archives
Save sets that never expire automatically are designated by appending *archives* to the save set line.

Migrations
Save sets that never expire automatically, and were created by a file migration application, are designated by appending *migrations* to the save set line.

The trailer lists four utilization statistics of the save set after **nsrim** has applied the policies to it. The four statistics are the total number of browsable files remaining in the online index, the total of files currently associated with the save set, and the amount of recoverable data out of the total of data associated with the save set. For example, **nsrim** may print the following output for one save set name:

```
mars:/usr, retention policy: Year, browse policy: Month, ad hocs
8481 browsable files of 16481 total, 89 MB recoverable of 179 MB total
```

```
mars:/usr, retention policy: Year, browse policy: Month, ad hocs
0 browsable files of 13896 total, 163 MB recoverable of 163 MB total

mars:/usr, retention policy: Year, browse policy: Month 43835
browsable files of 427566 total, 6946 MB recoverable of 7114 MB total
```

When the **−v** option is used, the following information is also printed for each save set: the save set id, creation date, level, file count, size, and status. A save set's status is one of the following:

browse The file entries for the save set are browsable (the save set files still exist in the online index). These files are easily restored using the NetWorker recover mechanisms.

recover The age of the save set does not exceed the retention policy for the save set, but its entries have been purged from the NetWorker online index. This means that save set is recoverable from the backup media using **recover**. (See **recover**(1m).) **scanner**(1m) may be also be used to recover the save set, but users should use **recover** first.

recycle The save set is older than its associated retention policy and may be overwritten (deleted) once its backup media is recycled. Until the media is recycled, the save set is also recoverable from the backup media. Recyclable save sets of disk family (see **nsr_device**(5)) volumes will be removed from the volumes and media database, the data in these save sets will no longer be recoverable.

delete The save set will be deleted from the media database. **nsrim** deletes only recyclable save sets that have zero files.

The save set status may be followed by any of the following modifiers:

(archive)
The save set never expires, and is exempt from any status change.

(migration)
The save set was created by a file migration application and never expires, and is exempt from any status change.

(scanned in)
The save set was restored using the **scanner** command, and is exempt from any status change.

(aborted)
A save set of questionable size, consuming backup media space.

If **nsrim** changes the status of a save set, then it prints the transition symbol −> followed by the new status. For example:

```
17221062  3/05/92  f  23115 files   158 MB recycle
17212499  3/19/92  f    625 files    26 MB recover(aborted)->recycle
17224025  5/23/92  i      0 files     0 KB recover->recycle->delete
17226063  6/05/92  f   3115 files    58 MB recover
17226963  6/09/92  f   3197 files   114 MB browse->recover
17227141  6/10/92  f   3197 files   115 MB browse
```

Once **nsrim** has processed all of the save sets, it flags the file index for cross-checking in **nsrindexd**(1m). If the **−l** flag is specified, the cross-check is attempted synchronously, otherwise, it is simply scheduled and **nsrindexd** performs the cross-check when the index is idle. At the same time, **nsrim** processes the status of any affected NetWorker volumes. With the absence of the **−q** flag, a line is printed for each

volume. The line includes the volume name, the amount of space used, the total number of save sets, and the status. The status will be one of the following:

appendable
: More save sets may be appended to the volume. The status may also be modified with *(currently mounted)* which signifies that the volume could transition to the *recyclable* state if it was not mounted for writing.

read-only, *full*
: No more save sets can be appended to the volume, nor can the volume be reused since it contains some valuable save sets.

recyclable
: No more save sets can be appended to the volume, and all save sets on the volume have expired.

In addition, the following modifier applies to all three of these states:

(manual-recyclable)
: The volume will not be automatically eligible for recycling when all of its save sets have expired. Instead, the volume may only be recycled by a manual relabel operation. Note that a *read-only* volume can still be recycled unless the *manual-recyclable* flag is also set. The manual-recyclable flag can be set using NetWorker Management Console or the **nsrmm**(1m) and **nsrjb**(1m) commands when volumes are labeled or at any time thereafter. This flag is never set automatically.

If the volume status changes, then **nsrim** appends −>**recyclable** to the status. If the volume contains some browsable save sets, then this is noted; recoverable save sets are also noted. The odd case where an appendable volume has only recyclable save sets is also noted. For example:

```
jupiter.20: 3474 MB used, 398 save sets, full->recyclable
jupiter.21: 4680 MB used, 440 save sets, full, 249 recoverable
jupiter.22: 4689 MB used, 351 save sets, full, 351 browsable
jupiter.24: 1488 MB used, 141 save sets, appendable, 141 browsable
```

RETENTION AND BROWSE POLICIES

Under normal circumstances, the association between browse or retention policies and client save sets is obvious. However, since a save set may be listed by more than one client resource with the same name, and each client resource may specify different browse and retention policies, determining the policies applicable to a save set is not always straight forward. **nsrim**(1m), uses the following steps to select an instance of a client resource with the client's name. Once the client resource is selected, its browse or retention policy is used for managing information about the save set.

1) Locate all the client resources which belong to the same group that the save set belongs to. Within this set of client resources, apply the following rules to get the best match. If no client resource belongs to the save set's group, or if the group no longer exists, or if the saveset is from a backup earlier than version 5 (when group information was not recorded in the save set), apply the following rules to all the client resources to get the best match.
2) Locate a client resource explicitly listing the save set. If more than one client resource lists the save set, choose the client resource with the longest policy.
3) Search for a client resource listing the save set "All". If more than one client resource lists the save set "All", choose the client resource with the longest policy.
4) Find the client resource listing a save set with the most common prefix (longest) of the target save set. If more than one client resource lists the save set with the

most common prefix, choose the client resource with the longest policy.

5) Among all of the client resources, choose the client resource with the longest policy.

Note that if two or more client resources with the same name exist, it is possible that the browse policy from one instance of the client resource and the retention policy of another instance may be used for managing save set information.

Save sets that have no corresponding NetWorker client resource use the NetWorker client resources of the server to determine the browse or retention policies.

A save set cannot be purged from the index or marked for recycling until all of its dependent save sets are also eligible for purging or recycling. See the *NetWorker Administrator's Guide* for an explanation of dependent save sets.

The last (and only) Full save set will not be purged from the online index until it is also marked for recycling. In this case, the header line of the save set omits the browse policy and prints a message stating that only one browsable cycle exists.

With the exception of the **−l** option, manual ad hoc save sets are treated as full save sets that have no dependents. However, unlike true Full save sets, the last manual save set is not given any special consideration with regard to index purging.

The retention time applied to save sets is rounded up to midnight when the elapsed time implied by the policies is greater than or equal to a day. Therefore, **nsrim** should produce the same results whether it is run at 8 a.m. or 5 p.m. on the same day.

OPTIONS

−c *client*
Only process the online file index for the specified client. Normally, all client indexes are processed. This option may be repeated to process multiple clients.

−C Force the compression of media database at the end of the nsrim run. Normally nsrim initiates database compression operation, but nsrmmdbd performs the operation based on the standard compression policy. By specifying this option, nsrmmdbd will ignore the standard policy and perform the compression immediately.

−l Removes the oldest full save and all save sets dependant on it from the online index. Browse and retention policies are ignored. The save set header information will print the number of browsable full cycles currently in the online index. *Archive* and *migration* save sets are ignored. With this option, *manual* save sets are treated as normal *incremental* save sets. This option also sets the utilization threshold to 30 percent.

−M Master mode (not advised for manual operation). Advises **nsrim** that it is being run by **nsrd**(1m) or another NetWorker daemon and that it should log messages with timestamps, and perform any other behavior expected by **nsrd**.

−N *save set*
Process only save sets named; all others are skipped. This option can be repeated to process multiple save sets.

−n Do nothing. Instead, emulate the actions of this command without the index cross-check. Note that trailer statistics reflect current (and not emulated) results.

−q Run quietly. This option will not generate header, trailer or save set messages.

−V *volume*
Specifies the name of the volume to be processed. This option can be repeated to process multiple volumes. **−c, −N** and **−l** options are ignored when this

option is specified.

−v Produce a more detailed report. This may produce a large amount of output. When both **−v** and **−q** are issued, they cancel each other.

−X Check the consistency of the data structures of the save set with the data structures of the volume. This is only required after a NetWorker crash.

NOTES

The standard policy for media database compression is to perform the compression every 22.5 days. By compressing the media database, the possibility of performance degradation due to fragmentation in the database can be reduced, but nsrmmdbd will not be available for service for the duration of the database compression. As a result, the -C option should not be used as a default option. The -C option will not used when **nsrim** is started by savegrp. If customer suspects performance degradation due to fragmentation of media database, **nsrim** can be run with -C option to reduce the fragmentation. Please select a time at which the Networker server is not busy to perform the database compression.

FILES

/nsr/tmp/.nsrim
nsrim locks this file to prevent more than one copy of itself from thrashing the media database.

/nsr/mm/nsrim.prv
nsrim updates this file to log the last time that it was started.

DIAGNOSTICS

You are not authorized to run this command
Only root may run **nsrim** to modify the online indexes. However, any user may invoke the command with the **−n** option.

nsrim has finished checking volume <**name**>
This notification message appears in the NetWorker messages window when **nsrim** completes and the command was invoked with the **−V** option.

nsrim has finished (cross) checking the media db
This notification message appears in the NetWorker messages window when **nsrim** completes and the command was invoked without the **−V** option.

SEE ALSO

nsr_client(5), **nsr_layout**(5), **nsr_policy**(5), **nsr**(1m), **nsrd**(1m), **nsrindexd**(1m), **nsrmm**(1m), **recover**(1m), **savegrp**(1m), **scanner**(1m)

NAME nsrindexasm − NetWorker module for recovering indexes

SYNOPSIS **nsrindexasm** [*standard-asm-arguments*]

DESCRIPTION The **nsrindexasm** is a standard, external ASM (Application Specific Module). It assists in the recovery of NetWorker on-line save record index files that were saved with NetWorker versions earlier than version 6.

See **uasm**(1m) for a general description of ASM and the [*standard-asm-arguments*]. It is intended that **nsrindexasm** be invoked by **uasm** during **nsrck**(1m) index recovery operations.

FILES **/nsr/index/***clientname***/db6**
This is directory whose data is recovered by this ASM.

SEE ALSO **nsr_layout**(5), **nsrck**(1m), **nsrindexd**(1m), **mmrecov**(1m), **uasm**(1m)

NAME nsrindexd − NetWorker file index daemon

SYNOPSIS **nsrindexd**

DESCRIPTION The **nsrindexd** daemon is started by the server **nsrd**(1m) daemon. It should not be started manually. The daemon provides an RPC-based service to the server **nsrd**(1m) daemon; direct network access to this service is not allowed. The RPC program and version numbers provided by **nsrindexd** are 390105 and 4, respectively.

The service provided to the NetWorker system is designed for high performance insertion and deletion of save records into indexes. This performance is obtained by keeping information cached in the **nsrindexd** process address space. When the NetWorker system wishes to commit a save session's records, it notifies the **nsrindexd** daemon (via a remote procedure call) to flush its volatile state to its file(s).

Since the daemon (or the server) may crash at any time, the index files may be left in an inconsistent state. Therefore, the maintenance program, **nsrck**(1m) is run automatically by the **nsrd** daemon before the NetWorker service is started.

When the NetWorker service is started, it starts the process nsrindexd which will invoke **nsrck -L 1** to perform a fast and efficient check for each of the configured client file indexes. Only the consistency of the index header and journal files are checked. It is generally not necessary (and very time consuming) to check every record and key file in the client file index at startup. If a problem is detected, a more thorough check will be automatically performed on client file index in question.

If you believe an index may be corrupt, you can manually run a higher level check on the index, for example:

nsrck -L 6

Running **nsrck -L 7** will not overwrite existing files in the client file index. So, if online client file index data already exists for a saveset for a particular save time, it must be removed before **nsrck -L 7** can be used to restore it from the backup media.

Since **nsrindexd** and **nsrck** are run at the same time, both programs use an advisory file-locking mechanism on the file **v6ck.lck** to synchronize their access to an index.

FILES

/nsr/index/*clientname***/db6**
: This directory is where the client's index header file and journal files are stored. The index record files (.rec) and the corresponding key files (.k0 and .k1) are stored in different subdirectories under db6 directory.

/nsr/index/*clientname***/db6/v6hdr**
: This is the name of the index header.

/nsr/index/*clientname***/db6/v6journal**
: This is the name of the journal file.

/nsr/index/*clientname***/db6/v6hdr.lck**
: This is the name of the lock file used for synchronizing access to the index header file and journal file.

nsr_layout(5), **nsr**(1m), **nsrck**(1m), **nsrd**(1m), **nsrim**(1m), **nsrindexasm**(1m), **nsrls**(1m), **nsrmm**(1m)

DIAGNOSTICS

Continuing without index header for client *clientname*
: This is an informative message to indicate that another program is accessing the same file that is required by this daemon. The daemon determined that it can continue its operation safely without the index header.

NAME nsrinfo − NetWorker file index reporting command

SYNOPSIS **nsrinfo** [−**vV**] [−**s** *server* | −**L**] [−**n** *namespace*] [−**N** *filename*] [−**t** *time*] [−**X** *application*] [−**x** *exportspec*] *client*

DESCRIPTION The **nsrinfo** command generates reports about the contents of a client file index. Given a required NetWorker client name and no options, **nsrinfo** will produce a report of all files and objects, one per line, in the *backup* name space for that client. It can also generate reports as follows: for a specific file index name space, for all name spaces at once, or for a particular XBSA application. Reports can also be restricted to a single time (the time at which the entry was entered into the file index, called the *savetime*).

For example, to generate a report of all files backed up in the most recent backup of the */usr* file system for the client *mars*, use the following sequence of commands (assuming the % character is the shell prompt):

```
% mminfo −r nsavetime −v −N /usr −c mars −ot | tail −1
809753754
% nsrinfo −t 809753754 mars
```

Note: The time used in the query is obtained by running the **mminfo**(1m) command with a custom report to print the save time for the most recent save set for */usr.* The time printed is passed to **nsrinfo** along with the name of the client (mars).

OPTIONS

−**v** Verbose mode. In addition to the filename, it prints the type of the file, the internal file index identifier (if any), the size (if a UNIX file), and the savetime. This option may be combined with the −**V** option.

−**V** Alternate verbose mode. In addition to the filename, it prints the offset within the save set containing the file, the size within the save set, the application name space (see the −**n** option for a list of values), and the save time. This option may be combined with the −**v** option.

−**s** *server*
Indicates the name of the NetWorker system to be queried. By default, the server on the local system is queried.

−**L** Opens a file index directly without using the server. This option is used for debugging, or to query the file index while NetWorker is not running.

−**n** *namespace*
Indicates the file index name space to query. By default the *backup* name space is used. The other recognized values are: *migrated*, *archive*, *nsr* (for internal use), *informix* (for INFORMIX data), *sybase* (for Sybase data), *msexch* (for Exchange data), *mssql* (for SQL Server data), *notes* (for Lotus Notes data), *db2* (for DB/2 data), *oracle* (for Oracle data), and *all*. The name space field is case sensitive.

−**N** *filename*
Indicates an exact filename to look for in the file index. Only index entries matching this name exactly print. Note that for some clients, such as NetWare, the name stored in the file index is often not made up of printable ASCII characters, giving this option limited use.

−**t** *time* Restricts the query to a single, exact save time. The time can be in any of the NetWorker **nsr_getdate**(3) formats. Every save set created by NetWorker has a unique save time; these times can be determined by using the **mminfo**(1m) command.

−**X** *application*
Restricts the query to list information for only a specific X/Open Backup

Services (XBSA) application. Valid application types are **All, Informix,** and **None.** The application type is not case sensitive. See the APPLICATION TYPES section of this man page for more information.

−**x** *exportspec*
As an alternative to the default human-readable output format, *exportspec* provides for two styles of program-readable output formats. The *exportspec* 'm' displays XML output, while *exportspec* 'c<separator>' displays values separated by any single character or string. For example, **'nsrinfo −xc,'** will produce comma-separated values.

FILE TYPES

The file index can store entries for all types of clients. Each index entry includes an index entry type. In general, only the client that created the index entry can decode the entry.

This section lists index entry types recognized by **nsrinfo**. However, even though these types are recognized, **nsrinfo** can only completely decode one entry type: the UNIX version decodes UNIX entry types, and the NT version decodes NT entry types. For other recognized types, some information may be incomplete.

old UNIX	Clients running versions earlier than 3.0 of NetWorker for UNIX.
UNIX	Clients running versions earlier than 4.0 of NetWorker for UNIX.
UNIX ASDF	Index entries including extended ASM Structured Data Format (ASDF) information for clients running versions 4.1 and later of NetWorker for UNIX.
UNIX ASDF v2	Index entries from agentless saves for clients running versions 4.2 and later of NetWorker for UNIX.
UNIX ASDF v3	Index entries for large files (files > 2 gigabytes) for clients running versions 5.1 for UNIX and later NetWorker for UNIX.
old DOS	DOS clients running versions 2.0 and earlier of NetWorker for DOS.
DOS	DOS, Windows, or OS/2 clients running version 2.0 of NetWorker for DOS, Windows, or OS/2.
DOS old ASDF	DOS, Windows, or OS/2 clients running version 2.0 of NetWorker for DOS, Windows or OS/2.
WIN ASDF	Windows or NT clients running NetWorker for Windows NT 4.2 and above.
WIN ASDF v2	Windows or NT clients running NetWorker for Windows NT 4.2 and above, created by using agentless saves.
old NetWare	NetWare clients running version 3.0 and earlier of NetWorker for NetWare.
NetWare	NetWare clients running version 3.0 and later of NetWorker for NetWare 3.0.
OSF 64bit	A client running OSF/1 with 64bit file sizes and offsets.
continuation	A special internal index entry, that is generated when a file crosses save set boundaries in a save set series.

APPLICATION TYPES

All	This application type prints out all of the X/Open Backup Services API (XBSA) information available for each object; only XBSA objects are printed. The **-v** and **-V** flags have the same effect here as they do on files.
Informix	This application type prints out only those objects recognized as Informix Database objects (XBSA ObjectOwner.bsaObjectOwner is

INFORMIX). The **-v** flag behaves as it does with files, while the **-V** flag prints out all the XBSA information about the object (see All, above), including the normal **-V** information.

None — This application type prints out objects that are *not* XBSA objects, but match the given criteria. For example, this option can be used to print a list of files backed up from a client.

PRIVILEGE REQUIREMENTS

A user is required to have the "Operate Networker" privilege to browse the indexes of any NetWorker Client. However, a user with 'Recover local data' privilege can browse NetWorker client indexes of the host from which this command is invoked. If the -L option is used, the user must be an administrator on the system where this command is invoked (that is, root on a UNIX system).

FILES

/nsr/index/*client***/db6**

SEE ALSO

nsr_getdate(3), **mminfo**(1m), **nsrck**(1m), **nsrindexd**(1m)

DIAGNOSTICS

bad time value '*time***'**
The time value specified in the −**t** option is not in a valid **nsr_getdate**(3) format.

cannot open index for client *client***:** *reason*
The file could not be opened using the −**L** option. The specific reason is printed, although there may be several. The most likely reasons are *permission denied* if the user is not the superuser, and *service busy, try again* if the file index is already locked (for example, by **nsrindexd**(1m)).

cannot create db scan on *client*
An internal error occurred while attempting to query the file index. Contact EMC Technical Support.

number **bad records for client** *client*
This diagnostic prints at the end of a report if any bad index records were detected. This is a sign that the index is damaged, and may need to be recovered.

cannot connect to server *server*
The index server is not available for one of many reasons. For example, the NetWorker server may be down, or **nsrinfo** may not be able to connect to a running server due to either a resource shortage or a network problem.

cannot start session with server *server*
The index server is running, but refused the connection. The exact reason is printed on the subsequent line of output. The most likely reasons are *permission denied* if the user is not a NetWorker administrator, and *service busy, try again* if the file index is locked (for example, by **nsrck**(1m)).

lookup failed to server *server*
The index server is running, but was unable to process the query. The exact reason is printed on the subsequent line of output.

LIMITATIONS

The command line options should be made as powerful as those of **mminfo**(1m).

The −**v** and −**V** reports are not formatted into columns.

A query for a specific time can take a very long time due to the schema of the file index.

The queries are limited due to the lack of a cross-platform browser.

NAME nsrjb − NetWorker jukebox control command

SYNOPSIS

nsrjb
[−**C**] [−**j** *name*] [−**s** *server*] [−**v**] [−**f** *device*] [−**S** *slots* | −**T** *Tags* | *volume names*]

nsrjb
−**L** [−**j** *name*] [−**s** *server*] [−**gimnqvG**] [−**Y** | −**N**] [−**B**] [−**b** *pool*] [−**f** *device* | −**J** *hostname*] [−**e forever**] [−**c** *capacity*] [−**o** *mode*] [[−**S** *slots* | −**T** *Tags*] [*volume names*]]

nsrjb
−**L** [−**j** *name*] [−**s** *server*] [−**gimnqvG**] [−**Y** | −**N**] −**R** [−**b** *pool*] [−**f** *device* | −**J** *hostname*] [−**e forever**] [−**c** *capacity*] [−**o** *mode*] [−**S** *slots* | −**T** *Tags* | *volume names*]

nsrjb
−**l** [−**j** *name*] [−**s** *server*] [−**nvqrG**] [−**R** [−**b** *pool*]] [−**f** *device* | −**J** *hostname*] [−**S** *slot* | −**T** *tags* | *volume names*]

nsrjb
−**u** [−**j** *name*] [−**s** *server*] [−**qv**] [−**f** *device*] [−**S** *slot* | −**T** *tags* | *volume names*]

nsrjb
−**I** [−**j** *name*] [−**s** *server*] [−**Evpq**] [−**I** | −**f** *device*] [−**S** *slots* | −**T** *tags* | *volume_names*]

nsrjb
−**p** [−**j** *name*] [−**s** *server*] [−**vq**] [−**f** *device*] [−**S** *slot* | −**T** *tag* | *volume name*]

nsrjb
−**o** *mode* [−**j** *name*] [−**s** *server*] [−**Y**] [−**S** *slots* | −**T** *tags* | *volume names*]

nsrjb
−**H** [−**j** *name*] [−**s** *server*] [−**EHvp**]

nsrjb
−**h** [−**j** *name*] [−**s** *server*] [−**v**]

nsrjb
−**U** *uses* [−**j** *name*] [−**s** *server*] [−**S** *slots* | −**T** *tags*]

nsrjb
−**V** [−**j** *name*] [−**s** *server*]

nsrjb
−**d** [−**j** *name*] [−**s** *server*] [−**v**] [−**N**] [−**Y**] [−**P** *ports*] [−**S** *slots*] [−**T** *tags*] [*volume names*]

nsrjb
−**w** [−**j** *name*] [−**s** *server*] [−**v**] [−**N**] [−**Y**] [−**P** *ports*] [−**S** *slots* | −**T** *tags* | *volume names*]

nsrjb
−**a** [−**j** *name*] [−**s** *server*] [−**vd**] [−**T** *tags* | [−**T** *tags*] *volume names*]

nsrjb
−**x** [−**j** *name*] [−**s** *server*] [−**vwX**] [−**T** *tags* | −**S** *slots*]

nsrjb
−**F** [−**j** *name*] [−**s** *server*] [−**v**] −**f** *device*

DESCRIPTION

The **nsrjb** program manages resources in two broad classes of jukeboxes, **remotely** managed jukeboxes and **locally** managed jukeboxes. Remotely managed jukeboxes are controlled through an external agent. **nsrjb** communicates with this agent to gain access to jukebox resources. The agent allows multiple applications, including multiple NetWorker servers, to share resources in the jukebox. Examples of agents are **AlphaStor** and **StorageTek's ACSLS . nsrjb** communicates directly with a locally managed jukebox, there is no intervening agent. Resources in a locally managed jukebox can be used by only one NetWorker server.

For a locally managed jukebox, the jukebox resource is used to track the state of the entire jukebox. The resource records the number of drives and slots in the jukebox. It is also used to track whether devices are loaded, whether there is media residing in the slots, the name of any volume on the media, as well as other information. See nsr_jukebox(5).

The jukebox resource for a remotely managed jukebox does not reflect the current state of the entire jukebox, only NetWorker's view. Media in remotely managed jukeboxes must be allocated before NetWorker may access it. For more details, see the description of the **−a** option. The number of slots in a remote jukebox resource increases as media is allocated for NetWorker's use and decreases as media is deallocated after NetWorker has no further use for the media. The order in which media is listed in the jukebox resource does not necessarily reflect physical location within the jukebox. The number of drives in a remote jukebox is an upper bound on the number of volumes in the jukebox that NetWorker may access simultaneously.

The **nsrjb** command is used to manage all jukeboxes for a NetWorker server. Use this command, rather than **nsrmm**(1m), to label, load, and unload the volumes contained within a jukebox. Multiple **nsrjb** commands may access a jukebox at any given time.

A **nsrjb** command which requires use of jukebox resources does not directly perform the requested operation. Instead the command makes a request of the NetWorker server process, **nsrd,** which forwards the request to **nsrmmgd** for processing.

Since **nsrjb** does not perform the operation directly, killing nsrjb will not cause the operation to be aborted. Provision for operation cancellation is built into nsrjb via an interrupt handler that is tied to SIGINT. This means that if you have a nsrjb command running, and you want the operation to be cancelled, then you may do it either by means of Control-C against the **nsrjb** process, or using the UNIX 'kill' command to send a SIGINT signal.

A single Control-C or SIGINT will cause the operation to be cancelled, with nsrjb still monitoring the status of the appropriate NSR jukebox operation status resource until it is clear that the operation has in fact terminated. A second Control-C or SIGINT will tell **nsrjb** to exit without waiting for confirmation of the operation's termination.

A *NSR jukebox operation status* resource will be automatically generated and managed by **nsrd** for each jukebox operation that is created, regardless of whether that operation was initiated automatically by **nsrd** or is created explicitly by invoking **nsrjb**

This *NSR jukebox operation status* resource tracks the current state of the operation, holds all messages (error, informational, or verbose) related to the operation, and generally acts as a communication path between the **nsrjb** process that invoked the operation, and the various Networker programs that carry the operation out. See the *nsr_op* man page for more details on this resource.

A *volume* resides on a side of a physical piece of media. Examples of piece of media are tape cartridges or optical disks. Tape cartridges have one side and therefore have one volume residing on each cartridge. Optical media may have two sides with a volume residing on each side of the media. Each volume within a jukebox and each jukebox has a name recognized by NetWorker. A *volume name* is specified when the volume is first labeled by NetWorker. You can change the volume name when a volume is relabeled. NetWorker refers to volumes by their volume names. For example, when requesting the mount of a volume, NetWorker asks for it by *volume name.*

Before using **nsrjb**, the jukebox and its device resources must be added to the NetWorker server. Use **jbconfig** to add the jukebox resource and its device resources to the NetWorker server. The jukebox resource is described in **nsr_jukebox**(5).

When a NetWorker server requires a volume for backup or recovery and an appropriate volume is not already mounted, the server checks the media database to verify whether a jukebox contains a volume that satisfies the media request. If so, **nsrd** sends a request to **nsrmmgd** to load the media into an idle device. The *Available Slots* attribute specifies the slots containing volumes available to automatically satisfy NetWorker requests for writable volumes. When automatically selecting a writable volume for backup, NetWorker only considers volumes from the list of available slots. It is important to note that the *Available Slots* attribute does not limit what slots the user running **nsrjb** can operate on.

nsrjb attempts to determine which jukebox to use based on the options **−j , −f ,** or a *volume name* . If one or more of these options do not uniquely identify a jukebox and one must be selected, the **nsrjb** program prompts you to select a jukebox. You can set the **NSR_JUKEBOX** environment variable to the name of the jukebox you want the **nsrjb** program to use by default.

NOTE: In a clustered configuration, either the **−f** *device* or the **−J** *hostname* option must be provided.

OPTIONS

Options are separated into two groups. The first are the options which specify the operation to be performed, e.g. label or load media. The second group list the additional options which provide arguments for the operation, e.g. specifying the media to be labeled or loaded. Note that option arguments that have spaces, for example, pool name, must be enclosed in double quotes.

OPERATION OPTIONS

−a This option is used in conjunction with the **−T** *tags* option, to allocate volumes in a remotely managed jukebox. A volume must be allocated before it can be labeled and used by a NetWorker server.

For STL silos a **−d** option can be added for silos that support depositing (also known as importing or entering) tapes from their I/O ports. The **−d** must appear *after* the **−a** on the command line. This function is usually handled by the silo management software, but is added here for ease of use. This option may not be supported on all silos supported by NetWorker.

There are two types of volumes which may be allocated or added to an AlphaStor jukebox resource: scratch or in-use. The term scratch is used to indicate volumes currently not being used by NetWorker. An in-use volume is one that was already used by NetWorker before being imported into AlphaStor.

Use **−a** in conjunction with **−T** *tags* option to allocate volumes for NetWorker's use. Both scratch and in-use volumes can be allocated this way. By specifying the barcode or physical cartridge label with this option, volumes from specific

media cartridges may be allocated. In-use volumes will be discovered by the jukebox inventory operation.

Use −**a** in conjunction with −**T** *tags* and **volume names** to directly add in-use volumes to an AlphaStor jukebox resource. The **tag** is the name given to the volume when it was imported into AlphaStor. The **volume name** is the volume name recorded in NetWorker's media database.

See −**x** for a description of how volumes are removed from a remote jukebox's list of volumes available for use by a NetWorker server.

−**C** Displays the current volumes in the jukebox and the devices associated with the jukebox. This is the default command option, used if no other command options are specified. It displays a list of slot numbers, volume names, media pools, optional bar code information, volume ids and volume modes. If the jukebox attribute *Bar Code Reader* is enabled and there are bar code labels on the media volumes, then the bar code label is included in the list. If *Bar Code Reader* is set and the volume does not have a bar code label, a dash prints, indicating that there is no bar code label on the media. By default the short volume id of a volume is displayed. Using the verbose option (-v) displays the long volume id along with other information described below. The -C option does *not* perform an actual jukebox inventory; **nsrjb** only reports on the volumes currently contained within the jukebox resource. Volumes may be succeeded by one of the following flags: an **(R)**, to indicate the volume is read-only; or an **(A)**, to indicate the volume is either an archive or a migration volume. When combined with the −**v** option, the capacity of the volumes that have been filled is also displayed. Volumes that are not contained in the NetWorker media database are marked with an asterisk, "*".

The Mode column contains additional information about the mode of the volume. The Mode field can have one of three values: *manually recyclable* to indicate that the volume will not be automatically recycled or relabeled; *recyclable*, to indicate that the volume is eligible for automatic recycling; or blank to indicate that neither of the other two values apply.

After the slot map prints, a line about each device is displayed. For each enabled device, the following information is provided: drive number, device pathname, slot number and name of the currently loaded volume, and an indication if NetWorker has the volume mounted. If the device is disabled, only the drive number and pathname are displayed, along with the message *disabled*. When several device resources share a physical drive in the jukebox, via the same *hardware id* attribute value, the drive number is only displayed on the first device pathname sharing the drive.

−**d** Deposits (loads into the jukebox) one or more cartridges from the cartridge access ports (also called import/export elements, mail slots, or I/E ports).

The number of cartridges to deposit is determined by the number of specified slots or tags. All empty slots in the jukebox are deposited, if slots or tags are not specified. Multiple destination slot ranges may be specified, full slots are skipped. If all available import ports are empty and there are cartridges to deposit, the operator will be prompted to fill the import ports. When the −**N** option is used in conjunction with the jukebox polling feature, the jukebox will poll for cartridges in the import ports until all of the cartridges are deposited or an error occurs. Exceeding the polling timeout waiting for additional

cartridges is considered an error.

Specifying volume names on the command line is not recommended. The inventory command should be run to accurately determine the volume names.

If **−d** is used with a **−T** *tags* option, then the command is assumed to be running on a silo, and is treated internally as if it had been run with the **−a** and **−d** options. Specified volume tags (barcodes) will be deposited into the silo and then NetWorker will attempt to allocate them for its use. Depending on the exact type of silo used, this allocation step may or may not succeed. You should verify the success of the allocation, and retry the command with just the **−a** option for all of the tag values specified. If the tags have already been allocated, you will see a message indicating this. This is not an error, and only means that the volumes had already been successfully allocated for use by NetWorker.

−F Releases a shared device contained within an STL silo. This option is only available for tape libraries with device sharing. See **nsr_jukebox**(5).

−h Displays the actions and results of the past 120 jukebox commands issued. These include commands issued on the command line by the user, or requests that were started automatically by NetWorker. If you wish to change the number of command lines saved in the history, you may set the environment variable **NSRJB_HISTORY_COUNT** to a value between **20** and **2000.** Values smaller than 20 will result in 20 being used, and values larger than 2000 will result in 2000 being used.

−H Resets the jukebox hardware (and the NetWorker database representing the jukebox) to a consistent state. The jukebox clears the transport and then unmounts and unloads volumes from the drives to slots. An actual inventory is not performed; (see the **−I** option). If the jukebox senses that the inventory is out-of-date, it prints an appropriate message.

For silos, only devices which NetWorker thinks are loaded are unloaded. You can use the silo controller to empty other drives.

For AlphaStor jukeboxes, resets the jukebox devices and the NetWorker database representing the jukebox to a consistent state. The operation synchronizes the state of the devices in the jukebox and the media in the jukebox resource with AlphaStor. NetWorker queries AlphaStor for information about volumes in the jukebox resource and which volumes are currently mounted. It uses this information to synchronize the jukebox and device resources to be consistent with the information reported by AlphaStor. If the **-p** option is also specified, a check operation will be performed on the loaded volumes.

NetWorker automatically queries AlphaStor to synchronize the jukebox and device resources whenever the server is started.

−I Performs an inventory on the jukebox's contents. Use this option to ensure that the mapping between slot number and *volume name* is correct. If necessary, the volumes in the specified slots may be loaded into a device, so their labels may be read. This option can take a long time to complete depending on the type of jukebox.

If a jukebox has a bar code label reader, and the jukebox resource attribute *Bar Code Reader* is set, and *Match Bar Code* is set, then the *volume name* associated with a slot is derived from the media bar code label.

Tapes are always loaded into drive for labels to be read in the following conditions:
1) jukebox does not have a barcode reader
2) jukebox has barcode enabled and tape's barcode is not in the media database
3) jukebox has barcode and match barcode enabled; tape's barcode is in the media database but location is empty

If a bar code label on the media has changed, then the NetWorker media database is updated with the new bar code label. Proper use of a jukebox's bar code reader can minimize the time it takes to perform an inventory.

The -II option can be used to perform a fast inventory which operates only on slots with volumes that can be verified without reading their labels. Since fast inventory does not involve reading the tapes, this option may not be combined with a device specification (−f).

The -Ip option forces tapes to be loaded into the drive for their label to be read even if the volume's label can be verified.

For jukeboxes that have element status capability you can use the −**E** option in conjunction with the −**I** option to reinitialize the jukebox's inventory state. The **-E** option increases the amount of time it takes to inventory a jukebox, because the hardware must check every component, including all slots and drives, for the presence of media. You should only use this option if you are manually swapping media in or out of a jukebox.

For AlphaStor jukeboxes, this operation is used to synchronize NetWorker and AlphaStor databases. It insures that AlphaStor and NetWorker agree to the state of all volumes allocated to this NetWorker server and listed in this jukebox resource. If the −**p** option is also specified, *nsrjb* requests the volumes be loaded so that labels on each volume may be verified.

To allocate slots in a jukebox for cleaning cartridges, set the jukebox resource attribute *Auto Clean* to *Yes* and the *Cleaning Slots* attribute to a non-empty range of slots. For further information see **nsr_jukebox**(1m). Volumes from slots that are reserved for cleaning cartridges are not loaded during the inventory of a jukebox. For jukeboxes that do not support element status or have a bar code reader, the −**U** *uses* option must be used to enter a cleaning cartridge into the jukebox's inventory. For jukeboxes that support element status or have a bar code reader, cleaning cartridge slots that were previously empty but now contain a cartridge have the number of uses for the cleaning cartridge is the value set in the jukebox attribute *Default Cleanings.*

−**l** Loads and mounts specified volumes. Volumes are specified by name, by the slot in which the volume resides, or for remote jukeboxes by the *tag* associated with the volume. The operation fails, if the number of volumes specified is greater than the number of available drives.

For AlphaStor jukeboxes, the command will attempt to mount volumes into devices accessible from the storage node upon which *nsrjb* is running. The −**J** option can be used to specify a different storage node.

The −**f** option can be used to specify media devices into which volumes are

loaded.

If loading a device located in a remote jukebox, an NDMP device for instance, the "-f" option with the "rd=" syntax in the device name must be specified.

−**L** Labels the volumes in the specified slots, or for remotely managed jukeboxes, by specified tags. Names for the volumes labeled are derived from media bar code labels, *volume names* specified on the command line, or generated by referencing the *label template* resource for the given **pool**. If you do not specify any slots, the range of slots is as described in the **NSR_jukebox** resource for the jukebox. Labeling a complete jukebox may take a long time.

If the jukebox has a bar code label reader, and the **NSR_jukebox** resource attributes *Bar Code Reader* and *Match Bar Code Labels* are set, then the volume label is derived from the bar code label on the media. If the jukebox resource attribute *Match Bar Code Labels* is not set, or the jukebox does not have a bar code reader, then the volume label is derived from *volume names* specified on the command line. If more volumes are being labeled then volume names specified on the command line, then the volume label is derived from the label template. No matter how the volume label is derived, if the media labeled has a media bar code label, the bar code is stored in the NetWorker media database so that it can be used during inventory operations.

Volumes names cannot be used without −**S** or −**R** options for regular jukeboxes. The reason for this is that the *volume names* do not exist in the *media database* in the case of the new or imported tapes.

Volumes located in slots set aside for cleaning cartridges cannot be labeled. See −**I** for a discussion of how the slots of a jukebox are set aside for cleaning cartridges.

If an empty slot is encountered, an informational message is displayed and the operation continues.

See the −**m** option if you want the volume to be automatically mounted after being labeled.

−**o** *mode*
Sets the mode of a volume or range of slots. The following mode values are available: **[not]recyclable, [not]readonly, [not]full,** or **[not]manual.** If the −**Y** option is not used, you are prompted to confirm the operation for each volume. See **nsrim**(1m) for a discussion of the per-volume flags.

−**p** Verifies and prints a volume label. A slot or for remotely managed jukeboxes a tag may be specified. The device used to read the volume may also may be specified. See **nsrmm**(1m).

−**u** Unloads a volume from a device. To unload a volume from a device, specify the name of the volume, the device in which the volume is loaded, or the slot from which the volume was loaded. If no volume, device or slot is specified, media is unloaded from all loaded devices.

−**U** *uses*
Sets the number of times a cleaning cartridge can be used. Slots can also be specified. Any slot specified must be in the range of slots set aside for cleaning cartridges in the jukebox. If a range of slots is not specified, all slots set aside for cleaning cartridges are updated. For slots that are currently empty in the

jukebox's inventory, this option updates the inventory to indicate that the slot is occupied by a cleaning cartridge. For a discussion of how slots of a jukebox are set aside for cleaning cartridges, see −**I.**

Uses must be either a positive integer, or the reserved words *remove* or *default.* The reserved word *remove* can be used (for example, **-U** *remove*) to delete the cleaning cartridge(s) from the NetWorker inventory. Specifying *default* sets the number of times a cleaning cartridge may be used to the value of the *default cleanings* attribute for the jukebox. See **nsr_jukebox**(5).

You can use the −**T** option in conjunction with the −**U** option to add cleaning cartridges to a Silo Tape Library (STL). This option sets aside a cleaning slot in the STL each time a cleaning cartridge is added. For a description of how to remove cleaning cartridges from an STL, see −**x.** See −**I** for a discussion of how slots in a non-STL jukebox are set aside for cleaning cartridges.

−**V** Display the current jukebox configuration.

−**w** Withdraws (ejects media from the jukebox) one or more cartridges to the cartridge access ports.

Cartridges must be specified by slot, volume name or tag. Multiple slot ranges and volume names may be specified, empty and duplicate slots are ignored. If the available export ports are full and there are cartridges to withdraw, the operator will be prompted to empty the export ports. When the −**N** option is used in conjunction with the jukebox polling feature, the jukebox will poll for empty export ports until all cartridges are withdrawn or an error occurs. Exceeding the polling timeout waiting for empty ports is considered an error.

If −**w** is used with a −**T** *tags* option, then the command is assumed to be running on a silo, and is treated internally the same as if it had been run with the −**x** and −**w** options. Specified volume tags (barcodes) are withdrawn from the silo. Then NetWorker deallocates them from its list of volumes for that silo. In general, you can only withdraw at most about 40 volumes from a silo at one time, although this limit differs on different silo models. If a given command does not cause any tapes to be withdrawn from the silo, try again using fewer tag values on the command line.

−**x** This option, when used in conjunction with the −**T** *tags* or −**S** *slots* option, is used to remove volumes from a remote jukebox. The specified volumes are removed from the remote jukebox's list of volumes available for use by a NetWorker server.

For STL silos, a −**w** option can be added to withdraw or eject tapes from the silo or to *physically* remove the tapes from the silo. The −**w** must appear *after* the −**x** on the command line. This function is normally handled by the silo management software, but is added here for ease of use. This option may not be supported on all silos supported by NetWorker.

See −**a** for a description of how volumes are allocated for use by a NetWorker server.

ADDITIONAL OPTIONS

−**b** *pool* Specifies the media pool to which the volume should belong. The pool may be any pool currently registered with the NetWorker server. The pool names can be viewed by selecting Media Pools from the left pane of **NetWorker**

Management Console's Media display. The pool name is referenced by the NetWorker server when determining what save sets can reside on the volume. If you omit this option the volume is automatically assigned to the *Default* pool. If you specify a pool name without a volume name, **nsrjb** will use the next volume name associated with the specified pool's *label template* resource. See **nsr_label**(5).

−**c** *capacity*
Overrides the volume's default capacity. See **nsrmm**(1m).

−**B** Verifies that the volume currently being labeled does not have a readable NetWorker label. Before labeling a volume, NetWorker attempts to read any existing labels written on the volume. If you specify this option and the volume has a NetWorker label that is readable by the device currently being used, the label operation is canceled and an error message is displayed. If the volume does not have a label, or has a label that is not readable by the current device, then the volume can be labeled. This option is used by **nsrd**(1m) to label volumes automatically when **nsrmmd**(1m) makes a request for a volume while saving data.

−**e forever**
Specifies the volume to be an Archive volume. (see **nsrmm**(1m)).

−**E** Initializes element status for jukeboxes that support this feature. You can use this option in conjunction with the −**I** or −**H** options. Some jukeboxes have the ability to keep track of whether or not there is media in a component in the jukebox. This feature is known as an "element status" capability. The −**V** option may be used to determine whether a jukebox has this capability. When swapping media into the jukebox where media was not previously loaded, it may be necessary to reinventory (−**I**) the jukebox with the −**E** option so the jukebox reinitializes its element status.

−**f** *media device*
Specifies a media device to be used for an operation. Use the pathname of the media device as it is configured in the jukebox resource. When more than a single media device has been configured for a jukebox, **nsrjb** selects available devices with the lowest value for the device resource attribute **accesses.** See **nsr_device**(5). When loading or verifying volumes, the number of devices available must at least be greater than or equal to the number of volumes specified for the operation. For other operations, the value of the jukebox attribute **max parallelism** is an upper bound on the number of devices that may be used by any nsrjb command. See **nsr_jukebox**(5). You can override the device selection by using the −**f** option. You can use this option multiple times, to specify more than one media device.

For AlphaStor jukeboxes, the device resource is not tied to a physical device. It is a logical device resource. An association between this logical device and the physical device lasts as long as media is loaded in the device. NetWorker never asks AlphaStor to load media into a particular device. It allows AlphaStor to choose the device into which the media is loaded. Then **nsrjb** creates an association between the actual device and NetWorker logical device resource by assigning values to the device's **logical name**, **logical type**, and **logical family** attributes. See **nsr_device**(5). AlphaStor and NetWorker have different names for device and media types. **nsrjb** maintains a table to map between AlphaStor and NetWorker names to be able to correctly set the values of these attributes. This table can be updated dynamically to support additional AlphaStor drive and/or media types. The file **/nsr/res/dmidevmap.txt** is used

to make additions to nsrjb's map table. Each line in this file contains four columns, AlphaStor cartridge type, AlphaStor bitformat, NetWorker device resource media type, and NetWorker device resource family type. The AlphaStor bitformat maybe a regular expression, all other values are strings. As an example the line,

DTL7000 DLT8000.* DLT8000 tape

maybe used for the DLT8000 device using AlphaStor DLT7000 cartridge type.

−**g** This option is kept for historical reasons only. It has no affect.

−**G** This option is used only by the server to have the autoloader mount or label a volume in a Network Data Management Protocol (NDMP) device.

−**i** This option is kept for historical reasons only. It has no affect.

−**j** *name*
Specifies a particular jukebox to use. The given *name* is the one assigned by the user when the jukebox resource is created. This option overrides the **NSR_JUKEBOX** environmental variable.

−**J** *hostname*
Specifies a particular hostname to use. Drive selection by **nsrjb** will be restricted to a drive on the given *hostname.* This option can be used with the −**l** (load) or −**L** (label) options, and cannot be used with the −**f** option.

If the jukebox that you manage is connected to a NDMP server (e.g., a NAS filer), you need to use this option to specify the NDMP server. See **Examples.**

−**m** Mount a volume after it has been labeled. There must be enough available drives to mount all volumes to be labeled.

−**n** Loads, but does not mount, the volume when specified with the −**l** option.

−**N** Tells **nsrjb** to skip the confirmation prompt when used in conjunction with the −**LRdw** options. When NetWorker recycles volumes, NetWorker prompts you to confirm that it is okay to overwrite any volumes considered to be nonrecyclable. See **nsrim**(1m) for a discussion of the per-volume flags.

−**P** *ports*
Specifies a cartridge access port or range of ports to deposit or withdraw volumes.

Ranges are specified as *low* to *high*. Both *low* and *high* must be integers; *low* must be less than or equal to *high*. Both numbers are checked for validity against the resource describing the jukebox. You can specify only one port range for a command.

−**q** Runs the **nsrjb** program in quiet mode. Turns off all of the messages normally produced when verifying, labeling, loading, or unloading volumes, or inventorying a jukebox. You can use this option only with the −**p** −**L**, −**l**, −**u** or −**I** options.

−**r** Loads the volume as read-only. You can use this option only with the −**l** option. See **nsrmm**(1m).

−**R** Recycles the volume. If the conditions as described for the -L option are met for using bar codes as labels, then the volume label is derived from the bar code label on the media (without exception). If the volume is recycled to a new pool, the label is generated by referencing the label template resource for the given pool. Otherwise the volume is relabeled using its current name. If

a volume is recyclable, you are not prompted for confirmation as to whether or not this volume may be overwritten. See **nsrmm**(1m) for a discussion of the per-volume flags.

−**s** *server*
Specifies the controlling server when **nsrjb** is used on a storage node. To use **nsrjb** on a storage node, the command must be run on the storage node. See **nsr_storage_node**(5) for additional information on storage nodes.

−**S** *slots*
Specifies a slot or range of slots on which to operate. Specify the slot range from low to high integer order. Both *low* and *high* must be integers; *low* must be less than or equal to *high*. Both numbers are checked for validity against the resource describing the jukebox. You can specify multiple slot ranges for a command.

−**T** *tags*
Specifies tags or barcodes of volumes in a remote jukebox. You can specify this option more than once for a command.

tags can specify a single volume tag or a volume tag template similar to a label template. See **nsr_label(5).** The volume tag Template is a list of template fields separated by slashes "/". A template field is a constant alphanumeric string or an alphabetic or numeric range represented by the low and high value separated by "-".

This template differs from the templates used in NetWorker GUI. Each portion of the template is entered into a separate line in the GUI's dialog box instead of using "/" as a separator.

The tag is used to identify the media when a request is made of the agent managing the remote jukebox. This identifier is determined by the remote agent. A tag often is a bar code label. When making a request to load media into a device, NetWorker sends the *tag* with the request to the agent to identify the media to be loaded. Volumes in a jukebox resource are listed in alphanumeric order of their tags. Therefore, the order in the jukebox resource may change as media is allocated and deallocated, and has no relation to the slot in which the media may reside in a physical library.

−**v** Set the verbosity level by the number of times this flag is specified on the command line. The maximum verbosity level supported is 5. See other arguments for specific details on the verbose output.

−**X** You can use this option in conjunction with −**x** to purge a volume from NetWorker's media database when the volume is being deallocated. A prompt is displayed to confirm that the volume is to be purged from the media database, unless −**Y** is also specified.

−**Y** Disables confirmation prompting. Rather than prompting for confirmation, a *yes* answer is assumed. Prompts are normally generated when a volume is being relabeled before its expiration date, or when a volume is still registered in the NetWorker media database. If the operation is to label (−**L**) a volume or to load (−**l**) a volume, with the −**R** option also specified, and the volume is recyclable, there is no prompt to confirm whether the volume may be overwritten.

volume name
Specifies the name to be used when labeling a volume. After a volume has been labeled, the volume name is used to select media for an operation.

Multiple volumes names may be specified for a single command, and must come at the end of the command line.

EXAMPLES *Labeling volumes*:

To label all of the volumes in a jukebox, use the −**L** option:

```
nsrjb −L
```

To specify a particular pool, use the −**b** option:

```
nsrjb −L −bOffsite
```

Labeling the volumes in slots 5 through 19:

To label the volumes in slots 5 through 19, use the −**S** option:

```
nsrjb −L −S 5−19
```

Labeling a volume with a non-standard name:

To label the volume in slot 20 with a name that does not match the label template associated with a pool, specify the name along with the −**L** option:

```
nsrjb −L −S 20 mars.special
```

When more than one volume is to be labeled, the name must match the label template associated with the pool. This ensures that **nsrjb** generates the subsequent names.

Mounting a volume after it has been labeled:

To mount a volume after it has been labeled use the −**m** option:

```
nsrjb −L −S 20 −m
```

To mount an NDMP volume after it has been labeled, use either of the following commands with the **-m** option:

```
nsrjb −J <ndmp client name> −L −S 20 −m
```

or

```
nsrjb −L −S 20 −m −f <ndmp device name>
```

The command fails if there are not be enough drives to mount all volumes to be labeled.

Labeling volumes with a standard name:

To label the volumes in slots 21 through 28, starting with a name different than that referenced by the label template associated with the pool resource, specify the first name along with the −**L** option. In order for **nsrjb** to generate the additional names, the specified name must match the layout of the label template.

```
nsrjb −L −bOffsite −S 21−28 Offsite.501
```

After labeling the volume in slot 21 with 'Offsite.501' **nsrjb** uses the label template to generate names for the volumes in slots 22 ('Offsite.502') through 28 ('Offsite.508'). If the next *volume name* in the sequence for a label template is already in use, the name is skipped.

Loading a volume:

To load volumes, use the −**l** option.

```
nsrjb −l
```

nsrjb will select volumes to load into selected devices. It will continue loading volumes until all of the devices are loaded.

Loading specific volumes:

To load a volume named *mars.001*, specify the *volume name* along with the −**l** option:

```
nsrjb −l mars.001
```

To load the volume in slot 5, use the −**S** option:

```
nsrjb −l −S 5
```

To load the selected volume into device */dev/nrst1,* include the −**f** option.

```
nsrjb −l −f /dev/nrst1 mars.005
```

Loading volumes in a jukebox connected to an NDMP server:

To load the volume in slot *1* of jukebox *mylibrary* (connected to NDMP server *10.31.32.220),* use the *-J* and *-j* options.

```
nsrjb −J 10.31.32.220 −j mylibrary −l −S 1
```

To load the volume in slot *1* of jukebox *mylibrary* (connected to NDMP server *10.31.32.220)* to a specific device *nrst0l,*

```
nsrjb −l −f "rd=10.31.32.220:nrst0l (NDMP)" −j mylibrary −S 1
```

Unloading a volume

You can unload a particular volume, slot, or device. To unload volume *mars.0028*, use the *-u* option:

```
nsrjb −u mars.0028
```

To unload the volume in slot *28*, use the −**S** option:

```
nsrjb −u −S 28
```

To unload the volume in device */dev/nrst3*, use the −**f** option.

```
nsrjb −u −f /dev/nrst3
```

Displaying the jukebox's current volumes

To display a list of slots and volumes, and which volumes are loaded in to a jukebox's devices, use the *-C* option:

```
nsrjb −C
```

The −**C** option is the default and is used when no other options are selected. A range of slots may also be specified. For example, to display the volumes in slots 10 through 23, use the **-S** option:

```
nsrjb −S 10−23
```

Setting the number of uses for a cleaning cartridge:

To set the number of times all cleaning cartridges in a jukebox may be used to 12, use the **-U** option:

```
nsrjb −U 12
```

To set the number of times the cleaning cartridge in slot 10 may be used, use the **-S** option:

```
nsrjb −U 25 −S 10
```

Slot 10 must be a slot set aside for cleaning cartridges in the jukebox.

Inventorying the volumes:

To reconcile the actual volumes and the list of volumes produced by **nsrjb,** use the −**I** option. Each volume may be loaded into a device and examined for a NetWorker label (depending on bar code settings and other factors). The internal list is then updated with the new information. After all volumes have been examined, the new list is compared to the NetWorker media database, and a message listing any volumes located in the jukebox but not in the database is produced. To inventory the volumes in slots 17 through 43, use the **-S** option:

```
nsrjb −I −S 17−43
```

Like labeling, volume inventory may take considerable time.

Using the NetWorker notification system:

When NetWorker needs a volume, a "media event" is generated. To have **nsrjb** automatically respond to these events, the NetWorker notification system is used. This notification resource is automatically generated.

Using the cartridge access port:

To withdraw cartridges from jukebox slot 7 through 11 to the cartridge access port 5 through 10, use the **-w** option along with the **-S** and **-P** options:

nsrjb −w −S 7−11 −P 5−10

To deposit cartridges into jukebox slot 8 through 10 from the cartridge access port 3 through 5, use the **-d** option along with the **-S** and **-P** options:

nsrjb −d −S 8−10 −P 3−5

Using barcode templates on tape libraries:

To add volumes with barcodes D001A, D002A, ..., D100A to the volumes available for NetWorker in the tape library, use the **-a** and **-T** options:

nsrjb −a −T D/001−100/A

To deposit tapes labeled with barcodes D001A, D002A, ..., D012A into the silo and also to make the volumes available for NetWorker in the tape library, use the **-a** and **-T** options along with the **-d** option:

nsrjb −a −T D/001−012/A −d

To remove volume with barcode D055A from the volumes available for NetWorker in the tape library, use the **-x** and **-T** options:

nsrjb −x −T D055A

To remove volume with barcode D055A from the volumes available for NetWorker in the tape library, and to withdraw it from the tape library physically (for example, for off-site storage), use the **-x** and **-T** options, along with the **-w** option:

nsrjb −x −T D055A −w

To label volumes with barcodes D010A, D011A, ... , D020A, use the **-L** and **-T** options:

nsrjb −L −T D0/10−20/A

To add cleaning cartridge with barcodes C010A, that can be used the default number of time for this jukebox, use the **-U** and **-T** options:

nsrjb −U default −T C010A

Forcing an unload of all drives on a tape library:

nsrjb −HH

ENVIRONMENT VARIABLES

NSR_JBOX_POLL_JUKEBOX_OP_STATUS

When **nsrjb** is run to initiate a jukebox operation, a request is submitted to **nsrmmgd,** for execution. Status of the operation is reported by **nsrmmgd** using a **NSR JUKEBOX_OPERATION_STATUS** resource. This resource is stored in the RAP database maintained by **nsrd.** Periodically **nsrjb** polls **nsrd** to determine the status of the request. The default is to poll every 10 seconds. Set this environment variable to modify the polling interval. Minimum interval is to poll every 5 seconds and the maximum interval is 30 seconds.

FILES

/nsr/mm/mmvolume
The NetWorker media database.

/nsr/res/nsrdb The configuration database containing resource descriptors.

/nsr/res/dmidevmap.txt
The file used to map from AlphaStor media and drive types to a NetWorker device resource **media type.** *jukebox*.

SEE ALSO

jbconfig(1m), **jbexercise**(1m), **mminfo**(1m), **mmlocate**(1m), **nsr**(1m), **nsrd**(1m), **nsrmmgd**(1m), **nsr_layout**(5), **nsr_device**(5), **nsr_jukebox**(5), **nsr_op**(5), **nsr_notification**(5), **nsr_storage_node**(5), **nsradmin**(1m), **nsrim**(1m), **nsrmm**(1m),

nsrmmd(1m), **nsrwatch**(1m)

DIAGNOSTICS

The exit code returned by the **nsrjb** command has one of four possible values:

0 *(success)*
A zero exit code indicates successful execution of the command.

1 *(not executed)*
Indicates that the command caused an error that prevented it from being submitted for execution. For example, an invalid command-line argument.

2 *(non-retryable)*
The command was submitted to **nsrmmgd** for execution, but a "non-retryable" error occurred. For instance, the named volume does not exist.

3 *(retryable)*
The command was submitted to **nsrmmgd** for execution, but a "retryable" error occurred. For instance, a required drive is busy.

In general, a "retryable" error indicates that if you simply retry the same **nsrjb** command again, there is a possibility that it would succeed this time. Conversely, a "non-retryable" error indicates that some user intervention is required in order to resolve the issue, before the **nsrjb** command should be retried.

NAME nsrjobd − NetWorker jobs monitoring daemon

SYNOPSIS **nsrjobd**

DESCRIPTION The **nsrjobd** daemon is one of the NetWorker server daemons. It is responsible for spawning and monitoring of ´jobs´ of NetWorker client binaries. All executions of **save**(1m), **savefs**(1m) and **savegrp**(1m) are considered jobs. Every execution of the binary is considered a separate job. **nsrjobd** allows for monitoring and long term recording of NetWorker activities. Ultimately, all functionality requiring monitoring or remote execution will be considered a job and handled with help of **nsrjobd.**

nsrjobd extends the capabilities previously offered by nsrexec. An example of where **nsrjobd**'s capabilities are taken advantage of is when **savegrp**(1m) requires execution of **save**(1m) and **savefs**(1m).
In addition to the remote spawning, **nsrjobd** collects run-time information to be used by the NetWorker GUI to report and monitor on both actively running as well as completed jobs.

For storing job related information, **nsrjobd** maintains its own RAP database in /nsr/res/jobsdb. To prevent this database from constantly growing, there is an upper limit of size and a retention period placed on the data in the database. Data pertaining to completed jobs will be migrated to an SQL database maintained by the NetWorker GUI, then deemed eligible for purging from the RAP database. The values for the retention period and database size are stored in the NSR Resource in nsrd's RAP database. They are configurable by the administrator. In contrast to NetWorker's RAP database, nsrjobd's database is considered an opaque data store for nsrjobd's private use, and thus no tools are provided for viewing or manipulating its contents.

nsrjobd is started and stopped automatically when **nsrd**(1m) is started and shut down respectively. It is not meant to be started manually.

nsrjobd uses the client side **nsrexecd**(1m) for remote execution, so the NetWorker server requesting command execution must be in the client's servers file.

FILES **/nsr/res/jobsdb**
Directory holding nsrjobd's RAP database.

SEE ALSO **nsrd**(1m), **nsrexecd**(1m)

NAME nsrlcpd − NetWorker library control program daemon

SYNOPSIS **nsrlcpd** −**s** *server* −**N** *mmgd_daemon_num* −**n** *lcpd_daemon_num*

DESCRIPTION The **nsrlcpd** daemon provides a uniform library interface to the NetWorker media management daemon, **nsrmmgd.** The **nsrlcpd** implements class specific protocols to communicate to the NetWorker supported library classes which include:

- o SCSI medium changers connected through SCSI cables, fibre-channel SANs or NDMP Services.
- o Microsoft Removable Storage Subsystems.
- o Standard Tape Libraries (STL), sometimes refered to as "SILO" Subsystems.
- o AlphaStor Media Management Subsystems.

The **nsrlcpd** manages the library subsystem media, slot, drive and port resources providing control to move and access the resources within the library subsystems.

The NetWorker media management service starts one **nsrlcpd** daemon for each virtual jukebox instance defined in the NetWorker server's configuration resource database. Each **nsrlcpd** daemon will be started on the NetWorker Storage Node which has access to the library subsystem interface. Once the **nsrlcpd** daemon is started, it provides the ability to:

- o Accept configuration information to access and control the library.
- o Report library components and characteristics as controllable resources.
- o Report accessible media within the library.
- o Allocate and deallocate media for use by the NetWorker application.
- o Load and unload media into read and writable devices.
- o Deposit and withdraw media into the library systems.

The **nsrlcpd** daemon provides an RPC-based library control program service across network boundaries. The RPC program number for **nsrlcpd** is 390429. To support multiple instances, the RPC version number used by the **nsrlcpd** during the RPC service registration is calculated by multiplying 100 by the **nsrlcpd** daemon number and adding 1, which is the base version. For example, a nsrlcpd process started with the command, "nsrlcpd -s NetWorkerServer -N 1 -n 2" would register with the program number 390429 and the version number 201.

OPTIONS

−**s** *server*
Specify the controlling NetWorker server.

−**N** *mmgd daemon num*
Specify the nsrmmgd daemon number.

−**n** *lcpd daemon num*
Specify the nsrlcpd daemon number.

FILES

/nsr/logs/daemon.raw
The file to which **nsrlcpd** and other NetWorker daemons send information about various error conditions that cannot otherwise be logged using the NetWorker event mechanism.

SEE ALSO **nsr**(1m), **nsr_service**(5), **nsr_render_log**(1m), **nsrmmgd**(1m)

NAME nsrlic − NetWorker license reporting command

SYNOPSIS **nsrlic** [−**vi**] [−**s** *server*]

DESCRIPTION **The nsrlic** command generates reports about all license information currently active on a NetWorker server. This command queries the NetWorker resource database, and formats and displays the results to standard output.

The **nsrlic** program reports the following information:

− The number of standard client licenses

− The number of standard client licenses used

− The number of standard client licenses borrowed by virtual client physical nodes

− The number of remaining standard client licenses

− The list of standard clients connected to the named NetWorker server

− The list of standard clients defined in the specified NetWorker server

− The number of clients, listed by platform

The **nsrlic** program reports the following information for virtual client physical hosts and for NDMP clients:

− The number of licenses

− The number of licenses used

− The number of remaining licenses

− The list of clients connected to the specified NetWorker server

When applications exist which require licensing, **nsrlic** also reports them in the same manner. In this case, however, the output will not contain any references unless either there are licenses available, or a connected client is utilizing a license count for such applications.

OPTIONS

−**i** Selects interactive mode. In this mode, you can request different reports, refresh the information, or switch to a different server. The information is requested once and cached until another **connect** command is issued.

−**s** *server*
Selects which NetWorker server to query. By default, the server on the local system is queried.

−**v** Selects verbose mode. In addition to the number of licenses or the number of clients, a list of connected and defined clients is gathered and displayed.

USAGE The following commands are supported in the interactive mode:

connect [*server*]
Connects to the named *server.* By default, this is the server on the local system.

detail
Produces a detailed report. Displays a list of connected clients (clients that saved to NetWorker) and a list of defined clients (clients that are defined in the NetWorker server, but not yet saved).

help
Displays a list of available commands.

summary
Displays a summary report.

?

Is the same as online help.

quit

Performs an immediate exit from **nsrlic.**

DIAGNOSTICS

nsrlic displays a "usage" message describing the available options when characters are used that are not valid for this command.

command not found

Indicates that the command is not a supported command.

RPC error: Remote system error RPC error: Program not registered

Indicates that some problems were encountered while connecting to the NetWorker server on the specified system. The **nsrlic** command requires that the NetWorker daemons be running. Start your NetWorker daemons **(nsrd)** and rerun **nsrlic.** If **nsrd** is already running, you have probably reached a resource limit on the server (for example, not enough memory or no more processes).

SEE ALSO

nsrd(1m), **nsradmin**(1m)

NAME nsrls − list statistics of NetWorker index files

SYNOPSIS **nsrls** [{ *clientname* ... | −**m** }]

DESCRIPTION When **nsrls** is used without any specified options being specified, the number of records in an online index and the usage of the online index with respect to the number of kilobytes allocated to its UNIX files is printed. Administrators can use this command to establish how many files have been saved by a client.

OPTIONS When invoked with the −**m** option, **nsrls** prints the information for the media database. The media database has the following four statistics associated with it: an internal file ID (**Fid**), the size of the file (**Size**), the number of logical records in the file (**Count**), and a descriptive name for the internal file (**Name**).

The internal files are interpreted as follows:

saveset files
These are the internal record files which store the actual data (for example, ss).

volume files
These are the internal record files which store the volumes (for example, vol).

index files
These internal b-tree index files hold the index records used for optimizing media database queries. The names of these files contain the extension "_i*" (for example, ss_i0 and vol_i1).

temporary files
These files (beginning with the filename "temp_*") contain temporary records used during sorting. Temporary files are present only while a database is currently being modified.

The number, name, function, and interpretation of the internal files can change at any time.

An empty argument list prints the statistics for all known clients.

EXAMPLE

```
% nsrls -m

Database id 0: /nsr/mm/mmvolume
 Fid |      Size |     Count | Name
 ----------------------------------------
   0 |     16 KB |         6 | vol
   1 |    136 KB |       484 | ss
   2 |     16 KB |         6 | vol_i0
   3 |     16 KB |         5 | vol_i1
   4 |     16 KB |         5 | vol_i2
   5 |     16 KB |         5 | vol_i3
   6 |     16 KB |         0 | vol_i4
   7 |     24 KB |       484 | ss_i0
   8 |     24 KB |       484 | ss_i1
   9 |     16 KB |       164 | ss_i2
  10 |     24 KB |       483 | ss_i3
  11 |      8 KB |         1 | temp_0

% nsrls jupiter

/space2/nsr/index/jupiter: 292170 records requiring 50 MB
```

/space2/nsr/index/jupiter is currently 100% utilized

SEE ALSO **nsr_layout**(5), **nsrindexd**(1m)

DIAGNOSTICS **... is not a registered client**
The client named is not a valid NetWorker client.

NAME nsrmm − NetWorker media management command

SYNOPSIS

nsrmm [−**C**][−**v** | −**q**][−**s** *server*][−**f** *device*]

nsrmm −**m** [−**v** | −**q**][−**s** *server*][−**f** *device*][−**r**][*volume*]

nsrmm −**l** [−**v** | −**q**][−**s** *server*][−**f** *device*][−**myB**][−**e** *forever*][−**c** *capacity*][−**o** *mode*][−**b** *pool*][−**R** | *volume*]

nsrmm −**H** −**f** *device* [−**v** | −**q**][−**s** *server*][−**y**]

nsrmm { −**u** | −**j** }[−**v** | −**q**][−**s** *server*][−**y**][−**f** *device* | *volume..*]

nsrmm −**p** [−**v** | −**q**][−**s** *server*][−**f** *device*]

nsrmm { −**d** | −**o** *mode* }[−**v** | −**q**][−**s** *server*][−**Py**][−**S** *ssid[/cloneid]* | −**V** *volid* | *volume ...*]

nsrmm −**S** *ssid[/cloneid]* [−**w** *browse-time*][−**e** *retention-time*][−**y**]

DESCRIPTION **nsrmm** a command line interface to manage the media and devices(tapes, disks, and files) used by NetWorker servers and storage nodes.

A *volume* is a physical piece of media, for example, a tape or disk cartridge. When dealing with file type devices, *volume* refers to a directory on a file system. NetWorker must have exclusive use of this directory, as files will be created and removed. The NetWorker system keeps track of which user files have been saved on which volumes, so they can be more easily recovered. Every volume managed by NetWorker has a *volume name* (also known as a *volume label*) selected by an operator. A volume name is specified when the volume is first introduced to the system. It can only be changed when a volume is relabeled. The volume should have an external label displaying its volume name for future reference. NetWorker refers to volumes by their volume names, for example, when requesting a volume for recovery.

The NetWorker system automatically manages an index that maps saved user files to volumes. NetWorker also keeps other attributes associated with a volume, including the expected capacity of the volume.

The NetWorker server requests that specific volumes be mounted by their name for recoveries, or any writable volumes for saves. These requests are submitted through the **nsr_notification(5)** mechanism. **NetWorker Management Console's** Administration window or the **nsrwatch**(1m) command can be used to monitor pending mount requests. Typically, the requests will also be written to the system console, or logged in a file. The same requests can be used as input for software that controls a *jukebox* (a device that automatically loads and unloads volumes).

Before the **nsrmm** command can be used(that is, before any data can be saved or recovered), at least one device must be configured for the NetWorker server. The NetWorker configuration may be modified with **NetWorker Management Console's** Administration window or the **nsradmin(1m)** command after NetWorker has been installed.

OPTIONS

−**B** Verifies that the volume you want to label does not have a readable NetWorker label. Before labeling the volume, an attempt is made to read any existing label the volume may already possess. If you specify this option and the volume has a valid NetWorker label that is readable by the device currently being used, the label operation is canceled and an error message is displayed. If the volume does not contain a label that is readable by the current device, the volume may be labeled. This option is used by **nsrd(1m)** when automatically labeling volumes on behalf of **nsrmmd(1m)** requests.

−**b** *pool* Specifies the pool to which the volume belongs. −**b** *pool* can name any pool

currently registered with **nsrd**. The possible values can be viewed by selecting Media Pools from the left pane of **NetWorker Management Console's** Media display or using the **nsradmin(1m)** command. The pool name is referenced by **nsrd** when determining which save sets can reside on the volume. If you omit this option, the volume is automatically assigned to the *Default* pool. If you specify a pool name without specifying a volume name, the next volume name associated with the pool's *label template* resource is used.

−**C** Displays a list of NetWorker configured devices and the volumes currently mounted in them. This list displays only the devices and volumes assigned to the server, not the actual devices and volumes. The −**p** option verifies the volume label. −**C** is the default option.

−**c** *capacity*
Overrides the default capacity of a volume. NetWorker normally uses built-in default capacities based on the device type. This option overrides these defaults. The format of the specification is *number multiplier. Multiplier* can be one of 'K' (1024 bytes), 'M' (1000 KB), or 'G' (1000 MB). Lower case letters are also accepted, as are extra characters like spaces, or an extra 'B' after 'K', 'M', or 'G'. *Number* may be any value, including an integer or real number, with up to three decimal places.

−**d** Deletes the client file indexes and media database entries from the NetWorker databases. It can be used in conjunction with −**S** ssid/cloneid to delete a specific saveset. Note that the ssid used can be the long format, to avoid ambiguity. The long format of ssid can be obtained by running the mminfo with −**r** "ssid(53)". The mminfo manpage has details of this usage.
The action does not destroy the volume: instead, it removes all references used by NetWorker to the volume and the user files contained on it. This option can be used to control the size of the NetWorker databases.

−**e** *time* When used in conjunction with the −**S** option, it sets the clone retention time of the specified save set or save set clone instance. The retention time should be specified in the format that is acceptable to the function **nsr_getdate**(1m)**.** If a clone identifier is not specified all clone instances will be updated with specified clone retention time. The save set retention time will reflect the longest recoverable clone instance retention time. It is possible for a clone instance to have a retention time less that browse time. However, the save set retention time may not be set such that the save set would become recyclable while it is still browsable. Refer to the −**w** option for more details on browse time.
When used in conjunction with volumes, the volume labeled will be an Archive volume if the value of *time* is **forever** (Archive volumes mean that the volume label never expires). Any other value of *time* are not applicable to a volume.

−**f** *device*
Specifies a device explicitly. When more than one device has been configured, **nsrmm** will select the first device by default. This option overrides the selection made by **nsrmm.**

−**H** Performs a software reset on the given device. Ongoing operations on the given device will be interrupted resulting in possible data loss. This option resets the internal Networker device state, not the physical device.

−**j** Ejects a volume from the device. This option is similar to performing an unmount operation, except that the volume is also physically ejected from the device, if possible. This feature is not supported by some device types, disk devices, and tapes. CAUTION: the −j option should be used only on devices that are in idle mode -- using the −j option on an active device may cause a

core dump.

−l Labels(initializes) a volume for NetWorker to use and recognize. Labeling must be performed *after* the desired volume is physically loaded into the device, either by an operator or a jukebox. When more than one enabled device exists, specify [**−f** *device*] to indicate which device to use for the label operation.

−m Mounts a volume into a device. Mounting is performed *after* a volume is placed into a device and labeled. You can mount only labeled volumes. When more than one enabled device exists, specify [**−f** *device*] to indicate which device to use for the mount operation. The labeling and mounting operations can be combined into a single command line. See the **EXAMPLES** section.

−o *mode*

Sets the mode of a volume, save set, or save set clone instance. The *mode* can be one of the following: **[not]recyclable**, **[not]readonly**, **[not]scan**, **[not]full**, **[not]offsite, [not]manual** or **[not]suspect**. The **[not]recyclable** mode applies to volumes, save sets and save set clone instances. Setting a volume to recyclable will also set the volume to full. A volume becomes recyclable when all the save sets on that volume become recyclable. A save set is recyclable when all the save set clone instances become recyclable. Therefore, setting the last not recyclable save set clone instance to recyclable can also cause the save set and volume to also become recyclable. Setting a recyclable save set clone instance to not recyclable will also force the associated save set and volume to become not recyclable. If a save set is not recyclable, at least one save set clone instance must be not recyclable. So, if all clone instances of a saveset have expired, and a particular clone instance needs to be recovered, that particular saveset clone instance needs to have its clone retention time reset to the future, by using the **-e** option along with **-S** ssid/cloneid, before the saveset can be made notrecyclable. Setting a save set to not recyclable is not recommended, since once a save set becomes recyclable it is possible that all of the volumes for an associated save set have been overwritten. Once a save set becomes recyclable, all associated save sets are not guaranteed to be available for recovery. For example, if an incremental save set depends on a full save set. The full save set will not be marked recyclable until all dependent save sets have also past their retention times. However, once the all the associated save sets have passed their retention times, all the save sets becomes recyclable. Any one of the save sets can be overwritten. Setting all the remaining save set not recyclable does not guarantee a complete recovering of the original data. Setting a save set not recyclable will only set the clone instances that have not past their retention time back to recyclable. The **[not]readonly**, **[not]scan**, **[not]offsite, [not]full** and **[not]manual** modes apply only to volumes. The **[not]manual** mode is the only valid mode when used with the **−l** option. The **[not]suspect** mode applies only to save set clone instances, meaning it must be specified along with **−S** *ssid/cloneid*, not just **−S** *ssid* by itself. (Remember that every instance of a save set has a clone identifier, even the original.) See **nsrim(1m)** for a discussion of the per-volume flags. The **suspect** flag is set automatically when a **recover(1m)** encounters a media error recovering data from a particular save set clone.

−P When used in conjunction with the **−d** option the corresponding file index entries are purged, without deleting the entries in the media database. The **scanner**(1m) command can then be used to recover the file index entries.

−p Verifies and prints a volume's label. To confirm that the external volume label matches the internal label, load a volume into a drive and use this option to

display the volume name in the label. Verifying a label unmounts mounted volumes.

−q Quiet mode. This option tells **nsrmm** to print out as little information as possible while performing the requested operation. Generally, only error messages are printed.

−R Relabels a volume. This option rewrites the volume label and purges the NetWorker indexes of all user files previously saved on the volume. Some of the volume usage information is maintained.

−r Mounts a volume as read-only. To prevent NetWorker from writing to a volume, specify the read-only flag when mounting the volume. Volumes marked as full and those in the read-only mode(**−o readonly**) are automatically mounted read-only.

−s *server*
Specifies the NetWorker server to perform the **nsrmm** operation on. See **nsr(1m)** for a description of server selection.

−S *ssid* Changes(**−o)** or removes(**−d)** a save set from the NetWorker databases, or used in changing the browse time (specified with **−w**) or the retention time (specified with **−e**) of the specified save set record. Note that the ssid used can be the long format, to avoid ambiguity. The long format of ssid can be obtained by running the mminfo with **−r** "ssid(53)". Check mminfo manpage for details on query and report of a saveset record in long format. The save set is identified by a save set identifier, *ssid*. A save set instance, or clone, can be specified using the format *ssid/cloneid* (but, it is ignored when used for the option **−w).** The **mminfo**(1m) program may be used to determine save set and clone identifiers.

−u Unmounts a volume. A volume should always be unmounted before you unload it from a device.

−V *volid*
Removes a volume from the NetWorker databases when used in conjunction with the **−d** option. The volume is identified by a volume identifier, or *volid*. The **mminfo**(1m) command can be used to determine volume identifiers.

−v Verbose mode. This option polls the NetWorker server to print out more information as the operation proceeds.

−w *browse time*
Specifies the browse time for the specified save set(supplied with the **−S** option). Note that once the save set becomes recoverable, the browse time may not be changed. The browse time should be specified in the format that is acceptable to the function **nsr_getdate**(1m)**.** The browse time has to be after the insert time in the save set record, but it cannot be after the retention time. If the option **−e** was not used, the existing retention time in the save set record is used for comparing with the specified browse time. See under the option **−e** for more details on retention time.

Labeling new tapes:
To introduce a new tape, named **mars.001**, to the NetWorker system, load the tape in an empty drive, then use the command:

```
nsrmm −l mars.001
```

The tape is labeled with **mars.001** and an entry is made in the appropriate NetWorker indexes. The **mminfo**(1m) command may be used to inspect the volume database and display information about the volumes:

```
mminfo −m
```

Mounting a tape:

To mount a NetWorker volume, use the −**m** option. Note that the volume must have been labeled previously and loaded in the drive:

```
nsrmm −m
```

When mounting, a volume name can also be specified:

```
nsrmm −m mars.001
```

The mount will fail unless the given volume name matches the one read from the media.

By mounting a volume, you make the volume available to NetWorker. When **nsrmmd(1m)** needs the volume, the label will be read again and confirmed, preventing accidental data loss. Volumes are also verified and mounted automatically if the server recovers after a crash.

Labeling and mounting a tape:

A volume may be labeled and mounted with a single **nsrmm** command by combining the −**m** and −**l** options. The following example labels a volume as **mars.003** and mounts it on device **/dev/nrst0:**

```
nsrmm −m −l −f /dev/nrst0  mars.003
```

Unmounting or ejecting a volume:

When a volume needs to be unmounted, use either the −**u** or −**j** option, depending on whether or not the device can physically eject a volume.

```
nsrmm −u
```

When more than one volume is mounted, you can specify either the volume name or device to select the desired volume. The following example ejects the volume named **mars.003**.

```
nsrmm −j mars.003
```

Displaying the current volumes:

The −**C** option displays the configured devices and the mounted volumes. This is the default option.

```
nsrmm −C
```

Deleting a volume:

To remove references to a volume and the user files saved on it from the NetWorker indexes, use the −**d** option. This option does not modify the physical volume, and should only be used when the physical volume is destroyed. By deleting a volume, you free up space in the NetWorker file index and the NetWorker media index, but not much more than if you had purged it. The amount of space released depends on the number of user files saved on the volume. The following example deletes the volume **mars.003:**

```
nsrmm −d mars.003
```

The **scanner(1m)** command can be used to rebuild the database entries.

Purging file index entries:

The file index contains information about each file saved by NetWorker. Due to size constraints, it may be necessary to purge information from the file index. When a volume or save set is deleted, the corresponding file index entries are also removed. It is also possible to preserve the media database entries of a volume while purging the file index by specifying the −**P** option when deleting.

The following example purges all of the file index entries for volume **mars.001**:

```
nsrmm −d −P mars.001
```

The **scanner**(1m) command can be used to recover the file index.

SEE ALSO **nsr(1m), nsr_getdate(3), nsr_layout(5), nsr_device(5), nsr_notification(5), mminfo(1m), mmlocate(1m), nsrmmd(1m), nsradmin(1m), nsrim(1m), recover(1m), scanner(1m)**

DIAGNOSTICS

type family volume **mounted on** *device*, **write enabled**
Message indicating that the **−m** (mount) option was successfully performed on a device with the given media *type* and media *family*, for example, 8mm tape.

'*saveset*' **is not a valid save set id**
The given save set identifier is not in the valid format. The format is either a single number(for the save set without reference to its instances), or two numbers separated by a slash(/) (representing a save set and clone(instance) identifier pair).

duplicate name; pick new name or delete old one
It is illegal to label two tapes with the same name. If you wish to reuse a name, remove that volume from the index using the **−d** option.

Are you sure you want to over-write *volume* **with a new label?**
An attempt is being made to relabel a volume. A positive confirmation will overwrite the existing data on that tape.

Purge file index entries for *type family volume***? ...**
After confirmation, the file index entries are removed.

volume **not in media index**
The media index has no entry associated with *volume,* so the **−m** command cannot be used. This problem may be caused by mistyping the volume name when the tape was originally labeled, or deleting it.

No valid *family* **label**
The tape or disk in the named device does not have a valid NetWorker label.

NAME nsrmmd − NetWorker media multiplexor daemon

SYNOPSIS **nsrmmd [−v] [−s server] [−r system]** *number*

DESCRIPTION The **nsrmmd** daemon is the storage node daemon and is responsible for network save and recover media multiplexing operations.

The **nsrmmd** daemon does the following:

- o Receives backup information from the NetWorker client.
- o Writes data to the devices (volumes).
- o Sends tracking information to the NetWorker server to track the data written to the devices (volumes).
- o Reads the data from the devices (volumes) at the request of the client during a recovery.

The **nsrmmd** RPC program ID is 390104 and the daemon version number is 5. To support multiple instances of nsrmmd (if the Concurrent Device Support feature is enabled), the daemon numbers are incremented by 100. The first daemon registered is 105, then 205, and so on.

One **nsrmmd** per enabled device is started automatically by **nsrd.** Additional **nsrmmd** daemons can be started when a mount request is pending. To change the number of daemons, alter the number of enabled devices.

OPTIONS

−**n** *number*
Specify the daemon number.

−**s** *server*
Specify the controlling server. This option is used on a storage node (see **nsr_storage_node**(5)).

−**r** *system*
Some **nsrmmd** programs run on the server but are controlling a device attached to a *Network Data Management Protocol (NDMP)* system. Such instances of **nsrmmd** have an optional −**r** argument specifying the system that is being controlled.

−**v** Verbose: Print out messages about what the daemon is doing.

SEE ALSO **nsr**(1m), **nsr_layout**(5), **nsr_service**(5), **nsr_storage_node**(5), **nsrd**(1m), **nsrmm**(1m), **mm_data**(5)

NAME nsrmmdbasm − NetWorker module for saving and recovering media databases

SYNOPSIS **nsrmmdbasm** [*standard-asm-arguments*]

DESCRIPTION The **nsrmmdbasm** is a standard, external ASM (Application Specific Module) that assists in the saving and recovering of the NetWorker media multiplexor's database files.

See **uasm**(1m) for a general description of ASMs and the [*standard-asm-arguments*]. It is intended that **nsrmmdbasm** only be invoked by **nsrmmdbd**(1m) or **mmrecov**(1m) operations.

Features of **nsrmmdbasm** performance specific to the NetWorker application during a **save** are:

Architecture independence:

The high speed access methods and data structures implemented by the database code are machine-dependent. This ASM saves only the records (and not access indexes) in an architecture-independent manner. Therefore, NetWorker media databases may be saved from one machine architecture and recovered to another.

Conservation:

Since only changed records are saved, and not internal indexes, considerable network bandwidth and tape space are conserved.

The **recover** operation of this ASM is the inverse of the **save** operation.

FILES

/nsr/mm/.nsr This directive file causes most files in the directory to be skipped during normal save operations. **nsrmmdbasm** ignores this directive.

/nsr/mm/mmvolume6
The directory containing the media database which is saved and recovered by this ASM.

/nsr/mm/mmvolume6.r
A temporary file that stores the contents of a recovered media database until **nsrmmdbd**(1m) has completed building a new media database.

/nsr/mm/mmvol<n> A temporary file that this ASM reads when backing up data. The unique file name is generated by appending <n>, a six digit hex number, to the file name.

/nsr/mm/volume.tmp A temporary file created when converting an older media database schema to the present schema during recovery.

SEE ALSO **nsr**(5), **nsr_layout**(5), **mmrecov**(1m), **nsrmmd**(1m), **nsrmmdbd**(1m), **nsrindexasm**(1m), **recover**(1m), **savegrp**(1m), **uasm**(1m)

NAME nsrmmdbd − NetWorker media management database daemon

SYNOPSIS **nsrmmdbd**

DESCRIPTION The **nsrmmdbd** daemon provides an RPC-based database service to the **nsrd**(1m) and **nsrmmd**(1m) daemons, and query-only access to the NetWorker clients. The RPC program number provided by **nsrmmdbd** is 390107. The RPC version numbers provided by **nsrmmdbd** are 3, 4, and 5. **Nsrmmdbd** is normally started by **nsrd**(1m).

The daemon manages a ''volume, save set and client id database'' located in the directory **/nsr/mm/mmvolume6**. The primary purpose of the database is to remember which save sets reside on which backup volumes. The database also provides the client name mapping to the internally used client identifier. Numerous access methods are provided to save set, volume and client id map records within the database.

FILES

/nsr/mm/mmvolume6
Directory containing the media database.

/nsr/mm/cmprssd
For performance and space reasons, the database is periodically rebuilt (or compressed). This file is created each time the database is compressed; its associated ctime is used to determine the next time that the database will be compressed. The database compression can be invoked by removing this file and running **nsrim.** This is not recommended while the NetWorker server is actively saving or recovering data.

/nsr/mm/mmvol<n>
This temporary file is created to hold the media database information that will be saved to a volume by **nsrmmdbasm**(1m). A unique file name is generated by utilizing <n>, a 6 digit hex number appended as part of the file name.

/nsr/mm/mmvolume6.r
The file (created by **nsrmmdbasm**) that is read when the media database information is being recovered.

/nsr/mm/volume.tmp
A temporary directory created when recovering or compressing the media database.

/nsr/mm/nsrim.prv
An empty file is used to track the last time that the program nsrim was started to perform maintenance on the NetWorker databases.

/nsr/logs/daemon.raw
The file to which Networker daemons writes the log messages.

SEE ALSO **mmrecov**(1m), **nsr**(1m), **nsrd**(1m), **nsrim**(1m), **nsrmmd**(1m), **nsrmmdbasm**(1m), **nsrmm**(1m), **nsr_render_log**(1m), **mminfo**(1m)

DIAGNOSTICS The **nsrmmdbd** diagnostic messages will normally be logged to the **/nsr/logs/daemon.raw** file.

Besides the messages listed below, **nsrmmdbd** may generate other diagnostics. Any diagnostics other than those listed below indicate a serious problem with the media database. It may be necessary to recover your media database using **mmrecov**(1m) if that occurs.

media db is converting, this may take a while

Any media databases created prior to the NetWorker 7.? (KeyStone) release have to be converted (once) to the new database format. The database is converted to current version.

media database conversion completed successfully
Printed when the conversion is completed successfully.

media conversion failed! *reason*
Printed when the conversion is terminates unsuccessfully. A more detailed reason may be appended to the message. NetWorker cannot work until the media database is converted successfully.

media db is saving its data, this may take a while
Printed when the daemon is dumping its records to a temporary file when the database is being backed up. The service is unavailable while the database is dumping.

media db is recovering, this may take a while
Printed when the daemon is reloading its database. The service is unavailable while the data is being reloaded.

media db is checking btrees
Printed each time the daemon is restarted. Upon start-up, the daemon performs sanity checks on the database search indexes.

media db is consistency checking the database
Printed each time the daemon is restarted. Upon start-up, after the database search indexes are checked, the database is checked for incomplete records.

media db is open for business
Printed after any of the previous messages are printed to indicate that the service is once again available.

media db is closed
Printed after the media database is successfully shutdown.

A copy of this process is already running!
Another copy of **nsrmmdbd**(1m) is currently running and has exclusive access to the media database. Only one **nsrmmdbd** process should be running on a given machine at a time. This can happen if the previous **nsrmmdbd** was not properly killed off. Use **nsr_shutdown**(1m) or **ps**(1) and **kill**(1) to identify and kill off all the NetWorker daemons before restarting **nsrd**(1m) again.

Cannot open lock file
An internal error, check the permissions on the **/nsr/tmp** and **/nsr/mm** directories.

NAME nsrmmgd − NetWorker daemon that manages jukebox operations

SYNOPSIS **nsrmmgd**

DESCRIPTION The **nsrmmgd** daemon provides an RPC-based service that manages all jukebox operations on behalf of the **nsrd** NetWorker server.

The **nsrd** server maintains all of the RAP resources that describe the state of any jukeboxes and their associated devices, pools, and operations. The **nsrmmgd** daemon is the process that is responsible for ensuring that the necessary jukebox operations actually get performed when needed by **nsrd.**

nsrmmgd runs on the same host as the **nsrd** server, and there will be at most one such daemon running. Multiple **nsrlcpd** daemons (one per enabled jukebox) may be started and controlled by **nsrmmgd** to handle the lower-level control of, and interface to, the various jukeboxes. The **nsrlcpd** processes that **nsrmmgd** manages may be distributed across multiple hosts, since **nsrlcpd** runs on the host that the jukebox is on.

The **nsrmmgd** daemon is invoked automatically by nsrd when needed, and never needs to be started directly by a user. If **nsrd** detects that there are any jukeboxes configured and enabled, then it will start **nsrmmgd** as part of the **nsrd** startup process. If no jukeboxes are enabled when **nsrd** starts up, then **nsrmmgd** will not be started until such time as a jukebox resource gets added, or an existing disabled jukebox resource is enabled.

The RPC program number for **nsrmmgd** is 390430.

FILES **/nsr/logs/daemon.raw**
The file to which **nsrmmgd** and other NetWorker daemons send information about various error conditions that cannot otherwise be logged using the NetWorker event mechanism.

SEE ALSO **nsr**(1m), **nsr_service**(5), **nsr_render_log**(1m), **nsr_op**(5), **nsrd**(1m), **nsrlcpd**(1m)

NAME nsrmon − command to remotely control NetWorker commands and daemons

SYNOPSIS **nsrmon**

DESCRIPTION The **nsrmon** command is run only by NetWorker daemons. **nsrd**(1m) starts the command to remotely control other commands and daemons on NetWorker storage nodes running **nsrexecd**(1m). Commands and daemons started remotely include **nsrjb**(1m) and **nsrmmd**(1m). See **nsr_storage_node**(5) for additional detail on storage nodes.

SEE ALSO **nsr**(1m), **nsr_storage_node**(5), **nsrd**(1m), **nsrexecd**(1m), **nsrjb**(1m) **nsrmmd**(1m)

NAME nsrndmp_clone − use NetWorker and Network Data Management Protocol(NDMP) to perform save set cloning

SYNOPSIS

nsrndmp_clone [−**v**] [−**p**] [−**s** *server*] [−**J** *recover-storage-node-name* +] [−**b** *pool*]] −**y** *retention*] { −**f** *file* | *volname...* }

nsrndmp_clone [−**v**] [−**p**] [−**s** *server*] [−**J** *recover-storage-node* +] [−**b** *pool*] [−**y** *retention*] −**S** { −**f** *file* | *ssid[/cloneid]...* }

nsrndmp_clone [−**v**] [−**p**] [−**s** *server*] [−**J** *recover-storage-node* +] [−**b** *pool*] [−**y** *retention*] −**S** −**t** *start time* [−**e** *end time*] [−**c** *client name*] [−**g** *group name*]

nsrndmp_clone [−**v**] [−**p**] [−**s** *server*] [−**J** *recover-storage-node* +] [−**b** *pool*] [−**y** *retention*] −**S** −**e** *end time* [−**t** *start time*] [−**c** *client name*] [−**g** *group name*]

nsrndmp_clone [−**v**] [−**p**] [−**s** *server*] [−**J** *recover-storage-node* +] [−**b** *pool*] [−**y** *retention*] −**V** { −**f** *file* | *volid...* }

The **nsrndmp_clone** program makes new copies of existing save sets. These copies are indistinguishable from the original, except for the volume(s) storing the copies. The copies are placed on different media volumes, allowing for higher reliability than a single copy provides. The copies may be made onto any kind of media (for example, save sets on an 8mm tape may be copied to an LGTO Ultrium 2 tape). However, all media used as the destination of an **nsrndmp_clone** operation must be in a *clone pool*. See **nsr_pool**(1m) for a description of the various pool types.

Although the command line parameters allow you to specify volume names or volume identifiers, **nsrndmp_clone** always copies complete save sets. Save sets that begin on a specified volume will be completely copied, so volumes may be requested during the cloning operation in addition to those specified on the command line. Conversely, save sets residing on the specified volumes that begin elsewhere are not cloned.

Note that **nsrndmp_clone** does not perform simple *volume duplication*, but rather, copies full save sets to a set of destination volumes in a given pool. If the first destination volume chosen cannot hold all of the save sets to be copied, another volume will be chosen. This allows you to use different kinds of media for each copy, allowing for variable sized volumes, such as tapes.

The **nsrndmp_clone** program, in conjunction with **nsrmmd**(1m), guarantees that each save set will have at most one clone on a given volume. When you specify a volume name or identifier, the copy of the save sets on that volume are used as the source. When save sets are specified explicitly, those with existing multiple copies are automatically chosen (copies of save sets that exist on volumes in a jukebox are chosen over those that require operator intervention). You can also specify which copy (clone) of a save set to use as the source (see the −**S** option description, in the Options section).

The **nsrndmp_clone** program can also be used to clone regular NetWorker savesets over NDMP. This is accomplished via the use of the −**p** option. The resulting clone savesets in this case are known as **opaque savesets** (their clone flag shows 'o' in the **mminfo** report). NetWorker treats **opaque** save sets the same way as regular save sets when dealing with data recovery and scanning.

Cloning between storage nodes is accomplished by an NDMP Tape Server on the source node reading from a volume, and another NDMP Tape Server on the target node writing to a volume. The source node is determined by the location of a source volume; where the volume is currently mounted, or by its "location" field if unmounted (see **mmlocate**(1m)). The target node of a clone is determined by the

"clone storage nodes" attribute of the client resource in descending priority. See **nsr_storage_node**(5) and **nsr_client**(5) for additional detail on how these attributes are used and for other storage node information.

If the save set to be cloned was backed up by *nsrndmp_save* via *nsrdsa_save* (i.e. the save set's flags have 'N' and 's'), then use *nsrclone* to clone these save sets; they are cloned to any NetWorker storage device other than an NDMP tape device. Cloning from a non-NDMP tape device to an NDMP tape device, and vice-versa, is not supported. See **mminfo**(1m) for more details on the 'N' and 's' save set flags.

OPTIONS

−**b** *pool* Specifies the media pool to which the destination clones should be sent. The pool may be any pool currently registered with **nsrd**(1m) that has its status set to *clone*. The possible values can be viewed by selecting Media Pools from the left pane of **NetWorker Management Console's** Media display. If you omit this option, the cloned save sets are automatically sent to the *Default Clone* pool.

−**f** *file* Instructs **nsrndmp_clone** to read the volume names, volume identifiers or save set identifiers from the file specified, instead of listing them on the command line. The values must be listed one per line in the input file. The *file* may be "-", in which case the values are read from standard input.

−**s** *server*
Specifies a NetWorker server. See **nsr**(1m) for a description of server selection. The default is the current system.

−**J** *storage-node*
Specifies the NetWorker recover storage node. You must use this option to specify the source for the clone, for Path-To-Tape cloning.

−**v** Enable verbose operation. In this mode, additional messages are displayed about the operation of **nsrndmp_clone**, such as save sets that cross volumes, or save set series expansions.

−**p** Enable cloning of regular NetWorker savesets over NDMP.

−**y** *retention*
Sets the date (in **nsr_getdate**(3) format) when the cloned data will become recyclable. The special value **forever** is used to indicate that a volume that never expires (i.e. an archive volume) must be used. By default, the server determines this date for the save set based on the retention policies in effect. This option allows overriding the existing policies.

−**S** Causes **nsrndmp_clone** to treat subsequent command line parameters as save set identifiers, not volume names. Save set identifiers are unsigned numbers. You can find out the save set identifier of a save set using the **mminfo -v** command (see **mminfo**(1m)). The −**S** option is useful when you want to copy individual save sets from a volume or all save sets matching an **mminfo** query (see the examples below). The save set identifiers may also specify exactly which copy of a save set with multiple copies to use as the source. To specify exact copies, use the *ssid/cloneid* format for each save set identifier. In this case, the *ssid* and the *cloneid* are unsigned numbers, separated by a single slash (/). You can find out the *cloneid* for a particular copy by using the **mminfo -S** report, or a custom report.

−**V** Causes **nsrndmp_clone** to treat subsequent command line parameters as volume identifiers, not volume names. Volume identifiers can be found using the **mminfo -mv** report, for example. This option can not be used in conjunction with the −**S** option.

−**e** *end time*

Specify the end time (in **nsr_getdate**(3) format) for selecting save set IDs for cloning. This option can only be used with -S option. If not specified, end time is set as current time. Please note that, -e 0 is same as -e today.

−**t** *start time*
Specify the start time (in **nsr_getdate**(3) format) for selecting save set IDs for cloning. This option can only be used with -S option. If not specified, start time is set as end time - 24 hours. Please note that, -t 0 is same as -t today. When specifying a time range, at least -t or -e option must be specified.

−**c** *client name*
If client name is specified, only the save sets belonging to that client will be selected. More than one client name can be specified by using multiple -c options. This option can only be used with -t or -e option.

−**g** *group name*
If a group name is specified, only the save sets belonging to that group will be selected. Only one group name can be specified. It can be used with -c option. This option can only be used with -t or -e option.

EXAMPLES

Copy all save sets that begin on the volume **mars.001** to a volume in the **Offsite Clone** pool:

nsrndmp_clone −b 'Offsite Clone' mars.001

Copy all complete save sets created during the previous weekend (recall that **nsr_getdate**(3) dates without time-of-day match midnight at the beginning of that day). Only complete save sets can be copied by **nsrndmp_clone**(1m):

nsrndmp_clone -S 'mminfo −r ssid \
-q '!incomplete,savetime>last saturday,savetime<last monday'‘

Copy a specific clone of a specific save set:

nsrndmp_clone -S 1538800517/770700786

Copy all save sets created between time 01/21/05 14:50:03 and 01/24/05 14:50:03 for the group Default

nsrndmp_clone -S -t '01/21/05 14:50:03' -e '01/24/05 14:50:03' \
-g Default

Copy all save sets created in the last 24 hours for clients "rose" and "seam".

nsrndmp_clone -S -e now -c rose -c seam

Clone a specific regular NetWorker save set 3517744106 using Path-To-Tape to a volume in the **ndmpclone** pool. The source save set resides in the VTL that supports Path-To-Tape capability. The (NDMP) storage node associated with this VTL is 192.168.0.1.

nsrndmp_clone -J 192.168.0.1 -p -b ndmpclone -S 3517744106

SEE ALSO

nsrclone(1m), **nsrndmp_save**(1m), **nsrndmp_recover**(1m), **nsr_getdate**(3), **nsr_pool**(5), **nsr_storage_node**(5), **mminfo**(1m), **nsr**(1m), **nsrd**(1m), **nsrmmd**(1m)

DIAGNOSTICS

The exit status is zero if all of the requested save sets were cloned successfully, non-zero otherwise.

Several messages are printed signaling that **nsrd**(1m) is unavailable for cloning data; these are self-explanatory. You may also see a message from the following list.

adding save set series which includes parent *ssid*
If running in verbose mode, this message is printed when **nsrndmp_clone** notices that a requested save set is continued, requiring the entire series to be cloned (even if only part of the series was specified in the command line parameters).

adding save set series which includes descendent *ssid*
If running in verbose mode, this message is printed when **nsrndmp_clone** notices that a requested save set is a continuation, requiring the entire series to be cloned.

Cannot contact media database
The media database (and most likely other NetWorker services as well) on the named server is not answering queries. The server may need to be started, or if it was just started, it needs to finish its startup checks before answering queries.

cannot clone save set *number***, series is corrupt**
The given save set is part of a save set series (used for saving very large files or filesystems), but not all of the save sets in the series were found in the media database. This can happen if, for example, you relabel a tape that contains part of a save set series.

cannot open nsrndmp_clone session with *server*
This message is printed when the server does not accept clone sessions.

cloning not supported; upgrade required
Another enabler is required to use this feature.

cloning requires at least 2 devices
Cloning requires at least one read/write device and one read-only or read/write device, since data is copied from one volume directly to another.

server **does not support cloning**
The named server is not capable of cloning.

each clone host needs at least two enabled devices
When cloning between two storage nodes that share the same physical drive, each node must have at least two enabled devices.

error, no valid clones of ssid *number*
The listed save set exists, but cannot be cloned because there are no complete copies of the save set. The save set was either aborted or is in progress. Only complete save sets can be copied.

error, user *username* **needs to be on administrator list**
error, user *username* **needs to be on archive users list**
Only NetWorker administrators are allowed to make clones of backup save sets. NetWorker administrators are listed in the NSR server resource, see **nsr_service**(5) for more information.

no complete save sets to clone
Only complete save sets can be copied, and no complete save sets were found matching the requested command line parameters.

number **is not a valid save set**
The given save set identifier is not valid. Two forms are understood: simple save set identifiers and those with a cloneid specified. Simple save sets are unsigned numbers. The save set with the cloneid form is specified as two unsigned numbers separated by a single slash (/).

pool is not a cloning pool
The pool specified with the **−b** *pool* option is not a clone pool. You must always use a pool with a type of "Backup Clone" for the **−b** option.

Volume *name* **has clone; requesting**
additional volumes
This message is printed in verbose mode when a specified save set has already been cloned to the volume specified in the error message. Since a save set can

have at most one clone per volume, nsrndmp_clone automatically requests additional volumes.

save set *number* **does not exist**
The given save set (from a −**S** save set list) does not exist. Verify your save set identifiers using **mminfo**(1m).

save set *number* **crosses volumes; requesting additional volumes**
This message is printed in verbose mode when volume names or IDs were specified, but the given save set is only partially resident on the listed volumes. Since only complete save sets can be cloned, **nsrndmp_clone** automatically requests additional volumes.

save set clone *number/cloneid* **does not exist**
A specific clone of a save set was specified, but that save set has no clones with that clone identifier. Verify your save set identifiers using **mminfo**(1m).

volume *name-or-number* **does not exist**
The given volume (either a volume name or a volume id specified in the −**V** option) does not exist in the media database.

waiting 30 seconds then retrying
A temporary error occurred and **nsrndmp_clone** will automatically retry the request until the condition is cleared. For example, an error will occur if all devices are busy saving or recovering and **nsrndmp_clone** must wait for these devices become available.

NAME nsrndmp_recover − use NetWorker and Network Data Management Protocol (NDMP) to recover data

SYNOPSIS **nsrndmp_recover** [−**c** *client*] [−**s** *server*] [−**J** *storage-node*] [−**R** *recover-target*] { **-r** *raw device* **-S** *ssid[/cloneid]* **-m** *mount point* [**-v** { *on*|*off* }] [**paths** [**paths...**]] | **-F** }

DESCRIPTION **nsrndmp_recover** is used to coordinate recover operations with NetWorker and a *Network Data Management Protocol* (NDMP) system. Only the super-user may run this command. There are two ways to recover data: destructive recovers and file-level recovers. Destructive recovers occur when a raw partition is specified by the −**r** option along with a save set ID (−**S**) option. Only a single save set can be specified at a time since the target *raw device* pathname must be specified. Users may opt to use the administrative user interface, ®to perform the destructive recover operation through the save set recover window. Users can determine save set IDs using the user interfaces or the **mminfo**(1m) command. File level recovers are specified by the −**F** option in conjunction with the use of the **nwrecover**(1m) or **recover**(1m) commands. Users should not specify this option.

The status of a recover can be monitored using the Java based **NetWorker Management Console** or the **curses**(3X) based **nsrwatch**(1m) program for other terminal types. Only volume information is available at this time. The amount of data that has been recovered is not provided.

nsrndmp_recover is not responsible for moving data on the NDMP system. All such activity is handled by the NDMP system. **nsrndmp_recover** receives messages from the NDMP system and processes them appropriately. Such messages could request a new tape be mounted or to post a log message. Refer to the NDMP specification and documentation available at *www.ndmp.org* for more details.

In order to recover data from another system, make sure the user performing the **nsrndmp_recover** operation is on the *remote access* attribute list of the client resource. See **nsr_client**(5).

Supports recovering data from a NetWorker storage device if the save set was backed up by *nsrndmp_save* via *nsrdsa_save.* Nsrndmp_recover program will spawn *nsrdsa_recover* locally if the save set's flags are identified to have 'N' and 's'. See **mminfo**(1m) for more details on 'N' and 's' save set flags.
Notes:
It should be noted that browsers such as *recover, nwrecover and winworkr* will spawn nsrndmp_recover locally. Therefore, for better performance, try to launch browsers based on a volume location that has save sets to be restored. For instance, if a backup was performed to a NetWorker storage node that is different from the NetWorker server, then launch the browser on the NetWorker storage node for better performance. If the browser is launched on the NetWorker server, then data will flow from the NetWorker storage node to the NetWorker server and from the NetWorker server to the NDMP system. All command line options mentioned below apply for recovering from NetWorker storage node too.

OPTIONS −**c** *client*
Client is the name of the machine that saved the files.

−**F** Specifies that a file-level recovery is going to be performed. This option should only be specified by **nwrecover**(1m) or **recover**(1m).

−**m** *mount point*
Specifies the destination directory to relocate recovered files. If not specified, the data will be restored to original location.

Notes:
If the NDMP Server is SnapImage, then the mount point of the raw device specified by the **-r** option. The filesystem will be unmounted for the recover operation and mounted after the operation is complete.

−**r** *raw device*
This option specifies the pathname of the raw device the data is to be recovered to. You must use this option only for destructive recovers with SnapImage product.

−**R** *recover-target*
This option specifies the name of the destination host the data will be recovered to. If not specified, the data will be restored to the source host.

−**s** *server*
This option selects which NetWorker server to use.

−**J** *storage-node*
This option selects which NetWorker storage node to use.

−**S** *ssid[/cloneid]*
This mandatory option is used to specify save set recover mode. This mode can be used to implement fast batch file recovery without requiring the NetWorker file index entries. *ssid* specifies the save set ID for the save set to be recovered. When there are multiple clone instances for a save set, the cloneid can also be specified to select the particular clone instance to be recovered from. The cloneid of a particular saveset can be obtained from mminfo(1m) output.

When no **path** arguments are specified, the entire save set contents will be recovered. One or more **path** arguments can be specified to limit which directories and files are actually recovered. If **path** arguments are supplied, then the beginning of each path name as it exists in the save set must exactly match one of the **path** before it will be recovered. Shell like filename, matching using meta characters like '*', '?', and '[...]', is not supported.

−**v** *on|off*
This option specifies the value of the verify flag. If the verify flag is turned **on** then, prior to sending to ndmp server for recovery, the **paths** would be verified as having existence in the index database for the given save set ID. Only those entries that are found in the index database would be sent to the NDMP server for recovery. On other hand if verify flag is **off** then **paths** would be sent across to the NDMP server without verification. The default value of this flag is **on.**

EXAMPLES

Sub-directory level restore to original location.

nsrndmp_recover -s server -c client -S ssid[/cloneid]
/fs/dir1 /fs/dir2 /fs/dir3 /fs/dir4/file1 ...
When indexes are not available for the specified paths
nsrndmp_recover -s server -c client -S ssid[/cloneid]
-v off /fs/dir1 /fs/dir2 /fs/dir3 ...

Restore to the original location

nsrndmp_recover -s server -c client -S ssid[/cloneid]

Relocate to a different location on the client

nsrndmp_recover -s server -c client -S ssid[/cloneid]
-m /destdir
where /destdir is the destination directory on the client

Relocate to a different location on the remote host

nsrndmp_recover -s server -c client -S ssid[/cloneid]
-m /destdir -R desthost
OR
nsrndmp_recover -s server -c client -S ssid[/cloneid]
-m desthost::/destdir
where desthost is another NDMP Client configured in the NW server

Destructive restore with SnapImage

nsrndmp_recover -s server -c client -S ssid[/cloneid]
-m /mntpoint -r /dev/rdsk/c2t2d0s1

DIAGNOSTICS

Skipping file due to incomplete save set: /core
The user has marked a file that is associated with an incomplete save set. The user should run **nsrim -X** to resynchronize the file index and media database.

Entry /core not found in index, skipping
The user specified directory / file (/core) in **path** arguments, which could not be found in the index database. The user should run **nsrndmp_recover** with **−v off** from the command line if this check is wished to bypass. This can be useful if the indexes are lost and the data exists in backup media.

SEE ALSO

mminfo(1m), **nsr_client**(5), **nsrndmp_save**(1m), **recover**(1m), **nwrecover**(1m)

NAME nsrndmp_save − use NetWorker and Network Data Management Protocol (NDMP) to save data

SYNOPSIS **nsrndmp_save** −**T** *backup-type* −**c** *client-name* [−**LL**] [−**M**] [−**P** *Proxy-host*] [−**I** *Index-host*] [−**g** *group*] [−**l** *level*] [−**b** *pool*] [−**m** *masquerade*] [−**N** *name*] [−**s** *server*] [−**J** *storage-node*] [−**t** *date*] [−**e** *expiration*] [−**w** *browse_time*] [−**y** *retention_time*] [−**W** *width*] **path**

DESCRIPTION **nsrndmp_save** coordinates the backup process with NetWorker and a target *Network Data Management Protocol* (NDMP) system. Only the super-user may run this command. The user must specify a *backup type*, *client-name*, *server*, and *path*.

The behavior of the backup depends on the NDMP system being protected. Certain environment variables may be needed depending upon the target system. Documentation for such backups should be consulted for more details.

The status of a backup can be monitored using the Java based **NetWorker Management Console** or the **curses**(3X) based **nsrwatch**(1m) program for other terminal types.

nsrndmp_save is not responsible for moving data on the NDMP system. All such activity is handled by the NDMP system. **nsrndmp_save** receives messages from the NDMP system and processes them appropriately. Such messages could request a new tape be mounted or a file index entry to be created. Refer to the NDMP specification and documentation available at *www.ndmp.org* for more details.

Details about handling media are discussed in **nsrmm**(1m) and **nsr_device**(5).

In order to save data for another system, make sure the user performing the **nsrndmp_save** operation is on the *remote access* attribute list of the client resource. See **nsr_client**(5).

OPTIONS −**c** *client-name*

Specifies the client name for starting the save session. This is useful on clients with multiple network interfaces, and hence multiple host names. It can be used to create multiple index databases for the same physical client. Note that this does not specify the network interface to use. This is specified in the **server network interface** attribute of the client resource. See **nsr_client**(5).

−**M** An NDMP client will be backed up to a NetWorker storage node by the **nsrdsa_save** program. This option provides most of the NetWorker storage node features, such as backup to disk, multiplexing, automedia verification, and staging. The **nsrndmp_save** program spawns **nsrdsa_save** locally. A NetWorker storage node hostname should be listed in the "storage nodes" attribute of server's *client-name* resource.
Notes:
Save sets that are backed up by **nsrndmp_save** via **nsrdsa_save** will be treated as regular NetWorker save set and will have save set flags as 'N' and 's'. 'N' indicates NDMP client and 's' indicates backed up to NetWorker storage node. See **mminfo**(1m) for more details.

−**P** *proxy-host*

nsrndmp_save spawns **nsrdsa_save** on Proxy-host. Proxy-host must be a valid NetWorker client and should be listed in the remote access list of *client-name* resource. This host will act as a proxy to NDMP Data Server to receive the data and save it to NetWorker storage device. This option is valid only with -M.

−**I** *index-host*

Used by **savegrp** to spawn **nsrndmp_save** on *Index-host.* Index-host must be a

valid NetWorker client and should be listed in the remote access list of *client-name* resource. This host is designated to perform NDMP backup initiation and index processing. This host must have 'Operate NetWorker' privileges.
This parameter is accepted but must be ignored by **nsrndmp_save.** The reason for this is **nsrndmp_save's** command line is specified in the Networker client resource. **savegrp** accesses the client resource and examines **nsrndmp_save's** command line before starting **nsrndmp_save.** If the -I flag is present **savegrp** knows to start **nsrndmp_save** on the node named *index-host.* Once **nsrndmp_save** has been started there is no further need for the flag and its argument but, it is passed on to **nsrndmp_save** which needs to ignore it.

−**g** *group*
Is used by **savegrp**(1m) and **savefs**(1m) to denote the group of the save. See **nsr_client**(5) and **nsr_group**(5). It is also used by the NetWorker server to select the specific media pool.

−**l** *level* Indicates the level of the save. This option is used by **savegrp**(1m) and **savefs**(1m) to specify a particular level for a scheduled save. Valid values are: full, incr, and 1 through 9 inclusive. The incr option is only supported when used with NDMP servers that support token based backups using the NDMP BASE_DATE/DUMP_DATE backup environment variables. Currently Network Appliance and EMC Celerra filerservers are recognized as supporting this. For other NDMP servers an incremental backup is forced to a full backup and a message noting this is included in **nsrndmp_save's** output. When the level value incr is specified the -t option must also be included and must specify the save time of a previously produced full or incremental backup. See the -t option for additional information. If the NDMP server's implementation is lower than version 4, incremental backups are forced to full backups.

When the level values 1-9 are specified nsrndmp_save will use the NDMP server's native dump levels 1-9 if there is no token based backup support. If the NDMP server does support token based backups, then the -t option specifies the savetime of a previously created saveset and the current backup will contain files created or modified since the previous backup. If token based backups are supported and -t is not specified the requested level backup will be forced to a full backup.

−**b** *pool* Specifies a particular destination pool for the save.

−**L** When two −**L** options are specified, this option causes an extra line to be printed at the end of the form ''complete savetime=number'', where number is the savetime of the save set created by this backup. Used by **savegrp**(1m).

−**m** *masquerade*
Specifies the tag to precede the summary line. This option is used by **savegrp**(1m) and **savefs**(1m) to aid in savegrp summary notifications.

−**n** Indicates no save. Not supported, but provided for compatibility.

−**N** *name*
Indicates the symbolic name of this save set. By default, the most common prefix of the *path* arguments is used as the save set name. The indexes get stored against the actual path name.

−**q** Indicates quiet. Not supported, but provided for compatibility.

−**s** *server*
Specifies which machine to use as the NetWorker server.

−**J** *storage-node*
Specifies which machine to use as the NetWorker storage node.

−**t** *date* Indicates the date (in **nsr_getdate**(3) format) after which files must have been modified before they will be saved. This option is used by **savegrp**(1m) and **savefs**(1m) to perform scheduled saves by consulting with the media database to determine the appropriate time value based on the previous saves for the save set and the level of the scheduled save. **nsrndmp_save** lists a valid (but cryptic) date that may be used in future runs requesting an incremental backup. Look for an output line like: "nsrndmp_save: browsable save-time=1290539475". The large number is the save time and may be used with this option.

−**T** *backup-type*
The type of backup on the NDMP server, for example *celestra*.

−**e** *expiration*
Set the date (in **nsr_getdate**(3) format) when the saved data will expire. When a save set has an explicit expiration date, the save set remains both browsable and non-recyclable until it expires. After it expires and it has passed its browse time, its state will become non-browsable. If it has expired and it has passed its retention time, the save set will become recyclable. The special value **forever** is used to indicate that a volume that never expires (i.e. an archive or a migration volume) must be used. By default, no explicit expiration date is used.

−**w** *browse*
Sets the date (in **nsr_getdate**(3) format) after which the saved data will no longer be browsable. By default, the server determines the browse date for the save set based on the browse policies in effect. This option allows overriding the existing policies on a save-by-save basis.

−**y** *retention*
Sets the date (in **nsr_getdate**(3) format) when the saved data will become recyclable. The special value **forever** is used to indicate that a volume that never expires (an archive or a migration volume) must be used. By default, the server determines this date for the save set based on the retention policies in effect. This option allows overriding the existing policies on a save-by-save basis.

−**W** *width*
The width used when formatting summary information output.

SEE ALSO **curses**(3X), **mminfo**(1m), **nsr_getdate**(3), **nsr_client**(5), **nsr_device**(5), **nsr_group**(5), **nsr_service**(5), **nsrd**(1m), **nsrim**(1m), **nsrindexd**(1m), **nsrmm**(1m), **nsrmmd**(1m), **nsrndmp_recover**(1m), **nsrwatch**(1m), **recover**(1m), **savefs**(1m), **savegrp**(1m)

NAME nsrports − port configuration tool

SYNOPSIS **nsrports** [−**s** *server*] [−**S** | −**C**] [*range ...*]

DESCRIPTION The **nsrports** command is used to display and set ranges of ports used by the NetWorker software. A range of ports may be either a single integer or two integers separated by a dash (-). Any integer used to define a range of ports must be from 0 to 65535. The range 0-0 is treated equivalent to 0-65535. The port ranges are stored by **nsrexecd**(1m) in the *NSR system port ranges* resource. When **nsrports** is executed without any options, the program displays the configured ranges for the system on which the command is being run.

Users executing nsrports can change the system ports to be used. There are also two additional options for viewing and setting the port ranges. The first is by using the **NetWorker Management Console.** The second is by using **nsradmin**(1m). Execute the program as follows:

```
# nsradmin -s server -p nsrexec
```

where *server* is the system for which ports are to be displayed. The administrator attribute for this resource can be modified with either the **NetWorker Management Console** or **nsradmin**(1m) by any user currently on the administrator list.

OPTIONS −**s** *server*
Specifies the system to contact.

−**S** Sets the system's service ports to the ranges specified. By default NetWorker defines a range of 7937-9936 for service ports.

−**C** Sets the system's connection ports to the ranges specified. By default NetWorker defines a range of 0-0 for connection ports.

EXAMPLES *Setting two service port ranges 7937-9936 and 9999*

```
nsrports −S 7937-9936 9999
```

SEE ALSO **nsrexecd**(5), **nsradmin**(1m)

NAME

nsrpush − remotely distribute and install client software from a centralized server to NetWorker clients

SYNOPSIS

nsr_push −**i** { −**all** | *clients* }

nsrpush −**a** { −**U** | −**W** } −**p** *product* −**v** *version* −**P** *platform* −**m** *media kit path* [−**R** *repository path*] [−**c** *cross platform client*] [−**C** *mount point*]

nsrpush −**r** −**p** *product* −**v** *version* −**P** *platform*

nsrpush −**u** −**p** *product* −**v** *version* { −**all** | *clients* }

nsrpush −**l**

nsrpush −**L** { −**U** | −**W** } −**m** *media kit path*

nsrpush −**s** [−**t**] { −**all** | *clients* }

nsrpush −**e** *clients*

nsrpush −**x** *clients*

nsrpush −**d**

DESCRIPTION

The **nsrpush** program allows the user to add, remove software packages and upgrade NetWorker clients. You might find it easier to use the **NMC GUI's Software Administration Wizard** to perform these operations.

After starting an operation such as upgrade if the user hits **Cntrl C,** the user is given the choice to either exit the CLI or cancel the operation. If the user selects to exit the CLI, the CLI program exits but the upgrade operation is not cancelled and it continues to run. The user can at a later point in time monitor the progress of the upgrade via the **NMC GUI**

OPTIONS

Options are seperated into two groups. The first are the options which specify the operation to be performed, e.g. inventory or upgrade clients. The second group list the additional options which provide arguments for the operation e.g. specifying the clients to be inventoried or upgrade.

OPERATION OPTIONS

−i Probes all specified NetWorker clients in a datazone to determine what EMC software is installed on each client. This step is required before a NetWorker client can be upgraded.

−a Adds software packages to software repository. The software repository is a software based cenetralized hierarchial directory of software packages that can be pushed to clients. The software repository is organized by product, platform, version and packages.

−r Removes software packages from software repository. This option can be used to remove previosly added software products from the software repository.

−u Upgrades NetWorker Clients software. It is required to do an inventory of the client before an upgrade.

−l Lists all the sofware packages in software repository. This option is used for listing names of the products, their version and platform that exists in the software repository. The product names, version and platform are required while using this program non-ineratctively for remove and upgrade operations.

−L Lists all the software packages on the distribution media kit. This option is used for listing names of the products, their version and platform that exists on the distribution CD. The product names, version and platform are required

while using this program non-ineratctively for add operations.

−s List the EMC software installed on each specified NetWorker client.

−e Adds specified clients to **exclude clients** attribute of **CP Master** resource. This specifies clients that are to be excluded from upgrade operation.

−x Removes clients from **exclude clients** attribute of **CP Master** resource

−d Lists all the clients in the **exclude clients** attribute of **CP Master** resource

ADDITIONAL OPTIONS

−p *product*
This option may be used in conjunction with repository and upgrade operations to specify the product name.

−v *version*
This option is used in conjunction with repository and upgrade operations to specify the version of the product.

−P *platform*
This option is used iin conjunction with repository operations to specify the platform of the software product.

−R *repos path*
This option is used to specify the location of the repository. If the repository already exists then this option will be ignored.

−m *media kit path*
This option is used to specify the path (mount point) of the distribution media.

−U This option is used in conjunction with **−a** option to add **Unix** products from the distribution media in the repository.

−W This option is used in conjunction with **−a** option to add **Windows** products from the distribution media in the repository.

−c *cross platform client*
This option is used to specify the cross platform client name. This option needs to be specified if the server and client are not both Unix based or Windows based. The distribution media should be mounted and simultaneously available on the server and the client via an NFS share.

−C *cross platform mount point*
This option is used to specify the cross platform path name. This option needs to be specified if the server and client are not both Unix based or Windows based.

−all This option is used in conjunction with **−i** and **−u** option to perform inventory and upgrade operation for all applicable clients.

−t This option is used in conjunction with **−s** option and displays the output in tabular format.

EXAMPLES

Inventoring clients:
To inventory all of the clients known to NetWorker, use the **−all** option:
```
nsrpush −i −all
```
To specify particular clients, specify the client names seperated by space:
```
nsrpush −i ledma153 ledma160
```

Adding to repository:
To add Unix based products to repository on a Unix NetWorker server, use the **−a** option in conjunction with the **−U** option:
```
nsrpush -a -U −p NetWorker −v 7.4.1
```

−P solaris_64 -m /cdrom/networker_vol_1

To add Windows based products to repository on a Unix NetWorker server, use the −**a** option in conjunction with the −**W** option. In addition the user should specify the windows client and windows mount point where the distribution media will be mounted using the −**c** and −**C** option respectively:

nsrpush -a -W −p NetWorker −v 7.4.1
−P win_x86 -m /cdrom/networker_vol_1 -c ledma170 -C "G:\"

Removing from repository:

To remove products from the repository, use the −**r** option:

nsrpush -r −p "NetWorker Module for Oracle" −v 4.5
−P linux_x86

Upgrading Clients

To upgrade clients, use the −**u** option:

nsrpush -u −p NetWorker −v 7.4.1 ledma170

FILES **/nsr/res/cpdb** The client push configuration database containing resource descriptors.

NAME nsrretrieve − retrieve NetWorker archive save sets

SYNOPSIS **nsrretrieve** [−**fnqu**] [−**i** {*nNyYrR*}] [−**d** *destination*] [−**s** *server*] { [−**S** *ssid[/cloneid]*]... [−**A** *annotation*]... } [*path ...*]

DESCRIPTION **nsrretrieve** is used to restore archive save sets from a NetWorker server. No browsing is available via **nsrretrieve**. Use of **nsrretrieve** is restricted to listed administrators and users of an archive client resource. See the **nsr_client**(5) man page for further details. When not running as root, only the files that the user owns can be retrieved.

When no *path* arguments are specified, the entire save set contents will be retrieved. To restrict the archive save set retrieval to only particular directories or files matching a given path prefix, exact matching *path*'s can be specified to limit which directories and files are retrieved.

OPTIONS

-A *annotation*
The *annotation* is a regular expression which uniquely identifies a single archive save set. See **nsrarchive**(1m). The regular expression is as used by **grep**(1). At least one annotation or ssid (see below) must be specified.

-S *ssid[/cloneid]*
The *ssid* specifies the save set IDs for the save sets to be retrieved. When there are multiple clone instances for an archive save set, you can specify the *cloneid* to select the particular clone instance to be retrieved. At least one annotation (see above) or ssid must be specified.

-d *destination*
Specifies the destination directory to relocate retrieved files.

-s *server*
Selects which NetWorker server to use.

-q The **nsrretrieve** command normally runs with verbose output. This flag turns off the verbose output.

-f Indicates that retrieved files will overwrite existing files whenever a name conflict occurs.

-n Does not actually create any directories or files while retrieving.

-i {**nNyYrR**}
Specifies the initial default overwrite response to use when retrieving files and the file already exists. You may specify only one letter. This option is the same as the **uasm -i** option when running in recover mode. See the **uasm**(1m) man page for a detailed explanation of this option.

-u Stop when an error occurs during retrieval. Normally, nsrretrieve treats errors as warnings and tries to continue to retrieve the rest of the files requested. However, when this option is used, nsrretrieve will stop recovering on the first error it encounters.

SEE ALSO **grep**(1), **nsrarchive**(1m), **nsr_client**(5), **nsr_service**(5), **nsr**(1m), **nsrd**(1m), **uasm**(1m)

DIAGNOSTICS **Exit Codes:**

0 Normal exit. This means that all of the requested data was successfully retrieved.

<>**0** Abnormal exit.

Messages:

The **nsrretrieve** command reports invalid options by printing a ‘‘usage’’ message describing the available options.

Cannot contact media database on *server*
: This message indicates that some problem was encountered connecting to the NetWorker server on the named machine.

cannot retrieve backup save sets
: The **nsrretrieve** command can only be used to restore archive save set data.

cannot retrieve migration save sets
: The **nsrretrieve** command can only be used to restore archive save set data.

more than one saveset have the annotation
: The specified annotation matched more than one archive save set. Use **nwretrieve**(1m) for retrieving save set that has non-unique annotation key.

cannot find saveset with unique annotation
: The specified annotation matched no archive save set.

NAME nsrscsi_recover − restores binary image to a host-accessible raw device from long-term storage with NetWorker

SYNOPSIS **nsrscsi_recover** −**S** *ssid/cloneid* −**T** *target-device* [−**V** *vendor-plugin-name*]

nsrscsi_recover −**I** *input-filename* [−**V** *vendor-plugin-name*]

DESCRIPTION The **nsrscsi_recover** program performs a restore to a raw device using SCSI commands directly accessing the device. It reads the data stream from NetWorker storage device (See **nsrmmd**(1m)) and stores the data blocks directly to the raw device. Only the super-user may run this command.

Before performing a recover, the user needs to determine the save set (ssid) and the destination device. The destination device must be specified even though it is an original device that was backed up.

If the user wants to know all the device sets, then attribute 'cover' can be used in **mminfo**. For instance, "mminfo -aVvot -q cover" would display all the cover save sets. If the user wants to recover a specific device set, then a media database query should be performed on the device set name to determine the device set's ssid. The device set is registered in media database as a cover save set (used as a container set), ssflags in **mminfo** should contain 'K' along with other flags which indicate the save set is a cover set. Next, a detailed media database query on this ssid should be performed using "mminfo -S -q "ssid=xxx" to determine all the connected save set ids (See **mminfo**(1m)). The connected save sets are backups of all the device entries of a device set.

The **nsrscsi_recover** program will fork a recover thread per save set id to be recovered. The recover thread is responsible for establishing a recover session with NetWorker server and restoring the data to the destination device. For instance, if the user gives ten ssids to be recovered and each ssid is targeted to restore to unique target device, then ten recover threads will be running in parallel. The total number of threads run are directly proportional to number of save set ids to be restored.

If the *target-device* is a SYMMETRIX device, then it must contain SYMMETRIX id and SYMMETRIX device id (See options section below for syntax of the *target-device*). The **nsrscsi_recover** program directly interacts with the SYMMETRIX device via SYMAPI library and discovers the host accessible raw device path for the corresponding SYMMETRIX device id. It then performs SCSI recover to the raw device path. (Refer to SYMMETRIX documentation for more details on SYMMETRIX and SYMAPI.)

OPTIONS −**S** *ssid/cloneid*

The saveset identifier points to the save set that needs to be recovered. This option will be ignored if -I is specified.

−**T** *target-device*

This option is mandatory when -S is supplied. The *target-device* is the raw device path or SYMMETRIX device (standard or BCV device). The SYMMETRIX format of a target device is SymID/DevId.

Examples of a *target-device*

raw device /dev/rdsk/c1t2d0s2

SYMMETRIX device 00343456567/0366
where SymId is 00343456567 and
DevId is 0366.

−**I** *input-filename*

If the user wants to recover multiple save sets at the same time or all the save sets that belong to a specific device set, then the user can include the save sets in a file and identify the file using this option.

Inputfile can be an absolute path to the file that needs to be read, or inputfile can be "-" which means read device entries from stdin.

Each entry in the *input-filename* must contain save set ids mapped to target devices in the following format:- SSID=> *target-device*

Examples:-

```
/nsr/res/ora-restore1.res  2343542342=>00343456567/0366
                           2363572344=>00343456567/0367
                           2373562345=>00343456567/0368
                           Please note separator "=>" is
                           required between the source
                           and the target

/nsr/res/ora-restore2.res  2343542342=>/dev/rdsk/c1t0d1s2
                           2343642343=>/dev/rdsk/c1t1d2s2
                           2343742344=>/dev/rdsk/c1t2d3s2
```

Notes:
The *target-devices* in the *input-filename* must be either raw device entries or SYMMETRIX device entries but cannot have both.

−**V** *vendor-plugin-name*

By default, the vendor used to interact with the device at the time of the backup will be used for the recover as well. The *vendor-plugin-name* should be supplied in case it is required to override original *vendor-plugin-name* used at the time of backup.

Notes:

1. Each save set to be recovered (ssid) should be mapped to a unique target device. Multiple save set recovers to the same target device would overwrite the device.

2. The disk capacity of the target has to be same as the source. If target capacity is less than source, then recover will not complete. If the target capacity is more than source, then the target will lose disk capacity as it will have to be relabeled with smaller capacity.

3. The nsrscsi_recover program checks whether target-device is a raw device or not. If target-device is a host accessible raw device, then nsrscsi_recover will ignore vendor-plugin-name. Otherwise, the nsrscsi_recover program uses vendor-plugin-name from the -V option or the media database.

4. Currently "emc_symm" is the only supported storage vendor.

For other vendors the raw device path can be attempted.

5. The recover of logical device set is not yet supported but there is a workaround to get the list of ssids from cover set using mminfo query based on device set name.

SEE ALSO **mminfo**(1m), **nsr_client**(5), **nsrscsi_save**(1m), **recover**(1m)

NAME nsrscsi_save − backs up binary image of a host-accessible raw device to long-term storage with NetWorker

SYNOPSIS **nsrscsi_save** [−**c** *client-name*] [−**g** *group*] [−**N** *save-set-name*] [−**I** *input filename*] [−**s** *server*] [−**b** *pool*] [−**e** *expiration*] [−**y** *retention time*] **Path**

DESCRIPTION **The nsrscsi_save program** performs raw device backup using SCSI commands directly accessing the device. It retrieves the data blocks directly from the device and sends the data stream to a NetWorker storage device (See nsrmmd(1m)). Only the super-user may run this command. The user must specify a *Path.*

If a backup of a raw device needs to be performed, then the *Path* must be the raw device path that is accessible to nsrscsi_save.

If a backup of a SYMMETRIX device needs to be performed, then the *Path* must contain SYMMETRIX id and SYMMETRIX device id (See options section below for syntax of the *Path).* The nsrscsi_save program directly interacts with the SYMMETRIX device via the SYMAPI library and discovers the host accessible raw device path for the corresponding SYMMETRIX device id. It then performs a SCSI backup of the raw device path. (Refer to SYMMETRIX documentation for more details on SYMMETRIX and SYMAPI.)

The status of a backup can be monitored using the Java based **NetWorker Management Console** or the **curses**(3X) based **nsrwatch**(1m) program for other terminal types.

The user can also specify a device set to be backed up where the device set is a set of device entries. If the *Path* specified is a *device-set-name* (See Options section below), then for each device entry in the device set a backup thread is forked to dedicate the backup operation for each device for optimized performance and better utilization of system and NetWorker resources. The total number of threads run are directly proportional to number of device entries in the device set. However, server, client and device parallelism attributes are enforced on the NetWorker server.

Details about handling media are discussed in **nsrmm**(1m) and **nsr_device**(5).

In order to save data for another system, make sure the user performing the **nsrscsi_save** operation is on the *remote access* attribute list of the client resource. See **nsr_client**(5).

Since it is a raw device backup, the save sets are not browsable. These save sets are recoverable via nsrscsi_recover command only (See nsrscsi_recover (1m)). Also, if the number of devices to be backed up is more than one, then *nsrscsi_save* will start and end a save session to create a container set(deduced from cover set). This container set would encompass all the save sets backed up by threads. The mminfo tool would display the container sets as any other regular save sets (See mminfo (1m)).

OPTIONS −**c** *client-name*

Specifies the client name for starting the save session. However, the client name is optional. If client name is unspecified, then the local host will be assumed to be the client. If *client-name* is specified, the save session is started against this client and the resulting save set is registered against *client-name.*

Note:- This client name is not necessarily the host where device is accessible. For instance, a SYMMETRIX BCV device is accessible on a different host for backup but the user wants to register the backup against the client where standard device is attached.

−**N** *save-set-name*

Symbolic name to the save set. By default, *save-set-name* is the *Path* name itself.

However, if the *Path* is a *device-set-name* , then -N will be ignored as *device-set-name* will be tagged against each device entry to form a save set name for that backup session.

−**I** *input-filename*

Input file name (eg. /tmp/testdisks.res) is the absolute file path name that must have the list of devices to be backed up. If the *input-filename* is not specified, then default *input-filename* is constructed from *device-set-name.* For example, if the *device-set-name* is oracledisks, the *input-filename* would be /nsr/res/oracledisks.res.

Inputfile can be an absolute path to the file that needs to be read, or inputfile can be "-" which means read device entries from stdin.

If *device-set-name* is specified in the *Path* and -I is not specified, the default location for input file is /nsr/res/device-set-name.res

Notes:
The *input-filename* should contain only the device entries that need to be backed up, and multiple device entries should be separated by newline. When -I is specified, *device-set-name* should also be specified.

Path The *Path* can be anyone of the following formats:-

"Raw device path" E.g. "/dev/rdsk/c1t2d0s2"

"{device-set-name}" E.g. "{OracleDevices}"
Note that the braces are mandatory to distinguish device-set-name from single device path.
In this case, device entries are read from /nsr/res/device-set-name.res file i.e., the resource file name should be same as the device-set-name.
Also note that the device entries in this file must be raw device paths.

"<<vendor>>{device-set-name}"
Same syntax as above. However, the difference is that the device entries must be in vendor specific format.

E.g. "<<emc_symm>>{OracleDevices}"
Currently, emc_symm is the only suppored vendor.

SYMMETRIX format of a device entry or a device path is SymID/DevId where SymId is 00343456567 and DevId is 0366.

"<<vendor>>vendordevice" E.g. "<<emc_symm>>00343456567/0366"
In this case, only one vendor device is backed up. Here, vendor is "emc_symm" and device to be backed up is

"SymID/DevId".

Note:- *Path* must be given in double quotes.

−**g** *group*
Is used by **savegrp**(1m) and **savefs**(1m) to denote the group of the save. See **nsr_client**(5) and **nsr_group**(5). It is also used by the NetWorker server to select the specific media pool.

−**b** *pool* Specifies a particular destination pool for the save. Note that all the save sessions go the same pool.

−**L** When two −**L** options are specified, this option causes an extra line to be printed at the end of the form ''complete savetime=number'', where number is the savetime of the save set created by this backup. Used by **savegrp**(1m).

−**m** *masquerade*
Specifies the tag to precede the summary line. This option is used by **savegrp**(1m) and **savefs**(1m) to aid in savegrp summary notifications.

−**n** Indicates no save. Not supported, but provided for compatibility.

−**q** Indicates quiet.

−**s** *server*
Specifies which machine to use as the NetWorker server.

−**e** *expiration*
Set the date (in **nsr_getdate**(3) format) when the saved data will expire. When a save set has an explicit expiration date, the save set remains non-recyclable until it expires. If it has expired and it has passed its retention time, the save set will become recyclable. The special value **forever** is used to indicate that a volume that never expires (i.e. an archive or a migration volume) must be used. By default, no explicit expiration date is used.

−**w** *browse*
Indicates the browse date. Not supported, but provided for compatibility.

−**y** *retention*
Sets the date (in **nsr_getdate**(3) format) when the saved data will become recyclable. The special value **forever** is used to indicate that a volume that never expires (an archive or a migration volume) must be used. By default, the server determines this date for the save set based on the retention policies in effect. This option allows overriding the existing policies on a save-by-save basis.

−**W** *width*
The width used when formatting summary information output.

SEE ALSO **curses**(3X), **mminfo**(1m), **nsr_getdate**(3), **nsr_client**(5), **nsr_device**(5), **nsr_group**(5), **nsr_service**(5), **nsrd**(1m), **nsrim**(1m), **nsrindexd**(1m), **nsrmm**(1m), **nsrmmd**(1m), **save**(1m), **nsrwatch**(1m), **recover**(1m), **savefs**(1m), **savegrp**(1m)

NAME nsrssc − NetWorker save set consolidation program

SYNOPSIS **nsrssc** −**c** *client* −**N** *saveset* [−**p** *pool*] [−**y** *retention*] [**-r**] [−**vq**]

DESCRIPTION **nsrssc** consolidates the most recent level 1 (partial) save set and its corresponding full-level save set into a new full-level save set. This consolidation process effectively achieves the same outcome as a full-level backup at the time partial backup was done.

Normally, **nsrssc** is invoked within **savegrp**(1m) as part of a consolidation-level backup. During the consolidation-level backup, **savegrp**(1m) automatically generates a level 1 backup, then calls **nsrssc** to create a consolidated backup, using the latest full-level save set.

Using **nsrssc** allows for greater flexibility in scheduling backups and save set consolidation. Unlike the **savegrp(1m)** command, which completes a consolidation backup promptly after the level 1 backup is completed, **nsrssc** allows you to schedule the consolidation at a different time. Scheduling a time between the full backup and consolidation backup frees up NetWorker to complete other processes.

If you execute **nsrssc** manually, the most recently backed-up save set must be a level 1 save set; otherwise, the consolidation will not be successful.

The **nsrssc** command requires at least two active devices. The consolidation process uses simultaneous device reads and writes to create its consolidated save set. This mechanism creates a restriction upon the location of the newly created save set. The new save set cannot be created on the same volume where the partial or full save set was derived. Also, volumes containing the previous full and level 1 save sets must reside on the same storage node.

OPTIONS −**c** *client*
Indicates the name of the client whose save set should be included for the consolidation process.

−**N** *saveset*
Indicates the name for the generated consolidated save set.

−**p** *pool* Specifies the destination media pool to build the consolidated full save set. The pool may be any pool currently registered with **nsrd**(1m). The selected pool *must* be the same pool type as the previous full-level save set. You can view pool values by selecting Pools from the Administration menu of **networker**(1m). Pool values are also listed in the **NSRpool** resource. See **nsr_pool**(5).

If this option is omitted, then the consolidated save sets are automatically built on volume(s) whose media pool is the same as that of the previous full-level save set.

−**r** Removes the level 1 save set. If the level 1 save set is on tape, then the save set will expire. If the level 1 save set is on [adv_]file type volume, then the save set (including its index entries, its media database entries, and the actual save set data on disk) is removed. Please note that **nsrssc** will *never* attempt to remove the level 1 if consolidation fails.

If the level 1 save set has already been cloned prior to the consolidation, then this option is ignored and the original level 1 save set will not expire, and it will not be removed from [adv_]file type volume.

−**v** Enables verbose operation. In this mode, additional messages may be

generated during the consolidation process.

−y *retention*
Sets the date (in **nsr_getdate**(3) format) when the consolidated data will become recyclable. The special value **forever** is used to indicate that a volume that never expires (i.e. an archive volume) must be used. By default, the server determines this date for the save set based on the retention policies in effect. This option allows overriding the existing policies.

−q Runs quietly. This is the default mode.

EXAMPLES

The following examples demonstrate how save set consolidation can be performed. In both examples, a save set defined in group name *elmanco* is consolidated for client *delepanto.* The save set data for group *elmanco* is */etc* and */users* .

Example:

To perform a save set consolidation, use the following commands:

```
savegrp −l 1 −I −G elmanco
nsrssc −c delepanto −N /etc
nsrssc −c delepanto −N /users
```

Note that this example is almost the same as doing a **savegrp −G elmanco −l c.** The only differences are:
1) No index and bootstrap is backed up after data is consolidated.
2) If there is a failure during the consolidated process, a full backup is not performed.

Example:

To direct level 1 data to a disk cache (file-type device) and have the level 1 save set removed after a full-level save set is built on tape, perform the following operations:

1. Set up a pool which only accepts level 1 data and its devices are only file type devices.
2. Run the following commands:

```
savegrp −G elmanco −l 1 −I
nsrssc −c delepanto −N /etc −r
nsrssc −c delepanto −N /users −r
```

This process removes the level 1 completely. Also, since fast media (disk-file type) is involved, this process may be a much faster way of generating a full-level save set when compared to doing a regular full-level backup.

DIAGNOSTICS

On successful completion, **nsrssc** returns zero; otherwise, a non-zero value is returned.

Some error codes are:

98 Failed because the level 1 and previous full are not in the same storage node.

99 Failed, most likely due to a renamed/deleted directory condition

You may also see one of the following messages:

You are not authorized to run this command
Only root may run **nsrssc**

Cannot contact media database
Most likely, **nsrmmd**(1m) is unavailable to answer queries, or an additional NetWorker daemon may have completed processing. In either case, the system administrator needs to determine if the NetWorker services need to be restarted. Note, there may be a small interval during startup when the services may be unavailable to answer any queries.

SEE ALSO **nsr_schedule**(5), **nsr_group**(5), **mminfo**(1m), **savegrp**(1m), **nsrclone**(1m)

NAME nsrstage − NetWorker save set staging command

SYNOPSIS

nsrstage [−**v**] [−**F**] [−**d**] [−**s** *server*] [−**J** *storage-node*] [−**b** *pool*] [−**y** *retention*] −**m** −**S** { −**f** *file* | *ssid[/cloneid]...* }

nsrstage [−**v**] [−**s** *server*] [−**J** *storage-node*] [−**y** *retention*] −**C** −**V** **volume**

DESCRIPTION The **nsrstage** program is used to migrate existing save sets on a manual basis. Migration is the process of moving one or more save sets between volumes. The process begins with a clone of the specific save sets to the new volume specified, followed by the deletion of cloned save set entries from the media database (see the **-S** description). Finally, the possible removal of the save sets from the original source volumes. The second and the third operations are triggered by the successful completion of the previous operation. The data is moved to new media volumes, making room for new data on the original volumes.

Migration can be onto any media type (for example, save sets on a disk family volume can be migrated to a tape volume). The **nsrstage** program does not perform simple volume migration; it migrates full save sets.

You can specify exactly which copy (clone) of a save set to use as the source. See the −**S** option description.

If the **nsrstage** program encounters an error after successfully cloning some of the specified save sets, then it will delete only those successful save sets from the source volume before it gets aborted.

OPTIONS

−**b** *pool* Specifies the name of the media pool to which the data should migrate. The pool may be any pool currently registered with **nsrd**(1m). You can view values by selecting Media Pools from the left pane of **NetWorker Management Console's** Media display. If you omit this option, the cloned save sets are automatically assigned to the Clone pool corresponding to the original pool. Eg., if staging a save set from the Default pool, it is assigned to the Default Clone pool; if staging a save set from an Archive pool, it is assigned to the Archive Clone pool.

−**m** Performs the actual migration operation.

−**s** *server*
Specifies a NetWorker server with save sets to migrate. See **nsr**(1m) for a description of server selection. The default is the current system.

−**J** *storage-node*
Specifies which host to use as the storage node for the recovery part of the staging process (see nsr_storage_node(5)).

−**v** Enables verbose operation. In this mode, additional messages are displayed about the operation of **nsrstage**, such as save sets that cross volumes, or save set series expansions. If concurrent **nsrstage** operations are performed on the same savesets, it is possible for the volume names to be inaccurate. In that case **nsrstage** will issue a warning. Please see **DIAGNOSTICS** for the exact warning message.

−**y** *retention*
Sets the date (in **nsr_getdate**(3) format) when the staged data will become recyclable. The special value **forever** is used to indicate that a volume that never expires (i.e. an archive volume) must be used. By default, the server determines this date for the save set based on the retention policies in effect. This option allows overriding the existing policies.

−**d** Deletes the input file that specifies the save set identifiers to be staged. This option must always be specified in conjunction with a **-f** option.

−**C** Instructs **nsrstage** to perform a volume cleaning operation. Scans a volume for save sets with no entries in the media data base and recovers their space. Space for recyclable and aborted save sets are also recovered from the volume with the save set entries removed from the media data base. You can perform this operation on disk family volumes.

−**S** *ssid* Causes **nsrstage** to treat subsequent command line parameters as save set identifiers. Save set identifiers are unsigned numbers. You can find out the save set identifier of a save set using the **mminfo -v** command. See **mminfo**(1m). The −**S** option is useful when you want to migrate individual save sets from a volume, or to migrate all save sets matching some **mminfo** query. The save set identifiers also specify exactly which copy of a save set to use as the source. To specify exact copies, use the *ssid/cloneid* format for each save set identifier. In this case, the *ssid* and the *cloneid* are unsigned numbers, separated by a single slash (/). You can find out the *cloneid* for a particular copy by referring to the **mminfo -S** report.
Notes:
If the **-S** *ssid/cloneid* format is used, then only the specific ssid instances will be removed from media database.
If the **-S** *ssid* is specified and no cloneid is specified, that all cloned instances of the ssid will be deleted from the media database, except the one being staged. If other disk family device clone instances were removed from the media database as a result of staging, then those save sets will also be removed from their respective volumes, and space will be recovered.

−**f** *file* Instructs **nsrstage** to read the save set identifiers from the file specified, instead of listing them on the command line. The values must be listed one per line in the input file. The *file* may be -, in which case the values are read from standard input.

−**F** If specified will force nsrstage to skip all invalid savesets and continue staging.

−**V** *volume*
Specifies the name of the volume to be cleaned. This option cannot be used with −**S** or −**m** options.

EXAMPLES

Migrate save sets **1234 and 4568** to a volume in the **Offsite Clone** pool:

```
nsrstage −b 'Offsite Clone' -m -S 1234 4567
```

Migrate clone instance **12345678 of save set 1234** to a volume in the **Default Clone** pool:

```
nsrstage −m −S 1234/12345678
```

Migrate all save sets created since last Saturday to a volume in the **Default Clone** pool:

```
nsrstage −m −S `mminfo −r ssid \
        -q 'savetime>last saturday'`
```

Recover space from volume **jupiter.013**:

```
nsrstage −C −V jupiter.013
```

Only complete save sets can be migrated by **nsrstage**(1m).

DIAGNOSTICS

The exit status is zero if all of the requested save sets migrated successfully; otherwise status is non-zero.

Several messages are printed denoting a temporary unavailability of **nsrd**(1m) for migrating data. These are self-explanatory. In addition, you may see one of the following messages:

Adding save set series which includes *ssid*

If running in verbose mode, this message prints when **nsrstage** notices that a requested save set is continued and requires the entire series to be migrated (even if only part of the series was specified by the command line parameters).

Cannot contact media database on *server*
The media database (and probably other NetWorker services as well) on the named server is not answering queries. The server may need to be started. Or if it was just started, it needs to finish its startup checks before answering queries.

Cannot open nsrstage session with *server*
This message prints when the server is not accepting migration sessions. A more detailed reason prints on the previous line.

number **is not a valid save set**
The given save set identifier is not valid. Two forms are understood: simple save set identifiers and those with a cloneid specified. Simple save set are unsigned numbers. You specify the save set with the cloneid form as two unsigned numbers separated by a single slash (/).

save set *number* **does not exist**
The given save set (from a **−S** save set list) does not exist. Verify your save set identifiers using **mminfo**(1m).

save set clone *number/cloneid* **does not exist**
You specified a specific clone of a save set, but that save has no clones with that clone identifier. Verify your save set identifiers using **mminfo**(1m).

volume *name* **does not exist**
The given volume (if you specified the **−V** option) does not exist in the media database.

waiting 30 seconds then retrying
A temporary error occur. **nsrstage** automatically retries its request until the condition is cleared. For example, if all of the devices are busy saving or recovering, **nsrstage** cannot use these devices and must wait for two of them to become free.

Space can only be recovered from disk family devices.
The given volume (if you specified the **−V** option) is not a disk family volume. This message is also printed after a successful migration of data from volumes of type other than disk family.

WARNING: Multiple concurrent cloning operations on the same savesets have been detected. The list of volumes reported below may not be accurate.
nsrstage prints this message when it detects more clone instances than it expected. This happens when more than one nsrstage commands are run on same saveset concurrently. Verify the clone volumes using **mminfo**(1m). Please note that the result of the staging operation is not affected by this warning.

SEE ALSO

nsr_stage(5), **nsrclone**(1m), **nsr_getdate**(3), **mminfo**(1m), **nsr**(1m), **nsr_device**(5), **nsr_pool**(5), **nsrd**(1m), **nsrmmd**(1m)

NAME nsrtask − execute an action based on a NetWorker nsr_task resource

SYNOPSIS **nsrtask** [*options*] [−**C**] [−**v**] [−**D** *debuglevel*] **task**

DESCRIPTION The **nsrtask** program executes actions on a periodic basis. It is normally run automatically by **nsrd**(1m) as specified by a **nsr_task**(5) resource.

The **nsrtask** program will query the NetWorker server for a nsr_task resource with the same name as passed on its command line. From that resource, nsrtask will access the **action** attribute and the **plan** attribute. The value of the **plan** attribute is used to determine if the action should be executed or skipped based on the current day of the period (see **nsr_task (5)**). The value of the **action** attribute is used to determine what actions to take to perform the task.

The nsrtask command will setup an RPC connection to **nsrjobd (1m)** to request execution of any commands related to the entries in the **action** attribute. This attribute contains the type and name of the command resource to use for task execution and the exact processing of a task is dependent upon the command resource type.

OPTIONS

−**C** Causes nsrtask to check the **plan** attribute to determine if it should run or not.

−**v** Increments the verbosity level of the output. This option can be provided multiple times and its effects are additive.

−**D** Sets the debug level.

−**?** Prints the usage statement.

task Specifies the NetWorker task resource to execute.

RESOURCE TYPES

NSR task The attribute *plan* determines whether the task's actions are executed or skipped. The attribute *action* determines what command resources are to be used for execution.

FILES

/nsr/logs/nsrtask.raw Log file containing messages from running the nsrtask program.

SEE ALSO **nsr_service**(5), **nsr_hypervisor**(5), **nsr_task**(5), **nsr_resource**(5), **nsr**(1m), **nsradmin**(1m), **nsrjobd**(1m), **nsrvim**(1m),

NAME nsrtrap − snmp notification scheme for NetWorker messages

SYNOPSIS **nsrtrap** [−**c** *community*] [−**i** *version*] [−**t** *trap-type*] [−**s** *specific-type*] [−**v**] **network_management_station**

DESCRIPTION **nsrtrap** is a mechanism to send NetWorker notifications using the Simple Network Management Protocol (SNMP) trap mechanism. A NetWorker administrator could create a custom NetWorker notification scheme based on nsrtrap, by configuring the NetWorker events and priorities.

A NetWorker administrator could create notification schemes to receive messages on different network management consoles by configuring the events and priorities and specifying the desired network management station as the location to receive the trap messages.

To create a new SNMP notification, follow the steps below:

1. Open the Notifications window from the Customize menu.
2. Choose the Details option under the View menu.
3. Click on the Create button.
4. Enter the name of the new notification in the Name field.
5. In the Action field, enter the command nsrtrap along with the network management station name to which the networker SNMP notification should be sent. For example:
   /usr/sbin/nsrtrap -c networker SNMPhost
   where SNMPhost is the hostname of the SNMP network management station.
6. Set the events and priorities desired.
7. Click on the Apply button.

OPTIONS

−**c** *community* The SNMP community string. This option allows you to specify the SNMP community that is authorized to receive traps from the NetWorker server. SNMP communities are configured on the SNMP server. This option defaults to "public".

−**i** *version* The SNMP version. This option allows you to specify the SNMP version. The value 1 for SNMPv1 and the value 2 for SNMPv2. This options defaults to "2".

−**s** *specific-type* This option is a generic setting that can be used to identify the type of trap the NetWorker server is sending. This option can be set to any integer value and may be used in conjunction with different SNMP notifications to distinguish different traps coming from the NetWorker server. For example, you can create multiple SNMP notifications: one for critical messages, another for warnings, and another for other events or priorities. You can then use the -s option to differentiate the various notifications so that the SNMP management software can determine which type of trap is being sent.

−**t** *trap-type* One of the SNMP trap types[0-6]. The default is 6, the "enterprise-specific" trap type.

−**v** Sets the Output mode to verbose. In verbose mode, nsrtrap echoes the community, trap type, specific trap type, and the hostname or IP address to the command line.

SEE ALSO **nsr**(1m) **nsr_notification**(5) **nsr_resource**(5)

NAME nsrvim − client binary for communicating with VMware VirtualCenter

SYNOPSIS **nsrvim**
[*options*] [−**v**] [−**d**] [−**s server**] [−**D debuglevel**] [−**-help**] [−**-version**] **hypervisor**

DESCRIPTION The **nsrvim** program communicates over SOAP with a VMware VirtualCenter instance using the VMware Infrastructure Methodology API. It is normally run automatically by **nsrtask**(1m) as specified by a **nsr_hypervisor**(5) resource.

The **nsrvim** program will query the NetWorker server for a **nsr_hypervisor**(5) resource with the same name as passed on its command line. From that resource, nsrvim will access the **username, password,** and **endpoint** attributes and use them to attempt to connect to a VMware VirtualCenter instance via SOAP.

Once connected, the nsrvim program will query the VirtualCenter instance for the list of virtual machines and important elements of the inventory (Datacenters, Clusters, Hosts and VirtualMachines). The program will also setup an RPC connection to **nsrjobd (1m)** and send the virtual machine list and inventory data back to the server through that connection. Note, nsrvim itself does **not** update the **nsr_hypervisor** resource; this should be done by the job management program (normally **nsrtask**).

OPTIONS

−**s** Hostname of the NetWorker server to query for the **nsr_hypervisor** resource.

−**v** Increments the verbosity level of the output. This option can be provided multiple times and its effects are additive.

−**D** Set the debug output level.

−**d** Dump the inventory to a file. The file name is the name of the nsr_hypervisor resource with a .xml extension.

−**-help** Print the usage statement and exit.

−**-version**
Print the version string and exit.

hypervisor
Specifies the NetWorker nsr_hypervisor resource to use.

RESOURCE TYPES

NSR hypervisor
The attributes *username* and *password* are used to log into the VirtualCenter instance. The attribute *endpoint* is used to connect to the SOAP interface of the VirtualCenter instance. The attribute *vm list* is used to determine if any new virtual machines have been found and to store the current list. The attribute *environment* is used to store an XML document describing elements of the VirtualCenter inventory as needed by NetWorker.

FILES

/nsr/logs/nsrtask.raw Log file containing execution messages routed through nsrtask.

SEE ALSO **nsr_service**(5), **nsr_hypervisor**(5), **nsr_notification**(5), **nsr_resource**(5), **nsr**(1m), **nsradmin**(1m), **nsrjobd**(1m), **nsrtask**(1m),

NAME nsrwatch − command for character-based display of NetWorker status

SYNOPSIS **nsrwatch** [−**s** *server*] [−**p** *polltime*]

DESCRIPTION The **nsrwatch** command displays a NetWorker server's status. The server's name is specified by the optional −**s** *server* argument. If no server is specified, it defaults to the same server that would be used by a command, such as **recover**(1m) in the current directory. If there is no NetWorker service on the selected machine, the command issues an error message. The polling interval is specified by the optional −**p** *polltime* argument (in seconds). The default is two seconds.

Users can run **nsrwatch** from any terminal that has enough **termcap**(5) capabilities for cursor positioning; it does not require any particular window system. The **nsrwatch** program gets its information via remote procedure calls to the specified server. This way it can be used from any machine that can access the server through the network.

The **nsrwatch** display is divided into a status summary followed by five subwindows: the *DEVICES* window, the *GROUPS* window, the *SESSIONS* window, the *MESSAGES* window, and the *PENDING* window. Subwindow sizes are adjusted depending on the size of the terminal or window being used. A subwindow can be brought into focus by using the Tab key. Once a subwindow is in focus, its content can be scrolled using the arrow keys. Visibility of the subwindows can be toggled using the **d, g, s, m** and **p** keys for the *DEVICES, GROUPS, SESSIONS, MESSAGES* and *PENDING* subwindows respectively.

The status summary displays the name of the server, NetWorker version, the server's start time, approximate response time of the server, and save and recover session totals.

The *DEVICES* window displays the devices known to the current server. For each device, the panel displays its name, the device type, the name of the mounted volume, or **(unmounted)** if no volume is mounted, the pool the volume belongs to and device status. The name may be followed by **(J)** if the device is configured as part of a jukebox device. Read-only devices can be hidden by pressing the **r** key. The *GROUPS* window lists active save groups running on the server. The *SESSIONS* window provides current save set information for each active session (saving, recovering, or browsing). The *MESSAGES* window displays a history of messages of general interest to the operator. Finally, the *PENDING* messages window displays messages that require operator intervention.

The **nsrwatch** program runs continuously until quit, stopped, or interrupted (Control-C, for example). Typing the **q** character quits the program, the Control-L forces a screen clear and redraw, while any other character forces the status to be updated.

The **nsrwatch** program checks for new devices at a slower rate than the polling rate, so it might take up to a minute after a new device is added before the device is noticed. To recognize the devices immediately, either restart the program or press Control-L. Deleted devices may cause a ''resource does not exist'' message temporarily, but otherwise they are noticed immediately.

The **nsrwatch** program adapts to changes in the screen size, if supported by the underlying environment. For example, if a window terminal emulator is resized, the size of each field may change to match the window. If the window is too small, all the devices, sessions, and messages, might not be displayed. For best results, use a window of at least 30 lines.

OPTIONS

−s *server*
Sets the current NetWorker server to *server*.

−p *polltime*
Sets the polling interval to be *polltime* seconds.

SEE ALSO

termcap(5), **nsr_notification**(5), **nsr_device**(5), **nsr_service**(5), **recover**(1m), **nsradmin**(1m), **nsr**(1m), **nsrd**(1m)

NAME nsr_archive_request − NetWorker resource type ''NSR archive request''

SYNOPSIS **type: NSR archive request**

DESCRIPTION Each NSR archive request is described by a single resource of type **NSR archive request** (see **nsr_resource**(5)). To edit the NSR archive request resources for a NetWorker server type:

```
nsradmin -c "type:NSR archive request"
```

See the **nsradmin**(8) manual page for more information on using the NetWorker administration program. The archive request resource may also be edited using **NetWorker Management Console.**

This resource allows administrators to set up an archive to occur later or to set up frequent archives of a set of data. The administrator can run an archive on a specified client within the next 24 hours. The archive is executed via the **nsralist**(8)) command.

ATTRIBUTES The following attributes are defined for resource type **NSR archive request**. The information in parentheses describes how the attribute values are accessed. **Read-only** indicates that the value cannot be changed by an administrator. **Read/write** means the value can be set as well as read. **Hidden** means it is an attribute of interest only to programs or experts. Hidden attributes can only be seen when the hidden option is turned on in **nsradmin**(8). **Choice** means that the value of the attribute can only be one from a list specific to that attribute (for example, status can be start now or start later). **Dynamic** attributes have values which change rapidly. **Encrypted** attributes contain data that is not displayed in its original form. The assumption is that the data is sensitive in nature and needs to be protected from accidental disclosure. Several additional attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the archive request.

annotation (read/write)
This attribute contains the annotation text associated with the archive save set generated from this archive request.
Example: annotation: Product Release 4.1;

archive clone pool (read/write)
This attribute indicates the archive clone media pool the archive request should use when cloning the archive save set generated by this archive request.
Example: archive clone pool: Archive clone;

archive completion (read/write)
A notification action to be executed to send status of the archive request to.
Example: archive completion: /usr/ucb/mail -s "Product Archive" systemadmin;

archive pool (read/write)
This attribute can be used to override the normal media pool selection applied to the archive save set generated from the archive request. Selecting a pool will direct the archive to that media pool.
Example: archive pool: Archive;

client (read/write)
This attribute indicates what NetWorker archive client the archive request is to be executed on.

Example: client: neptune;

clone (read/write)
This attribute controls whether the archive save set generated by the archive request is to be cloned. A value of *Yes* implies the archive save set should be cloned. A value of *No* does not imply cloning.
Example: clone: No;

cloned (read/write, hidden)
This attribute is unused.
Example: cloned: No;

completion time (read/write, hidden)
This attribute indicates when the archive request completed. The format is "day-of-week month day hours:minutes:seconds year".
Example: "Thu Oct 22 17:00:37 1994";;

directive (read/write)
This attribute specifies the directive to use when running the archive. The default value is nothing selected. The valid choices for the directive resource are names of the currently defined 'NSR directive' resources, see **nsr_directive**(5).
Example: directive: Default with compression;

grooming (read/write)
This attribute indicates any grooming actions to be taken once the archive save set generated by the archive request has been created, verified, and cloned. A value of *none* implies no action. A value of *remove* implies the files and directories specified in the **save set** attribute will be removed via the **rmdir**(2) and **unlink**(2) system calls.
Example: grooming: none;

log (read/write, hidden)
This attribute contains any information pertaining to the execution of the **nsralist** command.
Example: log:; **name** (read/write) This attribute specifies the name of this NetWorker archive request.
Example: name: Product Source Tree;

save set (read/write)
This attribute lists the path names to be archived on the archive client. The names should be separated by a comma and a space (", ").
Example: save set: /product/src, /accounting/db;

start time (read/write)
This attribute determines when the archive request will be run. The **status** attribute (see above) must be set to *start later* for the archive request to be scheduled. The 24 hour clock format is "hours:minutes".
Example: start time: 3:33;

status (read/write, choice)
This attribute determines if an archive request should be run. No value implies the archive request is not scheduled. Selecting *start now* causes the archive request to be run immediately. Selecting *start later* causes the archive request to be run at the time specified by the **start time** attribute.
Example: status:;

verified (read/write, hidden)
This attribute is unused.
Example: verified: No;

verify (read/write, choice)
This attribute indicates the archive request should verify the archive. See **nsr_archive**(5) for more information on archiving. Selecting the *Yes* choice causes the verification to occur. Selecting the *No* choice will not cause any verification. If the user also requests that the archive save set be cloned, the verification is done on the clone since the cloning operation will have verified the original archive save set.
Example: verify: Yes;

Save operations (read/write, string)
This attribute specifies the save operation instructions in the form of:
KEYWORD:TOKEN=STATE

This attribute is required if save set attribute of the archive client contains non-ASCII names. Specify:

I18N:mode=nativepath (for NetWorker 7.4 or later clients on UNIX platforms with non-ASCII save set names)

I18N:mode=utf8path (for pre-7.4 clients and NetWorker clients on Windows platforms with non-ASCII save set names)

Example: Save operations: I18N:mode=nativepath;

EXAMPLE

Note: the hidden options are not shown in this example.

A resource to define an archive request, called Product:

```
                type: NSR archive request;
                name: Product Source;
          annotation: Product Release 3.0;
              status: Start later;
          start time: "2:00";
              client: space;
            save set: /product/source;
           directive: Default with compression;
        archive pool: Archive;
              verify: Yes;
               clone: Yes;
  archive clone pool: Archive Clone;
            grooming: none;
  archive completion: mail -s Product Source Archive productteam;
```

SEE ALSO

nsr(5), **nsr_directive**(5), **nsr_resource**(5), **nsradmin**(8), **rmdir**(2), **unlink**(2)

NAME nsr_client − NetWorker resource type ''NSR client''

SYNOPSIS **type: NSR client**

DESCRIPTION Each NSR client is described by a single resource of type **NSR client** (see **nsr_resource**(5)). To edit the NSR client resources for a NetWorker server type:

```
nsradmin -c "type:NSR client"
```

See the **nsradmin**(8) manual page for more information on using the NetWorker administration program. The Client resource may also be edited using **NetWorker Management Console.**

For each NetWorker client, this resource describes which files should be saved, the schedule used to save these files, which directive should be used to omit files from the save, how long the files' index entries should be kept in the online file index and the media index, and who is allowed to back up, browse, and recover this client's files. A client may have more than one resource describing it.

ATTRIBUTES The following attributes are defined for resource type **NSR client**. The information in parentheses describes how the attribute values are accessed. **Read-only** indicates that the value cannot be changed by an administrator. **Read/write** means the value can be set as well as read. **Hidden** means it is an attribute of interest only to programs or experts. Hidden attributes can only be seen when the hidden option is turned on in **nsradmin**(8). **Dynamic** attributes have values which change rapidly. **Encrypted** attributes contain data that is not displayed in its original form. The assumption is that the data is sensitive in nature and needs to be protected from accidental disclosure. Several additional attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

Certain client attributes (such as "Client OS type", "CPUs", "NetWorker version" and "Enabler in use") do not get populated in the Client Setup/Information window of the NetWorker interface, when the NetWorker Server is running under Eval mode or an Enterprise license. However, when the NetWorker server has a Workgroup/NetWork/Power Edition enabler, these client attributes are refreshed appropriately in the window after the client backup.

name (read-only, single string)
This attribute specifies the hostname of this NetWorker client.
Example: name: venus;

client id (read-only)
The client id is used by the media database and for index backups to identify a save set with a specific client. Each client has a unique client id which is automatically generated by the NetWorker server (nsrd).

server (constant, single string)
This attribute specifies the hostname of this client's NetWorker server. The server's hostname will be used as the default value.
Example: server: jupiter;

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about this NetWorker client.
Example: comment: machine located in room 243;

scheduled backup (read/write, choice)
This attribute is provided for the administrator to disable this client for

scheduled backups. This value is specific to this client resource, it does not propagate to any other existing resources for the same client. By default this attribute is Enabled.
Example: scheduled backup: Disabled;

archive services (read/write, choice)
This attribute determines if this system can use archive services. This attribute can only be set if archive support has been enabled on the server. The choices are enabled or disabled. *Example:* archive services: enabled;

schedule (read/write, choice)
This attribute specifies the name of the schedule controlling the backup levels for the save sets listed in the 'save set' attribute. The default value is 'Default'. Any currently defined schedule names may be used, see **nsr_schedule**(5).
Example: schedule: Default;

browse policy (read/write, choice)
This attribute specifies the name of the policy controlling how long entries will remain in this client's online file index. The default value is 'Month'. Any currently defined policy name may be used as long as the period defined by the policy is not longer than the retention policy's period, see **nsr_policy**(5).
Example: browse policy: Month;

retention policy (read/write, choice)
This attribute specifies the name of the policy controlling how long entries will remain in the media index before they are marked as recyclable. The default value is 'Year'. Any currently defined policy name may be used as long as the period defined by the policy is not shorter than the browse policy's period, see **nsr_policy**(5). The pool resource also contains a retention policy attribute. If the pool that a clone save set will be written to also has a defined retention policy, the longer of the client and pool policy will be utilized. Cloned save sets will utilize the pool retention policy if one is defined.
Example: retention policy: Year;

directive (read/write, choice)
This attribute specifies the directive to use when backing up the client. The default value is NULL. The valid choices for the Directive resource are names of the currently defined 'NSR directive' resources, see **nsr_directive**(5).
Example: directive: UNIX with compression directives;

group (read/write, choice list)
This attribute specifies the group this client is a member of. The group controls the start time for automatic backups. The value may be one of the currently defined 'NSR group' resources, see **nsr_group**(5). The default value is 'Default'.
Example: group: Default;

save set (read/write, list)
This attribute lists the path names to be saved for this client. The names should be separated by comma space (,). The default value is 'All'. On all NetWorker supported clients, 'All' refers to the mounted file systems. Also, NetWorker supports *'wildcard'* at the *filesystem level.* For example, For a UNIX NetWorker client, '/*' refers to all mounted filesystems under '/'. And if '/space1' and '/space2' are valid filesystems, one could use '/space*' to get both these filesystems backed up on the particular client. Please note that *'wildcard ' matching* at the *subdirectory level, is not supported.* So, '/space1/subdir*' will not work.

When a client needs to have different file systems saved on different schedules,

a Client resource is needed for each set of file systems on a particular schedule. For all the Client resources with the same name in a group, a given path name may only appear once. When a Client resource lists the save set 'All', it must be the only Client resource with its name belonging to its group.
Example: save set: /, /usr, /usr/src;

Backup renamed directories (read/write, choice)
This attribute enables the support of renamed directories during scheduled backups. When enabled, the save program performs a lookup in the client file index to determine whether a directory has been renamed. If a directory has been renamed, all of the files and subdirectories under the directory will be backed up. The default value is 'Disabled'.
Example: Backup renamed directories: Enabled;

Checkpoint enabled (read/write, choice)
This attribute enables the checkpoint restart functionality during scheduled backups. When enabled, the savegrp program will attempt to continue interrupted save sets from the last checkpoint, rather than restart them from the begining. Savegrp will retry save sets up to 'client retries' times. See the **nsr_group**(5) man page for more information regarding 'client retries'.
Example: Checkpoint enabled: Enabled;

Checkpoint granularity (read/write, choice)
This attribute controls the frequency of checkpoints generated during a checkpoint restart enabled save.

The two options are:

Directory - guarantees that a checkpoint will be created after each directory is backed up. Checkpoints may be created at intermediate points depending on the number of files in the directory; however, this is not guaranteed. This is the default.

File - a checkpoint is generated after each file saved. This option introduces extra overhead and will slow down the save process.

Example: Checkpoint granularity: Directory;

priority (hidden, read/write, choice)
This attribute controls the backup priority of this client. Priority 1 is the highest, 1000 is the lowest. Automated savegrp's will attempt to back up clients with higher priorities before clients with lower priorities. Note that this is only one factor used to determine the next client. The **savegrp** command has many parameters to consider, and may choose a lower priority client while trying to balance the load.
Example: priority: 500;

remote access (read/write, string list)
This attribute controls who may back up, browse, and recover a client's files. By default this attribute is an empty list, signifying that only users on the client are allowed to back up, browse, and recover its files. Additional users, hosts, and netgroups may be granted permission to access this client's files by adding their names to this attribute. Netgroup names must be preceded by an ampersand ('&'). Each line specifies a user or a group of users, using one of these formats: *user/host@domain* , *group/host@domain* , *user@host* , *user@domain* , *group@host* , *group@domain* , *&netgroup* (only available on platforms that support netgroups) , *user_attribute=value[, ...]*.

where *user* is a user name; *host* is a host name; *group* is a user group name; *domain* is a domain name; *user_attribute* can be **user**, **group**, **host**, **nwinstname**, **nwinstancename**,

domain, or **domaintype** (type of the domain, **NIS** or **WINDOMAIN**).

The user attributes: **nwinstname** and **nwinstancename** are used to indicate a NetWorker instance name. The value that should be entered for either of these attributes is the value in the "name" field in the NSRLA resource for the machine where a matched user is connecting from.

value can be any string delimited by white space. If the value has space in it, then it can be quoted with double quotes. The value may contain wild cards, "*". Entering just a user name allows that user to administer NetWorker from any host (equivalent to *user*@* or **/user* or **user=***user*). Netgroup names are always preceded by an "&".

The format: *user_attribute=value[, ...]* is more secure because the format is not overloaded. **For example, if** *test@test.acme.com* **is entered, then any users in the** *test* **group or users named** *test* **and that are in the domain;** *test.acme.com* **or from the host;** *test.acme.com* **will match this entry.**
Example: The entries:

remote access: mars, *@jupiter, sam@pluto, */root;

remote access: host=mars, host=jupiter, "user=sam,host=pluto", user=root;

are equivalent.

remote user (read/write, string)
This attribute has several uses. For those clients that are accessed via the **rsh**(1) protocol (new clients use **nsrexecd**(8) instead), this attribute specifies the user login name the NetWorker server will use to authenticate itself with the client. The default value is NULL, implying that 'root' should be used. When **savegrp**-p (see **savegrp**(8)) is run on the NetWorker server, the server runs commands on the client to determine which files to save. Note that when the **nsrexecd**(8) protocol is used to access the client, the remote user attribute is not used for authentication.
Certain clients, such as NetWare fileservers, use this attribute along with the **password** attribute, below, to gain access to the files being backed up. Other clients that back up application data, such as Sybase databases, use this attribute along with the password to gain access to the application data. There may be a different value of this attribute for each resource that describes the same client.

NDMP clients use this attribute along with the **password** attribute to configure access to a NDMP server. The same username (**remote user** attribute) and **password** should be configured in the Device resource as they are configured for the NDMP server.
Example: remote user: operator;

password (read/write, encrypted)
The **savegrp** command uses this attribute when initiating the **savefs** and **save** commands on the client's machine. The **savefs** and **save** commands use the password to gain access to the files being backed up. If a password is given, then the "remote user" attribute for the Client resource must also be defined. There may be a different value of this attribute for each resource that describes the same client.

This attribute does not need to be set for existing UNIX clients that are not backing up any application specific data.
This attribute is also used in conjunction with the **remote user** attribute to

configure access to a NDMP server.

backup command (read/write, string)
The remote command to run to back up data for this client and save sets. This command can be used to perform pre and post backup processing and defaults to the **save** command. The value must not include a path and must start with the prefix "save" or "nsr".
Example: backup command: savemsg;

Save operations (read/write,**string)**
This attribute specifies the save operation instructions in the form of:
KEYWORD:TOKEN=STATE[;KEYWORD:TOKEN=STATE;...]

This attribute is required if save set attribute of this client contains non-ASCII names. Specify:

I18N:mode=nativepath (for NetWorker 7.4 or later clients on UNIX platforms with non-ASCII save set names)

I18N:mode=utf8path (for pre-7.4 clients and NetWorker clients on Windows platforms with non-ASCII save set names)

This attribute can also be used to configure the VSS saves on Windows 2003. The savegrp program uses the following attributes "Backup renamed directories", "Checkpoint granularity", and "Save operations" to pass the save instructions via the **-o save_operations** option to the save program. See **save (8)** for more information.
Example: Save operations: I18N:mode=nativepath;

Pool (read/write, choice list)
This attribute sets the media pool for data target selection during scheduled backup of the save sets specified in this client. This attribute is supported on 7.6.1 or later clients. Backup may fail on older (pre-7.6.1) clients for non-NULL value. This pool specification overrides any other pool criteria associated with the group or save set for this client. The default value is NULL.
Example: Pool: Default;

executable path (read/write, string, hidden)
This attribute specifies the path to use when the NetWorker server is executing commands on the client. When no path is specified, the "remote user's" $PATH is used.
Example: executable path: /etc/nsr;

server network interface (read/write, string, hidden)
The name of the network interface on the server to be used for saves and recovers.
Example: server network interface: mars-2;

aliases (read/write, string list, hidden)
This attribute is a list of aliases (nicknames) for the client machine that queries can match. If this list is empty, match on client name alone.
Example: aliases: mars;

owner notification (read/write, hidden)
A notification action to be executed to send the contents of status messages to the owner/primary user of a machine (for example, savegrp completion messages).
Example: owner notification: /usr/ucb/mail -s "mars' owner notification" carl@mars;

statistics (constant, hidden, dynamic)
This attribute contains three values: the size of the client's online file index in

kilobytes, the number of kilobytes actually used, and the number of entries in the index.
Example:
statistics: elapsed = 1761860, index size (KB) = 776,
amount used (KB) = 680, entries = 2216;

index save set (update-only, hidden, dynamic)
This attribute specifies the client file index save set to purge when the index operation is set to purging oldest cycle.
Example: index save set: /;

index path (read/write, hidden)
This attribute is used to allow the NetWorker administrator to balance NetWorker online file index disk utilization across multiple disk partitions. If set, this attribute contains the full path to the directory containing the client's online file index. Note that the last component of the path must match the *name* attribute of the Client resource (see above). If left blank, the index path defaults to the path /nsr/index/*name*, where *name* is the name attribute from the Client resource.
Example: index path: /disk2/index/venus;

index message (update-only, hidden, dynamic)
This attribute contains the ending status message for the previous index operation. This attribute is typically blank, indicating that the previous operation completed successfully.
Example: index message:;

index operation start (update-only, hidden, dynamic)
This attribute contains the starting time of the current index operation. This attribute is a null string ("") when the operation is 'Idle'. The format is weekday followed by hour and minutes.
Example: index operation start: Wednesday 02:45;

index progress (update-only, hidden, dynamic)
This attribute contains the progress the index has made towards finishing the current task. This attribute is blank when the operation is 'Idle'. The progress is expressed as a percentage.
Example: index progress: 45;

index operation (update-only, hidden, dynamic)
This attribute contains the current index operation. It is normally 'Idle'.
Example: index operation: Reclaiming space;

parallelism (read/write, hidden)
This attribute specifies the maximum number of saves that should be run at the same time for the client.
Example: parallelism: 2;

archive users (read/write, string list)
This attribute specifies a list of users that are allowed to use the archive services on the client. This attribute can only be set if archive support has been enabled on the server. To schedule an archive request for a client, root (or equivalent) must be on that client's Archive users list, or else root@client must be in the server's Administrator list. If no users are listed and the client resides in same machine as the server, only administrators and the local root user (that is, root@server) are allowed to use the archive services on the client. A value of '*' implies any user is allowed to archive or retrieve data. The '/' and '@' characters are not allowed as part of the user name.
Example: archive users: paul;

application information (read/write, hidden, string list)
This attribute contains client application information. The use of this attribute is client specific and should be utilized as indicated by the documentation received with the product. NDMP clients fill in various parameters and values in this attribute separated by an equals sign ('=').
Example: application information: HIST=yes;

ndmp (read/write, choice)
This attribute indicates whether or not the Client resource is configured for NDMP backups. If the client is used for NDMP backups, the **remote user** and **password** attributes must be filled in. The **application information** attribute may also be used.
Example: ndmp: yes;

storage nodes (read/write, string list)
This attribute is an ordered list of storage nodes for the client to use when saving its data. Its saves are directed to the first storage node that has an enabled device and a functional media daemon, **nsrmmd**(8). The default value of 'nsrserverhost' represents the server. In addition to storage node names the keyword 'curphyhost' could also be entered into the list. The entry 'curphyhost' denotes the current physical host. It is only used for virtual clients on a cluster. It should not be used on physical clients or on the client which is tied with the virtual server. Using the curphyhost keyword would enable the virtual clients backup to be directed to the storage node on which the virtual client is currently residing on. See **nsr_storage_node**(5) for additional detail on storage nodes.

clone storage nodes (read/write, string list)
This attribute specifies the hostnames of the storage nodes that are to be selected for the 'save' side of clone operations. Cloned data originating from this client is directed to the first node listed in the 'clone storage node' list that has both an enabled device and a functional media daemon, **nsrmmd**(8). There is no default value. If this attribute has no value, the server's **'clone storage nodes'** will be consulted. If this attribute also has no value, then the server's **'storage nodes'** attribute will be used to select a target node for the clone. See **nsr_storage_node**(5) for additional detail on storage nodes.

recover storage nodes (read/write, string list)
This attribute is an ordered list of storage nodes for the client to use when recovering its data.

Exclusions:

If the volume being recovered from is already mounted, then the recover storage node list is ignored and the volume is used from its existing location.

If the volume is in a jukebox and the "read hostname" attribute is set, then the volume will be mounted on the designated host.

During regular recover and clone operations, if the environment variable FORCE_REC_AFFINITY is set to "Yes" or "yes", it will force the broker to use the Recover Storage Affinity, even if the requested volume is mounted, to determine the "read" host.

In a virtual tape environment(VTL libraries have "virtual jukebox" attribute set to "yes"), during Cloning, whether or not the FORCE_REC_AFFINITY is set,

the behavior is always going to be as if that environment variable is set to true(yes).

While cloning, if volume is not mounted and the volume is not in a jukebox with the "read hostname" set, the Server's Client resource is checked for the value of the "Recover Storage Node" for the "read" host. Then, the Client resource of the "read" host is checked for the value of the "Clone Storage Node" attribute, to determine where the "write" should go to.

licensed applications (read-only, string list)
This attribute contains names of the licensed applications used by the client. By default, this field is blank.

EXAMPLES **Note:** The hidden attributes are not shown in these examples.

A resource to define a client, called venus, backing up all of its files to the NetWorker server mars:

```
                type: NSR client;
                name: venus;
              server: mars;
archive services: Disabled;
            schedule: Full Every Friday;
  browse policy: Month;
retention policy: Quarter;
           directive: UNIX with compression directives;
               group: Default;
            save set: All;
Backup renamed directories: Disabled;
        remote access: ;
          remote user: ;
             password: ;
      backup command: ;
                Pool: ;
             aliases: venus, venus.emc.com;
        archive users: ;
        storage nodes: nsrserverhost;
clone storage nodes: ;
```

The resources for a client backing up different file systems on different schedules with the support of renamed directories enabled:

```
                type: NSR client;
                name: saturn;
              server: mars;
archive services: Disabled;
            schedule: Default;
  browse policy: Month;
retention policy: Quarter;
           directive: ;
               group: engineering;
            save set: /,  /usr,  /usr/src;
Backup renamed directories: Enabled;
        remote access: venus, sam@*, jupiter/john;
          remote user: operator;
             password: ;
```

```
      backup command: ;
               Pool: ;
            aliases: saturn.emc.com;
       archive users: ;
       storage nodes: nsrserverhost;
clone storage nodes: ;

               type: NSR client;
               name: saturn;
             server: mars;
  archive services: Disabled;
           schedule: Full on 1st Friday of Month;
      browse policy: Month;
  retention policy: Quarter;
          directive: UNIX standard directives;
              group: Default;
           save set: /usr/src/archive;
     remote access: sam@venus, &netadmins, root@*;
       remote user: operator;
           password: ;
      backup command: ;
               Pool: ;
            aliases: saturn.emc.com;
       archive users: ;
       storage nodes: nsrserverhost;
clone storage nodes: ;
```

SEE ALSO **rsh**(1), **ruserok**(3), **nsr**(5), **nsr_schedule**(5), **nsr_directive**(5), **nsr_group**(5), **nsr_policy**(5), **nsr_pool**(5), **nsr_storage_node**(5), **save**(8), **savegrp**(8), **savefs**(8), **nsradmin**(8), **nsrexecd**(8)

NAME nsr_crash − recover from a disaster with NetWorker

DESCRIPTION NetWorker can be used to recover from all types of system and hardware failures that result in loss of files.

When a NetWorker client has lost files, the **recover** command can be used to browse, select, and recover individual files, selected directories, or whole filesystems. If the NetWorker **recover** command is lost or damaged, it will have to be copied either from a NetWorker client or from the NetWorker distribution media.

When recovering a large number of files onto a filesystem that was only partially damaged, you may not want to overwrite existing versions of files. To do this, wait until **recover** asks for user input to decide how to handle recovering an existing file. You can then answer **N** meaning ''always no'' to cause recover to avoid overwriting any existing files, or **n** if you want to protect this file but you want **recover** to ask again on other files.

If you do want to replace the existing version of a file or set of files with the saved versions, answer **Y** or **y** when **recover** asks if it should overwrite existing files (**Y** means ''always yes'' for future overwrite cases; **y** means just overwrite this one file).

For more information on using the **recover** command, see the **recover**(1m) manual page.

If the NetWorker server daemons or commands are lost, it may be necessary to re-install the server from the NetWorker distribution media. Once the NetWorker server is installed and the daemons are running, other NetWorker server files can be recovered using the **recover** command. When re-installing NetWorker you must be sure to install the */nsr* directory in exactly the same place as it was originally installed. The machine used to recover files may be different that the one used to save the files, but it must have the same hostname as the original machine. Recovery of the NetWorker server and client indexes requires that the destination machine be of the same kind as the one used to save the indexes.

If the NetWorker server's media database is lost, it will be necessary to recover the **bootstrap** from media. **mmrecov** recovers the **bootstrap** which contains the media database and the NetWorker server resource files. Since the resource files cannot be restored on top of the ones the NetWorker server is using, it is necessary to shut down NetWorker, rename the recovered resource files, and restart NetWorker. The save set identifier and other information about the bootstrap save set is printed by **savegrp** at the end of each scheduled save. It can also be displayed using **mminfo -B** or **scanner -B.**

See the **savegrp**(1m), **mminfo**(1m), and **scanner**(1m) man pages for more details.

If the index of any NetWorker server or client is lost, the index must be recovered from backup media before the **recover** command can be used to browse and recover files that were saved from that client. To recover the NetWorker server or any other client's index once the media database and server resource files have been recovered, use the **nsrck** command. The **nsrck** command recovers the lost index for a NetWorker server or client by locating the **index:clientname** save set produced by the **savegrp**(1m) command at the end of a scheduled save. **nsrck** queries the media database to determine which save sets to extract from which volumes to recover the index to the latest time. See the **nsrck (1m)** man page for more details.

To summarize, these are the steps you must do to recover your server after **mmrecov** completes.

1. Shut down your NetWorker server (**nsr_shutdown -a**). For Windows, you would stop the NetWorker services.
2. Change to the */nsr* directory (**cd /nsr**). For Windows, cd to the install location

(default C:\Program Files\nsr).

3. Save the temporary resource directory created when you reinstalled the NetWorker server (**mv res res.save**). For Windows, use "My Computer" or "Windows Explorer" to rename the res directory to res.save.
4. Move the recovered resource directory into place (**mv res.R res**). For Windows, use "My Computer" or "Windows Explorer" to rename the res.R directory to res.
5. Restart the NetWorker daemons on the Server by running the platform dependent startup script. For example, on Solaris, this is "/etc/init.d/networker start". For Windows, you would start the NetWorker services.
6. After verifying that the recovered resources are valid, remove the temporary resource directory (**rm -r /nsr/res.save**). For Windows, use "My Computer" or "Windows Explorer" to send the res.save directory to the recycle bin.
7. Recover your server and client indexes (**nsrck -L7**).

NOTE: The **mmrecov** command is only used to recover the NetWorker server's media database and resource files. Use **nsrck** to recover the server and client indexes.

Once the media database and server resource files have been recovered, you may recover any of your server or client indexes in any order. It is not necessary to recover the server's index before recovering the clients' indexes. Moreover, if your clients have the NetWorker client installed, you may run on-demand and scheduled saves once the media database and server resource files have been recovered. However, you will not be able to browse the saves for a client until you recover the client's file index. You may use save set **recover** to recover files before a client's file index has been recovered.

See the **recover**(1m) man page for details on running recover by save set.

If the server is damaged so badly that it will not run at all, you will need to follow the manufacturer's instructions for re-installing and rebooting a multiuser system. Once you have the system up and running in multiuser mode, you can re-install NetWorker (that is extract NetWorker from the distribution media and install it, using **pkgadd**(1M) or any other installation utility depending on your system), use **mmrecov** to recover the media database and resource files, and use **nsrck** to rebuild the on-line indexes for the server and each client. Finally, you will want to recover files which previously existed on the machine, but which do not exist on the manufacturer's distribution media. This may include system files which had been customized, a specially tailored kernel, new special device entries, locally developed software, and users' personal files.

SEE ALSO **nsr_layout**(5), **nsr**(1m), **nsrck**(1m), **recover**(1m), **savegrp**(1m), **mmrecov**(1m), **scanner**(1m)

NAME nsr_data − data formats for NetWorker Save and Recover

DESCRIPTION All data in the NetWorker system is encoded using the *eXternal Data Representation* (XDR) standard. When files are passed between client (see **save**(8) and **recover**(8)) and server (see **nsrd**(8)) and media (see **nsrmmd**(8)), they are represented as a *savestream*, which is encoded as a linked list of *savefiles*. There are currently 2 different *savefile* formats. A magic number at the start of each file indicates the particular type of the following *savefile* thus allowing for self identifying *savestreams* containing more than one *savefile* type. Logically each *savefile* consists of some header information followed by file data. The original *savefile1* format uses a doubly wrapped set of client attributes describing the file attributes and the file data is encoded as a *bucketlist*. The newer *savefile2* format uses an alternate singularly wrapped client attributes with the file data encoded as a bucket-less succession of self describing sections each containing a type, a length, and bytes of data. The file data section of a file is terminated by an ending section with a type of 0 (NSR_ASDF_END).

The XDR language description of the OS independent portion of the *savestream* data structures is shown below.

```
const NSR_IDLEN = 1024;                 /* length of file id */
const NSR_MAXNAMELEN = 1024;            /* max length of file system name */
const NSR_MAXCATTRSIZE = 8192;          /* max size of client specific attributes */
const NSR_MAXBUCKETDATA = 8192;/* max size of file bucket's data (w/o slop) */
const NSR_MAXBUCKETSIZE = 9000;  /* max total size of file bucket (w/ slop) */
const NSR_MAXCLNTSIZE = 16384;          /* max size of a clntrec */

typedef opaque fileid<NSR_IDLEN>;                   /* file identifier */
typedef string nsrname<NSR_MAXNAMELEN>;             /* file name type */
typedef opaque clientattr<NSR_MAXCATTRSIZE>;  /* client attributes */
typedef opaque wraposaverec<NSR_MAXCLNTSIZE>;/* wrapped osaverec */
typedef uint32_t checksum;                          /* 4 bytes for checksum */
typedef u_long sfid_t;                              /* savefile id (offset) */

struct id {
        string id_str<>;                /* id string */
        id *id_next;                    /* next such structure */
};

struct asmrec {
        id *ar_info;                    /* name and args to ASM */
        nsrname *ar_path;               /* not currently used */
        asmrec *ar_next;                /* next such structure */
};

const NSR_MAGIC1 = 0x09265900;          /* older format using buckets & ssaverec's */

struct osaverec {
        nsrname sr_filename;            /* name of this file */
        fileid sr_fid;                  /* client specific file id */
        asmrec *sr_ar;                  /* ASM list for this file */
        u_long sr_catype;               /* client specific attribute type */
        clientattr sr_cattr;            /* client specific file attributes */
};
```

```
struct ssaverec {
        sfid_t sr_id;                   /* savefile id in the savestream */
        u_long sr_size;                 /* size of encoded savefile */
        uint32_t sr_savetime;           /* savetime of this saveset */
        wraposaverec sr_wcr;            /* a wrapped osaverec */
};

/*
 * File data for older style savestream is logically
 * expressed as a linked list of file buckets.
 */
struct bucketlist {
        bucket bl_bucket;
        bucketlist *bl_next;
};

/*
 * XDR description of the original savefile1 format.
 */
struct savefile1 {
        u_long sf_magic;                /* magic number (must be NSR_MAGIC1) */
        u_long sf_chksumtype;           /* file checksum type */
        ssaverec sf_saverec;            /* wrapped file attributes */
        bucketlist *sf_data;            /* file data in buckets */
        checksum sf_checksum;           /* checksum value */
};

/*
 * Newer savestream defines and structures.
 */
const NSR_MAGIC2 = 0x03175800;  /* newer bucketless format */

const NSRAPP_BACKUP = 1;        /* backup application name space */
const NSRAPP_HSM = 2;           /* HSM application name space */
const NSRAPP_ARCHIVE = 3;       /* Archive application name space */

struct saverec2 {
        sfid_t sr_id;                   /* savefile id in the savestream */
        u_long sr_size;                 /* size of encoded savefile */
        uint32_t sr_savetime;           /* savetime of this saveset */
        uint32_t sr_appid;              /* application id */
        nsrname sr_filename;            /* name of encoded file */
        fileid sr_fid;                  /* client specific file id */
        asmrec *sr_ar;                  /* ASM list for this file */
        u_long sr_catype;               /* client specific attribute type */
        clientattr sr_cattr;            /* client specific file attributes */
};

/*
 * Current 64-bit savestreams
 */
typedef struct uint64_t unsigned long long;
typedef struct lg_time64_t unsigned long long;
```

```
const NSR_MAGIC3 = 0x03175803;   /* 64-bit format */
struct saverec3 {
        uint64_t sr_id;          /* savefile id in the savestream */
        uint64_t sr_size;        /* size of encoded savefile */
        lg_time64_t sr_savetime; /* savetime of this saveset */
        uint32_t sr_appid;       /* application id */
        nsrpath sr_filename;     /* full path of encoded file */
        fileid sr_fid;           /* client specific file id */
        asmrec *sr_ar;           /* ASM list for this file */
        u_long sr_catype;        /* client specific attribute type */
        clientattr sr_cattr;     /* client specific file attributes */
};


/*
 * Defines for self describing data sections.
 * The NSR_ASDF_END type defines the end of the file data.
 * The NSR_ASDF_FILE_DATA_TYPE type has the file data preceded by an
 * uint32_t that is the relative offset from the last block into the file.
 */
const NSR_ASDF_END = 0x0;                   /* end of ASDF data */
const NSR_ASDF_FILE_DATA_TYPE = 0x100;      /* normal file data */

/*
 * Describes a section of NetWorker "file data" when
 * using ASM Structured Data Format (ASDF) sections.
 */
struct asdf_hdr {
        uint32_t typevers;       /* type of file data */
        uint32_t length;         /* section length */
};

/*
 * Pseudo XDR description of the newer savefile2 format.
 * The new savefile2 format uses the unwrapped saverec structure
 * and a "bucketless" file data format that is based on ASDF.
 * The data portion ends with a 0 sized section of type NSR_ASDF_END.
 */
struct savefile2 {
        u_long sf_magic;         /* magic number (must be SF_MAGIC2 or SF_MAGIC3) *
        u_long sf_chksumtype;    /* file checksum type */
        saverec sf_saverec;      /* new saverec structure */
        <asdf_hdr & data>        /* ASDF section sans buckets */
            ...
        <asdf_hdr & data>        /* ASDF section sans buckets */
        <asdf_hdr.typevers = 0>  /* final ASDF section type = NSR_ASDF_END */
        <asdf_hdr.length = 0>    /* final ASDF section len = 0 */
        checksum sf_checksum;    /* checksum value */
};
```

SEE ALSO **mm_data**(5), **nsr**(8), **nsrmmd**(8), **nsrd**(8), **recover**(8), **save**(8), **xdr**(3n)
RFC 1014 XDR Protocol Spec

NAME nsr_device − NetWorker resource type "NSR device"

SYNOPSIS **type: NSR device**

DESCRIPTION Each storage device used by a NetWorker server is described by a single resource of type **NSR device**. See **nsr_resource**(5) for information on NetWorker resources. To edit the **NSR device** resources run:

```
nsradmin -c "type:NSR device"
```

Be sure to include quotation marks and to insert a space between "NSR" and "device". See **nsradmin**(8) for information on using the NetWorker administration program. The mounting and unmounting of individual volumes (tapes or disks) is performed using the **nsrmm**(8), and **nsrjb**(8), commands and **NetWorker Management Console.**

ATTRIBUTES The following attributes are defined for resource type **NSR device**. The information in parentheses describes how the attribute values are accessed. **Read-only** indicates that the value cannot be changed by an administrator. **Read/write** indicates a value that can be set as well as read. **Hidden** indicates a hidden attribute of interest only to programs or experts. These attributes can only be seen when the hidden option is turned on in **nsradmin**(8). **Static** attributes change values rarely, if ever. **Dynamic** attributes have values that change rapidly. For example, an attribute marked **(read-only, static)** has a value that is set when the attribute is created and never changes.

name (read-only, static)

This attribute specifies the path name of the device. Only non-rewinding tape devices are supported. For systems that support "Berkeley style" tape positioning, use the BSD tape device name. The name given to Optical disks is typically the name given to the "c" partition of the raw device.

A logical device type has been defined to facilitate interaction with external media management services. When interacting with external media management services, the device name may be determined by the media management service associated with the device where a volume is loaded. The logical device is used to define a NetWorker device resource. The number of device resources that can exist is limited by the number of volumes managed by the service that NetWorker may access simultaneously. The name given to a logical device is not related to any specific device, but is required to be a unique name for the device. For logical devices, both the media type and the family are set to **logical.** The name, type, and family are determined after the media management service has loaded a volume into a device in response to a request made by NetWorker. The name, type, and family of the actual device are then stored in the attributes **logical name, logical type,** and **logical family,** respectively. The association between the logical device and the actual device only exists when the volume is loaded into the device and allocated for use by NetWorker.

Specify UNC (\\<server-name>\<share-point-name>) path when creating an adv_file device of a network filesystem on Windows storage node.

When defining a remote device on a storage node, include the prefix "rd=hostname:", in the path name; where *hostname* is the system to which the device is directly attached (the storage node). For more information, see **nsr_storage_node**(5).

Example: name: /dev/rmt/0hbn;

comment (read/write)

This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the device.

description (read/write)
This attribute is used to store a brief description about the device. The description is used to help administrators identify the device, and it can be in any format.
Example: description: DLT8000 tape drive in Engineering Lab rack #2;

message (read-only, dynamic, hidden)
This attribute specifies the last message received from the NetWorker server regarding this device. The values for this attribute may include information on the progress or rate of the operation.
Example: message: "Tape full, mount volume mars.017 on /dev/nrst8";

event id (read-only, dynamic, hidden)
This attribute specifies the unique tag for the alerts or notifications generated due to device ordering issues.
Example: event id: "<host_name> Potential device ordering issue : /dev/rmt/2cbn";

volume name (read-only, dynamic, hidden)
This attribute monitors the mounting and unmounting of volumes for a device. When a volume is mounted, the value is the volume name, otherwise there is no value.
Example: volume name: mars.017;

media family (read-only, static, hidden)
This attribute describes the class of storage media, as determined from the media type. The only legal values are: **tape** − tape storage device; **disk** − disk storage device; **logical** − used when interacting with an external media management service.
Example: media family: tape;

media type (read-only, static)
This attribute indicates the type of media a device uses. The *media type* varies depending on the operating system/platform (See the online NetWorker Hardware Compatibility Guide, which is referenced in the NetWorker Administration Guide in the Device and Media Management chapter, for a comprehensive list of media types supported on your platform.). Potential values, their meaning, and default capacities are:
4mm − 4mm digital audio tape (1 GB); **8mm** − 8mm video tape (2 GB); **8mm 5GB** − 8mm video tape (5 GB); **Data Domain** − Data Domain device; **adv_file** − advanced file type device, standard UNIX file system is supported; **dlt** − digital linear tape cartridge (10 GB); **vhs** − VHS data grade video tape (14 GB); **3480** − high-speed cartridge tape (200 MB); **qic** − quarter inch data cartridge (150 MB); **himt** − half inch magnetic tape (100 MB); **tk50** − DEC TK50 cartridge tape (94 MB); **tk70** − DEC TK70 cartridge tape (296 MB); **optical** − optical disks, Write Once Read Many (WORM), Erasable Optical Disks (EOD), or standard UNIX files are supported; **file** − file type device, standard UNIX file system is supported; **logical** − used when interacting with an external media management service.

Example: media type: 8mm 5GB;

enabled (read-write)
This attribute indicates whether a device is available for use. The value for this attribute is either **Yes**, **No** or **Service**. If the value is set to **Yes** the device is fully operational and can be used for all operations. This is the default setting for a device. If the value is set to **No**, the device is disabled and may not be used. A device cannot be set to **No** if a device is mounted, since it would cause the mounted volume to become inaccessible to NetWorker until the device is set back to **Yes**. If the value is

set to **Service**, the device may not be mounted for save or recover operations. This state is used to reserve a device for maintenance. The device can be used for administrative purposes such as volume verification, labeling or inventories if the device is selected using the -f option. A device set in **Service** mode can not be used for save or recover operations. If the device is set to **Service** while the device is in use, all current sessions will be allowed to complete normally, but no new sessions will be assigned to the device. If the device is a jukebox device, the device will be unloaded after the sessions have completed.
Example: enabled: yes;

shared devices (read-write, hidden)
This attribute enables, disables or service modes all devices that have the same value for their **hardware id** attribute, and so are sharing the same physical drive. Possible values are **enable all**, **disable all**, **service all** or **done**. After the value is set to either **enable all**, **disable all** or **service all** and the action is performed, the value will be reset to **done**. The action will enable, disable or service mode as many devices as it can, regardless of any error conditions. For example, it is not possible to disable a device that has a mounted volume. **So when this attribute is set to disable all**, as many devices as possible will be disabled, excluding those with mounted volumes. For such cases, an error message will be logged.
Example: shared devices: done;

write enabled (read/write, dynamic, hidden)
This attribute indicates whether writing to the current volume is allowed. The value for this attribute may be set to **yes** or **no**. This value can only be set when a volume is not mounted.
Example: write enabled: no;

read only (read-write)
This attribute indicates whether a device is reserved for read-only operations, such as recover or retrieve. The value for this attribute can be either **yes** or **no**. If the value is set to **yes**, only read operations are permitted on the device. This value cannot be changed if a volume is mounted.
Example: read only: yes;

target sessions (read/write)
This attribute indicates the target number of sessions that will write to a device. When all devices on a host have the same value for this attribute, sessions are assigned to a device, until the device's **target sessions** is reached; then sessions are assigned to the next device on the host. Once all devices have reached their **target sessions,** new sessions are assigned equally across all devices.
When this attribute has different values for devices on a host, and the **nsrmmd**(8) has not yet been assigned to a device, then sessions are assigned to an **nsrmmd**(8) based on the lowest attribute value among the host's devices. Once the **nsrmmd**(8) is assigned to a device, the **target sessions** value for the assigned device is used.
Use higher values to multiplex more clients onto each tape. This attribute is not a maximum number for a device, but is used for load-balancing.
Example: target sessions: 3;

max sessions (read/write)
This attribute indicates the maximum number of sessions that will write to a device. When all devices have reached their target sessions, new sessions are assigned across all devices. For those devices where the attribute max sessi ons is set not more than the specified number of sessions is assigned to the device. If the number is reached the device is omitted for new sessions. This attribute

is used to avoid overlaod situation of a nsrmmd process and to allow to optimize recover performanc (with high multiplex factor the recover performance may decrease).
Example: max sessions: 8;

volume label (read/write, dynamic, hidden)
This attribute is set by the **Verify label** operation and can be performed before the **Label** operation. If this attribute is blank during the labeling process, then the volume's current label is reused.

volume default capacity (read/write, static, hidden)
This attribute is used by the **Label** operation when the **volume current capacity** attribute is blank. To override the volume default capacity associated with the media type, you must enter a specific value, such as 1 or greater. The value of this attribute must end with K, M, or G, where K represents kilobytes, M represents megabytes, and G represents gigabytes. The actual capacity written to the volume may be slightly lesser or greater than the value specified. This hidden attribute can be modified by a user, and can be used to override default sizes when using devices (and/or tapes) with different capacities than the defaults.
Example: To override the default capacity of a tape drive to 10 Gb for all future volume label operations, set the value as follows:
volume default capacity: 10G;

volume current capacity (read/write, dynamic, hidden)
If the attribute's value is non-blank, it determines the capacity of a volume during the **Label** operation. Its format is the same as **volume default capacity.**
Example: volume current capacity: 5G;

volume expiration (read/write, dynamic, hidden)
This attribute is set by the **Verify label** operation and can also be used by the **Label** operation. The value for this attribute is specified in **nsr_getdate**(3) format. A blank value causes the default expiration to be used during labeling.
Example: volume expiration: next year;

volume pool (read/write, hidden)
This attribute indicates the pool that a mounted volume belongs to. This attribute can be set right after a device has been created and prior to a volume has been labeled to specify or display the default pool selection for the **Label** operation. If this attribute is set during a **Label** or **Label without mount** operation, this value will indicate the pool a volume is being assigned to. In order to change the volume pool assignment, **Label** operation must be performed by specifying a different pool. Manual updates to this attribute in Devices resource has no effect on the pool assignment. See **nsr_pool**(5) for more information on volume pools.
Example: volume pool: myPool;

volume flags (read/write, hidden)
This attribute displays the new flags for the volume being operated on. This attribute is used during "Label" or "Label without mount" operations.

volume operation (read/write, dynamic, hidden)
The *volume operation* attribute manipulates the media (volume) currently located inside the device. This attribute can be set to one of the following values: **Unmount**, **Mount**, **Verify label**, **Verify write time**, **Label**, **Label without mount**, **Eject**, or **Monitor device**. Each of these operations may require parameters to be set.
When the value is **Unmount**, NetWorker releases the device. The **Unmount** operation is asynchronous.

When the value is **Mount**, NetWorker mounts the loaded volume into the device. The **Mount** operation is asynchronous.
When the value is **Verify label**, the volume's label is read by NetWorker, and the **volume label** and **volume expiration** attributes are set. The **Verify label** operation is synchronous, and therefore the operation may take a long time to complete.
When the value is **Verify write time,** the volume's label is read by NetWorker, and the attributes **volume label**, **volume expiration**, and **volume write time** are set. The **Verify write time** operation is synchronous, and therefore the operation may take a long time to complete.
When the value is **Label** or **Label without mount**, the volume receives a new label as determined by the attributes below. When the value is **Label,** the volume is then mounted. These operations are asynchronous.
When the value is **Eject**, NetWorker ejects the volume from the device. The **Eject** operation is asynchronous.
When the value is **Monitor device** and the device is idle (no volume loaded into the device), NetWorker will periodically check the device to determine whether a volume has been loaded into the device. When a volume containing a readable NetWorker label is loaded, the volume is placed into the NetWorker media database. The volume can then be written to by NetWorker if the volume is mounted with write permissions turned on; otherwise, the volume is mounted as read-only, and cannot be written to by NetWorker. When a volume without a readable NetWorker label is loaded into the device, the device's **unlabeled volume loaded** attribute is set to **yes**, and the volume may be labeled at a later date. The **Monitor device** operation is never performed on jukebox devices, because NetWorker only monitors non-jukebox devices.

volume write time (read-only, dynamic, hidden)
This attribute indicates the time that a save set was first written to the volume.

volume error number (read-only, dynamic, hidden)
This attribute indicates the last error number reported for this device. This is a numeric value encoded with the source, severity and the actual error number. Processes check for this value only on error in a media operation when the media operation is known to update this field, e.g., a label verify. The error number is not reset on a successful media operation, so it is not an indication of the status of the last media operation, but just the last error number reported for this device.

volume block size (read-only, dynamic, hidden)
This attribute indicates the block size of the currently mounted volume.

volume id (read-only, dynamic, hidden)
This attribute indicates the volume id for the currently mounted volume.

long volume id (read-only, dynamic, hidden)
This attribute indicates the volume id for the currently mounted volume in the long globally unique format.

accesses (read-only, hidden)
This attribute indicates the total number of operations performed on the device since it was configured as a NetWorker device. Changes to this attribute are propagated to all devices that have the same **hardware id** value.

access weight (read/write, hidden)
This attribute indicates the weight of a single operation performed on the device. The "accesses" attribute will be incremented by "access weight" each time an operation is performed on the device. The higher the weight, the less often the device will be selected for new operations. Changes to this attribute are

propagated to all devices that have the same **hardware id** value.

consecutive errors (read-only, dynamic, hidden)
This attribute indicates the current number of consecutive errors on a device. Changes to this attribute are propagated to all devices that have the same **hardware id** value.

max consecutive errors (read/write, hidden)
This attribute indicates the maximum number of consecutive errors allowed before disabling the device. Changes to this attribute are propagated to all devices that have the same **hardware id** value.

operation arg (read-only, dynamic, hidden)
This attribute indicates extra parameters to be used during device operations. Parameters are packed into a string and parsed by the associated operation's function.

volume message (read-only, dynamic, hidden)
This attribute indicates the result of the last volume operation.

event tag (read/write, single number, hidden)
This attribute contains the tag (unique identifier) of the last notification event sent to the **nsrd (8)** daemon. The tag is used to clear the previous event. This attribute is used to pass information between NetWorker programs, and should not be changed manually by the administrator.

NSR operation (read-only, dynamic, hidden)
This attribute indicates the current operation being performed by a device. The valid values for this attribute are: **Idle**, **Write**, **Read**, **Eject**, **Verify label**, or **Label**.
Example: NSR operation: Write;

minor mode (read-only, dynamic, hidden)
This attribute indicates the current state of a device. The **NSR operation** attribute is the major mode. The valid values for this attribute are: **idle**, **reading**, **writing**, **rewinding**, **moving forward**, **moving backward**, **error**, **done**, **writing eof**, or **finding eom**.
Example: minor mode: moving forward;

jukebox device (read/write, dynamic, hidden)
This attribute indicates the media device that is part of a jukebox device. This value can be either **yes** or **no**.

statistics (read-only, dynamic, hidden)
This attribute reports the statistics for the operation of this device. The statistics include:
the time of operation ("**elapsed**"), the number of errors ("**errors**"), the last writing rate ("**last rate**"), the maximum number of concurrent clients ("**max clients**"), the number of file marks written ("**file marks**"), the number of rewinds ("**rewinds**"), the number of files skipped ("**files skipped**"), the number of records skipped ("**records skipped**"), the current file number ("**current file**"), the current record number ("**current record**"), the relative number of files being spaced over ("**seek files**"), the relative number of records being spaced over ("**seek records**"), the total estimated amount read/written on the volume, in KB ("**estimated KB**", to be implemented in a future release), the total amount read/written on the volume, in KB ("**amount KB**"), the current amount read/written on this file, in KB ("**file amount KB**"), and the current number of sessions assigned to this device ("**sessions**").

cleaning required (read/write)
This attribute indicates whether a device needs to cleaned. The value for this attribute may be either **yes** or **no**. If the value of this attribute changes from yes to

no and the value of **date last cleaned** attribute is not updated, then the **date last cleaned** attribute is set to the current time. NetWorker might set this attribute to yes if, at the time the device is next scheduled to be cleaned, it is not available to be cleaned. In this case, the following message is displayed: **device cleaning required.** This message indicates that the device needs to be cleaned. This attribute can only be used for a device whose **media family** is **tape** and **jukebox device** is **yes**. For all other devices, the value of this attribute is always **no**.

cleaning interval (read/write)
This attribute indicates the amount of time from the **date last cleaned** until the next scheduled cleaning for the device. This value can be specified in **days**, **weeks**, or **months**. One day, week, or month is implied if a number is not specified. If this attribute is set and **date last cleaned** is blank, **date last cleaned** is set to the current time. This attribute may only be used for a device whose **media family** is **tape** and **jukebox device** is **yes**.
Example: cleaning interval: 2 weeks;

date last cleaned (read/write)
This attribute indicates the time and day a device was last cleaned. Input may be in any format acceptable to **nsr_getdate**(3). Some values acceptable to **nsr_getdate**(3) are relative, for example, **now**. For that reason all input is converted into **ctime**(3) format, weekday, month, day, time, year. As noted in the description of **cleaning required** and **cleaning interval ,** the value of this attribute might be set automatically by NetWorker. This attribute can only be used for a device whose **media family** is **tape**

auto media management (read-write)
This attribute indicates whether "automated media management" is enabled for a device. For jukebox devices this value is always **no**. See **nsr_jukebox**(5) for a description of **auto media management** for a jukebox. For non-jukebox devices, this value can be either **yes** or **no**. If this value is set to **yes**, then any recyclable volumes loaded into the device might be automatically re-labeled by NetWorker for re-use, and unlabeled volumes loaded into the device can be automatically labeled. When NetWorker is labeling a volume that is not expected to have a valid NetWorker label, it verifies that the volume is unlabeled before labeling the volume. A volume is considered to be unlabeled if the volume does not contain a label that may be read by this device.
Note: If a volume contains a label, but the label is written at a density that cannot be read by the associated device, the volume is considered to be unlabeled. If the volume contains data written by an application other than NetWorker, it most likely does not have a label recognizable by NetWorker, and the volume is considered to be unlabeled. With this attribute enabled, care should be taken when loading any volume considered to be unlabeled or recyclable into the device. The volume might be re-labeled and the data previously on the volume over-written by NetWorker.
When this attribute is set to **yes** for a device, and the device is idle (no tape loaded into the device), NetWorker will monitor the device and wait for a volume to be loaded. See the description of **Monitor device** in the discussion of the **volume operation** attribute.
Example: auto media management: yes;

NDMP (read-only)
This attribute is used to note which devices are associated with NDMP servers. This attribute cannot be changed after the resource has been created. The resource must be deleted and recreated if the user needs to change this attribute for this device. The same username (**remote user** attribute) and **password**

should be configured in the device resource as they are configured for the NDMP server.
Example: NDMP: yes;

max active devices (read-write)
This attribute set the maximum number of devices NetWorker may use from the storage node associated with the device. All devices on a storage node must have the same value for this attribute. When this attribute is modified, the new value is propagated to all other devices on the storage node. The default value for this attribute is NULL = no limit on the number of active devices on the storage node. For newly created devices the value of this attribute is inherited from other devices on the same storage node or set to the default value of NULL if there are no other devices on the storage node. Value for this attribute must be an integer greater than the number of non-shared devices on the storage. This attribute can only be set on storage nodes with shared devices.

dedicated storage node (read-write)
The value for this attribute can be either **yes** or **no**. The value of this attribute determines whether a storage node is a dedicated storage node. A dedicated storage node can only back up its local data. All devices on a storage node must have the same value for this attribute. When a device is created or the value of this attribute is modified, the value of this attribute is propagated to all other devices on the storage node.
Example: dedicated storage node: yes;

remote user (read/write, string)
This attribute is used when the NDMP attribute is set to a value of yes. The value entered for this attribute should be the username configured for the NDMP server. This attribute is also used when creating an adv_file device of a network drive on Windows storage node. Specify remote user attribute as the user name for the Windows storage node (nsrmmd) to connect to the network drive.
Example: remote user: root;

password (read/write, encrypted)
This attribute is used in conjunction with the **remote user** attribute to configure access to a NDMP server. This attribute is also used in conjunction with the **remote user** attribute to connect to an adv_file device of a network drive on Windows storage node.
Example: password: ;

unlabeled volume loaded (read-only, dynamic, hidden)
This attribute indicates whether a volume loaded into the device has a readable NetWorker volume label. This value can be either **yes** or **no**. This attribute is set to **yes** when NetWorker is monitoring the device, a volume is loaded into the device, and the volume does not have a valid NetWorker label that can be read by this device. This attribute is set to **no** when the volume in the device is labeled or ejected from the device.

logical name (read-only, hidden, no create)
This attribute indicates the name of the actual device associated with the logical device. This attribute is only used for logical devices.
Example: logical name: /dev/rmt/0hbn;

logical family (read-only, hidden, no create)
This attribute indicates the family of the actual device currently associated with the logical device. The values that can be associated with this attribute are the values that are valid for the **media family** attribute. The only exception is that

the value of this attribute cannot be set to **logical.** This attribute is only used for logical devices.
Example: logical family: tape;

logical type (read-only, hidden, no create)
This attribute indicates the actual device type associated with the logical device. The values that can be associated with this attribute are the values that are valid for the **media type** attribute. The only exception is that the value of this attribute cannot be set to **logical.** This attribute is only used for logical devices.
Example: logical type: 8mm 5GB;

hardware id (read/write)
This attribute represents the unique identification of a shared physical drive, which can be accessed by multiple device resources. Each device resource that shares the same physical drive must have the same value for this attribute. It can only be updated when the device is disabled and not within a jukebox resource. When a value is defined for this attribute, corresponding device messages will contain a number that uniquely represents the **hardware id** attribute, and will be visible in administrator commands, such as **NetWorker Management Console** and **nsrwatch**(8). This number identifies the devices that share the same physical drive.

save mount timeout (read/write, hidden, no create)
This attribute indicates the timeout value for an initial save mount request for the storage node on which a device is located. If the request is not satisfied within the indicated time, the storage node will be locked from receiving save processes for the "save lockout" time. See **nsr_storage_node**(5) for a description of storage nodes. This attribute can be used for local devices as well, but "save lockout" cannot be changed from its default value of zero. Hence, local devices cannot be locked out from save requests.

save lockout (read/write, hidden, no create)
This attribute indicates the number of minutes a storage node will be locked from receiving save assignments after it reaches the **save mount timeout** time during a save mount request. A value of zero indicates that the node will not be locked. This attribute cannot be changed for local devices.

CDI (read-write, no create)
This attribute indicates whether CDI (Common Device Interface) is used to communicate with this device and if so, which CDI method is used. The value for this attribute is one of:

Not used
NetWorker will use the same method to communicate with a device as in versions up to and including 6.x. This setting is mainly for debugging purposes, since selecting **Not used** essentially turns off the use of CDI.

SCSI commands
NetWorker will use the CDI interface to send explicit SCSI commands to tape drives. This allows the best control of and status collection from a device and is the default for SCSI or SCSI-like tape drives directly under NetWorker's control.

Tape driver commands
NetWorker will use all available functions that are present in the operating system's tape driver interface. There are many capabilities that tape drivers lack which are unavailable to NetWorker if this setting is selected. Also, different OSes support different functions, so

devices may behave differently on different platforms if this choice is selected.

Generic tape driver commands
NetWorker will use only those functions that are present on all Unix platforms. This will give results similar to the **Not used** selection, except that CDI code will be used instead of the pre 7.0 code.

NDMP
NetWorker will use NDMP to control NDMP connected tape devices. The exact functions available will likely depend on the NDMP server being used.

iSCSI
NetWorker will use iSCSI commands to control iSCSI connected tape devices.

Example:
CDI: SCSI commands;

device block size (read-write, single number, hidden)
This attribute allows you to override the device's default block size on a per-device basis. Allowable values are **handler default** (the default setting for this attribute), **32kB**, **64kB**, **96kB**, **128kB**, **160kB**, **192kB**, **224kB**, **256kB**, **384kB**, **512kB**, **640kB**, **768kB**, **896kB**, and **1024kB**. Note that the block size for a NetWorker volume is set when the volume is labeled, so a change to this attribute will not have any effect until a volume is recycled. Also, resetting this attribute to **handler default** will not have any effect until the nsrmmd daemon for this device is restarted. Also, hardware platform limitations may result in the use of a block size smaller than that selected in this attribute, as some SCSI adapters or adapter drivers place limits on the maximum size of a SCSI transfer. These limits are usually silently enforced by NetWorker.

device file size (read-write, single number, hidden)
This attribute allows you to override the default tape file size used by NetWorker for this device. The file size is the number of blocks (tape records) that NetWorker will write before writing a filemark to the tape. Allowable values are between 100 and 3,000,000.

device load time (read-write, single number, hidden)
This attribute allows you to override the default load time used by NetWorker for this device. The load time is the number of seconds that NetWorker will continue trying to open a tape device after loading it into a tape drive. Allowed values are between 10 and 900 seconds.

This attribute is only used when **CDI** is set to **Not used**.

device eject time (read-write, single number, hidden)
This attribute allows you to override the default tape eject time used by NetWorker for this device. The eject time is the number of seconds that NetWorker will wait for a tape drive to eject a tape before trying to remove the tape from the drive. Allowed values are between 30 and 900 seconds.

device poll interval (read-write, single number, hidden)
This attribute allows you to override the default tape polling interval used by NetWorker for this device. The polling interval is the number of seconds that NetWorker will wait between successive attempts to determine whether a tape drive is ready for use. Allowed values are between 1 and 30 seconds.

device min load tries (read-write, single number, hidden)
This attribute allows you to override the load try limit used by NetWorker for

this device. The load try limit is the minimum number of times that NetWorker will attempt to determine if a tape drive is ready for use. This is primarily intended for operating systems where the open() system call to a tape drive that is not yet ready takes a very long time to fail. Allowable values are between 2 and 120.

This attribute is only used when **CDI** is set to **Not used**.

device tape flags (read-write, hidden)
This attribute allows you to override the default flags used by NetWorker for this device. The flags are settings that control major aspects of how NetWorker interacts with a device. Allowable values are **TAPE**, **NOEOM**, **PHYSREC**, **SIZED**, **NOBSF**, **FILE**, **FILESYS**, **32K**, separated with either spaces or an 'or' symbol (|), and possibly preceded by an 'or' symbol (|).

WARNING: You should only use this attribute if you have been instructed to by EMC Technical Support, as misuse can cause the loss of data on any tapes that get loaded into the drive when this attribute is set to anything other than the default empty value.

device default capacity (read-write, single number, hidden)
This attribute allows you to override the default capacity used by NetWorker for this device. The default capacity is a typical value for the uncompressed storage capacity for drives of a given device type, and does not take different tape sizes or compression methods into account. It is only used for rough informational purposes and has no effect whatever on the amount of data that NetWorker will write to a tape.

This attribute is only used when **CDI** is set to **Not used**.

TapeAlert Critical (read-write, no create)
This attribute stores a list of any Critical TapeAlert flags that may have been collected from a tape drive during operations. Critical flags are those that might result in data loss. Often, these will require user intervention to resolve.

There are several Critical flags that will automatically be cleared by NetWorker when the particular flag no longer pertains to the drive. (The actual TapeAlert flag number is in parentheses after the flag name):

Media (4)
unrecoverable read, write or positioning error caused by tape

Write protect (9)
Attempt to write to a write-protected tape

Recoverable snapped tape (13)
tape has snapped in a drive where the tape can be ejected

Forced eject (16)
The tape was manually ejected from the drive

Clean now (20)
The tape drive needs to be cleaned

Note that all of the TapeAlert attributes really should be "delete-only" lists, since they are used purely to report problems that the hardware reports to us. However, NetWorker does not have a "delete only" attribute, so these are read-write. NetWorker does use the values held in these attributes to suppress repeated warnings about a problem that has already been reported in the daemon log, messages file and any NetWorker administrative GUI that might be open.

TapeAlert Warning (read-write, no create)

This attribute stores a list of any Warning TapeAlert flags that may have been collected from a tape drive during operations. Warning flags are those that do not indicate the immediate danger of data loss, but do represent some aspect of device operation that may lead to data loss in the future.

There are several Warning flags that will automatically be cleared by NetWorker:

Read warning (1)
: The drive is having problems reading from the tape. No data has been lost but performance may suffer

Clean periodic (21)
: The drive is due for routine cleaning

TapeAlert Information (read-write, no create)
: This attribute stores a list of any Information TapeAlert flags that may have been collected from a tape drive during operations. Information flags represent occurrences that should be noted but which will not lead to loss of data.

There are several Information flags that will automatically be cleared by NetWorker:

No removal (10)
: An attempt was made to eject a tape when the drive was in use

Cleaning media (11)
: The tape in the drive is a cleaning tape and cannot be used for data

Unsupported format (12)
: The tape in the drive is a format that is not supported by the drive

Nearing media life (19)
: The tape cartridge is nearing the end of its specified life

autodetect id (read/write, hidden)
: This attribute is for identifying auto-detected devices. It is used by NetWorker programs only, and should not be changed manually by the administrator.

server network interface (read/write, hidden)
: This attribute defines the network address or the hostname which is used to communicate with mmd. This field is only relevant, if the device is connected to a storage node.

EXAMPLE A complete example follows:

```
                type: NSR device;
                name: /dev/nrst8;
             message: writing, done
         volume name: mars.017;
        media family: tape;
          media type: 8mm 5GB;
             enabled: Yes;
       shared devices: done;
dedicated storage node: No;
        write enabled: Yes;
           read only: No;
```

```
                target sessions: 4;
                   max sessions: 8;
                   volume label: mars.017;
         volume default capacity: ;
         volume current capacity: 5000 MB;
              volume expiration: "Thu Sep 21 17:23:37 1996";
                    volume pool: Default;
                   volume flags: ;
               volume operation: ;
              volume write time: ;
              volume block size: 32 KB;
                   volume id: 32449;
                    accesses: 199;
               access weight: 1;
            consecutive errors: 0;
        max consecutive errors: 20;
                 operation arg: ;
              volume message: ;
               NSR operation: ;
                  minor mode: idle;
              jukebox device: Yes;
                  statistics: elapsed = 257572, errors = 0, last rate = 397,
                              max clients = 3, file marks = 22, rewinds = 4,
                              files skipped = 1976, records skipped = 0,
                              current file = 2389, current record = 162,
                              seek files = 0, seek records = 0,
                              estimated kb = 0, amount kb = 6273,
                              file amount kb = 6273, sessions = 1;
             cleaning required: No;
              cleaning interval: 2 weeks;
             date last cleaned: "Tue Apr 11 15:10:32 1995";
        auto media management: No;
      unlabeled volume loaded: No;
                 logical name: ;
                 logical type: ;
               logical family: ;
       connection process id: ;
          connection message: ;
           connection status: ;
                 hardware id: ;
           save mount timeout: 30;
                 save lockout: 0;
                         CDI: SCSI commands;
device block size: handler default;
device default capacity: 20GB;
          device eject time:;
          device file size:;
        device load time: 120;
      device min load tries:;
       device poll interval:;
          device tape flags:;
        TapeAlert Critical: Media, Cleaning;
   TapeAlert Information: Read warning;
       TapeAlert Warning: Cleaning media;
```

FILES */nsr/res/nsrdb* − files in this directory should never be edited directly. Use **nsrmm**(8), **nsradmin**(8), or **NetWorker Management Console** instead.

SEE ALSO **nsr_getdate**(3), **ctime**(3), **nsr_client**(5), **nsr_resource**(5), **nsr_pool**(5), **nsr_schedule**(5), **nsr_service**(5), **nsr_storage_node**(5), **nsr_render_log**(8), **nsr**(8), **nsrmmd**(8), **nsrmm**(8), **nsradmin**(8)

NAME nsr_directive − NetWorker resource type ''NSR directive''

SYNOPSIS **type: NSR directive**

DESCRIPTION Each NSR directive is described by a single resource of type **NSR directive** (see **nsr_resource**(5)). To edit the NSR directive resources for a NetWorker server, use **nsradmin**(8) or **NetWorker Management Console.** See the corresponding manual page for more information on the use of these NetWorker administration programs.

These resources are used by the NetWorker ASM (Application Specific Module) family of commands when processing files; see **uasm**(8) and **nsr**(5). Directives can be used to improve the efficiency of backups by controlling which files get saved and specifying special handling on certain types of files.

Names of the currently defined 'NSR directive' resources are listed in the **directive** attribute of the 'NSR client' resource for selection which will be used to backup the client (see **nsr_client**(5)).

ATTRIBUTES The following attributes are defined for resource type **NSR directive**. The information in parentheses describes how the attribute values are accessed. **Create-only** indicates that the value cannot be changed after the resource has been created. **Read/write** means the value can be updated by authorized administrators. **Hidden** means it is an attribute of interest only to programs or experts, and these attributes can only be seen when the hidden option is turned on in **nsradmin**(8). **Dynamic** attributes have values which change rapidly. Several additional attributes such as, administrator, are common to all resources, and are described in **nsr_resource**(5).

name (create-only)
The names of directive resources are displayed as choices when creating or updating NetWorker client resources, see **nsr_client**(5). The name can generally be chosen at the administrator's convenience, but it must be unique for this NetWorker server. The directive resource named '**UNIX standard directives**' may be modified, but it may not be deleted. Other directives can only be deleted if no clients or archive lists are using them.
Example: name: UNIX standard directives;

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the directive.

directive (read/write)
This attribute contains the rules defining the directive. The value of this attribute is similar to the contents of a **.nsr** file except that absolute path names must be specified for each << *path* >> directive. See **nsr**(5) for more information on the format of this attribute.
Example: directive: "<< / >> skip : core";

NetWorker comes with several directive resources already defined. Examples include "UNIX standard directives", "UNIX with compression directives", "NT standard directives" and "Encryption directive". The first two are meant for use with clients running on UNIX platforms. The next directive is intended for use with clients running on Windows platforms. The last directive is for clients requiring encryption with Advanced Encryption Standard (AES). There may also be two other directives "Default" and "Default with compression". These are old names for "UNIX standard directives" and "UNIX with compression directives", respectively. NetWorker will remove the directive resources using the old names when they are no longer being used.

EXAMPLE An example NSR directive resource, named 'UNIX directive', follows:

```
type:   NSR directive;
name:   UNIX directive;
directive: "
            << / >>
                +skip : core
                skip : tmp
            << /usr/spool/mail >>
                mailasm : *
            << /nsr >>
                allow
           ";
```

SEE ALSO **nsr**(5), **nsr_resource**(5), **nsr_client**(8), **savegroup**(8), **savefs**(8), **uasm**(8), **nsradmin**(8)

NAME nsr_getdate − convert time and date from ASCII

SYNOPSIS

```
#include <sys/types.h>

time_t nsr_getdate(buf)
char *buf;
```

DESCRIPTION The **nsr_getdate()** routine converts most common time specifications to standard UNIX format. It takes a character string containing time and date as an argumant and converts it to a time format.

The character string consists of zero or more specifications of the following form:

tod A *tod* is a time of day, which is of the form *hh*[:*mm*[:*ss*]] (or *hhmm*) [*meridian*] [*zone*]. If no meridian − *am* or *pm* − is specified, a 24-hour clock is used. A *tod* may be specified as just *hh* followed by a *meridian*. If no zone (for example, GMT) is specified, the current timezone, as determined by the second parameter, *now*, is assumed.

date A *date* is a specific month and day, and possibly a year. The acceptable formats are *mm/dd*[*/yy*] and *monthname dd*[, *yy*]. If omitted, the year defaults to the current year. If a year is specified as a number in the range 70 and 99, 1900 is added. If a year is in the range 00 and 30, 2000 is added. The treatment of other years less than 100 is undefined. If a number not followed by a day or relative time unit occurs, it will be interpreted as a year if a *tod*, *monthname*, and *dd* have already been specified; otherwise, it will be treated as a *tod*. This rule allows the output from **date**(1) or **ctime**(3) to be passed as input to *nsr_getdate.*

day A *day* of the week may be specified; the current day will be used if appropriate. A *day* may be preceded by a *number,* indicating which instance of that day is desired; the default is *1*. Negative *numbers* indicate times past. Some symbolic *numbers* are accepted: **last**, **next**, and the ordinals **first** through **twelfth** (**second** is ambiguous, and is not accepted as an ordinal number). The symbolic number **next** is equivalent to **2**; thus, **next monday** refers not to the immediately coming Monday, but to the one a week later.

relative time
Specifications relative to the current time are also accepted. The format is [*number*] *unit*; acceptable units are **decade**, **year**, **quarter**, **month**, **fortnight**, **week**, **day**, **hour**, **minute**, and **second**.

The actual date is formed as follows: first, any absolute date and/or time is processed and converted. Using that time as the base, day-of-week specifications are added; last, relative specifications are used. If a date or day is specified, and no absolute or relative time is given, midnight is used. Finally, a correction is applied so that the correct hour of the day is produced after allowing for daylight savings time differences.

nsr_getdate accepts most common abbreviations for days, months, and so forth; in particular, it will recognize them with upper or lower case first letter, and will recognize three-letter abbreviations for any of them, with or without a trailing period. Units, such as **week**s, may be specified in the singular or plural. Timezone and meridian values may be in upper or lower case, and with or without periods.

SEE ALSO **ctime**(3), **date**(1), **ftime**(3c), **localtime**(2), **time**(2)

BUGS The grammar and scanner are rather primitive; certain desirable and unambiguous constructions are not accepted. Worse yet, the meaning of some legal phrases is not what is expected; **next week** is identical to **2 weeks**.

The daylight savings time correction is not perfect, and can become incorrect if provided times between midnight and 2:00 am on the days that the time changes.

Because **localtime**(2) accepts an old-style time format without zone information, passing **nsr_getdate** a current time containing a different zone will probably fail.

NAME nsr_group − NetWorker resource type ''NSR group''

SYNOPSIS **type: NSR group**

DESCRIPTION Each NetWorker group is described by a single resource of type **NSR group** (see **nsr_resource**(5)). To edit the **NSR group** resources for a NetWorker server type:

```
nsradmin −c "type:NSR group"
```

or use **NetWorker Management Console.** See the **nsradmin**(8) manual page for more information on using the NetWorker administration program.

These resources control when a group of NetWorker clients begin saving data and whether backups are started automatically each day. Each **NSR client** resource (see **nsr_client**(5)) lists the groups of which that client (or save sets for that client) is a member. Groups can only be deleted if no clients are members of them.

ATTRIBUTES The following attributes are defined for resource type **NSR group**. The information in parentheses describes how the attribute values are accessed. **Create-only** indicates that the value cannot be changed by an administrator once the resource is created. **Read/write** means the value can be set as well as read at any time. **Choice** indicates that the value can only be selected from a given list. **Yes/no** means only a yes or no choice is possible. **Static** attributes change values rarely, if ever. **Dynamic** attributes have values which change rapidly. **Hidden** means it is an attribute of interest only to programs or experts, and these attributes can only be seen when the hidden option is turned on in **nsradmin**(8). For example, an attribute marked **(create-only, static)** has a value which is set when the attribute is created and never changes. Several additional attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

name (create-only)
This attribute contains the name of the group defined by this resource. The name must be unique for this NetWorker server, but otherwise can be anything that makes sense to the administrator. This name will appear as a choice attribute of each **NSR client** and **NSR pool**(5) resource. The **NSR group** resource named 'Default' may be modified, but it may not be removed. The name can only be specified when the group is created.
Example: name: marketing;

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the group.

snapshot (read/write, yes/no)
This attribute determines whether the group represents snapshot backups. NOTE: This feature is enabled with EMC PowerSnap modules only.

autostart (read/write, choice)
This attribute determines if this group will be saved automatically every day. It may be one of three values: **Enabled**, **Disabled** or **Start now.** When the value is **Enabled**, the members of this group will start saving data at the time specified in the **start time** attribute. When the value is **Disabled**, the member of this group will not automatically start saving their data. When the **Start now** value is specified, the member clients will start saving their data immediately. The attribute will then return to its prior value.
Example: autostart: Enabled;

autorestart (read/write, choice, hidden)
This attribute controls whether this group should be automatically restarted

when an incomplete run (due to a power failure or administrator intervention) is noticed during NetWorker server startup. Like the **autostart** attribute, setting this attribute's value to **Restart now** causes NetWorker to restart the group immediately. Enabling **autorestart** only has an effect if **autostart** is also enabled.

stop now (read/write, choice, hidden)
Setting this value to 'True' when this group is running causes this group to abort all of its saves immediately. Once the group is stopped, the value is set back to 'False'. These are the only valid values.

start time (read/write)
This attribute specifies the time of day when this group will start saving. The NetWorker server's local time is used. The time is specified as "*hours*:*minutes*". Note that the quotes may be necessary when using character-based administration tools such as the **nsradmin** program because of the colon in the value. The hours may range from 0 to 23 (using a 24 hour clock) and the minutes range from 0 to 59.
Example: start time: "4:53";

last start (read/write, hidden)
The last time this group was started. If multiple instances were attempted to run, only one instance will update this attribute. The "last end" attribute will be cleared when this is set. This attribute is for informational purposes only, and changing it has no effect on the system.

last end (read/write, hidden)
The last time this group ended or completed. This attribute is for informational purposes only, and changing it has no effect on the system. This attribute gets updated only if last start attribute is updated. If the savegrp program or the machine terminates abnormally, the "last end" attribute may not be updated.

interval (read/write, static, hidden)
This attribute specifies how often this group is to be run automatically by NetWorker. Manually starting a group overrides the interval. The default value is 24:00, which means run once a day.

restart window (read/write, static, hidden)
This attribute is used for automatically restarted groups by NetWorker. This defines the amount of time for which restarts are valid. If the restart window has elapsed from the previous start of group, the restart attempt is converted into a fresh start. Also individual savesets are considered elapsed even if successfully completed previously but the start time is older than this window. These savesets are considered eligible for restart. This attribute helps maintain the start time of all savesets completed as part of the group fall within this time window. The *restart window* attribute has to be less than *interval* and setting it to half of *interval* helps avoid overruning restart attempts beyond the next scheduled start. The default value is 12:00, which is half of default value of *interval* (24:00).

success threshold (read/write, choice)
This attribute helps set the criteria for reporting the success of all savesets within a group. The default value is *Warning* which means if any saveset has warning(s) during backup it will be reported in the *Successful Save Sets* section. The client(s) of these savesets will be reported as *successful with warning(s)* in the completion report's header summary. If set to *Success* , any savesets completed with warning(s) will be reported in the *Unsuccesful Savesets* section. Please note any failures will invoke a retry on the saveset if retry count is not

0. The client will also be reported as failed in the savegroup completion report's header summary.

snapshot policy (read/write, choice)
Snapshot policy associated with the group, if this group happens to have snapshot set to Yes. NOTE: this feature is enabled with PowerSnap modules only.

snapshot pool (read/write, choice)
Pool to which Snapshot meta data needs to be backed up. NOTE: this feature is enabled with EMC PowerSnap modules only.

force incremental (read/write, static, hidden, choice)
Setting this attribute to 'Yes' will force an incremental level of a savegroup, when the **interval** attribute is less than 24 hours and the group is started after *12:00am+interval*. Please note that 12:00am is midnight. The default value is 'Yes.' A value of 'No' means that all backups are done at pre-determined level (as specified in the group resource's level attribute). This attribute applies only to automatically started groups.

The following example shows how to use this attribute in conjunction with "interval". Assume that a group g1 has to be backed up every 6 hours and the first backup must be a level backup. For g1, set the interval to 06:00, set autostart to "enabled" and set force incremental to 'Yes.' Also set the level to the desired level. In this case the first scheduled run of group g1 (before 6:00am) will be run at the level configured. Subsequent scheduled runs of group g1 on the same day (after 6:00am) will be run at incremental level.

savegrp parallelism (read/write)
If this value is non-zero, then the **savegrp** program eschews all other parallelism policies and attempts to keep that number of saves running.

client retries (read/write)
The number of times failed clients should be retried before **savegrp** gives up and declare them failed. Zero means do not retry. Abandoned saves are not retried, because they may eventually complete. A client's save sets are retried by **savegrp** whenever it would otherwise not be able to start a new save set. That is, **savegrp** prefers to start new save sets first, and only retries when there is nothing else to do.
Example: client retries: 1;

clones (read/write, static, yes/no, choice)
Setting this value to 'Yes' causes saves of this group to automatically make a clone of every save set backed up. The save set clones will be sent to the pool named in the **clone pool** attribute.

clone pool (read/write, static, choice)
The pool to which save set clones should be sent when 'clones' is 'Yes'. Only pools of type 'Backup Clone' are allowed (see **nsr_pool**(5)).

options (read/write, static, hidden)
The values specify flags with which this group will be run. The values No Monitor, No index save, No save, Index only, Verbose, Estimate, and Preview map to the **savegrp** command line flags **-m, -I, -n, -O, -v, -E,** and **-p** respectively. Some of these values (Preview and No save) are automatically reset when a run of **savegrp** completes normally.
Example: options: Verbose;

level (read/write, hidden, choice)
This is an explicit level the savegroup will use when started automatically by

NetWorker. *This hidden attribute can be modified by a user.* This value is not cleared automatically, that is, if one sets this attribute to full, this savegroup will run with a full level until this value is manually cleared. When not specified (the normal case), the NSR Schedule for each client filesystem will be used to determine the level. Manually running **savegrp** from the command line overrides this value. The choices are the standard level identifiers 'full', 'consolidate', 'incr', 'skip', and the number levels '1' through '9'.

printer (read/write, static, hidden)
The printer to which the bootstrap save set information will be printed, if one is generated by the run of this group. *This hidden attribute can be modified by a user.* If an invalid printer name is specified, bootstrap information will be included in the savegroup completion information piped through the *savegroup completion* notification (see **nsr_notification**(5)).
Example: printer: ps;

schedule (read/write, choice, hidden)
The schedule to use for determining what level of save to perform. *This hidden attribute can be modified by a user.* This value is not cleared automatically, that is, if one sets this attribute to a particular schedule, all clients which are part of this group will have their schedules overridden until this value is manually cleared. This overrides the schedule specified for individual clients. See **nsr_schedule**(5)).

schedule time (read/write, hidden)
An explicit time can be specified when looking at a schedule to determine which level of save to perform. A null value (normal setting) means use the current date to determine the level.
Example: schedule time: "3:00 am 01/11/93";

expiration time (read/write, hidden)
An explicit expiration time can be specified for the savesets saved with this group. This value will be forwarded to every save and backup command (see save(3)). A null value (normal setting) means no expiration time is set.

inactivity timeout (read/write, static, hidden)
The number of minutes that the **savegrp** command waits for any kind of activity on the server before concluding that a **savegrp** descendant is hung. *This hidden attribute can be modified by a user.* Once a hang is detected, **savegrp** prints a message indicating that a **save** is being aborted, kills or aborts the backup, and moves on to its next task. Inactivity is defined as the last time a client has sent data to the server. If a client has a very large filesystem and an incremental is run, it is possible for **savegrp** to abort a save set that only appears to be hung. In these cases, the inactivity timeout should be increased to accommodate the particular client.
Example: inactivity timeout: 30;

soft runtime limit (read/write, static)
The number of minutes since the start time of the group after which no new child processes will be launched. Index and bootstrap saves are exempt and will be started regardless of this setting. An empty string or a value of 0 indicates that no soft timeout will be enforced. Default value is 0.
Example: soft runtime limit: 240;

hard runtime limit (read/write, static)
The number of minutes since the start time of the group, after which all processes launched by the group will be sent a termination request. An empty string or a value of 0 indicates that no hard timeout will be enforced. Default value is 0.

Example: hard runtime limit: 480;

work list (read/write, dynamic, hidden)
The list of saves still not completed. These come in sets of three values: the client name, the level of save, and the path to save.
Example: work list: mars, incr, /usr, mars, incr, /g, mars, venus, /usr

completion (read/write, dynamic, hidden)
The status of each save set that has been completed. These come in sets of four values: the client name, the path saved, a status message (succeeded, failed, or unexpectedly exited), and the output from the save.
Example: completion: "mars", "/usr", "succeeded", "mars: / level=full, 6577 KB 00:06:41 625 files"

progress file name (read/write, dynamic, hidden)
The name of a (private) file where **savegrp** stores the current representation of the group's work list and completion information. Using a file avoids the constant pushing of massive amounts of information between **savegrp** and the RAP database owner, **nsrd**.

status (read-only, dynamic, hidden)
The current status of this NSR group. Currently, this can have the values 'idle', 'running' and 'cloning'. The value 'idle' is set when the group is not active, it is 'running' while backups are in progress, and it is 'cloning' when the backups are complete and clones are automatically being made.

EXAMPLE

The default NSR group resource automatically starts its members at 33 minutes past 3 o'clock in the morning:

```
type: NSR group;
name: Default;
autostart: Enabled;
start time: "3:33";
administrator: root;
```

A complete example follows, with the hidden attributes shown with values reasonable for an active group:

```
type: NSR group;
name: Default;
autostart: Enabled;
start time: "3:33";
options: Restartable;
printer: lp2;
inactivity timeout: 30;
work list: mars, incr, /g, mars, incr, index,
venus, incr, /usr, venus, incr, index,
jupiter, full, /, jupiter, full, /usr,
jupiter, full, index
completion: mars, /, succeeded,
"mars: / level=incr, 31 KB 00:01:01 72 files
",
mars, /usr, succeeded,
"mars: /usr level=incr, 2 KB 00:00:48 5 files
",
venus, /, succeeded,
"venus: / level=incr, 7711 KB 00:04:37 29 files
```

```
";
        administrator:    root, &operator;
```

SEE ALSO **nsr**(8), **nsr_notification**(5), **nsr_pool**(5), **nsr_resource**(5), **nsr_schedule**(5), **nsradmin**(8), **savegrp**(8)

NAME nsr_hypervisor − NetWorker resource type "NSR hypervisor"

SYNOPSIS **type: NSR hypervisor**

DESCRIPTION A resource of type **NSR hypervisor** is used to manage parameters for communicating with the virtual machine monitor of a virtualization platform such as VirtualCenter from VMware. A NetWorker hypervisor also stores information queried from the monitor such as the names of virtual machines found and details about the virtualization environment. See **nsr_resource**(5) for more information on NetWorker resources. To edit the NSR hypervisor resources type:

```
nsradmin −c "type:NSR hypervisor"
```

or use **NetWorker Management Console.** See **nsradmin**(8) for more information on using the NetWorker administration program.

ATTRIBUTES The following attributes are defined for resource type **NSR hypervisor**. The information in parentheses describes how the attribute values are accessed. **Create-only** indicates that the value cannot be changed by an administrator, except when the resource is created. **Read/write** means the value can be changed at any time by authorized administrators. **Choice list** means that any number of values can be chosen from the given list. **Single string** means that only a single value is allowed. **Static** attributes change values rarely, if ever. **Hidden** means it is an attribute of interest only to programs or experts, and these attributes can only be seen when the hidden option is turned on in **nsradmin**(8). For example, an attribute marked **(create-only, static)** has a value set when the attribute is created and never changes. Several additional attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

name (create-only, static)
This attribute holds the name of the hypervisor resource and should be the name of the machine hosting the Hypervisor Management software (for instance, VMware VirtualCenter).

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the resource.

service (read/write, choice list)
This attribute specifies the type of hypervisor monitor to be accessed. Its value is used to generate the default endpoint.

endpoint (read/write)
This attribute contains the URL endpoint used to communicate with the hypervisor web service interface. An empty value means use the default endpoint determined from the selection in the **service** attribute.

username (read/write)
This attribute contains the user name to log into the virtual machine monitor with.

password (read/write)
This attribute contains the password to log into the virtual machine monitor with.

command (read/write)
This attribute specifies the discovery command to execute. An empty value means use the default command based on the selection in the **service** attribute.

proxy (read/write)
This attribute contains the name of the machine to execute the discovery

command on. If empty, the host machine specified by the **name** attribute is used.

server (read/write, hidden)
This attribute contains the name of the machine to use as the NetWorker server. The default value is the host machine of the executing NetWorker server. This is also the value if the attribute is empty.

vm list (read only, hidden)
This attribute contains names of the virtual machines found during the last execution of the discovery command.

environment (read only, hidden)
This attribute contains information about the environment of the virtual machine monitor found during the last execution of the discovery command. The format of this value is dependent upon the type of hypervisor monitor being accessed.

A complete example of resource creation follows:

```
      type: NSR hypervisor;
      name: maelstrom.corp.emc.com;
  username: myusername;
  password: mypassword;
```

FILES */nsr/res/nsrdb* − files in this directory should never be edited directly. Use **NetWorker Management Console or nsradmin instead.**

SEE ALSO **nsr_resource**(5), **nsr_task**(5), **nsr**(8), **nsradmin**(8), **nsrtask**(8), **nsrvim**(8),

NAME nsr_jukebox − NetWorker resource type ''NSR jukebox''

SYNOPSIS **type: NSR jukebox**

DESCRIPTION Each jukebox known to NetWorker is described by a single resource of type **NSR jukebox**. A jukebox keeps track of the resources, volumes and devices that are being managed by an external media management service and are available to this NetWorker server. An example of an external media management service is OpenVault. This resource describes the physical characteristics of a jukebox. See **nsr_resource**(5). To edit the NSR jukebox resources for a NetWorker server, type:

```
nsradmin -c "type:NSR jukebox"
```

or use **NetWorker Management Console.** See the **nsradmin**(8) manual page for more information on using the NetWorker Administration program.

ATTRIBUTES The following attributes are defined for resource type **NSR jukebox**. The information in parentheses describes how the attribute values are accessed. **Create-only** indicates that the value cannot be changed by an administrator, except when the resource is created. **Read-only** indicates that the value cannot be changed by an administrator. **Read/write** means the value can be set as well as read at any time. **Choice list** means that any number of values can be chosen from the given list. **Yes/no** means only a yes or no choice is possible. **Single string** means that only a single value is allowed. **Number** means that only numeric values are allowed. **Static** attributes change values rarely, if ever. **Dynamic** attributes have values which change rapidly. **Hidden** means it is an attribute of interest only to programs or experts, and these attributes can only be seen when the hidden option is turned on in **nsradmin**(8). For example, an attribute marked **(read-only, dynamic)** has a value which cannot be changed by the administrator but which may change each time it is retrieved from the NetWorker server due to underlying state changes. Several additional attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

name (create-only, single string)
This attribute specifies the name of this jukebox. The value of this attribute may follow the "rd=hostname:" syntax of a remote device, when the jukebox is defined on a storage node. See **nsr_storage_node**(5) for additional detail on storage nodes.
Example: name: Huntington;

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the jukebox.

description (read/write)
This attribute is used to store a brief description about the jukebox. The description is used to help administrators identify the jukebox, and it can be in any format.
Example: description: DLT Changer drive in Engineerning Lab;

model (create-only, single string)
This attribute specifies the jukebox model.
Example: model: ADIC-VLS;

physical slots (read-only, list of numbers, hidden)
This attribute specifies the first and last physical slot numbers in the jukebox. The first slot number must be less than or equal to the last slot number. The numbers must also be specified as two separate attribute values.

For Silo Tape Libraries (STL), this attribute is equal to the number of volumes allocated to this NetWorker server, **nsrjb**(5) −**a** or −**x.** The number of physical slots changes as volumes are added or removed from the STL.
Example: physical slots: 1, 54;

control port (read/write, single string)
This attribute specifies the path of the control port for the jukebox robotics. Control commands (load slot 47 into drive b, for example) are sent to the jukebox via the control port.

For an STL, this attribute specifies the information required to set up a connection to the STL server. Form and contents of the attribute depend on the type of the STL, but most often it merely contains the hostname of STL server.

The value of this attribute may follow the "rd=hostname:" syntax of a remote device, when the jukebox is defined on a storage node. See **nsr_storage_node**(5) for additional detail on storage nodes.
Example: control port: scsidev@0.6.0;

devices (read/write, list of strings)
This attribute lists the device pathnames of the devices in the jukebox. Each entry that appears in this attribute must have a corresponding **NSR device** resource. Unless any of the drives are being shared by multiple device resources, there must be the same number of entries in the *devices* attribute as there are physical drives in the jukebox. In addition, they must be listed in the same order as they are physically installed in the jukebox. The entries are specified as separate attribute values.
Example: devices: /dev/rmt/0mbn, /dev/rmt/1mbn;

This attribute is updated by **jbedit** when adding or deleting a shared device or a physical drive.

number devices (read/write, single number, hidden)
The number of configured devices in the jukebox. This value corresponds to the number of entries in the **devices** attribute.
Example: number devices: 2;

This attribute is incremented or decremented by **jbedit** depending on whether a device is being added to or deleted from the jukebox.

number drives (read/write, single number, hidden)
The number of unique physical drives configured in the jukebox. When multiple device resources share a physical drive, each drive is represented by a unique **hardware ID** attribute, that is specified in all of the device resources sharing the same drive.
Example: number drives: 2;

This attribute is updated by **jbedit** when adding or deleting a unique physical drive but left unmodified when changing a shared device.

device hardware ids (read-only, hidden)
The hardware ids of the jukebox's devices. For each entry in the **devices** attribute of the jukebox resource, there will be a corresponding entry in the **device hardware ids** attribute. The **hardware id** entries of those devices sharing a physical drive will have the same value.

slot tags (read-only, hidden)
The tags of the jukebox's slots as reported by the **nsrlcpd (8)** process. This attribute will have a **tag** entry for each configured slot.

drive tags (read-only, hidden)
The tags of the jukebox's drives as reported by the **nsrlcpd (8)** process. For

each entry in the **devices** attribute of the jukebox resource, there will be a corresponding entry in the **drive tags** attribute. The **drive tags** entries of those devices sharing a physical drive will have the same value.

idle device timeout (read/write, hidden)
This attribute specifies the number of minutes to wait before unmounting a volume in an idle device (both shared and unshared). Setting this attribute's value to zero disables unmounting idle volumes. The function of this attribute only applies to SmartMedia jukeboxes, or silo and native jukeboxes with device sharing enabled.
Example: idle device timeout: 10;

SmartMedia update interval (read/write, hidden)
This attribute specifies the number of hours between calls to update the SmartMedia server's database. The SmartMedia database contains information copied from the NetWorker media database. The information includes the pool to which a volume belongs, whether the volume is full, and so forth. This information is used by the SmartMedia server when selecting a volume for writing. Since this information may change over time, it is necessary to periodically make sure that the data replicated in the SmartMedia server's database is current. This attribute determines the time period between attempts to update the SmartMedia server's database. This attribute only applies to SmartMedia jukeboxes.
Example: SmartMedia update interval: 12;

write enabled (read/write, yes/no, hidden)
This attribute indicates whether writing can be done to the mounted volume. This attribute is only used during a jukebox ''Load'' operation. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: write enabled: Yes;

bar code reader(read/write, yes/no)
This attribute indicates whether NetWorker should use the barcode label from the media if the jukebox has a barcode label reader. This should only be enabled if the jukebox has a barcode label reader.
Example: bar code reader: No;

match bar code labels (read/write, yes/no)
This attribute indicates whether NetWorker should use the barcode label instead of a label template when labeling media volumes. This should only be enabled if the jukebox has a barcode label reader and the attribute "bar code reader" is enabled.
Example: match bar code labels: No;

verify label on unload (read/write, yes/no)
This attribute indicates whether NetWorker should verify that a label exists at the beginning of every tape before it is unloaded. If this attribute is set to Yes and this label does not exist, all savesets on the volume are marked suspect and the volume is marked full.

volume expiration (read/write, single string, hidden)
This attribute specifies the expiration time of a volume currently being labeled. For jukeboxes interacting with external media management services, this attribute specifies the minimum expiration time for the volume to be loaded. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.

available slots (read/write, list of numbers)

This attribute specifies the slots containing volumes available to automatically satisfy NetWorker requests for writable volumes. When automatically selecting a writable volume, **nsrjb**(8) will only consider volumes from the list of available slots. The slots are specified as a list of ranges, one range per attribute value. A range may be a single slot number or a pair of slot numbers separated by a dash. The first number of a pair must be less than or equal to the second.

For Silo Tape Libraries, this attribute is automatically updated when adding or removing volumes, **nsrjb**(8) −**a** or −**x.**

When satisfying requests to mount a particular volume (that is, by its volume name) or slot, all of the volumes in the slots listed in *physical slots* can be used. This allows the jukebox to be partitioned, with saves restricted to a group of volumes while all of the volumes contained within the jukebox are accessible for recovers.
Example: available slots: 1-10;

enabler code (read-only, single string, hidden)
This attribute lists the enabler code for the **NSR license** resource (see **nsr_license**(5)) corresponding to this jukebox resource. A jukebox cannot be used until a license enabler has been loaded to control that jukebox.
Example: enabler code: 123456-123456-123456;

enabled slots (read-only, single string, hidden)
The value of this attribute is the number of slots enabled for this jukebox. This attribute's value is set by the server when an enabler code is loaded to the jukebox.
Example: enabled slots: 8;

operation (read/write, choice list, hidden)
This attribute shows the operation currently being performed on the jukebox. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation: Load;

command (read-only, single string, hidden)
This attribute shows the command that caused the current operation to be performed on the jukebox. If the operation was generated via a **nsrjb** command then this attribute shows the entire **nsrjb** command line, otherwise it shows a description of the operation. The attribute is used to pass information between NetWorker programs, and to help the administrator track the status of individual operations (see **nsr_jbop**(5)). This attribute should not be changed manually by the administrator.
Example: command: nsrjb -lnv -S 2;

operation message (read-only, single string, hidden)
This attribute displays an error message after a jukebox operation fails.
Example: operation message: ;

operation device (read/write, single string, hidden)
This attribute passes the name of the device to which the current operation refers. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation device: /dev/rmt/0mbn;

operation drive element address (read/write, hidden)
This attribute passes the element address of the drive to which the current operation refers. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.

Example: operation drive element address: D:082;

operation slots (read/write, single string, hidden)
This attribute passes the slots on which the current operation will be performed. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation slots: 1-10;

operation options (read/write, single string, hidden)
This attribute passes the mode of the volume used when the current operation will be performed, **nsrjb**(5) −**o** option. This attribute is used to pass information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation options: manual;

operation barcodes (read/write, list of strings, hidden)
This attribute passes the volume tags or barcodes on which the current operation will be performed. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator. This attribute is only used for Silo Tape Libraries and is only defined on platforms which provide support for Silo Tape Libraries.
Example: operation barcodes: A01B, A0/3−5/B;

operation response (read/write, choice list, hidden)
This attribute designates a default response to questions that may be asked while performing the operation. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation response: Yes;

operation report mode (read/write, choice list, hidden)
This attribute designates the amount of output generated during the execution of the operation. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation report mode: verbose;

operation label state (read/write, choice list, hidden)
This attribute designates whether a volume being labeled is to be recycled or is expected to be unlabeled. If a volume is to be recycled, it must already have a NetWorker label. You can recycle a volume while it is being mounted. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation label state: recycle;

operation volume capacity (read/write, single string, hidden)
This attribute specifies the capacity of a volume being labeled. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation volume capacity: 10G;

operation volume type (read/write, choice list, hidden)
This attribute specifies types of volumes that may be considered when allocating a volume. It is only used when interacting with an external media management service. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation volume type: 8mm, dlt;

operation ineligible (read/write, hidden)
This attribute specifies volumes which are ineligible for the current operation. Only used when interacting with an external media management service. This

attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation ineligible: ;

operation task (read/write, choice list, hidden)
This attribute designates a secondary task or operation to be performed with the current operation. For example, choosing the **mount after label** task will cause the volume to be mounted after it has been labeled. Currently, this attribute is only used when interacting with an external media management service. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation task: mount after label;

operation instance (read/write, single string, hidden)
This attribute designates an instance number to be associated with the operation. The instance must be unique for all current operations.

operation hostname (read/write, single string, hidden)
This attribute designates the name of the machine on which the operation is to be executed. This attribute is only used for those jukeboxes which support devices attached to multiple hosts. The host machine may be inferred from other attributes for the operation, such as **operation device.** If a device is specified, the operation will be executed on the host for the device. Otherwise the host will be inferred from the name of the jukebox, unless a value is specified for this attribute. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation hostname: host1;

operation dev hostname (read/write, single string, hidden)
This attribute designates the name of the machine from which a device is to be selected for the operation. It applies to shared jukeboxes, which can have drives attached to multiple hosts. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation dev hostname: host1;

operation template (read/write, single string, hidden)
This attribute shows the template that the label operation will use. The verify operation sets this to the volume name found on a piece of media. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation template: Default;

operation volume pool (read/write, choice list, hidden)
This attribute specifies the default volume pool to use when labeling. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation volume pool: NonFull;

operation source pool (read/write, choice list, hidden)
This attribute specifies the pool from which a volume may be selected when recycling a volume. This attribute is only supported on jukeboxes for volumes being managed by an external media management package. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation source pool: Default;

operation uses left (read/write, single string, hidden)

This attribute sets the number of times a cleaning cartridge may be used. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: operation uses left: 12;

volumes (read/write, list of strings, hidden)
This attribute contains a list of resident volume names. The order corresponds to the slot number. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: volumes: mars.001, mars.002, mars.003, mars.004;

volume ids (read/write, list of strings, hidden)
Every volume labeled by NetWorker is assigned a volume identifier, often referred to as a volid. This attribute contains a list of volume identifiers for the resident volumes. The volume identifiers stored could be the new long volume IDs or the older and shorter volume IDs. The type of volume identifiers stored depends on whether the storage node on which the device belonging to the jukebox resides on, supports the new long volume id or not. The order corresponds to the slot number.
This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: volumes: 24198, 24199, 24200, 24197;

volume cartridge ids (read/write, list of strings, hidden)
Some jukeboxes track volumes that are managed by external media management services. There may be multiple volumes on the same media, for example, a volume on each side of an optical disk. This attribute is used to track the identifier for each cartridge on which a volume resides. The order corresponds to the slot number. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.

loaded volumes (read/write, list of strings, hidden)
This attribute contains the names of the volumes currently loaded on the jukebox devices. The order is with respect to the *devices* attribute. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.

This attribute is updated by **jbedit** when adding or deleting a shared device or a physical drive.
Example: loaded volumes: mars.089, mars.003;

Using the names specified in the previous *devices* attribute, *mars.089* is loaded in '/dev/rmt/0mbn' and *mars.003* is loaded in '/dev/rmt/1mbn'.

loaded bar codes (read/write, list of strings, hidden)
This attribute contains the barcodes of the loaded volumes, if the use of barcodes is enabled for the jukebox. The order is with respect to the *devices* attribute. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.

This attribute is modified by **jbedit** when a device is added to or removed from the jukebox resource.
Example: loaded barcodes: 12345, 67890;

Using the names specified in the previous *devices* attribute, the volume with barcode 12345 is loaded in '/dev/rmt/0mbn' and the volume with barcode 67890 is loaded in '/dev/rmt/1mbn'.

loaded slots (read/write, list of numbers, hidden)
This attribute contains the slot numbers of the loaded volumes. The order is with respect to the *devices* attribute. This attribute passes information between

NetWorker programs, and should not be changed manually by the administrator.

This attribute is modified by **jbedit** when a device is added to or removed from the jukebox resource.
Example: loaded slots: 48, 3;

Using the names specified in the previous *devices* attribute, the volume in slot 48 is loaded in '/dev/rmt/0mbn' and the volume in slot 3 is loaded in '/dev/rmt/1mbn'.

event tag (read/write, single number, hidden)
This attribute contains the tag (unique identifier) of the last notification event sent to the **nsrd**(8) daemon. The tag is used by **nsrjb**(8) to clear the previous event. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: event tag: 6319962287;

event message (read/write, single string, hidden)
This attribute contains the text of the last notification event sent to the **nsrd**(8) daemon. The **nsrjb**(8) command will send a notification event to **nsrd** when operator intervention is needed before **nsrjb** can proceed. This attribute passes information between NetWorker programs, and should not be changed manually by the administrator.
Example: event message: could not unload device /dev/rmt/1mbn into slot 4;

messages (read/write, list of strings, hidden)
This attribute contains a log of messages reflecting previous operations **nsrjb**(8) has done. Generally, an entry is made each time **nsrjb** is invoked and for each mechanical operation. Each entry is timestamped. This attribute is used to pass information between NetWorker programs, and should not be changed manually by the administrator.
Example: messages: 04/01/91 01:15:08 loaded slot 4 into drive a;

minimum space (read/write, single string, hidden)
This attribute contains the low water mark for remaining space. When the remaining space on the volumes contained in the available slots is less than the minimum space, irrespective of whether there is only one volume or more in the Jukebox, an alert notification is sent that states that the volumes in the named jukebox has passed the low water mark, the first time its noticed. After this, during subsequent checks, if the situation has not changed, that is, remaining space is still less than or equal to the limit specified in "minimum space", a notification is sent to **nsrd** about the available space on the volume(s). If there is just one volume, then, even if the limit has not been reached, the notification about the available space on the volume, is sent. This hidden attribute can be modified by a user. *This hidden attribute can be modified by a user.*

The minimum space may be specified as a number of gigabytes or megabytes. Either 'G' or 'g' may be used for gigabytes, 'M' or 'm' for megabytes.
Example: minimum space: 7g;

jukebox options (read-only, list of strings, hidden)
This attribute contains a list of the options for this jukebox. This option is automatically set after jukebox creation.
Example: jukebox options: two_sided;

auto clean (read/write, yes/no)
This attribute specifies whether automatic cleaning of devices in the the jukebox is enabled.

Example: auto clean: Yes;

cleaning slots (read/write, list of numbers)
This attribute designates a range of slots in the jukebox that has been set aside for cleaning cartridges. A range may be a single slot number or a pair of slot numbers separated by a dash. If a pair of slot numbers is given, the first number of the pair must be less than or equal to the second. Only one range of slots may be set aside for cleaning cartridges. If **auto clean** is set to **no**, the value of cleaning slots is ignored and these slots may contain regular volumes. When **auto clean** is set to **yes**, the range of slots specified for this attribute are assumed to contain cleaning cartridges, and the range of slots specified by **available slots** and this attribute must not overlap.

For Silo Tape Libraries this attribute should not be changed directly. This attribute is automatically updated, when adding (**nsrjb −U**) or removing (**nsrjb −x**) cleaning cartridges.
Example: cleaning slots: 9-10;

default cleanings (read/write, single number)
This attribute designates the number of uses assigned to a new cleaning cartridge during an inventory of a jukebox by **nsrjb**(8). A cleaning cartridge is considered to be new when a slot set aside for cleaning cartridges that was empty is discovered to be full during an inventory of a jukebox.
Example: default cleanings: 12;

auto media management (read-write)
This attribute indicates whether automated media management for the jukebox is enabled. The value can be **yes** or **no**. If the value is set to **yes**, then unlabeled volumes in the jukebox may be automatically labeled by NetWorker. NetWorker verifies that the volume is unlabeled before labeling the volume. A volume is considered to be unlabeled if the volume does not contain a label that may be read by the device in the jukebox into which the volume is loaded. Note that if the volume contains a label, but the label is written at a density that cannot be read by the device the volume is considered to be unlabeled. If the volume contains data written by an application other than NetWorker, it most likely does not have a label recognizable by NetWorker and the volume is considered to be unlabeled. With this attribute enabled, care should be taken when loading any volume considered to be unlabeled into the jukebox. The volume may be re-labeled and the data previously on the volume over-written by NetWorker. For devices in a jukebox the value of their **auto media management** attribute is always **no**.
Example: auto media management: yes;

STL device names (read/write, list of strings)
This attribute lists the corresponding Silo device names of the devices listed in the **devices** attribute. If several device resources are sharing the same physical Silo drive, as indicated by a common **hardware ID** value, this attribute will only have an entry for each of the physical drives. This attribute is only used for Silo Tape Libraries and is only defined on platforms which provide support for Silo Tape Libraries.

STL interface lib (read/write, single string)
The pathname of the dynamically linked interface library. This attribute is only used for Silo Tape Libraries and is only defined on platforms which provide support for Silo Tape Libraries.
Example: STL interface lib: /usr/lib/libstl.so.1;

STL device sharing (read/write, single string)
This attribute specifies how to handle device sharing. Device sharing means automatic, load dependent, device switching of devices in a Silo Tape Library

between different hosts connected to the library. This feature can only be used if it is supported by the **STL interface lib**. Possible values for this attribute are an empty string (device sharing disabled) or "perm-max", where perm and max are numbers with perm < max. The perm value is the number of devices that can be reserved permanently (do not require releasing). The max value is the maximum number of devices that can be reserved. This attribute is only used for Silo Tape Libraries and is only defined on platforms which provide support for Silo Tape Libraries.
Example: STL device sharing: 2-4;

STL barcodes (read/write, list of strings, hidden)
The barcodes of the volumes in the library, which are available for NetWorker. This attribute maintains the volume names used by the Silo Tape Libraries for the corresponding volumes in the **volumes** attribute. This attribute is only used for Silo Tape Libraries and OpenVault virtual jukeboxes. The attribute is only defined on platforms which provide support for Silo Tape Libraries or OpenVault.

STL device reservation (read/write, list of strings, hidden)
This list contains the reservation state of shared devices in a tape library. The possible states are "Yes" (device is reserved), "No" (device is not reserved) and "Error" (an error occurred during release of this device). The order of the reservation state matches the 'devices' attribute. This attribute is only used for Silo Tape Libraries with device sharing enabled and is only defined on platforms which provide support for Silo Tape Libraries.

application name (read/write, encrypted, hidden)
This attribute is only used for OpenVault jukeboxes. OpenVault requires any application to identify itself when submitting a request. This is the name used by this server to identify itself to OpenVault when submitting a request to access resources listed in this jukebox.

application key(read/write, encrypted, hidden)
This attribute is only used for OpenVault jukeboxes. OpenVault requires any application to identify itself when submitting a request. This is the key used by this server to identify itself to OpenVault when submitting a request to access resources listed in this jukebox.

read hostname (read/write, single string)
The hostname that is used in selecting a storage node for recover and read-side clone requests. For recover requests, if the required volume is not mounted, and the client's "storage nodes" attribute does not match one of the owning hosts in the jukebox, then this attribute is used. For clone requests, if the required volume is not mounted, then this attribute is used.

NDMP jukebox (read-only, yes/no)
This attribute specifies that the jukebox robotics is controlled by the NDMP Tape Server host.

NDMP type (read-only, Choice list, hidden)
This attribute specifies the type of control that the NDMP jukebox provides. Other than Celestra 1.6 on Solaris, all other jukeboxes that are controlled by NDMP Tape Server will have the value "Logical Handle Device".

NDMP hostname (read-only, single string)
This attribute specifies the NDMP Tape Server hostname that is controlling the jukebox robotics.

remote user (read/write, single string, hidden)

This field is no longer used. The NDMP user name on the NDMP Tape Server host is used from **NSR storage node** (5) resource.

password (read/write, single string, hidden)
This field is no longer used. The password for the NDMP user on the NDMP Tape Server host is used from **NSR storage node** (5) resource.

NDMP jukebox handle (read-only, single string)
This attribute specifies the jukebox handle on the NDMP Tape Server host to control the jukebox robotics.

NDMP bus number (read-only, number, hidden)
This attribute specifies the BUS number of the Jukebox on the NDMP Tape Server host. This field maintained for backward compitiability.

autodetect id (read/write, hidden)
This attribute is for identifying autodetected devices. It is used by NetWorker programs only, and should not be changed manually by the administrator.

server network interface (read/write, hidden)
This attribute defines the network address or the hostname which is used to communicate with nsrjb. This field is only relevant, if the jukebox is connected to a storage node.

jukebox serial number (read only)
This attribute indicates the serial number of the jukebox.

hardware id (read only, hidden)
This attribute indicates the hardware identifier for the auto-detected jukebox.

ASCAPI (read/write, yes/no, hidden)
This attribute is kept for historical reasons only. It has no affect.

debug trace level(read/write, single number, hidden)
The level of debug messages generated & displayed during a jukebox operation may be set on a per jukebox basis. The value default value is 0, which means no debug information collected or displayed. Higher values represent increasingly larger amounts of debug information being captured.

operation timeout(read/write, single number, hidden)
The time, in seconds, that a jukebox operation may sit idle before it is automatically cancelled. The default is 1800 seconds (30 minutes).

operation lifespan(read/write, single number, hidden)
The time, in seconds, that the jukebox operation resource (for a completed jukebox operation) is kept in **nsrd** before automatically being deleted. The default value is 1800 seconds (30 minutes)

ready (read/write, yes/no, hidden)
Indicates whether the jukebox is ready to accept operations to be executed. The value is automatically set to "No" when Networker is started, and will be changed to "Yes" once Networker determines that the jukebox has finished initializing, and is ready to begin work. Any reset of the jukebox will also change the *ready* attribute back to "No" until the rest has completed.

virtual jukebox (read only, yes/no)
Indicates whether NetWorker has detected the jukebox as a virtual library device. This value will be set to "Yes" if, and only if, NetWorker has detected that the configured library is a virtual library. A facility for NetWorker to determine whether a library is virtual must be provided by the VTL vendor. Default value is "No".

virtual jukebox frameid (read/write, single string)
The unique identifier of the physical device providing the virtual library

services. Any number of virtual libraries are allowed to be created in a given virtual library server. For detected virtual libraries, the default value is "default vtl location". For non virtual libraries, the default value is an empty string.

unconfigured devices (read only, hidden)
Indicates the list of device names belongs to this jukebox, which are discovered by auto-detect during last scan however not yet configured in the library.

library drive base address (read only, hidden)
Indicates the base element address of the drives in the jukebox. The device detection program, **dvdetect,** updates the resource.

existing drive ids (read only, hidden)
Inidcates the serial numbers information of the currently existing drives in the jukebox. The device detection program, **dvdetect,** updates the resource.

existing drive addresses (read only, hidden)
Indicates the element addresses of the currently existing drives in the jukebox. The device detection program, **dvdetect,** updates the resource.

unconfigured drive ids (read only, hidden)
Indicates the serial numbers information of the unconfigured drives (detected in last scan however not yet configured) in the jukebox. The device detection program, **dvdetect,** updates the resource.

unconfigured drive addresses(read only, hidden)
Indicates the element addresses of the unconfigured drives (detected in last scan however not yet configured) in the jukebox. The device detection program, **dvdetect,** updates the resource.

jukebox features (read/write, hidden)
List of possible features the jukebox may contain or provide. Some of these features are described below.

auto_inventory:
nsrlcpd will automatically perform an inventory when a change in the hardware status is detected. Changes in hardware status may occur due to cap or door accesses, and other reasons.

autoeject:
The jukebox does not require an eject operation for data catridges prior to receipt of a move medium command. The default is that data cartridges are always automatically ejected.

barcode:
The jukebox contains a barcode reader.

volume_tags:
Volume tags are supported by the library. These are usually, but not necessarily, tape barcode labels. Volume identification information may be obtained by reading MAM, or by other means that may be vendor specific.

auto_cc_unload:
Cleaning cartridge is automatically unloaded by the library. Set based on jukebox model.

cc_eject:
Eject cleaning cartridge. NOTE: this must be set in conjunction with 'auto_cc_unload'. If 'cc_eject' is set, and 'auto_cc_unload' is _not_ set, NetWorker will eject the cleaning cartridge from the device. The default is cleaning catridges are automatically ejected.

elements_status:

Jukebox has an initialize element status feature. This implies the jukebox has the ability to determine what is in a slot.

has_range:
Jukebox has 'range' initialization. Default is based on the jukebox. See the jukebox Users Guide for further details.

ies_no_barcode:
Set based on jukebox manufacturer and model.

need_align
This option is used by Adic VLS jukeboxes. The jukebox has a magazine separated from the devices in it by a partition. This partition contains a window. To load or unload volumes into a device, both the slot and device have to be aligned to the window prior to an eject or move media operation.

EXAMPLE A resource defining a jukebox named *Huntington* is shown. The *model* attribute specifies a 'Exabyte 210' jukebox. The *control port* attribute specifies the bus, target, and LUN ID for the robotics device 'scsidev@0.6.0'. The *device* attribute lists the pathnames of the two tape devices in the jukebox, '/dev/rmt/0mbn' and '/dev/rmt/1mbn'. Since the jukebox has a bar code reader, the two bar code yes/no attributes are both set to 'Yes'. The *available slots* attribute lists the slots to consider when automatically selecting a volume to load for writing. The available slots are 2 through 11. The hidden attributes are displayed. *auto clean* is yes so automatic cleaning of devices is enabled for this jukebox. *cleaning slots* is set to slot 1. This slot is reserved for a cleaning cartridge.

```
                  type: NSR jukebox;
                  name: Huntington;
                 model: EXB-210;
          physical slots: 1-11;
          control port: scsidev@0.6.0;
               devices: /dev/rmt/0mbn, /dev/rmt/1mbn;
         number devices: 2;
          number drives: 2;
    device hardware ids: "", "";
idle device timeout: 10;
SmartMedia update interval: 12;
          write enabled: Yes;
        bar code reader: Yes;
    match bar code labels: Yes;
      volume expiration: ;
        available slots: 2-11;
           enabler code: 012345-6789ab-cdef00;
              operation: ;
      operation message: ;
       operation device: ;
        operation slots: ;
        operation ports: ;
      operation options: ;
     operation barcodes: ;
     operation response: ;
  operation report mode: ;
```

```
                operation label state: ;
           operation volume capacity: ;
                 operation volume type: ;
                  operation ineligible: ;
                       operation task: ;
                   operation instance: ;
                   operation hostname: ;
              operation dev hostname: ;
                   operation template: ;
                operation number uses: ;
                operation volume pool: ;
                operation source pool: ;
                              volumes: -, -, -, -, -, -, -, -, -, -, -;
                           volume ids: "", "", "", "", "", "", "", "", "", "", "";
                        STL barcodes: ;
                   STL device sharing: ;
               STL device reservation: ;
                     STL interface lib: ;
                            event tag: ;
                        event message: ;
                             messages: "09/12/03 11:50:56 CREATED";
                        minimum space: 7g;
                      jukebox options: ;
                           auto clean: Yes;
                       cleaning slots: 1;
                     default cleanings: 12;
           auto media management: Yes;
                          reset class: initialize unload;
                     application name: ;
                      application key: ;
                        read hostname: hostname;
                   debug trace level: 0;
                   operation timeout: 1800;
                               ready: Yes;
```

SEE ALSO **nsr**(5), **nsr_device**(5), **nsr_storage_node**(5), **nsradmin**(8), **nsrd**(8), **nsrjb**(8), **dvdetect**(8), **NSR storage node**(8)

NAME NSRLA − resource for the NetWorker client execution service

SYNOPSIS **type: NSRLA**

DESCRIPTION The NSRLA resource is used by NetWorker client execution service **nsrexecd** (see **nsrexecd**(8)). To edit the **NSRLA** resources run:

```
nsradmin -s host_name -p nsrexec -c type:NSRLA
```

or

```
nsradmin -s host_name -p 390113 -v 1 -c type:NSRLA
```

See **nsradmin**(1m) for information on using the NetWorker administration program.

ATTRIBUTES The following attributes are defined for resource type **NSRLA** The information in parentheses describes how the attribute values are accessed. **Hidden** means it is an attribute of interest only to programs or experts, and these attributes can only be seen when the hidden option is turned on in **nsradmin**(1m). **Static** attributes change values rarely, if ever. For example, an attribute marked **(read-only, static)** has a value which is set when the attribute is created and may never change. Not all attributes are available on all Operating Systems.

name (read-only, single string)
The name attribute specifies the NW instance name of the local install. This value is is a shorthand for the NW instance name. The value in this attribute should be entered where ever a NetWorker instance needs to be referred to. The default value is hostname. The value should be unique throughout the data zone.

NW instance ID (read-only, hidden, single string)
The NW instance ID. This value will be used to identify the local NetWorker install whenever a NetWorker program needs to communicate with another NetWorker program. This value has a one to one correspondence with the NetWorker instance name. The value is created by NetWorker by default. It should be unique throughout the data zone.

certificate (read-only, hidden, single string)
The certificate for the local NetWorker installation. The certificate is used by remote NetWorker installations to validate the identity of the local NetWorker install.

private key (read-only, hidden, single string)
The private key for the local NetWorker installation. The private key is used to validate the identity of the local NetWorker install to remote NetWorker installations. No users can view this attribute.

NW instance info operations (read-write, dynamic, choice)
One can use this attribute to import or export the NW instance information, or request that NetWorker generate a new private key and certificate. The NW instance information includes: the **name**, **NW instance ID**, **certificate**, and **private** key attributes. Setting this field to: **Export** will cause NetWorker to export the NW instance information to a file specified in the **NW instance info file** attribute. Setting this field to: **Import** will cause NetWorker to import the NW instance information from the file specified by the **NW instance info file** attribute and set the corresponding values in the **nsr_la(5)** resource. Setting this field to: **New Keys** will cause NetWorker to generate a new private key and certificate and store the values in the **private key** and **certificate** fields. The previous values in these fields will be overwritten. This

field will be reset to blank after NetWorker uses the value.

NW instance info file (read-write, dynamic, single string)
This field is used to specify a file name where NetWorker should load/store information when the **NW instance info operations** attribute is set to **Export** or **Import**. This field will be reset to blank after NetWorker uses the value.

version (constant, single string)
The software release version string. The string is made up of four parts: the company name, the product name, the release identifier, and the release timestamp.

servers (constant, list of string)
The list of NetWorker servers allowed to back up this client.

disable directed recover (read-write, hidden, choice yes/no)
Determines whether remote hosts are allowed to direct recoveries to this host. The default setting is **no** which means that remote hosts are allowed to direct recoveries to this host. To disable the ability of remote hosts to direct recoveries to this host, set this attribute to **yes.**

auth methods (read-write, list of strings)
This field will be used to specify the list of authentication methods that are allowed for communicating with the listed peer. The allowed authentication methods can be specified on a per peer basis.

The auth methods field is a multivalued attribute whose values are of the following format (in regular expression format):

IP(/mask)?,auth1(/authN)*

where *IP* = the peer's IP address or a network IP address, mask = the network mask (in the 255.255.0.0 format or a number indicating the number of 1's on the left side of the mask), and auth1, ..., authN are of the values: *nsrauth* or *oldauth*.

The value: *oldauth* indicates that AUTH_NONE, AUTH_UNIX, and AUTH_LGTO will be accepted by a TCP server and tried by a TCP client for the indicated set of peers. The value of *nsrauth* indicates that GSS EMC v1 authentication or nsr logon authentication (used by the GUI) will be tried/accepted.

The "auth methods" field is order dependent. That is, the client/server will look for a match for the peer in the list starting with the first entry. It will continue looking through the list until it finds a match. If it does not find a match, then it will use the default value of: "0.0.0.0/0,nsrauth/oldauth".

For example:

The following entries (combined) mean that the host: 10.101.1.1 can use AUTH_NONE, AUTH_UNIX, and AUTH_LGTO to authenticate, anything on the network: 137.69.0.0/16 network can use AUTH_NONE, AUTH_UNIX, AUTH_LGTO, and GSS EMC v1 to authenticate, and all other machines must use GSS EMC v1 to authenticate.

"10.101.1.1,oldauth", "137.69.0.0/16,nsrauth/oldauth", "0.0.0.0/0,nsrauth"

types created (constant, hidden, list of strings)
The type of resources that can be created by the **NSRLA** server. (see **hagentd**(8)).

administrator (read-write, list of strings)
The **administrator** list contains users and user netgroups that are allowed to add, delete, and update the **NSRLA** resources. **Netgroups** are indicated by preceeding the name by an ampersand **('&')** character. The **NSRLA** administrator list is used by default for all other **NSRLA** resources. Updating this list will also update the other resources on this server that have not been explicitly changed.

arch (constant, single string)
The instruction set architecture determines which user programs are binary compatible.

kernel arch (constant, single string)
The kernel architecture determines which operating system kernels are binary compatible.

CPU type (constant, single string)
The CPU type of this machine.

machine type (constant, single string)
The high-level type of this machine, e.g. server or workstation.

OS (constant, single string)
The operating system and version number.

NetWorker version (constant, single string)
The version of **NetWorker** client.

client OS type (constant, single string)
The operating system running on this machine.

CPUs (constant, number)
The number of CPUs on the system.

disks (const, number)
The number of secondary storage devices on the system that can be backed up.

MB used (dynamic, number)
The current total amount of disk storage used in MB.

IP address (constant, list of string)
Dotted-decimal Internet Protocol address list.

hostname (constant, hidden, single string)
The hostname of the machine on which the service that controls this resource is running. It is used internally and cannot be changed by the administrator.

ONC program number (constant, hidden, number)
The Open Network Computing (sunrpc) identification number for the client to server protocol provided by this service.

ONC version number (constant, hidden, number)
The Open Network Computing (sunrpc) identification number for the client to server protocol provided by this service.

ONC transport (constant, hidden, list of choice)
The Open Network Computing (sunrpc) transport protocols supported are **TCP** (the Transport Control Protocol) or **UDP** (the User Datagram Protocol).

SEE ALSO

nsradmin(1m), **nsrexecd**(8), **hagentd**(8)

NAME nsr_label − NetWorker resource type ''NSR label''

SYNOPSIS **type: NSR label**

DESCRIPTION Each NSR label template is described by a single resource of type **NSR label** (see **nsr_resource**(5)). To edit the NSR label resources for a NetWorker server, type:

```
nsradmin -c "type:NSR label"
```

or use **NetWorker Management Console.** See the **nsradmin**(8) manual page for more information on using the NetWorker administration program.

This resource describes the templates used to generate volume labels.

ATTRIBUTES The following attributes are defined for resource type **nsr_label**. The information in parentheses describes how the attribute values are accessed. **Read-only** indicates that the value cannot be changed by an administrator. **Read/write** means the value can be set as well as read. **Choice** means that the value of the attribute can only be one from a list specific to that attribute (for example, separator can be '-', or '.'). Several additional attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the label.

fields (read/write, list of strings)
This attribute specifies the constituent fields of a *label template*. When generating a volume name, the current value of each field is concatenated. The first field is considered the most significant, the last field the least. If there is a *separator* (see below) defined, then it will be placed between fields as they are concatenated to form a volume name. The fields are separated by commas.

There are four different types of fields: 'numeric range', 'lower-case range', 'upper-case range', and a 'list of strings'. A 'list of strings' consists of space (' ') separated strings. The other types are specified as starting and ending values separated by a dash ('-') . The starting and ending values of a range must have the same number of characters.

The *next* attribute (see below) contains the current position or value of each field. After a volume name has been assigned to a volume, the *next* attribute is incremented. When the ending value is reached, the current value will wrap around to the starting value. A 'list of strings' field is incremented by selecting the next string in the list. A numeric range field is incremented by adding 1 to its current value. Lower-case and upper-case ranges are incremented by moving on to the next letter in the least significant position. In the example below, after aa.99, the next label would be ab.00.
Example: fields: aa-zz, 00-99;

name (create only, single string, static)
This attribute specifies the name of this label template. The label template is referred to by its name in the jukebox resource, see **nsr_jukebox**(5).
Example: name: Default;

next (read/write, single string)
This attribute specifies the next volume name to use. After it is assigned to a volume, the next volume name will be generated and remembered here. The attribute consists of a component for each of the specified fields and the separator.
Example:

next: aa.00;

Using the separator and field attributes shown above, the *next* attribute would show: next: aa.01;
This would be followed by: next: aa.02;

separator (read/write, single choice, null ok)
This attribute specifies the character to use to separate the label fields. It may be one of '.', '_', ':', '-' or NULL.
Example: separator: .;

EXAMPLES

A label resource named *engineering* is shown below. (Hidden options are not shown.) There are two range-type fields defined, the first ranging from 'aa' to 'zz', the second from '00' to '99'. The *separator* attribute has the value '.' and it will be inserted in between the two fields. The *next* attribute holds the next name that will be generated by this template. After *aa.00* is used, the *00* will be incremented. The new name will be *aa.01*. After 98 more names have been generated, the *next* attribute will hold the name *aa.99*. When this name is incremented, the *next* attribute will hold *ab.00*. After generating 67,500 more names, the *next* attribute will hold *zz.99*. This will be followed by *aa.00*.

```
      type: NSR label;
      name: engineering;
      fields: aa-zz, 00-99;
 separator: .;
      next: aa.00;
```

A label resource named *accounting* is shown below. The field attribute defines five component fields. The *separator* attribute has the value '.'. It will be inserted in between adjacent fields. The *next* attribute holds the next name that will be used with this template. After *0.23.aa.AA.first* is used, the fifth field will be incremented. The new name will be *0.23.aa.AA.second*. This will be followed by *0.23.aa.AB.first*. After 1349 more volume names, the name will be *0.23.aa.ZZ.second*. This will be followed by *0.23.ab.AA.first*. After using *9.45.zz.ZZ.second*, the name will wrap around to *0.23.aa.AA.first*.

```
      type: NSR label;
      name: accounting;
      fields: 0-9, 23-45, aa-zz, AA-ZZ, first second;
 separator: .;
      next: 0.23.aa.AA.first;
```

SEE ALSO

nsradmin(8), **nsrjb**(8), **nsrmm**(8), **nsr**(8), **nsr_jukebox**(5)

NAME nsr_layout - NetWorker file layout

SYNOPSIS **type: NSR layout**

DESCRIPTION The NetWorker server filesystem has a directory called **/nsr** that contains log files, on-line indexes, and configuration information. This directory can be created in any filesystem with /nsr set up as a symbolic link to the actual directory (this is determined at installation time). The format of this directory is as follows:

/nsr/logs
Contains server logging messages. The files in this directory are in UTF8 format (generally in ASCII, a subset of UTF8).

/nsr/res Contains the configuration files for various components of the NetWorker server. For example, the server stores configuration files in **/nsr/res/nsrdb**.

/nsr/mm
Contains the media index. Information about the contents of this index file can be printed with the **nsrls**(8) command. See the **nsrmm**(8) and **mminfo**(8) manual pages on how to view and manipulate the media index information.

/nsr/index
This directory contains subdirectories with names that correspond to the NetWorker clients that have saved files. Each index directory contains files that allow the NetWorker server to provide an on-line database of the client's saved files. The most important element is the **db6** directory which contains the NetWorker save records and access indexes to those records. The disk space utilized by the index grows with the number of files saved by the NetWorker service. Administrators should plan to use about 200 bytes per saved file instance placed in this index. There are no practical limits on the maximum size of an online index, except that it must reside entirely within a single file system.

The format of the **db6** directory is subject to change, and is accessible only through an RPC interface to **nsrindexd**(8). However, the **nsrls**(8) command can be used to obtain some useful statistics from this directory. The **nsrck**(8) command is used for checking and rebuilding index files as well as recovering index files from backup media.

The data in the files in the **db6** directory are stored in platform-independent order, so these files may be migrated from one NetWorker server to another. Moving the media database from one NetWorker server to another of unlike architecture is not currently supported.

The files in the **db6** directory include the files listed below. Note that these files are for the internal use of the server and are not to be modified or changed for any purposes.

<**savetime**>**.rec**
These files contain the index records for each file saved at the savetime, where <**savetime**> is a hexadecimal representation of the time.

<**savetime**>**.k0**
These files contain the keys on the <**savetime**>**.rec** file based on file name.

<**savetime**>**.k1**
These files contain the keys on the <**savetime**>**.rec** file based on inode. These may be zero length files if the client file index is for a windows client.

<**savetime**>**.sip**

This is a save-in-process file and only exists when a save has been started and is not yet complete. Once the save is complete, this file is renamed to <**savetime**>**.rec.**

v6hdr This file contains a summary of all the <**savetime**>**.rec** files that exist in a client's **db6** directory.

v6journal
This file contains updates to the **v6hdr** file that are waiting to be merged into the **v6hdr** file. Any index operation includes the entries here as well as the ones in the **v6hdr.**

v6ck.lck
This file is a lock that **nsrck** uses to ensure that only one **nsrck** operates on a client's index at any given time.

v6hdr.lck
This file locks the **v6hdr** for reading and the **v6journal** for reading and writing.

v6tmp.ptr
This file refers to the working directory in which conversion and recovery will take place.

tmprecov
This is a working directory in which conversion and recovery take place.

recovered
This directory contains intermediate results of an index that has been converted or recovered. The results here are complete and will be integrated into the file index when **nsrck** is run against this client file index.

/nsr/cores
Contains directories that correspond to the NetWorker server daemons and certain executables. Each directory may contain core files from NetWorker server daemons or executables that have abnormally terminated.

/nsr/drivers
This directory may contain any device drivers for use with NetWorker.

/nsr/tmp
This directory contains temporary files used by the NetWorker system.

The executables for the NetWorker system are usually installed in the **/usr/etc** or **/usr/bin**, directories though alternate locations may be chosen during installation. See **pkgadd**(1M) for details on alternate executable locations for Solaris 2.x.

FILES

/nsr	NetWorker indexes, log files, and configuration information.
/usr/etc, /usr/bin	Where NetWorker executables for the native architectures are normally installed.
/usr/bin, /usr/sbin, /usr/lib/nsr	Where NetWorker executables for Solaris 2.x are normally installed.

SEE ALSO

nsrck(8), **nsrindexd**(8), **nsrls**(8), **nsrmm**(8), **mminfo**(8)

NAME nsr_license − NetWorker resource type ''NSR license''

SYNOPSIS **type: NSR license**

DESCRIPTION A resource of type **NSR license** is used to describe each feature enabled in your NetWorker installation. See **nsr_resource**(5) for more information on NetWorker resources. To inspect the NSR license resources type:

```
nsradmin −c "type:NSR license"
```

or use **NetWorker Management Console.** NSR license resources may be created, enabled and authorized from the GUI, but the **nsrcap**(8) command must be used to update an existing license resource. See **nsradmin**(8) for more information on using the NetWorker administration program.

ATTRIBUTES The following attributes are defined for resource type **NSR license**. The information in parentheses describes how the attribute values are accessed. **Create-only** indicates that the value cannot be changed by an administrator, except when the resource is created. **Read/write** means the value can be changed at any time by authorized administrators. **Read-only** means the value cannot be changed by authorized administrators. **Static** attributes change values rarely, if ever. **Dynamic** attributes may change often. **Hidden** means it is an attribute of interest only to programs or experts, and these attributes can only be seen when the hidden option is turned on in **nsradmin**(8). For example, an attribute marked **(create-only, static)** has a value which is set when the attribute is created and never changes. Several additional attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

name (create-only, static)
This attribute holds the name of the license resource.

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the license.

enabler code (create-only, static)
This code is identical to the code entered into the **nsrcap**(8) command to enable the feature named in this resource. The enabler code consists of 18 hexidecimal digits, printed in groups of 6 digits.
Example: enabler code: 123456-123456-123456;

host id (read-only, dynamic)
The unique host id associated with the computer or licensed operating system on which the enabler has been loaded. This value will often be an 8 digit hexidecimal number; however, other formats are possible, depending on specific platform requirements.
Example: host id: 7260d859;

expiration date (read-only)
The date on which this enabler will expire, if the enabler is an evaluation enabler or an otherwise un-registered license enabler. The enabler expires at 12:00:01 am on the date listed. The special prefix *G* means that a grace period has been allowed for this enabler. Enablers with the grace period allowed should be registered immediately. If the enabler has been registered, and the **auth code** is filled in with a valid value, the value will be as shown in the example, below.
Example: expiration date: Authorized - no expiration date;

auth code (read/write)
An 8 digit hexidecimal value used to permanently authorize an enabler. The

unique, valid authorization code for an enabler is obtained from EMC by registering each purchased license enabler. Evaluation enablers cannot be permanently authorized. If the server's host id changes, all authorization codes will immediately be invalidated, and the enablers must be re-registered with EMC to obtain new authorization codes.
Example: auth code: abcdef00;

license type (create-only, hidden)
A special code, used internally to describe the specific feature or features enabled by this license enabler.
Example: license type: J16;

checksum (read/write, hidden)
A coded checksum used to maintain consistency of a NSR license resource, and between license resources.

EXAMPLE Below is a complete NSR license resource for an authorized base enabler:

```
type:            NSR license;
name:            NetWorker Advanced/10;
enabler code:    123456-123456-123456;
host id:         7260d859;
expiration date: Authorized - no expiration date;
auth code:       abcdef00;
license type:    B10;
checksum:        xxxxxxxxxxxxxxxxxxxxx;
```

FILES */nsr/res/nsrdb* − files in this directory should never be edited directly. Use **NetWorker Management Console** instead.

SEE ALSO **nsr_resource**(5), **nsr**(8), **nsradmin**(8), **nsrcap**(8)

NAME nsr_mount_request − NetWorker resource type "NSR mount request"

SYNOPSIS **type: NSR mount request**

DESCRIPTION When a nsrmmd processes requires media for a session, e.g. save or recover, it sends a request to nsrd. If there is no media currently mounted which meets the requirements of the nsrmmd process, nsrd starts a nsrjb command to mount media. Before nsrd starts the nsrjb command a resource of the type **NSR mount request** is created. The resource exists until the corresponding nsrjb command exits. This resource and all its attributes are read only. A resource of this type may not be created or deleted using any administrative interface. See **nsr_resource**(5) for information on NetWorker resources. To view the **NSR mount request** resources run:

```
nsradmin -c "type:NSR mount request"
```

Be sure to include quotation marks and to insert a space between "NSR", "mount", and "request".

ATTRIBUTES The following attributes are defined for the resource type **NSR mount request**. The information in parentheses describes how the attribute values are accessed. **Read-only** indicates that the value cannot be changed by an administrator. **Hidden** indicates a hidden attribute of interest only to programs or experts. These attributes can only be seen when the hidden option is turned on in **nsradmin**(8). **Static** attributes change values rarely, if ever. **Dynamic** attributes have values that change rapidly. For example, an attribute marked **(read-only, static)** has a value that is set when the attribute is created and never changes.

name (read-only, static)
The name attribute corresponds to the instance of the nsrjb command that was created to load the media described by this resource. The instance of the nsrjb command is specified using the **−O** command line option.
Example: name: 15;

nsrmmd id (read-only/ static)
This attribute records the number that identifies the nsrmmd process which requested the media.
Example: nsrmmd id: 2;

nsrmmd host (read-only/static)
This attribute indicates the name of the host on which the nsrmmd process which requested the media is running.
Example: nsrmmd host: host1;

nsrmmd mount id (read-only/static/null ok)
Some operations, e.g. cloning, require two nsrmmd processes and two volumes be mounted at the same time. The value of this attribute will be the same for all mount requests submitted by such nsrmmd processes. When the nsrmmd which requested media is not participating in an operation which requires another nsrmmd process to complete, the value of this attribute is null.
It is possible that only one cooperating nsrmmd process may be requesting media be mounted. For example, this may occur while cloning data. If the nsrmmd writing data reaches end of media while the nsrmmd process reading data is still has data to be cloned, then the nsrmmd process writing data will request a new volume be mounted.
Example: nsrmmd mount id: 5;

operation (read-only/static)
This attribute indicates the operation being performed by the nsrmmd requesting media be mounted. The value for this attribute attribute can be either

reading, **writing**, or **media**. If the value is set to **reading**, the nsrmmd process is reading data from media. It could either be recovering data or the process that is reading data from media as part of a cloning operation. If the value is set to **writing**, the nsrmmd process is writing data to media. It could either be saving data or the process that is writing data to media as part of a cloning operation. If the value is set to **media**, the nsrmmd is performing an operation that involves only the media, e.g. labeling media.
Example: operation: writing;

data operation (read-only, static, null OK)
This attribute has a value when the **operation** attribute is set to either **reading** or **writing**. If the **operation** attribute is set to **media** this attribute has no value. Values for this attribute can be either **save**, **recover**, **archive**, **retrieve**, **migrate**, **clone**, or **consolidate**.
Example: data operation: save;

media operation (read-only, static)
This attribute indicates operations which are expected to be performed on the media after it is loaded into a drive. Values for this attribute can be either **mount**, **label**, **relabel**, or **verify**. This attribute may have multiple values. For example media may be labeled and then mounted.
Example: media operation: label, mount;

pool (read-only, static)
The pool to which the requested media belongs or the pool to which the media is to be added if the **media operation** attribute includes **label** or **relabel** values. Valid values for this attribute are any currently defined pool.
Example: pool: Default;

volume (read-only, static, null OK)
This attribute records the name of the volume being mounted. The value of the attribute is NULL if the volume name is not known at the time the operation is started. This can occur when labeling scratch media.
Example: volume: vol1;

client (read-write, static, null ok)
This attribute is used to record the name of the NetWorker client from which data is being read while saving (archiving) data or to which data is being written while recovering (retrieving) data.
Example: client: host1;

save set (read-only, static, null ok)
This attribute specifies the data being read from client and written to media (save) or the data read from media and written to client (recover).
Example: save set: /usr;

group (read-only, static, null ok)
If the operation being performed by the nsrmmd which requested the mount is the result of a command initiated by **savegrp (8),** this attribute records the name of the group. Otherwise there is no value. Valid values for this attribute are any currently defined group.
Example: group: Default;

EXAMPLE

A complete example follows:

```
          type: NSR mount request;
          name: 10;
      nsrmmd id: 2;
    nsrmmd host: host1;
```

```
nsrmmd mount id:  tape;
       operation:  writing;
  data operation:  save;
 media operation:  label, mount;
          volume:  vol1;
            pool:  Default;
          client:  host1;
        save set:  /usr;
           group:  Default;
```

FILES */nsr/res/nsrdb* − files in this directory should never be edited directly. Use **nsradmin**(8), or **NetWorker Management Console** instead.

SEE ALSO **nsr_resource**(5), **nsr_pool**(5), **nsr_group**(5), **nsr_service**(5), **nsr**(8), **nsrmmd**(8), **nsradmin**(8)

NAME nsr_notification − NetWorker resource type ''NSR notification''

SYNOPSIS **type: NSR notification**

DESCRIPTION A resource of type **NSR notification** is used for each combination of an event, priority, and action handled by the NetWorker notification system. A NetWorker notification consists of a single event type, a single priority, and a message. The notification system posts each message to the action of each **NSR notification** resource (by executing the command listed in the action, with the message on standard input) that includes that event type and priority. See **nsr_resource**(5) for more information on NetWorker resources. To edit the NSR notification resources type:

```
nsradmin −c "type:NSR notification"
```

or use **NetWorker Management Console.** See **nsradmin**(8) for more information on using the NetWorker administration program.

ATTRIBUTES The following attributes are defined for resource type **NSR notification**. The information in parentheses describes how the attribute values are accessed. **Create-only** indicates that the value cannot be changed by an administrator, except when the resource is created. **Read/write** means the value can be changed at any time by authorized administrators. **Choice list** means that any number of values can be chosen from the given list. **Single string** means that only a single value is allowed. **Static** attributes change values rarely, if ever. **Hidden** means it is an attribute of interest only to programs or experts, and these attributes can only be seen when the hidden option is turned on in **nsradmin**(8). For example, an attribute marked **(create-only, static)** has a value which is set when the attribute is created and never changes. Several additional attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the event.

action (read/write, single string)
The value is a command line to be executed when the given event occurs. The command line is run (see **popen**(3s)) with the event information connected to standard input. Typical actions are to log the message with the **syslog**(3) package, or send electronic mail to a system operator.
Example: action: /usr/ucb/mail −s "savegroup completion" root;

event (create-only, choice list, hidden)
Each value is a class of events that will trigger the given notification. More than one class may be selected. Valid values are: **Media** for events related to the media multiplexor subsystem, **Savegroup** for events generated by the **savegrp**(8) command (usually the nightly automatic backups), **Index** for events related to the on-line file index subsystem, **Registration** for events caused by changes in the product's registration status (for example, a license that will soon time out), and **Server** for other NetWorker server events, such as restarting.
Example: event: Media;

name (create-only, static)
This attribute holds the name of the notification resource.

priority (create-only, choice list, hidden)
Each value is a priority at which the notification will be triggered. More than one priority may be selected. The valid values in increasing priority order are **Info** − supplies information about the current state of the server; **Notice** − an

important piece of information; **Warning** − information about a non-fatal error; **Waiting** − the server is waiting for an operator to perform a routine task, such as mounting a tape; **Critical** − the server detected an error condition that should be fixed by a qualified operator; **Alert** − a severe error condition that demands immediate attention; **Emergency** − a condition that may cause NetWorker to fail unless corrected immediately.
Example: priority: Notice;

mail home (read-only, hidden)
This attribute indicates whether a notification resource is associated with the mail home feature.

enabled (read/write)
This attribute indicates whether an email shall be sent to EMC customer support if the event named in the notification resource occurs. This attribute applies only to mail home notifications.

immediate (read/write)
This attribute applies only to mail home notifications. This attribute indicates whether the notification is emailed immediately or deferred until the day specified by the mail home day attribute.

mail home day (read/write, choice list)
This attribute applies only to mail home notifications. This attribute indicates the day of the week for sending deferred mail home alerts.

pending event info (read-only, hidden)
This attribute contains mail home event information for which email has not been sent to EMC. This attribute applies only to mail home notifications.

A complete example follows with two resources, one for mail and one using the syslog mechanism:

```
          type: NSR notification;
          name: savegroup completion;
 administrator: root;
        action: /usr/bin/mail -s \"savegroup completion\" root;
         event: Savegroup;
      priority: Notice;

          type: NSR notification;
          name: log default;
 administrator: root;
        action: /usr/bin/logger -p daemon.notice -f -;
         event: Media, Savegroup, Index, Server, Registration;
      priority: Info, Notice, Warning, Waiting,
                Critical, Alert, Emergency;
```

FILES

/nsr/res/nsrdb − files in this directory should never be edited directly. Use **NetWorker Management Console** instead.

SEE ALSO

nsr_resource(5), **nsr_service**(5), **nsr_device**(5), **nsr**(8), **nsrmm**(8), **syslog.conf**(5), **syslog**(3), **nsradmin**(8), **nsrmmd**(8),

NAME nsr_op − NetWorker resource type ''NSR operation status''

SYNOPSIS **type: NSR operation status**

DESCRIPTION Some of the operations performed by a NetWorker server (e.g jukebox operations) are tracked by means of a single resource of type **NSR operation status** per operation. The resource is used by the calling program (e.g. nsrjb) both for tracking purposes (to know when the operation is complete, to follow error and verbose messages etc.), and for control purposes (cancellation, prompt responses, etc).

See **nsr_resource**(5) for information on NetWorker resources. To view the **NSR operation status** resources within nsradmin, run:

```
nsradmin -c "type:NSR operation status"
```

Be careful to include the spaces between the words that make up the resource type name, as well as the surrounding quotes. See **nsradmin**(1m) for information on using the NetWorker administration program.

The **NSR operation status** resources are *transient* resources - that is, they exist only as long as is deemed necessary in order to track the status of the operation in question. Each resource will be created when the operation is initiated, and will be removed a certain amount of time after it has completed.

Note that because the initiating client program (e.g. nsrjb) polls nsrd to get information on the current state of its operation, there must be a grace period between when the operation completes, and when nsrd removes the Nsr operation status resource, otherwise the client program may not see the completion information. For jukebox operations, this grace period is defined on a per-jukebox basis, by means of the "operation lifespan" attribute in the "Nsr jukebox" resource. The lifespan is in seconds, and defaults to 1800 (equal to 30 minutes). For non-jukebox operations, a fixed grace period of 5 minutes applies.

All NSR operation status resources will also be deleted automatically during Networker's start-up and shut-down phases, since no operation may continue beyond a single run of Networker. If nsrmmgd (which is responsible for controlling all jukebox operations on behalf of a nsrd server) terminates unexpectedly for any reason, then nsrd will also automatically mark all jukebox related operations as aborted.

ATTRIBUTES The following attributes are defined for resource type **NSR operation status**. The information in parentheses describes additional information about how the attribute values are accessed. **Hidden** means that these attributes can only be seen when the hidden option is turned on in **nsradmin**(1m)**. single string** means that these attributes can only have a single value, whereas **multiple strings** means that the attribute may have multiple values. **choice** means that the attribute value may only be selected from a series of well defined choices.

Note that none of the attributes of the **NSR operation status** resource should be changed by the user or administrator - they are for use by other Networker programs only. Manual changing any of the NSR operation status resources or their attributes, may cause unexpected behavior.

operation source (single string)
Indicates the source of the operation - e.g. "nsrjb", "GUI jb op", "nsrd jb op", "jbverify", "dvdetect", etc. This is used by Networker to handle any aspects of the operation that vary according to the origin of the operation, as well as being used by the GUI to allow sorting, filtering etc., of operations based on their origin.

name (single string)
If this is a jukebox operation, then the name attribute specifies the name of the jukebox that this operation is being performed on. For non-jukebox operations, this attribute may be left blank.

operation instance (single number)
This attribute is a number which is used to uniquely identify a given operation. The instance number may be "wrapped around" such that a lower instance number does not necessarily indicate a resource that was created before another such resource that has a higher instance number. (See the "start time" attribute" for determining relative ages of NSR operation status resources).

status (choice)
Defines the current status of the operation. Possible values are:
queued: The default state for a new operation, this indicates that the operation has been sent to the appropriate daemon where it will be performed.
running: Indicates that the controlling daemon is currently working on the operation.
succeeded: The operation has completed, and was successful.
failed: The operation has been terminated without it being completely successful. Note that this status value covers the entire operation, so if you tried to label 10 volumes and 9 of the 10 were successful, the operation would still indicate failure due to the fact that it was not **completely** successful.

retryable: Like "failure", except that Networker believes that there is a reasonable chance that the operation would complete successfully if simply retried.

completion code (single number)
This attribute is not always used (depending on the operation type and origin), but when it is non-NULL it contains a numeric value that gives more information about the completion status of the operation that the simple "status" attribute provides.

command (single string)
This attribute indicates what the command line was that initiated the operation. It is provided mainly to help the user track which operations are in which state, including knowing which were uncompleted when Networker shut down (a list of such uncompleted operations will be printed out during shutdown).

progress (multiple string)
This attribute is not always used (depending on the operation type and origin), but when it is non-NULL it contains information about the current progress of the operation. The multiple values of this attribute may be used so that the

first value indicates e.g. percentage completion of the operation, while the second value gives a description of the current task being performed for that operation.

error message (mutliple string)
A list of error messages associated with the operation. Note that the operation does not have to be in the "failed" status for there to be error messages contained in this attribute. This is because the operation may contain multiple parts which do not all need to be aborted once a single part has experienced an error. For example, if you issue a nsrjb command to label 10 tapes, and there is an error that prevents the first tape from being labeled, the error message will be logged in the "error message" attribute, but the operation will continue in the "running" state while the attempts to label the other 9 tapes proceeds. (Unless the user elects to cancel the operation due to the initial failure).

prompt (single string)
This attribute's value is normally empty. If it is non-empty, then this indicates that the operation is awaiting user-input in order to continue. The prompt will be shown by the client application that started the operation.

prompt response (single string)
The response that the user gave, to the prompt. Once a prompt response is entered, the prompt string will automatically be cleared to show that no prompt need be shown to the user anymore.

cancellation (choice)
Defaults to "none". If set by the administrator to either "full" or "immediate", this will cause the controlling daemon (e.g. **nsrmmgd**) to cancel the operation. Note that some stages of certain operations may take a significant amount of time to cancel. The type of cancellation (full or immediate) determines whether the controlling daemon waits for the operation to be properly cleaned up before removing the operation from its queue and marking it as complete. Consider the example of a nsrjb operation. It is possible that **nsrmmgd** is waiting for a response from **nsrlcpd** or **nsrd** when the cancellation request comes in. In such cases, a "full" cancellation tells **nsrmmgd** to wait for any pending responses from other processes in order to correctly set the appropriate values in the jukebox resource that indicate the true state of the system. By contrast, an "immediate" cancellation tells **nsrmmgd** to not wait for such responses. An "immediate" cancellation may cause the jukebox resource to mismatch the actual jukebox status for a while, so should only be done in those cases where a full cancellation is not working (e.g. **nsrmmgd** is awaiting a response from **nsrlcpd** but nsrlcpd has already been killed and restarted).

If the operation was initiated via nsrjb, then the cancellation attribute will be set to "full" if you cancel the operation by pressing Control-C. If you do not wish to wait for the cancellation to be completed and acknowledged, a second Control-C to nsrjb will cause nsrjb to exit without showing the progress of the cancellation, but the cancellation type will still remain "full".

If the controlling daemon terminates unexpectedly, then **nsrd** will automatically mark all outstanding operations as cancelled, by setting the cancellation type to "immediate".

messages (mutliple string)
A list of informational messages associated with the operation. This attribute is used for verbose logging of the progress of the operation. The higher the level of verbosity associated with the operation (typically set in the client application's command line), the greater the number of entries in the messages attribute is likely to be for a given operation).

start time (single string, hidden)
A machine-readable time-stamp indicating when the operation was initiated and this NSR operation status resource was created.

last update (single string, hidden)
A machine-readable time-stamp indicating when the last update to this NSR operation status resource was made. This value is updated when the resource is changed to e.g. require the issuing of a prompt, a prompt response, a verbose or error message being added, the operation being cancelled, or any other change in the status of the operation being tracked.

source (single string, dynamic, hidden)
Used by Networker to determine the source of the last change to the resource. This is used to cut down on unnecessary network traffic. This attribute is only used for certain operations.

EXAMPLE

The following example shows a resource that defines a label operation on jukebox "adic", in which a nsrjb command has been issued to label the volume in Slot 2. The volume already has a label of 'XYX', (as can be see in the verbose messages attribute), and so a prompt is issued to confirm with the user whether the (destructive) re-label should proceed.

```
                type: NSR operation status;
      operation source: nsrjb;
                name: adic;
    operation instance: 3;
              status: queued  [running]  succeeded  failed  retryable;
     completion code: ;
             command: nsrjb -L -S 2;
            progress: ;
       error message: ;
            messages: "Loaded volume ABC from Slot 2"
              prompt: "Confirm re-label of volume 'ABC' to 'XYZ' ? [Yes/No]";
      prompt response: ;
  operation cancelled: [No]  Yes ;
          start time: 1070557031;
         last update: 1070557031;
              source: ;
```

SEE ALSO

nsr(5), **nsr_resource**(5), **nsr_jukebox**(5), **nsrjb**(1m), **jbedit**(1m)

NAME NSR peer information − resource containing NW instance information about peers

SYNOPSIS **type: NSR peer information**

DESCRIPTION The NSR peer information resource is used by NetWorker authentication daemon **nsrexecd** (see **nsrexecd**(8)). To edit the **NSR**peer**information** resources run:

```
nsradmin -s host_name -p nsrexec -c "type:NSR peer information"
```

or

```
nsradmin -s host_name -p 390113 -v 1 -c "type:NSR peer information"
```

See **nsradmin**(1m) for information on using the NetWorker administration program.

DESCRIPTION Resources of this type are populated/created by NetWorker. They are used to hold the identity and certificate of remote NetWorker installations that the local installation communicated with in the past. These resources are similar to known_hosts file used by **ssh**(1). Once a NetWorker installation (client, server, or storage node) communicates with a remote NetWorker install (client, server, or storage node), a NSR peer information resource will be created on each host and will contain information about the peer (i.e. identity and certificate). During this initial communication, each host will send information about itself to the peer. This information includes the NW instance name, NW instance ID, and the certificate. After this initial communication, each NetWorker install will use the registered peer certificate to validate future communications with that peer.

This resource is only used if the two machines (the local machine and the one described by the **name** attribute) are using GSS EMC v1 authentication.

ATTRIBUTES The following attributes are defined for resource type **NSR peer information** The information in parentheses describes how the attribute values are accessed. **Hidden** means it is an attribute of interest only to programs or experts, and these attributes can only be seen when the hidden option is turned on in **nsradmin**(1m). **Static** attributes change values rarely, if ever. For example, an attribute marked **(read-only, static)** has a value which is set when the attribute is created and may never change. Not all attributes are available on all Operating Systems.

name (read-only, single string)
The name attribute specifies the NW instance name of a remote machine running NetWorker. This value is is a shorthand for the NW instance name of the remote machine. The value in this attribute should be entered where ever a NetWorker instance needs to be referred to. The value should be unique throughout the data zone.

NW instance ID (read-only, hidden, single string)
The NW instance ID. This value will be used to identify the remote NetWorker install whenever a NetWorker program needs to communicate with another NetWorker program. This value has a one to one correspondence with the NetWorker instance name. It should be unique throughout the data zone.

certificate (read-only, hidden, single string)
The certificate for the remote NetWorker installation. The certificate is used by the local NetWorker installations to validate the identity of the remote NetWorker install indicated by the **name** attribute in the current NSR peer information

resource.

Change certificate (read-write, dynamic, choice)
This attribute is used to import or clear the certificate in the resource. Valid values are: **Clear certificate** and **Load certificate from file**.

If **Clear certificate** is selected, then NetWorker will clear the certificate entry in the current NSR peer information resource. This will cause the initial communication between the local install and the peer described by the **name** attribute to reoccur on the next connection between the two hosts. Setting **Change certificate** to **Clear certificate** has the same effect as deleting the resource instance.

Setting **Change certificate** to **Load certificate from file**, causes NetWorker to attempt to load the peer certificate located in the file specified by the **certificate file to load** attribute.

This field will be reset to blank after NetWorker uses the value.

certificate file to load (read-write, dynamic, single string)
This field is used to specify a file name where NetWorker should load the peer certificate from when the **Change certificate** attribute is set to **Load certificate from file**. The file is expected to contain a certificate in PEM format. This field will be reset to blank after NetWorker uses the value.

administrator (read-write, list of strings)
The **administrator** list contains users and user netgroups that are allowed to add, delete, and update the **NSR peer information** resources. The default value for this field is the value of the administrator attribute in the NSRLA field at the time of creation of the first NSR peer information resource. The value of the **administrator** field is the same for all **NSR peer information** resource instances. When the **administrator** is changed for one instance of the **NSR peer information**, it will get changed for all instances. Each line specifies a user or a group of users, using one of these formats: *user/host@domain* , *group/host@domain* , *user@host* , *user@domain* , *group@host* , *group@domain* , *&netgroup* (only available on platforms that support netgroups) , *user_attribute=value[, ...]*.

where *user* is a user name; *host* is a host name; *group* is a user group name; *domain* is a domain name; *user_attribute* can be **user**, **group**, **host**, **nwinstname**, **nwinstancename**, **domain**, or **domaintype** (type of the domain, **NIS** or **WINDOMAIN**).

The user attributes: **nwinstname** and **nwinstancename** are used to indicate a NetWorker instance name. The value that should be entered for either of these attributes is the value in the "name" field in the NSRLA resource for the machine where a matched user is connecting from.

value can be any string delimited by white space. If the value has space in it, then it can be quoted with double quotes. The value may contain wild cards, "*". Entering just a user name allows that user to administer NetWorker from any host (equivalent to *user*@* or **/user* or **user=***user*). Netgroup names are always preceded by an "&".

The format: *user_attribute=value[, ...]* is more secure because the format is not over-**loaded. For example, if** *test@test.acme.com* **is entered, then any users in the** *test* **group or users named** *test* **and that are in the domain;** *test.acme.com* **or from the host;** *test.acme.com* **will match this entry.**

nsradmin(1m), **nsrexecd**(8), **nsr_la**(5)

NAME nsr_policy − NetWorker resource type ''NSR policy''

SYNOPSIS **type: NSR policy**

DESCRIPTION Each NetWorker policy is described by a single resource of type **NSR policy** (see **nsr_resource**(5)). To view the **NSR policy** resources for a NetWorker server, enter nsradmin at the command prompt to start the nsradmin program. At the nsradmin prompt, enter:

```
nsradmin>print type:NSR policy
```

See **nsradmin**(8) for more information on using the NetWorker administration program.

These resources control how long entries remain in a client's on-line file index and when to mark a save set as recyclable. Each **NSR client** resource (see **nsr_client**(5)) uses two policies, a browse policy and a retention policy. Policies can only be deleted if no clients are using them.

Each policy defines an amount of time. The amount of time is determined by the *period* and the *number of periods*.

ATTRIBUTES The following attributes are defined for resource type **NSR policy**. The information in parentheses describes how the attribute values are accessed. **Create-only** indicates that the value cannot be changed by an administrator once the resource is created. **Read/write** means the value can be set as well as read at any time. Several additional hidden attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

name (create-only)
This attribute contains the name of the policy defined by this resource. The name must be unique for this NetWorker server, but otherwise can be anything that makes sense to the administrator. This name will appear as a choice attribute of each NSR client resource. The **NSR policy** resources named "Quarter" and "Year" may be modified, but may not be removed. The name can only be specified when the group is created.
Example: name: life cycle;

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the policy.

number of periods (read/write)
The number of periods attribute specifies the number of base units to use.
Example: number of periods: 3;

period (read/write)
The period attribute determines the base unit for this policy. It may be one of four values: **Days**, **Weeks**, **Months** or **Years.** A week is defined as 7 days, the number of days in a particular month depends upon the calender, and a year as 366 days.
Example: period: Months;

EXAMPLE The following **NSR policy** resource named "Quarter" defines a period of 3 months, or one quarter of a year:

```
type: NSR policy;
name: Quarter;
period: Months;
```

number of periods: 3;

SEE ALSO **nsr**(8), **nsrim**(8), **nsr_resource**(5), **nsr_client**(5), **nsradmin**(8)

NAME nsr_pool − NetWorker resource type ''NSR pool''

SYNOPSIS **type: NSR pool**

DESCRIPTION Each NSR pool is described by a single resource of type **NSR pool** (see **nsr_resource**(5)). To edit the NSR pool resources for a NetWorker server type:

```
nsradmin −c "type:NSR pool"
```

Be careful to include the quotes and the space between ''NSR'' and ''pool''. See the **nsradmin**(8) manual page for more information on using the NetWorker administration program.

These resources are used by NetWorker to determine what volumes save sets should reside on depending upon the characteristics, for example, Group or Level, of the save. Consult your *NetWorker Administration Guide* for more guidelines on using pools.

There are four types of pools. *Backup* pools accept data from **savegrp** and manual backups. *Archive* pools accept archive data. Data cloned from a backup pool can be directed to a *backup clone* pool. Likewise, archive data can be cloned to an *archive clone* pool.

Further, backup pools can be created without any selection criteria supplied. While they may look like the Default pool, these "empty" pools are ignored by the server when determining which pool to use based on selection criteria. Other from that restriction, "empty" pools are fully functional and can be used to label volumes and store data. The caveat is an empty pool must be explicitly designated such as, by using the -b option of the save command. If an empty pool is updated to include selection criteria (by adding a client, group, level or save set) they become "regular" pools are included in automated pool selection.

There are ten pools shipped pre-enabled with NetWorker. The *Default* pool is meant to collect any backup data not directed to a pool a user creates with selection criteria. Any archive data not directed to a pool with selection criteria is collected in the *Indexed Archive* pool. While *Archive* pool is the counterpart of *Indexed Archive* pool that does not store index entries. When cloning data, the user must select a destination pool for the operation. The *Default clone* pool is available for users to clone backup data to. Both *Indexed Archive clone* pool and *Archive clone* pool are available for users to clone archive data to, with *Indexed Archive clone* pool designed for users to clone indexed archives and *Archive clone* pool designed for cloning non-indexed archives. The *PC Archive* pool is designed for the PC archive data, whereas the *PC Archive Clone* is available for users to clone PC archive data to. Similarly, the *Migration* and *Migration Clone* pools are designed for migration data and cloning of migration data respectively.

There are also a few pools shipped with NetWorker that are not enabled by default. The *Full* and *NonFull* pools can be used to segregate full level backups from other backups, for example, fulls versus incrementals. The *Offsite* pool can be used to generate offsite backups, because no index entries are stored for the media pool and will not be referenced during normal recovers. Note that one can also clone media to produce copies of data to be taken offsite. Save sets that are generated without index entries can still be recovered using the ''Save Set Recover'' feature of **nwrecover (8)** or **recover**(8).

ATTRIBUTES The following attributes are defined for resource type **NSR pool**. The information in parentheses describes how the attribute values are accessed. **Create-only** indicates that the value cannot be changed after the resource has been created. **Read/write** means the value can be updated by authorized administrators. **Yes/no** means only a yes or no choice is possible. **Choice** indicates that the value can only be selected from a given list. **Hidden** means it is an attribute of interest only to programs or experts, and these

attributes can only be seen when the hidden option is turned on in **nsradmin**(8).

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the pool.

archive only (read/write, yes/no, hidden, create)
If **yes** is selected, only archive saves are allowed to this pool. *This hidden attribute can be modified by a user.*
Example: archive only: no;

auto media verify (read/write, yes/no, choice)
If set to **yes**, NetWorker verifies data written to volumes from this pool. Data is verified by re-positioning the volume to read a portion of the data previously written to the media and comparing the data read to the original data written. If the data read matches the data written, verification succeeds; otherwise it fails. Media is verified whenever a volume becomes full while saving and it is necessary to continue onto another volume, or when a volume goes idle because all save sets being written to the volume are complete. When a volume fails verification, it is marked full so NetWorker will not select the volume for future saves. The volume remains full until it is recycled or a user marks it not full. If a volume fails verification while attempting to switch volumes, all save sets writing to the volume are terminated.
Example: auto media verify: yes;

clients (read/write, choice)
What *clients* (**nsr_client**(5)) are allowed in this pool. If a *group* is specified, only clients that are members of that group are allowed to be listed.
Example: clients: mars;

devices (read/write, choice)
This attribute lists the **ONLY** devices that volumes from this pool are allowed to be mounted onto. If no devices are listed, volumes from this pool may be mounted on any device.
Example: devices: /dev/nrst8;

enabled (read/write, yes/no, choice)
If set to **yes**, this pool is considered for determining what pools a save set should be saved to when performing backup volume selection. If set to **no**, this pool is completely ignored.
Example: enabled: yes;

groups (read/write, choice)
What *groups* (**nsr_group**(5)) are allowed in this pool.
Example: groups: Accounting;

label template (read/write, choice)
Determine what *label template* (**nsr_label**(5)) is referenced when generating volume names for this pool.
Example: label template: Accounting;

levels (read/write, choice)
What *levels* (**nsr_schedule**(5)) are allowed in this pool.
Example: levels: full;

name (create-only)
The names of pool resources are used when labeling volumes and when determining what volumes a save set should reside on. The name can be chosen at the administrator's convenience, but it must be unique for this NetWorker server. The pool resources named **Default**, **Default Clone**, **Indexed Archive**, **Indexed Archive Clone**, **Archive**, **Archive Clone**, **PC Archive**, and **PC Archive**

Clone cannot be modified or deleted. The pool resource named **Full** and **Non-Full** cannot be deleted. Other pools can only be deleted if no volumes still reference them.
Example: name: Accounting;

pool type (create-only)
This attribute determines how volumes that are members of this pool will be used.
Example: pool type: Backup;

recycle from other pools (read/write, yes/no, choice)
This attribute determines whether or not a given pool can recycle volumes from other pools when it exhausts all its write-able and recyclable volumes.
Example: recycle from other pools: yes;

recycle to other pools (read/write, yes/no, choice)
This attribute determines whether or not a given pool allows other pools to recycle its recyclable volume for their use.
Example: recycle to other pools: yes;

retention policy (read/write, choice)
This attribute specifies the name of the policy controlling how long entries will remain in the media index before they are marked as recyclable. The default value is blank meaning there is no pool policy. In the absense of a pool policy the client policy is used to determine the save set expiration date. A defined pool policy is always considered if it is a not the only instance of the save set (a clone). The pool policy is considered for original save sets but is utilized only if the period defined by the policy is longer the retention policy's period defined by the client resource, see **nsr_policy**(5).
Example: retention policy: Year;

save sets (read/write, choice)
What save sets (**nsr_client**(5)) are allowed in this pool. Save sets can be matched using the regular expression matching algorithm described in **nsr_regexp**(5)).
Example: save sets: /, /usr, C:\\windows\\system, *.JPG ;

store index entries (read/write, yes/no, choice)
If set to **yes**, entries are made into the file indexes for the backups. Otherwise, only media database entries for the save sets are created.
Example: store index entries: yes;

media type required (read/write, choice)
If a required type is specified, it is the only media type that can be mounted, labeled or written to in this poool.
Example: media type required: adv_file;

volume type preference(read/write, choice)
This attribute is used as a selection factor when a request is made for a writable volume. The preferred type will be considered first within a priority level such as *jukebox* or *stand alone device* .
This attribute name may be renamed to "media type preferred" when displayed in the NMC GUI.
Example: media type preferred: 4mm;

max parallelism (read/write, hidden)
This attribute can be used to impose an upper limit for the number of parallel sessions saving to a media belonging to the pool. Fewer parallel save session written to media reduces the time required to recover data from a saveset. Value of zero imposes no limit on number of parallel save sessions written to

media belonging to this pool.

mount class (read/write)
This attribute is kept for historical reasons only. It has no effect.

EXAMPLE A complete NSR pool resource, named 'Default', follows:

```
                type: NSR pool;
        archive only: No;
   auto media verify: Yes;
             clients: ;
             comment: ;
             devices: ;
             enabled: Yes;
              groups: ;
      label template: Default;
              levels: ;
                name: Default;
           pool type: Backup;
           save sets: ;
   store index entries: Yes;
recycle from other pools: Yes;
  recycle to other pools: Yes;
    retention policy: ;
  media type required: adv_file;
volume type preference: ;
```

SEE ALSO **nsr**(5), **nsr_label**(5), **nsr_resource**(5), **nsradmin**(8), **nwrecover**(8), **recover**(8), **savegroup**(8), **savefs**(8), **uasm**(8)

NAME nsr_regexp − regular expression syntax

DESCRIPTION This page describes the regular expression handling used in NetWorker. The regular expressions recognized are described below. This description is essentially the same as that for **ed**(1).

A regular expression specifies a set of strings of characters. A member of this set of strings is said to be matched by the regular expression.

Form	Description
1.	Any character except a special character matches itself. Special characters are the regular expression delimiter plus a backslash(\), brace(**[**), or period(**.**) and sometimes a carat(^), asterik(∗), or dollar symbol(**$**), depending upon the rules below.
2.	A **.** matches any character.
3.	A \ followed by any character except a digit or a parenthesis matches that character.
4.	A nonempty string **s**, bracketed string **[s]** (or **[^s]**) matches any character in (or not in) **s**. In **s**, \ has no special meaning and **]** may only appear as the first letter. A substring **a-b**, with **a**and**b** in ascending ASCII order, stands for the inclusive range of ASCII characters.
5.	A regular expression of form 1 through 4 followed by ∗ matches a sequence of 0 or more matches of the regular expression.
6.	A bracketed regular expression **x** of form 1 through 8, **\(x\)**, matches what **x** matches.
7.	A \ followed by a digit **n** matches a copy of the string that the bracketed regular expression beginning with the **n**th **\(x\)** matched.
8.	A regular expression **x** of form 1 through 8 followed by a regular expression **y** of form 1 through 7 matches a match for **x** followed by a match for **y**, with the **x** match being as long as possible while still permitting a **y** match.
9.	A regular expression of form 1 through 8 preceded by ^ (or followed by **$**), is constrained to matches that begin at the left (or end at the right) end of a line.
10.	A regular expression of form 1 through 9 picks out the longest among the leftmost matches in a line.
11.	An empty regular expression stands for a copy of the last regular expression encountered.

SEE ALSO **ed**(1), **nsr_client**(5)

NAME nsr_render_log − output NetWorker log file data in human-readable form

SYNOPSIS

nsr_render_log
[−**acdeghlmprty**] [−**L** *locale*] [−**S** *start_time* | "l locale_start_time"] [−**E** *end_time* | "l locale_end_time"] [−**N** *number_of_lines*] [−**x** *export_spec*] [−**T** *thread_id*] [−**P** *process_id*] [−**G** *program_name*] [−**B** *start_line*] [−**M** *message_id*] [−**H** *hostname*] [−**J** *hostname_referenced*] [−**A** *activity_id*] [−**C** *category*] [−**F** *devicename*] [−**Y** *severity*] log_file_name

DESCRIPTION

nsr_render_log reads messages from the NetWorker log file **log_file_name** , filters and renders them according to the command line options, and sends the output to stdout. The default language is English. If a locale is specified, the messages are output in the specified language, and the time stamps are formatted to that locale. If the messages cannot be rendered in the specified locale language then the messages are rendered in english. nsr_render_logs assumes that the proper fonts are configured by the user.

OPTIONS

−**a** Do not output the activity ID.

−**c** Do not output the category.

−**d** Do not output the timestamp.

−**e** Do not output the error number.

−**g** Do not output the program name.

−**h** Do not output the host name.

−**l** Output the header information line before the log output.

−**m** Do not output the message ID.

−**p** Do not output the process ID.

−**t** Do not output the thread ID.

−**y** Do not output the severity.

−**x** <*export_spec*>
As an alternative to the default human-readable output format, *export_spec* provides for a user defined output format separator. The *export-spec 'c<separator>'* displays values separated by <*separator*>. For example, **'nsr_render_log -x 'c** \t' <**file_name**>**,** will produce tab separated values.

−**L** *locale*
Output is translated, and time is formatted, to the specified locale.

−**S** **[start_time** | "l locale_start_time"]

Messages dated earlier than start_time will not be output. The date specified can be in nsr_getdate(3) format, or locale date format when "l locale_start_time" is specified. Note that the surrounding quotes and a blank space after "l" are required for locale date format. The input locale date/time follows the format display of **'ls -l'** command of **recover** on UNIX and **'dir'** command of **recover** on windows. For example, in an English locale and on a UNIX machine **'nsr_render_log -S** " **l may 30 4:00** " <**log_file_name**>**'** will produce all the messages dated later than May 30th 4:00. With the 'l' specification if no start time or an invalid start time is specified then the time is mapped to the start of the day (00:00:00).

See **changetime** command in recover man page for more information on the locale_date format supported.

−**E** **[end_time** | "l locale_end_time"]

Messages dated later than end_time will not be output. The date specified can be in nsr_getdate(3) format, or locale date format when "l locale_end_time" is specified. Note that the surrounding quotes and a blank space after "l" are required for locale date format. The input locale date/time follows the format display of **'ls -l'** command of **recover** on UNIX and **'dir'** command of **recover** on windows. For example, in an English locale and on a UNIX machine **'nsr_render_log -E " l may 30 4:00** " <**log_file_name**>**'** will produce all the messages dated earlier than May 30th 4:00. With the 'l' specification if no end time or an invalid end time is specified then the time is mapped to the end of the day (23:59:59).

See **changetime** command in recover man page for more information on the locale_date format supported.

−**N** *lines*
The maximum number of lines that are required to be printed. If there are not sufficient number of lines then the available number of lines shall be printed. If this option is specified along with the Start time then N lines are rendered after the Start time stamp.

−**B** *start_line*
Output only messages starting with this line number. If the given **'start_line' value is negative then it** lines from the end of file.

−**T** *thread_id*
Output only messages written by the specified thread ID(s). Multiple TIDs up to a limitof 8 may be entered. Each shall be separated by a space and the whole set enclosed within double quotes.

−**P** *process_id*
Output only messages written by the specified process ID(s). Multiple PIDs up to a limit of 8 may be entered. Each shall be separated by a space and the whole set enclosed within double quotes.

−**G** *program_name*
Output only messages written by the specified program name(s). Multiple Program Names up to a limit of 8 may be entered. Each shall be separated by a space and the whole set enclosed within double quotes.

−**M** *message_id*
Output only messages having the specified message ID(s). Multiple Message IDs up to a limit of 8 may be entered. Each shall be separated by a space and the whole set enclosed within double quotes.

−**H** *hostname*
Output only messages written by the specified host(s). Multiple HostNames up to a limit of 8 may be entered. Each shall be separated by a space and the whole set enclosed within double quotes.

−**A** *activity ID*
Output only messages having the specified activity ID. Multiple activity IDs up to a limit of 8 may be entered. Each shall be separated by a space and the whole set enclosed within double quotes.

−**C** *category*
Output only messages having the specified message category. Multiple message categories up to a limit of 8 may be entered. Each shall be separated by a space and the whole set enclosed within double quotes.

−**F** *device name*
Output only messages having a reference to the specified device name. Multiple devie names up to a limit of 8 may be entered. Each shall be separated by a space and the whole set enclosed within double quotes.

−**J** *host name referenced*
Output only messages havinga reference to the specified host name. Multiple host names up to a limit of 8 may be entered. Each shall be separated by a space and the whole set enclosed within double quotes.

−**Y** *severity*
Output only messages having a severity level equal to or more than the specified severity level.

log_file_name
Read messages from the file named by the path name **log_file_name**

EXAMPLES

Example 1: To translate log to Japanese, with a header line.

nsr_render_log -l -L ja_JP.utf8 /nsr/logs/daemon.raw

Example 2: To render log messages generated by pid's 41, 1064 and 1065, suppressing thread and activity/eventID output, searching only the last 50 lines of the log file and redirect output

nsr_render_log -ta -P "41 1064 1065" -B -50 ./recover.raw > recover.txt

Example 3: To render first 99 log messages between 11:00 AM and 12:00 AM on Jan 30, 2006 with a severity level 2.

nsr_render_log -S "Jan 30 11:00" -E "Jan 30 12:00" -N 99 -Y 2 /nsr/logs/daemon.raw

NOTES

1: For each backslash (" \ ") character in the path, enter two backslashes in succession.

Example: c:\\dir_one\\dir_two\\daemon.raw

2: If the user is rendering the messages to a locale other than the locale of the shell, then the output will be UTF-8 encoded. If this UTF-8 output is piped to a file, then it is the responsibility of the user to make sure that the file supports UTF-8 encoding.

SEE ALSO

nsr_getdate(3)

NAME nsr_resource − NetWorker resource format

SYNOPSIS

resource ::= *attribute list <blank line>*
attribute list ::= *attribute* [**;** *attribute*]*
attribute ::= *name* [**:** *value* [**,** *value*]*]
name, value ::= *<printable string>*

DESCRIPTION

The NetWorker system uses files containing *resources* to describe itself and its clients. Each resource represents a component of the NetWorker system that might need administration. Devices, schedules, and clients are examples of NetWorker resources. The system administrator manipulates resources to control the NetWorker system. The file and the resources in them are accessible through **NetWorker Management Console** and **nsradmin**(8) programs. They can also be viewed with a normal text editor.

The files all share a common format. The same format is used by the **nsradmin**(8) program. Each resource is described by a list of attributes, and ends in a blank line. Each attribute in the attribute list has a name and an optional list of values. The attribute name is separated from the attribute values by a colon (**:**), attribute values are separated by commas (**,**), and each attribute ends in a semicolon (**;**). A comma, semicolon or back-slash (\) at the end of a line continues the line. A line beginning with a pound-sign (#) is a comment and the rest of the line is ignored. The back-slash character can also be used to escape the special meaning of other characters (comma, semicolon, pound-sign, and back-slash).

The attribute name and values can contain any printable character. Upper and lower case is not distinguished on comparisons, and extra white space is removed from both ends but not from inside of names and values. For example,

```
Name: this is a test;
```

matches

```
name   :   This Is A Test ;
```

but is different than

```
Name: this      is      a      test;
```

EXAMPLES

In the following example resource, there are eight attributes. They are **type**, **name**, **server**, **schedule**, **directive**, **group**, **save set**, and **remote access**. The **remote access** attribute has no value.

```
            type: NSR client;
            name: venus;
          server: earth;
        schedule: Default;
       directive: Unix standard directives;
           group: Default;
        save set: All;
   remote access: ;
```

In the following resource, there are six attributes. The **administrator** attribute has three values: **&engineering**, **root, and operator**. Note that the three values are separated by commas. The **action** attribute has one value: **incr incr incr incr incr full incr**. Note that this is a single value − it just happens to have spaces separating its words.

```
            type: NSR schedule;
          action: incr incr incr incr incr full incr;
   administrator: &engineering, root, operator;
            name: engineering servers;
        override: ;
          period: Week;
```

SPECIAL ATTRIBUTES

Each NetWorker resource includes seven special attributes: **type**, **name**, **administrator**, **hostname**, **ONC program number**, **ONC version number**, and **ONC transport**. The **type** and **name** attributes are normally visible, but the others attributes are hidden. That an attribute is hidden indicates that it is infrequently used and perhaps esoteric. Hidden attributes should usually not be changed by the user.

The **type** attribute defines which other attributes a resource can contain. For example, a **resource with type NSR client** will always include the attribute **server**, while a resource of type **NSR schedule** does not.

The **name** attribute is a descriptive name of the object that a resource represents. In the first example above, the **name** attribute is the name of the NetWorker client machine. In the second example, the **name** attribute describes a schedule used to back up the the servers in the engineering department.

The **administrator** attribute is the list of users that have permission to modify or delete this resource. This attribute is inherited from the **type: NSR** resource when a new resource is created. The administrator of the **NSR** resource also controls who has permission to create and delete NetWorker resources.

The **hostname** attribute specifies the hostname of the machine on which the service that controls this resource is running. It is used internally and cannot be changed by the administrator.

The remaining attributes (**ONC program number**, **ONC version number**, and **ONC transport**) specify the Open Network Computing information for this service. They should never be changed manually.

In some cases, the resource identifier will be visible. Although it may look like an attribute, it is an internal value that is set and used by the NetWorker system to provide unique identification of each resource. When new resources are created in the **edit** command of **nsradmin**(8), the resource identifier attribute should be left off. This signals that this is a new resource and a new identifier will be assigned.

NetWorker resources are implemented by the EMC Resource Administration Platform, which is described in the **resource**(5) manual page. This flexible architecture means that in future releases of NetWorker, more resource types or attributes may be added, and the administration tools in this release will automatically be able to use them. To make this possible, each server provides *type descriptors* that are used internally to describe the attributes of each type, between the administration tools and the services. These type descriptors may cause limitation on the values, such as only allowing a single value, allowing no value, or only numeric values.

RESOURCE TYPES

This release of NetWorker defines the following types of resources:

NSR This resource describes a NetWorker server. It contains attributes that control administrator authorization, information about operations in progress, and statistics and error information about past operations. For more information see the **nsr_service**(5) manual page.

NSR client
This resource describes a NetWorker client. It includes attributes that specify the files to save, which schedule to use, and which group this client belongs to. There may be more than one client resource for a NetWorker client. This allows a client to save files on different schedules. For more information see the **nsr_client**(5) manual page.

NSR device
This resource type describes a storage device. It includes attributes that specify a particular device name (for example, /dev/nrst1), media type (for example, 8mm), and the name of the currently mounted volume. It also

provides status and statistics on current and past operations. For more information see the **nsr_device**(5) manual page.

NSR directive
This resource describes a directive. Directives control how a client's files are processed as they are being saved. For more information see the **nsr_directive**(5), **nsr**(5) and **uasm**(8) manual pages.

NSR group
This resource specifies a logical grouping of NetWorker clients and a starting time. Each day, at the specified time, all members of the group will start their saves. For more information see the **nsr_group**(5) manual page.

NSR jukebox
This resource type describes a jukebox. It includes attributes such as the jukebox model, the first and last slot numbers in the jukebox, and the names of the devices within the jukebox. For more information see the **nsr_jukebox**(5) manual page.

NSR label
This resource type specifies a template describing a sequence of names to be used when labeling volumes. For more information see the **nsr_label**(5) manual page.

NSR license
This resource contains licensing information for each feature currently enabled in this NetWorker installation. It contains various enabler and authorization codes that are used by NetWorker to validate licensed capabilities. For more information see the **nsr_license**(5) and **nsrcap**(8) manual pages.

NSR notification
A notification specifies an action to be performed when a particular type of NetWorker event takes place. For more information see the **nsr_notification**(5) manual page.

NSR policy
Policy resources are used as part of the index management process in NetWorker. These policies control how long entries remain in a client's on-line file index and when to mark a save set as recyclable. For more information see the **nsr_policy**(5) manual page.

NSR pool
This resource type is used by NetWorker to determine what volumes save sets should reside on based on the characteristics of the save (for example, group or level). For more information see the **nsr_pool**(5) manual page.

NSR schedule
Schedule resources define a sequence of save levels and an override list. The override list is made up of pairs of levels and dates. The level controls the amount of data saved when a client is backed up. For more information see the **nsr_schedule**(5) manual page.

NSR stage
Each stage resource describes a staging policy. The resource includes attributes that define control parameters for the policy, and devices managed by the policy. For more information see the **nsr_stage**(5) manual page.

FILES

/nsr/res/nsrdb Holds the NetWorker server's resources. Files in this directory should never be edited directly. Use **nsradmin**(8) or **NetWorker Management Console** instead.

SEE ALSO **resource**(5), **nsr**(5), **nsr_client**(5), **nsr_device**(5), **nsr_directive**(5), **nsr_group**(5), **nsr_jukebox**(5), **nsr_label**(5), **nsr_license**(5), **nsrcap**(8), **nsr_notification**(5), **nsr_policy**(5), **nsr_pool**(5), **nsr_schedule**(5), **nsr_service**(5), **nsr_stage**(5), **nsr**(8), **savegroup**(8), **savefs**(8), **nsradmin**(8), **uasm**(8)

NAME nsr_schedule − NetWorker resource type "NSR schedule"

SYNOPSIS **type: NSR schedule**

DESCRIPTION Each NetWorker schedule is described by a single resource of type **NSR schedule** (see **nsr_resource**(5)). To edit the **NSR schedule** resources for a NetWorker server, type:

```
nsradmin −c "type:NSR schedule"
```

See **nsradmin**(8) for more information on using the NetWorker administration program.

This resource describes a sequence of levels controlling the amount of data saved by NetWorker clients (see **nsr_client**(5)). There is one **NSR schedule** resource for each NetWorker schedule.

ATTRIBUTES The following attributes are defined for resource type **NSR schedule**. The information in parentheses describes how the attribute values are accessed. **Read-only** indicates that the value cannot be changed by an administrator. **Read/write** means the value can be set as well as read. Several additional hidden attributes (e.g., administrator) are common to all resources, and are described in **nsr_resource**(5).

name (read/write)
This attribute specifies the schedule's name. The schedule is referred to by its name in client resources.
Example: name: monthly_fulls;

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the schedule.

period (read-only)
This attribute specifies the length of the schedule's period. It may be either "Week" or "Month". "Week" schedules repeat every 7 days and start on Sunday. "Month" schedules start over at the first of each month. The default is "Week."
Example: period: Month;

action (read/write)
This attribute specifies the sequence of save levels making up the schedule. One entry is used for each day of the schedule. The entries must be separated by whitespace, i.e., blanks or tabs. The valid levels are "consolidate," "full," "incr," "skip," and the numbers 1 through 9. The actions consolidate, full, incr, and skip may be abbreviated "c," "f," "i," and "s," respectively.

When the action attribute does not contain enough entries to account for every day in the period, NetWorker will repeat the list of actions when the end of the action list is reached.
Example: action: f i i i i i i;

There are 12 levels: **full**, levels **1** through **9**, **incr**, and **skip**. **Full** specifies that all files are to be saved. It is analogous to a level 0 dump in **dump**(8). **Incr** specifies incremental saves in which only those files that have been modified since the most recent save, at any level, are saved. This level has no exact analogue in **dump**(8) since the last save at any level, including previous incremental saves, are considered when determining what to save. **Skip** causes no files to be saved. The levels **1** through **9** cause all files to be saved which have been modified since any *lower* level save was performed. As an example, if you did a full save on Monday, followed by a level 3 save on Tuesday, a

subsequent level 3 save on Wednesday would contain all files modified or added since the Monday full save. By default, the save level is determined automatically from the NetWorker client's schedule (NSR schedule). By using the history of previous saves maintained by **nsrmmdbd**(8) on the NetWorker server, the needed time for the given level can correctly be computed. By using media information on the server, times computed for saves that are based on previous save levels will automatically be adjusted as required when tapes are deleted.

override (read/write)

This attribute specifies a list of actions and dates overriding the actions specified in the *action* attribute. The format of the override specification is *action date*. *action* must be one of "full," "incr," "skip," or one of the numbers 1 through 9. *date* must be either a *fixed date* or *recurring date. Fixed date* is of the form "month/day/year." Month and day are 2 digit numbers, year may be either 2 or 4 digits. If the year is 2 digits, numbers in the range 70-99 are assumed to be offsets from 1900, those in the range 00-69 are assumed to be offset from 2000. *Recurring date* is of the form '[*number*] *weekday* every [*number*] *period'. number* can be a number (1, 2, 3, etc.) or an ordinal (first, second, third, etc.), and it is optional. *weekday* must be one of "monday," "tuesday," "wednesday," "thursday," "friday," "saturday," "sunday." The use of "last" for *number* is not supported (for example, "last friday" cannot be used to refer to the last Friday of the month). *period* must be one of "week," "month," "quarter," or "year." Action/date pairs are separated by commas (','). *Example:* override: full 1/1/1994, full first friday every 2 week;

EXAMPLE

The following defines a **NSR schedule** resource named "Default." The Default schedule may be modified, but it may not be deleted. Each NetWorker server must have a Default schedule. This schedule has a period of one week, does a full save on Sunday, followed by 6 incremental saves. There are no override actions specified.

```
    type: NSR schedule;
    name: Default;
  period: Week;
  action: f i i i i i i;
override:;
```

The following defines a schedule named "quarterly." It has a period of one month. The action attribute specifies level 5, 9, and incremental saves. In the override attribute, full saves are specified for the first day of each quarter. Note that there are only 7 entries in the action attribute. Upon reaching the end of the list, NetWorker will start over at the beginning of the list, performing a level 5 save.

```
    type: NSR schedule;
    name: quarterly;
  period: Month;
  action: 5 incr incr incr 9 incr incr;
override: f 1/1/1994, f 3/1/1994, f 6/1/1994, f 9/1/1994, f 1/1/1995;
```

SEE ALSO

nsr(8), **nsrmmdbd**(8), **savefs**(8), **mminfo**(8), **nsradmin**(8), **nsr_client**(5), **nsr_policy**(5), **nsr_resource**(5)

NAME nsr_service − NetWorker server resource type ''NSR''

SYNOPSIS **type: NSR**

DESCRIPTION Each NetWorker server is described by a resource of type **NSR**. See **nsr_resource**(5) for general information on NetWorker resources. To edit the **NSR** resource use the command:

```
nsradmin −c "type:NSR"
```

See **nsradmin**(8) for information on using the NetWorker administration program.

ATTRIBUTES The following attributes are defined for the **NSR** resource. The information in parentheses describes how the attribute values are accessed. **Read-only** indicates that the value cannot be changed by an administrator. **Read/write** means the value can be set and read. **Choice list** means that any number of values can be chosen from the given list. **Static** attributes change values rarely. **Dynamic** attributes have values which can change rapidly. **Hidden** means it is an attribute of interest only to programs or experts, and these attributes can only be seen when the hidden option is turned on in **nsradmin**(8). For example, an attribute marked **(read-only, static)** has a value which is set when the resource is created and never changes, or is changed only by the server.

name (read-only, static)
This attribute specifies the hostname of this NetWorker server.
Example name: mars;

version (read-only, dynamic)
This is the software version of the NetWorker server daemon, **nsrd**(8). This includes a slash and the number of clients currently licensed.
Example: version: NetWorker 4.1 Turbo/110;

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about this NetWorker server.
Example: comment: machine located in room 243;

save totals (read-only, dynamic, hidden)
Save statistics. A string containing the total number of save sessions, the number of saves with errors (if any) and the total number of bytes saved (if any). This attribute is updated after each save session completes.
Example: save totals: "37 sessions, 457 MB total";

recover totals (read-only, dynamic, hidden)
Recovery statistics. A string containing the total number of recover sessions, the number of recovers with errors (if any) and the total number of bytes recovered (if any). This attribute is updated after each recover session completes.
Example: recover totals: "347 sessions, 48 MB total";

totals since (read-only, dynamic)
The time statistics collection started. This is usually the last time the NetWorker server was rebooted.
Example: totals since: "Fri Jun 1 09:35:02 1992";

NSR operation (read-only, choice list, hidden)
This attribute is currently unused and is provided for backward compatibility.

parallelism (read/write, static)
This attribute sets the number of concurrent save sessions that this server will allow. The value can be set by an administrator. Use higher values for better

performance on a fast system with a lot of main memory and swap space. Use lower values to avoid overloading a slow system, or systems with little main memory and/or swap space. **Warning**: due to defects in some versions of UNIX, high values of parallelism may cause the system to lock up.
Example: parallelism: 4;

session statistics (read-only, dynamic, hidden)
This attribute reports the statistics of each active session. There are 14 values for each set of statistics, namely, **id** (session's unique identifier), **name** (session's name), **mode** (read, write, browse), **pool** (current pool), **volume** (current volume), **rate kb** (current data transfer rate for save session), **amount kb** (current amount read/written by session), **total kb** (total amount to be read by session), **amount files** (current number of files recovered; to be implemented in a future release), **total files** (current number of files to recover; to be implemented in a future release), **connect time** (time session has been connected), **num volumes** (number of volumes to be used by recover session), **used volumes** (number of volumes processed by recover session), and **completion** (running, complete, or continued)
Example: sessions statistics: ;

monitor rap (read/write, hidden)
This attribute allows the administrator to enable logging changes made to the configuration resources into the /nsr/logs/rap.log.

manual saves (read/write, hidden)
This attribute allows the administrator to disable manual backups to the server. Scheduled backups continue to work normally.

public archives (read/write)
This attribute determines whether a user can retrieve archived files that are owned by another user.

volume priority (read/write)
If a NetWorker server has volumes in locally managed jukeboxes and volumes being managed by SmartMedia, this attribute allows the administrator to assign a priority for volume selection when saving data. This attribute determines whether the server has a preference for volumes being managed by SmartMedia, **SmartMedia Priority,** or whether the server has a preference for volumes in a locally managed jukebox, **NearLine Priority.** The default value is **NearLine Priority.**
Example: volume priority: NearLine Priority;

SmartMedia save mount (read/write)
This attribute controls the form of the request made to SmartMedia to mount a volume for saving data. Setting the value of this attributes to **volume by characteristics** causes NetWorker to request a volume meeting specified criteria, and lets SmartMedia select an appropriate volume from all media which satisfy the criteria specified. When this attribute's value is set to **volume by name ,** NetWorker will request the volume by name and SmartMedia mounts the volume requested. The default value is **volume by characteristics .**
Example: SmartMedia save mount: volume by characteristics;

license server (read/write, hidden)
The name of the server on which a NetWorker license manager is installed and running. This attribute used to be called "GEMS server". You can set the value to a GEMStation where a GEMS license manager is running or to a machine where a NetWorker license manager is running. *Example:* license server: jupiter;

update licenses (choice, read-only, dynamic, hidden)
Set to 'Yes' if this server should attempt to re-synchronize its externally managed licenses immediately.

message (read-only, dynamic, hidden)
The last message of any kind logged. A timestamp is included at the start of the string.
Example: message: "Mon 12:25:51 Tape full, mount volume mars.001 on /dev/nrst1";

message list (read-only, dynamic, hidden)
A list of recent messages, with a timestamp and a string message for each value.
Example: message: "Mon 12:25:51 Tape full, mount volume mars.001 on /dev/nrst1";

server message (read-only, dynamic, hidden)
Lists recent, concise general messages about the status of the server.
Example: message: "Tape full, mount volume mars.001 on /dev/nrst1";

sequence number (read-only, dynamic, hidden)
The sequence number of the corresponding server message.

server message time (read-only, dynamic, hidden)
The time at which the server message was generated.

server message priority (read-only, dynamic, hidden)
The priority of the server message. Currently these are for NetWorker internal use.

server message source (read-only, dynamic, hidden)
This attribute names the NetWorker component, such as a client or device, that generated the message on the server. This can be a client, a device etc.

server message category (read-only, dynamic, hidden)
This attribute is the category the server message belongs to. Currently these are for NetWorker internal use.

session (read-only, dynamic, hidden)
The value of this attribute is a list of session information strings. Each string includes the NetWorker client name, type of operation (saving, browsing, or recovering) and information about the save set, including name, number of bytes, and number of files. All sizes and rates are in bytes per second, kilobytes (1024), Megabytes (a thousand Kilobytes), etc.
Example:
session: "venus:/usr saving to mars.001 20MB",
"mars:/usr/src done saving 24MB";"

session message (read-only, dynamic, hidden)
Concise session message string of the session attribute described above.
Example:
session: "venus:/usr saving to mars.001",
"mars:/usr/src done saving";"

session client name (read-only, dynamic, hidden)
The name of each client for which each session is active.

session rate (read-only, dynamic, hidden)
The rate of data transfer for the active session.

session rate label (read-only, dynamic, hidden)
Unit of data transfer for the active sessions.

session device name (read-only, dynamic, hidden)
The name of the device on which the session is active.

pending (read-only, dynamic, hidden)
A list of events pending with the NetWorker event notification system (see **nsr_notification**(5)). The first three fields are the time, priority, and event name.
Example: pending: "Fri 14:40:15 alert: media mount of mars.001 suggested on /dev/nrst1";

status (read-only, dynamic, hidden)
A list of status flags for the NetWorker server. These flags are only for use by NetWorker server-side programs (for example, **savegrp**) and list various features enabled in the running server. The format is currently *name=boolean* (true or false). The listed features and their states can change at any time.

statistics (read-only, dynamic, hidden)
A list of strings of the form *name=number* that give a number of server statistics.

LSSV reg code (read-write, hidden)
Customer registration code, for "EMC Single Server Version for Oracle".

Job inactivity timeout (read/write, static)
Global setting for the number of minutes since a job has been heard from last, after which it will be declared inactive and will be terminated.
This setting is enforced by nsrjobd and replaces environment variable **NSR_UNRESPONSIVE_JOB_TIMEOUT.**
Unlike the group inactivity timeout which applies only to **save** processes maintaining connection to **nsrmmd,** this timeout applies to all processes throughout runtime. For example, if a **save** process were to hang in argument processing, group inactivity setting would never trigger its termination, however if this attribute is set, it will result in terminating such a hang process after number of minutes set in this attribute has passed.
An empty string or a value of 0 indicates that no such timeout is in effect.

Jobsdb retention in days (read/write, static)
Minimum time to keep completed job records in the active jobs database, in days. After this time, records will be purged out of the active jobs database. NMC server should retrieve job records within this time for long-term monitoring purposes. Make sure that the NMC server does not stay off-line for a period of time longer than the interval specified by this attribute.
Warning: large values of this attribute may cause the active jobs database to grow very large on busy servers, potentially causing performance problems. Adjust with caution.

Jobsdb maximum size (read/write, static)
Maximum size that the active jobs database is allowed to grow to. If the database size exceeds this value, the database is purged until the size drops at least 10% below the maximum set by this attribute. Records are selected from oldest to youngest. This value supercedes the retention time. Setting this value too low will cause records to be purged prematurely.
Warning: large values of this attribute may cause the active jobs database to grow very large on busy servers, potentially causing performance problems. Adjust with caution.

types created (read-only, static)
A list of all the other resource types this NetWorker server can create and about which clients can query.
Example: types created: NSR device, NSR group;

administrator (read/write, static)
This is a list of names (or netgroups) of users who are allowed to administer

NetWorker. Users in this list are members of the Administrators user group. This list is inherited from the Users attribute in the Administrators user group resource. Only users with the *Change Security Settings* privilege can see or modify this attribute. The user "root" on the local host of the server is always an administrator. Entries specifying other administrators are of the form:

user, *user@host*, *user@domain*, *group@host*, *group@domain*, *host/user* **&***netgroup*, or *user_attribute=value[, ...]*

where *user* is a user name; *host* is a host name; *group* is a user group name; *domain* is a domain name; *user_attribute* can be **user**, **group**, **host**, **nwinstname**, **nwinstancename**, **domain**, or **domaintype** (type of the domain, **NIS** or **WINDOMAIN**).

The user attributes: **nwinstname** and **nwinstancename** are used to indicate a NetWorker instance name. The value that should be entered for either of these attributes is the value in the "name" field in the NSRLA resource for the machine where a matched user is connecting from.

value can be any string delimited by white space. If the value has space in it, then it can be quoted with double quotes. The value may contain wild cards, "*". Entering just a user name allows that user to administer NetWorker from any host (equivalent to *user*@* or **/*user* or **user=***user*). Netgroup names are always preceded by an "&".

The following example grants NetWorker Administrators membership to, "root" from any host, the user "operator" from the hosts "jupiter" and "mars", the user "admin" from any host, and all <user name, user's hostname, server's domain> in the netgroup "netadmins".
Example: administrator: root, operator@jupiter, mars/operator, admin@*, &netadmins;

The following example grants NetWorker Administrator membership to the user "root" on host "pluto", users in group "Backup Operators" from windows domain "Accounting", and user "joe" in NIS domain "YP.fubar.COM".
Example: administrator: "user=root,host=pluto", "group=\"Backup Operators\", domain=Accounting, domaintype=windomain", "user=joe, domain=YP.fubar.COM, domaintype=NIS";

contact name (read/write, static)
This attribute is used for product licensing/registration purposes. It must be specified before printing the registration information from the registration window.
Example: contact name: contact_name;

company (read/write, static)
This attribute is used for product licensing/registration purposes. Your company name must be specified before printing the registration information from the registration window.
Example: company: EMC Corporation;

street address (read/write, static)
This attribute is used for product licensing/registration mailing purposes. Specify your mailing street address.
Example: street address: 176 South Street;

city/town (read/write, static)
This attribute is used for product licensing/registration mailing purposes.
Example: city/town: Hopkinton;

state/province (read/write, static)
This attribute is used for product licensing/registration mailing purposes.
Example: state/province: MA;

zip/postal code (read/write, static)
This attribute is used for product licensing/registration mailing purposes.
Example: zip/postal code: 01748;

country (read/write, static)
This attribute is used for product licensing/registration mailing purposes.
Example: country: USA;

phone (read/write, static)
This attribute is used for product licensing/registration purposes. This attribute must to be specified before printing the registration information from the registration window.
Example: phone: 877-534-2867;

fax (read/write, static)
This attribute is used for product licensing/registration purposes.
Example: fax: 650-745-1477;

email address (read/write, static)
This attribute is used for product licensing/registration purposes.
Example: email address: support@emc.com;

server OS type (read/write, static)
This attribute is used for product licensing/registration purposes.
Example: server OS type: Solaris;

purchase date (read/write, static)
This attribute is used for product licensing/registration purposes. It specifies the purchase date of the product enabler code. This attribute must be specified before printing the registration information from the registration window.

product serial number (read/write, static)
This attribute is used for product licensing/registration purposes. It must be specified before printing the registration information from the registration window.

mm op message (read/write, dynamic, hidden)
This attribute lists the descriptive message for the most recently completed media database operation. The NetWorker program (such as **nsrmm**(8)) that requested the operation clears this attribute as soon as it has read the result. An administrator should never change this attribute manually.

mm operation value (read/write, dynamic, hidden)
This attribute is used by programs such as **nsrmm**(8) to pass the desired media database operation location or flags to the NetWorker server. The value is automatically cleared when the operation completes. An administrator should never change this attribute manually.

mm operation (read/write, choice list, dynamic, hidden)
This attribute is used by programs such as **nsrmm**(8) to pass the appropriate media database operation type to the NetWorker server. The possible choices are: purge volume, purge save set, delete volume, delete save set, mark volume, mark save set, unmark volume, unmark save set, specify volume location, specify volume flags, and specify save set flags. The server serializes such operations and performs the appropriate queries on **nsrmmdbd**(8). The value is automatically cleared when the operation completes. An administrator

should never change this attribute manually.

mm operation id (read/write, dynamic, hidden)
This attribute is used by programs such as **nsrmm**(8) to pass the desired media database operation identifier to the NetWorker server. The value is automatically cleared when the operation completes. An administrator should never change this attribute manually.

nsrmon info (read/write, dynamic, hidden)
This attribute is used by programs such as **nsrmon**(8) to pass information about remote daemon requests to the NetWorker server. The value is automatically cleared when the request completes. An administrator should never change this attribute manually. See **nsr_storage_node**(5) for a description of storage nodes and remote daemons.

nsrmmd count (read-only, dynamic, hidden)
This attribute is used by programs such as **nsrd**(8) to track the number and location of the media daemons, **nsrmmd**(8).

nsrmmd polling interval (read/write, hidden)
This attribute specifies the number of minutes between polling events of a remote **nsrmmd**(8). **nsrd**(8) polls a remote **nsrmmd**(8) at this interval, to determine whether it is running. If it determines from this poll that the daemon is no longer running, it will restart the **nsrmmd**(8), with a delay set by the 'nsrmmd restart interval'; see the **nsrmmd** restart interval description. See **nsr_storage_node**(5) for additional details on this attribute and storage nodes.

nsrmmd restart interval (read/write, hidden)
This attribute specifies the number of minutes between restart attempts of a remote **nsrmmd**(8). When **nsrd**(8) determines that a remote **nsrmmd**(8) has terminated, it periodically attempts to restart the remote daemon. A value of zero for this attribute means the daemon should be restarted immediately. See **nsr_storage_node**(5) for additional details on this attribute and storage nodes.

nsrmmd control timeout (read/write, hidden)
This attribute specifies the number of minutes **nsrd**(8) waits for storage node requests.

enabler code (read/write, dynamic, hidden)
This attribute specifies the enabler code for the base enabler of the server software.

vendor ID (read/write, hidden)
Identifier of the vendor that built and shipped this server.

SS cutoff size (read/write, hidden)
This attribute sets the default "save set cut off size" to be used when saving. A blank value uses the built in default value. A non blank value for this attribute consists of a number followed by KB, MB, or GB signifying kilobytes, megabytes, or gigabytes. Note that this field only affects clients older than Release 6.0. Continuation save sets have been eliminated for Release 6.0 and above.

User ID (dynamic, hidden)
This attribute contains the user ID by which the NetWorker server authenticated you. The user ID is usually in the format *user/host@domain*.

Privileges (dynamic, hidden)
This attribute lists the privileges that you have.

Member of (dynamic, hidden)
This attribute lists the user groups that you are a member of.

hostname (read-only, hidden)
This attribute gives the hostname of the machine on which the service that controls this resource is running. It is used internally and cannot be changed by the administrator.

ONC program number (read-only, hidden)
The Open Network Computing (sunrpc) identification number for the client to server protocol provided by this service.

ONC version number (read-only, hidden)
The version of the above protocol.

ONC transport (choice, read-write, hidden)
The Open Network Computing (sunrpc) transport protocols supported are TCP (Transport Control Protocol) or UDP (User Datagram Protocol).

timezone offset (read-only, hidden)
This attribute displays the server's timezone offset from GMT.

EXAMPLE

A complete example follows:

```
                type:  NSR;
                name:  mars;
             version:  "NetWorker 4.1 Turbo/110";
          save totals:  "84 sessions, 3597 MB total";
       recover totals:  "1 session";
        totals since:  "Fri Oct 14 12:41:31 1994";
      NSR operation:  Idle;
         parallelism:  4;
        manual saves:  Enabled;
         Monitor RAP:  Disabled;
             message:  \
"Mon 14:37:25 media alert event: recover waiting for 8mm tape mars.001";
        message list:  \
"Mon 07:10:12 media info: loading volume man.001 into /dev/nrst11",
"Mon 07:10:33 /dev/nrst11 mount operation in progress",
"Mon 07:11:15 /dev/nrst11 mounted 8mm 5GB tape man.001";
             session:  "mars:george browsing",
"mars:/home/mars starting recovery of 9K bytes";
  session statistics:  ;
             pending:  \
"Mon 14:40:15 media alert:  recover waiting for 8mm tape mars.001";
              status:  disabled=false, jukebox=true, dm=true,
                       archive=true, cds=true, turbo=true,
                       single=false;
          statistics:  elapsed = 257415, saves = 1176, recovers = 12,
                       save KB = 12050007, recover KB = 28272839,
                       bad saves = 0, bad recovers = 0,
                       current saves = 1, current recovers = 0,
                       max saves = 12, max recovers = 1, mounts = 0,
                       recover delays = 0, saving daemons = 0,
                       recovering daemons = 0, idle daemons = 0;
       types created:  NSR device, NSR group, NSR directive,
                       NSR notification, NSR client, NSR policy,
                       NSR schedule, NSR pool, NSR label, NSR jukebox,
                       NSR license, NSR archive client,
                       NSR archive list;
```

```
administrator:          root;
contact name:           Technical Support;
company:                "EMC Corporation";
street address:         176 South Street;
city/town:              Hopkinton;
state/province:         MA;
zip/postal code:        01748;
country:                USA;
phone:                  877-534-2867;
fax:                    650-745-1477;
email address:          support@emc.com;
purchase date:          ;
product serial number:  ;
mm op message:          ;
mm operation value:     ;
mm operation:           ;
mm operation id:        ;
nsrmon info:            ;
nsrmmd count:           "mars:2";
nsrmmd polling interval: 3;
nsrmmd restart interval: 2;
nsrmmd control timeout: 5;
enabler code:           ;
SS cutoff size:         ;
timezone offset:        GMT-0800;
```

FILES */nsr/res/nsrdb* − files in this directory should never be edited directly. Use **nsradmin**(8) instead.

SEE ALSO **netgroup**(5), **nsr**(5), **nsr**(8), **nsr_device**(5), **nsr_group**(5), **nsr_notification**(5), **nsr_resource**(5), **nsr_storage_node**(5), **nsradmin**(8), **nsrd**(8), **nsrmm**(8), **nsrmmdbd**(8), **nsrmon**(8), **recover**(8), **save**(8)

NAME nsr_shutdown − stop NetWorker services

SYNOPSIS **nsr_shutdown** [−**fq** | −**n** | −**l**][−**t timeout**][**service ...**]

DESCRIPTION Use **nsr_shutdown** to stop NetWorker services intelligently. By default **nsr_shutdown** will stop all NetWorker services running on a host. Optionally, individual services may be specified on the command line.

OPTIONS

−**f** Force **nsr_shutdown** to terminate NetWorker services if a graceful shutdown fails.

−**l** List all currently running NetWorker services and their child processes.

−**n** Run through the service termination sequence, but don't actually shutdown NetWorker services.

−**q** Do not print diagnostic messages during shutdown.

−**t timeout**
Specify a total time constraint in seconds for **nsr_shutdown** to shutdown all NetWorker services specified. If **nsr_shutdown** fails to gracefully shutdown all listed NetWorker services, and the **-f** flag was not specified, it will exit with an error message listing each service which failed to terminate. The default **timeout** value is 180 seconds.

SEE ALSO **ps**(1), **kill**(1), **nsr**(1m), **nsrd**(1m), **nsrexecd**(1m)

NOTES Only processes running as process group leaders may be specified on the **nsr_shutdown** command line.

NAME nsr_stage − NetWorker resource type ''NSR stage''

SYNOPSIS **type: NSR stage**

DESCRIPTION Each staging policy used by a NetWorker server is described by a single resource of type **NSR stage**. See **nsr_resource**(5) for information on NetWorker resources. To edit the **NSR stage** resources run:

```
nsradmin -c "type:NSR stage"
```

Be careful to include the space between ''NSR'' and ''stage'' and the surrounding quotes. See **nsradmin**(1m) for information on using the NetWorker administration program.

ATTRIBUTES The following attributes are defined for resource type **NSR stage**. The information in parentheses describes how the attribute values are accessed. **Read-only** indicates that the value cannot be changed by an administrator. **Read/write** means the value can be set as well as read. **Hidden** means it is an attribute of interest only to programs or experts, and these attributes can only be seen when the hidden option is turned on in **nsradmin**(1m). **Static** attributes change values rarely, if ever. **Dynamic** attributes have values which change rapidly. For example, an attribute marked **(read-only, static)** has a value which is set when the attribute is created and may never change. Additional attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

name (read-only, single string)
The name attribute specifies the staging policy name.

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the staging policy.

enabled (read/write, choice)
The enabled attribute determines whether or not save sets are automatically staged from devices associated with this policy. It also enables and disables the periodic recover space operations. It may be one of two values: *Yes, No*

high water mark (%) (read/write)
The point at which save sets should be *staged,* measured as the percentage of available space used on the file system. Staging will continue until the lower mark is reached.

Example: high water mark (%): 90;

low water mark (%) (read/write)
The point at which the staging process should *stop,* measured as the percentage of available space used on the file system.

Example: low water mark (%): 80;

save set selection (read/write, choice)
Save set selection criteria for staging. It may be one of four values:

largest save set,
smallest save set,
oldest save set,
or

youngest save set.

destination pool (read/write)
The pool to which save sets should be sent (see **nsr_pool**(5)). Note, the source pool and destination pool must match in terms of pool 'type'. That is, saveset clone instances can only be staged from backup pools to backup pools (either regular or clone) or from achive pools to archive pools (again, either regular or clone). Due to the dynamic nature of volume/clone selection, this is only detectable at the time of the staging operation so checking is not done during resource configuration.

devices (read/write, multiple choice)
This attribute lists the *disk family* devices (see **nsr_device**(5)) associated with this policy to be used as the source of staging.

max storage period (read/write)
Specifies the maximum duration for a save set in a given volume before it is *staged* to a different volume.

max storage period unit (read/write, choice, hidden)
Specifies the unit for **max storage period.** It may be one of two values: *days, hours.*

recover space interval (read/write, hidden)
The interval between recover space operations for recyclable, aborted save sets and save sets with no entries in the media database from disk family devices.

recover space unit (read/write, choice, hidden)
Specifies the unit for **recover space interval.** It may be one of two values: *hours, minutes.*

file system check interval (read/write, hidden)
The interval between file system check operations. Stage operation is invoked by NetWorker server every file system check interval to determine whether the high water mark or max storage period has been reached to stage the data from the devices associated with the policy.

file system check unit (read/write, choice, hidden)
Specifies the unit for **file system check interval.** It may be one of two values: *hours, minutes.*

start now (read/write, choice)
Updating this attribute will cause the selected operation to be triggered immediately on all devices associated with this policy. The attribute value will not actually change. Operation can be one of the following:

Check file system - check file system and stage data if necessary.
Recover space - recover space for save sets with no entries in the media database.
Stage all save sets - stage all save sets to the destination pool.

EXAMPLES

Note: the hidden options are not shown in the first example.

The following example shows a resource that defines a stage policy called 'test stage1'. Save sets will be staged from device '/disk/fd0' to pool 'Default Clone' when the file system is 90% full or 7 days after the date of the backup, whichever comes first. The largest save set will be the first to stage to the destination pool:

```
type: NSR stage;
name: test stage1;
comment: ;
```

```
                enabled: No   [Yes];
      high water mark (%): 90;
       low water mark (%): 85;
       save set selection: largest save set;
         destination pool: Default Clone;
                  devices: /disk/fd0;
       max storage period: 7;
              start now: ;
```

The following example shows a resource that defines a stage policy called ‘test stage2’. Save sets will be staged from device ‘/disk/fd2’ to pool ‘Default’ when the file system is 95% full or 14 days after the date of the backup, whichever comes first. The smallest save set will be the first to stage to the destination pool. The file system will be checked every 3 hours and a staging operation will be triggered if necessary. A recover-space operation will be triggered every 8 hours on all devices associated with the policy:

```
                   type: NSR stage;
                   name: test stage2;
                comment: ;
                enabled: No   [Yes];
      high water mark (%): 95;
       low water mark (%): 80;
       save set selection: smallest save set;
         destination pool: Default;
                  devices: /disk/fd2;
       max storage period: 14;
  max storage period unit: Hours   [Days];
   recover space interval: 8;
       recover space unit: Minutes   [Hours];
file system check interval: 3;
   file system check unit: Minutes   [Hours];
              start now: ;
          administrator: root@omni;
               hostname: omni;
```

SEE ALSO **nsr**(5), **nsr_device**(5), **nsrstage**(1m), **nsrclone**(1m), **nsradmin**(1m)

NAME nsr_storage_node − description of the storage node feature

SYNOPSIS The *storage node* feature provides central server control of distributed devices for saving and recovering client data.

DESCRIPTION A *storage node* is a host that has directly attached devices that are used and controlled by a NetWorker server. These devices are called *remote devices,* because they are remote from the server. Clients may save and recover to these *remote devices* by altering their "storage nodes" attribute (see **nsr_client**(5)). A *storage node* may also be a client of the server, and may save to its own devices.

The main advantages provided by this feature are central control of remote devices, reduction of network traffic, use of faster local saves and recovers on a storage node, and support of heterogeneous server and storage node architectures.

There are several attributes which affect this function. Within the NSR resource (see **nsr_service**(5)) there are the "nsrmmd polling interval", "nsrmmd restart interval" and "nsrmmd control timeout" attributes. These attributes control how often the remote media daemons (see **nsrmmd**(8)) are polled, how long between restart attempts, and how long to wait for remote requests to complete.

Within the "NSR device" resource (see **nsr_device**(5)) the resource's name will accept the "rd=hostname:dev_path" format when defining a *remote device.* The "hostname" is the hostname of the storage node and "dev_path" is the device path of the device attached to that host. There are also hidden attributes called "save mount timeout" and "save lockout," which allow a pending save mount request to timeout, and a storage node to be locked out for upcoming save requests.

Within the "NSR client" resource (see **nsr_client**(5)), there are "storage nodes", "clone storage nodes", and "recover storage nodes" attributes:

The "storage nodes" attribute is used by the server in selecting a storage node when the client is saving data.

During a cloning operation (which is essentially a recover whose output data is directed straight into another save operation), the "clone storage node" attribute of the (first) client whose data is being cloned is consulted to determine where to direct the data for the save side of the operation.

The "recover storage nodes" attribute is used by the server in selecting a storage node to be used when the client performs a recover (or the recover side of a clone operation). Note that if the volume in question is already mounted, it will be used from its current location rather than being unmounted and remounted on a system that is in the "recover storage node" list. If the volume in question is in a jukebox, and the jukebox has a value set for its "read hostname" attribute then that designated system will be used instead of consulting the "recover storage node" list, unless the environment variable FORCE_REC_AFFINITY is set to "yes".

The "NSR jukebox" resource (see **nsr_jukebox**(5)), contains the "read hostname" attribute. When all of a jukebox's devices are not attached to the same host, this attribute specifies the hostname that is used in selecting a storage node for recover and read-side clone requests. For recover requests, if the required volume is not mounted, and the client's "storage nodes" attribute does not match one of the owning hosts in the jukebox, then this attribute is used. For clone requests, if the required volume is not mounted, then this attribute is used.

INSTALL AND CONFIGURE

In order to install a storage node, choose the client and storage node packages, where given the choice. For those platforms that do not have a choice, the storage node binaries are included in the client package. In addition, install any appropriate device driver packages. If not running in evaluation mode, a storage node enabler must be configured on the server for each node.

As with a client, ensure that the **nsrexecd**(8) daemon is started on the storage node. To define a device on a storage node, from the controlling server define a device with the above mentioned "rd=" syntax. For a remote jukebox (on a storage node), run **jbconfig**(8) from the node, after adding root@storage_node to the server's administrator list, (where root is the user running **jbconfig**(8) and storage_node is the hostname of the storage node). This administrator list entry may be removed after **jbconfig**(8) completes.

In addition to **jbconfig**(8), when running **scanner**(8) on a storage node, root@storage_node must be on the adminstrator list.

When a device is defined (or enabled) on a storage node, the server will attempt to start a media daemon (see **nsrmmd**(8)) on the node. In order for the server to know whether the node is alive, it polls the node every "nsrmmd polling interval" minutes. When the server detects a problem with the node's daemon or the node itself, it attempts to restart the daemon every "nsrmmd restart interval" minutes, until either the daemon is restarted or the device is disabled (by setting "enabled" to "no" in the device's "enabled" attribute).

In addition to needing a storage node enabler for each storage node, each jukebox will need its own jukebox enabler.

OPERATION

A storage node is assignable for work when it is considered functional by the server - **nsrexecd**(8) running, device enabled, **nsrmmd**(8) running, and the node is responding to the server's polls. When a client save starts, the client's "storage nodes" attribute is used to select a storage node. This attribute is a list of storage node hostnames, which are considered in order, for assignment to the request.

The exception to this node assignment approach is when the server's index or bootstrap is being saved - these save sets are always directed to the server's local devices, regardless of the server's "storage nodes" attribute. Hence, the server will always need a local device to backup such data, at a minimum. These save sets can later be cloned to a storage node, as can any save set.

If a storage node is created first (by defining a device on the host), and a client resource for that host is then added, that hostname is added to its "storage nodes" attribute. This addition means the client will back up to its own devices. However, if a client resource already exists, and a device is later defined on that host, then the client's hostname must be added manually to the client's "storage nodes" attribute. This attribute is an ordered list of hostnames; add the client's own name as the first entry.

The volume's location field is used to determine the host location of an unmounted volume. The server looks for a device or jukebox name in this field, as would be added when a volume resides in a jukebox. Volumes in a jukebox are considered to be located on the host to which the jukebox is connected. The location field can be used to bind a stand-alone volume to a particular node by manually setting this field to any device on that node (using the "rd=" syntax). For jukeboxes which do not have all of their devices attached to the same host, see the previous description of the "read hostname" attribute.

There are several commands that interact directly with a device, and so must run on a storage node. These include **jbconfig**(8), **nsrjb**(8) and **scanner**(8), in addition to those in the device driver package. Invoke these commands directly on the storage node rather than on the server, and use the server option ("-s server_host", where server_host is the controlling server's hostname).

CLONING FUNCTION

A single clone request may be divided into multiple sub-requests, one for each different source machine (the host from which save sets will be read). For example, suppose a clone request must read data from volumeA and volumeB, which are located on storage nodes A and B, respectively. Such a request would be divided into two sub-requests, one to read volumeA from storage node A and another to read volumeB from storage node B.

A clone request involves two sides, the source that reads data and the target that writes data. These two sides may be on the same host or on different hosts, depending on the configuration. The source host is determined first and then the target host. If the volume is mounted, the source host is determined by its current mount location. If the volume is not mounted at the time of the clone request and it resides in a jukebox, then the source host is determined by the value of the jukebox's "read hostname" attribute.

Once the source host is known, the target host is determined by examining the "clone storage nodes" attribute of the client resource of the source host. If this attribute has no value, the "clone storage nodes" attribute of the server's client resource is consulted. If this attribute has no value, the "storage nodes" attribute of the server's client resource is used.

LIMITATIONS

A server cannot be a storage node of another server.

SEE ALSO

jbconfig(8), **mmlocate**(8), **nsr_client**(5), **nsr_device**(5), **nsr_jukebox**(5), **nsr_service**(5), **nsrclone**(8), **nsrexecd**(8), **nsrjb**(8), **nsrmmd**(8), **nsrmon**(8), **scanner**(8)

NAME nsr_storage_node_resource − NetWorker resource type ''NSR storage node''

SYNOPSIS **type: NSR storage node**

DESCRIPTION Each NSR storage node defined to NetWorker is described by a single source of type **NSR storage node** (see **nsr_resource**(5)). To edit the NSR storage node resources for a NetWorker server type:

```
nsradmin −c "type:NSR storage node"
```

Be sure to include quotation marks and to insert a space among "NSR", "storage", "node". See the **nsradmin**(8) manual page for more information on using the NetWorker administration program. The Storage Node resource may also be edited using NetWorker Management Console.

This resource keeps track of the configured devices, configured libraries, unconfigured devices, and unconfigured libraries that can be seen on the storage node. Every time device auto-detection **(ddmgr)** program is started, all the entries in unconfigure devices and unconfigured libraries, and all of their corresponding entries in the related attributes will be removed. A set of newly detected devices and libraries entries will be updated in these attributes.

ATTRIBUTES The following attributes are defined for resource type **NSR storage node**. The information in parentheses describes how the attribute values are accessed. **Read-only** indicates that the value can not be changed by an administrator. **Read/write** indicates the value can be updated by authorized administrators. **Choice list** means that any number of values can be chosen from the given list. **Yes/no** means only a yes or no choice is possible. **Single string** means that only a single value is allowed. **Hidden** indicates it is an attribute of interest only to programs or experts. These attributes can only be seen when the hidden option is turned on in **nsradmin**(8). Several additional attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

name (create-only, single string)
This attribute specifies the name of this storage node.
Example: name: polarbear;

type of storage node (read/write, choice)
This attribute specifies the type of this storage node currently is configured. There are three types of storage node which are SCSI, NDMP, or SILO. Default is set to *'scsi'*.
Example: type of storage node: [scsi] ndmp silo ;

storage node is configured (read/write, yes/no)
This attribute indicates whether this storage node is currently configured. The value will be set to 'yes' by the program if there is at least one device or library has been configured on this storage node. If the last device or library is removed, it will be set to 'no'.
Example: storage node is configured: [Yes] No ;

date of registration (read-only, single string)
This attribute indicates the date and time when the storage node is registered/created, the date and time when the first device is created on this storage node, or the date and time when the last device configured on it is removed.
Example: date of registration: "Fri Nov 18 18:20:28 2005";

number of devices (read-only, single number)
This attribute indicates the count of the devices defined on this storage node. This number will be updated every time a device is added or deleted.
Example: number of devices: 4;

number of libraries (read-only, single number)
This attribute indicates the count of the libraries defined on this storage node. This number will be updated every time a library is added or deleted.
Example: number of libraries: 1;

list of configured devices (read-only, list of string)
This attribute indicates the devices currently defined for this storage node. This list will be updated every time a device is added or deleted. Each entry in this attribute must have a corresponding **NSR device** resource.
Example: list of configured devices:
"rd=polarbear:\\.\Tape3",
"rd=polarbear:\\.\Tape4",
"rd=polarbear:\\.\Tape5",
"rd=polarbear:\\.\Tape6";

list of configured libraries (read-only, list of string)
This attribute indicates the libraries currently defined for this storage node. This list will be updated every time a library is added or deleted. Each entry in this attribute must have a corresponding **NSR jukebox** resource.
Example: list of configured libraries:
"rd=polarbear:ADIC@7.7.0";

list of configured silos (read-only, list of string)
This attribute indicates the Silos currently defined for this storage node. This list will be updated every time a Silo is added or deleted.
Example: list of configured silos:
"rd=s6hp11fc:mysilo";

type of the configured libraries (read-only, list of string)
This attribute indicates the type of tape libraries defined on this storage node which is corresponding to the list of configured libraries defined. This list will be updated every time a library is added or deleted.
Example: type of the configured libraries:
Standard SCSI Jukebox;

type of configured silos (read-only, list of string)
This attribute indicates the type of Silos defined on this storage node which is corresponding to the list of configured silos defined. This list will be updated every time a silo is added or deleted.
Example: type of configured silos: ACSLS_SILO;

max active devices (read/write, single number)
This attribute indicates the maximum number of devices that NetWorker may use from this storage node.

unconfig device names (read/write, list of string, hidden)
This attribute indicates the device names that were discovered by auto-detect program during the last device scanning on this storage node. The order of each entry in this attribute will have its corresponding entries in the following attributes:
unconfig device descriptions
unconfig device model types
unconfig device serial numbers
unconfig device library names

Once a device is configured, its name and its corresponding entries in the above 4 attributes will be removed. Also at each device scanning, all the entries in the unconfig device names and the above 4 attributes will be removed so that an up-to-date detected devices will be updated in these attributes.
Example: unconfig device names:
"\\.\Tape0", "\\.\Tape1";

unconfig device descriptions (read/write, list of string, hidden)
This attribute indicates the description of the devices that were discovered by auto-detect program during the last device scannning. The order corresponds to unconfig device names.
Example: unconfig device descriptions:
<EXABYTE Mammoth2 v05e at SCSI Port 7 Target 2 LUN 0>,
<EXABYTE Mammoth2 v05e at SCSI Port 7 Target 3 LUN 0>;

unconfig device model types (read/write, list of string, hidden)
This attribute indicates the model of the devices that were discovered by auto-detect program during the last device scannning. The order corresponds to unconfig device names.
Example: unconfig device model types:
8mm Mammoth-2, 8mm Mammoth-2;

unconfig device serial numbers (read/write, list of string, hidden)
This attribute indicates the serial numbers of the devices that were discovered by auto-detect program during the last device scannning. The order corresponds to unconfig device names.
Example: unconfig device serial numbers:
"Serial Numbers:WWNN=100000D080001721:ATNN=EXABYTE Mammoth2 0062041830:WWPN=100000D080001722:0062041830",
"Serial Numbers:WWNN=100000D0800012AC:ATNN=EXABYTE Mammoth2 0062034656:WWPN=100000D0800012AD:0062034656";

unconfig device library names (read/write, list of string, hidden)
This attribute indicates the library names to which the devices that were discovered by auto-detect program during the last device scannning belong. The order corresponds to unconfig device names. If the device resides in a library, there will be an entry of the unconfigured library name corresponding to unconfig library name attribute, and which may have a corresponding entry in **NSR unconfigured library** resouce. Default value is *'none'*.
Example: unconfig device library names:
DELL PV-132T WWNN=205000604517079D, none;

unconfig library names (read/write, list of string, hidden)
This attribute indicates the unique unconfigured library names that were discovered by auto-detect program during the last device scannning on this storage node. Each entry may or may not have a corresponding entry in **NSR unconfigured library** resource. If the library returns serial number of itself, there will be an entry in the **NSR unconfigured library** resource. If the library does not return serial number, there will not be an entry in the **NSR unconfigured library** resource. And device auto-configuration program will not configure this library automatically.

The order of each entry in this attribute will have its corresponding entries in the following attributes:
unconfig library reference names
unconfig library descriptions
unconfig library control ports
unconfig library models
unconfig library serial numbers

Once a library is configured, its name and its corresponding entries in the above 5 attributes will be removed. Also at each device scanning, all the entries in the unconfig library names and the above 5 attributes will be removed so that an up-to-date detected libraries will be updated in these attributes.
Example: unconfig library names:
EXABYTE Exabyte X80 WWNN=100000D080001E9B;

unconfig library reference names (read/write, list of string, hidden)
This attribute indicates the unique unconfigured library names that were discovered by auto-detect program during the last device scannning. The order corresponds to unconfig library names.
Example: unconfig library reference names:
"rd=polarbear:EXABYTE@7.6.0";

unconfig library descriptions (read/write, list of string, hidden)
This attribute indicates the description of the unconfigured libraries that were discovered by auto-detect program during the last device scannning. The order corresponds to unconfig library names.
Example: unconfig library descriptions:
<EXABYTE Exabyte X80 3.03 at SCSI Port 7 Target 6 LUN 0>;

unconfig library control ports (read/write, list of string, hidden)
This attribute indicates the control port of the unconfigured libraries that were discovered by auto-detect program during the last device scannning. The order corresponds to unconfig library names.
Example: unconfig library control ports:
scsidev@7.6.0;

unconfig library models (read/write, list of string, hidden)
This attribute indicates the model of the unconfigured libraries that were discovered by auto-detect program during the last device scannning. The order corresponds to unconfig library names.
Example: unconfig library models:
Exabyte Jukebox;

unconfig library serial numbers (read/write, list of string, hidden)
This attribute indicates the serial number of the unconfigured libraries that were discovered by auto-detect program during the last device scannning. The order corresponds to unconfig library names.
Example: unconfig library serial numbers:
WWNN=100000D080001E9B;

date of last scan (read/write, list of string)
This attribute indicates the date that auto-detect process was last run.
Example: date of last scan:

"Mon Nov 28 18:41:06 2005";

skip scsi targets (read/write, list of string)
This attribute indicates the scsi addresses that need to be skipped by the auto-detect process. The targets are in the format of 'bus.target.lun', where the target and/or lun fields can be wildcards. When specify multiple scsi addresses, enter one address per line. The maximum number of scsi address can be excluded is 63. This attribute can be used for a lot of different reasons such as skipped detecting the broken hardware, skipped detecting devices owned by another application other than NetWorker, speeded up the device auto-detection process.
Example: skip scsi targets: 7.9.6;

AFTD Allowed Directories (read/write, list of string)
This attribute indicates what base directories are allowed to create AFTD for a given NSR Storage Node. If it's not empty, then any AFTD directory must be subdirectory of one of the listed base directories (including the base directory itself), and the base directory must be valid and exists, otherwise AFTD config will be rejected. Therefore system administrator should create those base directories first in order to create AFTD underneath. If the list is empty, then a given AFTD directory must be valid before it can be used. It's strongly encouraged to specifiy the list of AFTD Allowed Directories for each NSR Storage Node to safeguard underlying file system interity, and control AFTD backup directories. NetWorker will promptly notify you on any conflict that can exclude existing AFTD devices when list of AFTD Allowed Directories is changed.
Example: AFTD Allowed Directories: /backup/;

search all luns (read/write, yes/no)
This attribute indicates whether search all luns is enable. If set to 'yes' NetWorker will search all the luns for every scsi target. It may cause device auto-detection taking a very long time to complete. By setting it to 'no', NetWorker will stop searching for devices at the first un-used/empty lun. Default is set to *'no'*.
Example: search all luns: Yes [No];

use persistent names (read/write, yes/no)
This attribute indicates whether NetWorker should use any available persistent device names when it searches for tape drives and medium changers. If set to yes, and the storage node's platform supports persistent names that are usable by NetWorker, any detected and/or configured libraries or tape drives will be configured using those persistent names. Any devices that *do not* have persistent names but **do** have normal device names will be found and/or configured using the available normal names. Default is set to *'no'*.
Example: search all luns: Yes [No];

At this time, the only names that NetWorker can automatically find and use are on linux and are of these forms:
/dev/tape/by-id/<<tapeID>>-nst
/dev/tape/by-id/<<changerID>>-generic
/dev/tape/by-id/scsi-<<changerID>>
/dev/generic/by-id/<<changerID>>-generic
/dev/generic/by-id/scsi-<<changerID>>
Setting this attribute to 'yes' on a platform that does not have NetWorker-usable persistent names will have no effect.

visible silo controllers (read/write, list of string, hidden)

This attribute contains all the Silo controllers being tested or configured as visible from this storage node.
Example: visible silo controllers: acsls2;

silo controller types (read/write, list of string, hidden)
This attribute contains all the Silo controller types being tested or configured as visible from this storage node. Each silo controller type in the list is corresponding to the visible silo controllers defined. The values can be: *ACSLS_SILO, DAS_SILO,* or *3494_SILO*
Example: silo controller types: ACSLS_SILO;

silo connection status (read/write, list of string, hidden)
This attribute indicates the status of the Silo connections being tested Each silo connection status is corresponding to the visible silo controllers defined. The value of each of them will be updated at each phase as the silo connection test is requested, running, and finished. The values can be: *Do Test*, *Testing*, *OK,* or *Failed*
Example: silo connection status: OK;

silo names (read/write, list of string, hidden)
This attribute contains the user assigned Silo library names during silo auto configuration. Each Silo library name in the list is corresponding to the visible silo controllers defined.
Example: silo names: mysilo;

silo das client names (read/write, list of string, hidden)
This attribute contains the user input Das client names during silo auto configuration. This value will only appear to the corresponding Das Silo in the visible silo controllers defined.

last error number (read/write, single string)
This attribute contains the error number, if any, logged by the detection **(dvdetect)** process during the last time device auto-detection was run on this storage node.

remote user (read/write, single string)
This attribute contains the user name used to connect to the NDMP server. This value inputs by the user either via Storage node properties, or duing device auto-detection or auto-configuration.

password (read/write, single string)
This attribute contains the password for the user name used to connect to the NDMP server.

last error message (read/write, single string)
This attribute contains the error message, if any, logged by the detection **(dvdetect)** process during the last time device auto-detection was run on this storage node.

device sharing mode (read/write, choice list)
This attribute contains a list of choices for the device sharing at the storage node level. Device sharing controls what NetWorker's autoconfiguration code will do when it encounters a tape library where the tape drives in the library are visible to more than one Storage Node.
If the value is set to *'server default'* then whatever value is set in the server's **Device Sharing Mode** attribute will be used by this storage node as described below.
If the value is *'no sharing'* then autoconfig will do its best to configure the jukebox as an un-shared jukebox with any drives visible from the storage node that can see and control the jukebox itself being configured for NetWorker's

use. No other storage nodes will have access to the drives in that library. If the value is *'maximal sharing'* then autoconfig will configure all storage nodes that can see any drive in the library to use that drive resulting in the maximum possible **Dynamic Drive Sharing** configuration for that library.

Device sharing mode set at this level will override device sharing mode setting at the server level. Default value is set to *'Server default'*.
Example: device sharing mode:
no sharing maximal sharing
[server default];;

EXAMPLE A complete NSR Storage node resource follows:

```
                    type: NSR Storage Node;
                    name: polarbear;
      type of storage node: [scsi]  ndmp  silo ;
 storage node is configured: [Yes]  No ;
       date of registration: "Fri Nov 18 11:10:22 2005";
         number of devices: 4;
       number of libraries: 1;
 list of configured devices: "rd=polarbear:\\.\Tape3",
                             "rd=polarbear:\\.\Tape4",
                             "rd=polarbear:\\.\Tape5",
                             "rd=polarbear:\\.\Tape6";
list of configured libraries: "rd=polarbear:ADIC@7.7.0";
   list of configured silos: ;
types of the configured libraries: Standard SCSI Jukebox;
  types of configured silos: ;
        max active devices: ;
     unconfig device names: "\\.\Tape0", "\\.\Tape1";
unconfig device descriptions:
<EXABYTE Mammoth2        v05e at SCSI Port 7 Target 2 LUN 0>,
<EXABYTE Mammoth2        v05e at SCSI Port 7 Target 3 LUN 0>;
 unconfig device model types: 8mm Mammoth-2, 8mm Mammoth-2;
unconfig device serial numbers:
"Serial Numbers:WWNN=100000D080001721:ATNN=EXABYTE Mammoth2
0062041830:WWPN=100000D080001722:0062041830",
"Serial Numbers:WWNN=100000D0800012AC:ATNN=EXABYTE Mammoth2
0062034656:WWPN=100000D0800012AD:0062034656";
unconfig device library names: none, EXABYTE Exabyte X80 WWNN=100000D080001E9B;
     unconfig library names: EXABYTE Exabyte X80 WWNN=100000D080001E9B;
unconfig library reference names: "rd=polarbear:EXABYTE@7.6.0";
unconfig library descriptions:
<EXABYTE Exabyte X80     3.03 at SCSI Port 7 Target 6 LUN 0>;
unconfig library control ports: scsidev@7.6.0;
    unconfig library models: Exabyte Jukebox;
unconfig library serial numbers: WWNN=100000D080001E9B;
          date of last scan: "Wed Nov 30 20:28:08 2005";
          skip scsi targets: 7.9.6;
            search all luns:  Yes  [No];
   visible silo controllers: ;
     silo controller types: ;
    silo connection status: ;
```

```
                silo names: ;
        silo das client names: ;
            last error number: 0;
                remote user: ;
                  password: ;
           last error message: ;
        device sharing mode:  no sharing    maximal sharing   (server default);
```

SEE ALSO **nsr**(5), **nsr_device**(5), **nsr_jukebox**(5), **nsr_resource**(5), **nsr_unconfigured_library**(5), **nsradmin**(8)

NAME nsr_task − NetWorker resource type "NSR task"

SYNOPSIS **type: NSR task**

DESCRIPTION A resource of type **NSR task** is used to create automatically reaccuring actions. The actions themselves are defined using other resources. This resource is used for secheduling. See **nsr_resource**(5) for more information on NetWorker resources. To edit the NSR task resource type:

```
nsradmin −c "type:NSR task"
```

or use **NetWorker Management Console.** See **nsradmin**(8) for more information on using the NetWorker administration program.

ATTRIBUTES The following attributes are defined for resource type **NSR task**. The information in parentheses describes how the attribute values are accessed. **Create-only** indicates that the value cannot be changed by an administrator, except when the resource is created. **Read/write** means the value can be changed at any time by authorized administrators. **Choice list** means that any number of values can be chosen from the given list. **Single string** means that only a single value is allowed. **Static** attributes change values rarely, if ever. **Hidden** means it is an attribute of interest only to programs or experts, and these attributes can only be seen when the hidden option is turned on in **nsradmin**(8). For example, an attribute marked **(create-only, static)** has a value set when the attribute is created and never changes. Several additional attributes (for example, administrator) are common to all resources, and are described in **nsr_resource**(5).

name (create-only, static)
This attribute holds the name of the resource and uniquely identifies it.

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the resource.

action (read/write)
This attribute specifies the action to be taken by providing the resource type and name of the command resource in a colon seperated string. Note, you must provide quotes around the value due to the embedded colon character.
Example:

```
action: "NSR hypervisor:vcserver.mydomain.com";
```

autostart (read/write, choice list)
This attribute controls whether the task should be automatically started at the specified start time. If the **start now** choice is available, selecting it will cause the task to be executed immediately (when the resource is updated), but the attribute value will remain unchanged.

start time (read/write)
This attribute specifies when to automatically execute the task. The format is expected to be hh:mm in a 24 hour clock (so 23:00 would initiate the task at 11:00 pm.
Example:

```
start time: "23:00";
```

interval (read/write)
This attribute is used to specify how often the task is executed. The defaul value is 24:00, meaning run once per day.

period (read/write, choice list)
This attribute controls when the plan cycle will repeat. For example, a week cycle repeats the plan actions every 7 days while monthly does so once per

month.

plan (read/write)
This attribute contains a white-space separated sequence of *exec* and *skip* specifing if the task should be performed on a given day of the cycle defined by the **period** attribute. Only the first character of each entry is inspected so a weekly cycle could be abbreviated *s e e e e e* When the **plan** attribute does not contain enough actions to account for every day in the period, NetWorker will repeat the actions begining from the first entry.

last start (read only)
The attribute holds the last time the task was started.

last end (read only)
This attribute contains the last time the task ended.

last message (read only)
This attribute holds any message from the last time the task was executed.

job id (read only)
This attribute contains the job identifier for a running task.

status (read only)
This attribute holds the current status of the task. It can have the values **disabled, idle** and **running.**

A complete example of resource creation follows:

```
type:   NSR task;
name:   mytask;
action: "NSR hypervisor:maelstrom.legato.com";
```

FILES

/nsr/res/nsrdb − files in this directory should never be edited directly. Use **NetWorker Management Console or nsradmin instead.**

SEE ALSO

nsr_resource(5), **nsr_hypervisor**(5), **nsr**(8), **nsradmin**(8), **nsrtask**(8), **nsrvim**(8),

NAME nsr_unconfigured_library - NetWorker resource type "NSR unconfigured library"

SYNOPSIS **type: NSR unconfigured library**

DESCRIPTION Each medium changer that is discovered during a **scan for devices** operation has information about it initially stored in the **NSR unconfigured library** resource. This resource type is hidden, so it will not be diplayed in any administrative application unless the appropriate options (if available) are selected for that application to display hidden resources.

This resource is, in effect, a holding pen for the information that NetWorker needs to properly autoconfigure a tape library. We place all of the relevent information that we can collect during the device detection process into this resource before we attempt to actually configure any devices for NetWorker's use. The resource is also modified during the autoconfiguration process, so some values may change as that process progresses.

See **nsr_resource**(5) for information on NetWorker resources. To edit the **NSR unconfigured library** resources run:

```
nsradmin -c "type:NSR unconfigured library"
```

Be sure to include quotation marks and to insert a space between "NSR" and "unconfigured" and "library". See **nsradmin**(8) for information on using the NetWorker administration program.

ATTRIBUTES The information in parentheses describes how the attribute values are accessed. **Read-only** indicates that the value cannot be changed by an administrator. **Read/write** indicates a value that can be set as well as read.

Hidden indicates a hidden attribute of interest only to programs or experts. These attributes can only be seen when the hidden option is turned on in **nsradmin**(8). **Static** attributes change values rarely, if ever. **Dynamic** attributes have values that change rapidly. For example, an attribute marked **(read-only, static)** has a value that is set when the attribute is created and never changes.

type (read-only)
This attribute indicates that this resource is for an unconfigured library and the value is always **NSR unconfigured library**

name (read-only)
This attribute is an automatically generated unique name, usually derived from the SCSI inquiry data for the library along with serial number information also retrieved from the library.

library types
This attribute is a listing of the connection method(s) used to communicate with this library. There is one entry in the list for each storage node that can see the library. They are usually all the same, but that is not a requirement. The possible values include *scsi, silo* and *ndmp.*

silo library

This attribute is the filename of the (oddly enough, the dynamically loaded) software library that is used by NetWorker to communicate with a silo controller (if this is a silo-type library).

reference names (read-only)
This attribute is a listing of the names that will be used to describe this library for the various storage nodes that are able to see it.

storage nodes (read-only)
This attribute is a list of the storage nodes that have reported to the NetWorker server that they were able to communicate with this library.

descriptions (read-only)
This attribute is a list of descriptions of the library for each storage node that can see the library. They are usually based on inquiry data and SCSI addresses.

control ports (read-only)
This attribute is a list of the control ports that the various storage nodes reported using to communicate with this library. There may be more than one port for any given storage node due to the existence of multiple communications channels between the storage node and the library.

model (read-only)
This attribute is the model name used internally by NetWorker. Most often it will be "Standard SCSI Jukebox" which is our internal name for a library that behaves just as described in the SCCI Medium Changer specification.

serial number (hidden, read-only)
This attribute is the serial number of the library itself. Only the best choice identifier is used if the library reutrns multiple identifiers. (i.e. World Wide Node Name is the "most preferred", followed by World Wide Port Name, IEEE Node Name, IEEE Port Name, Vendor Unique Node Name, and VPD Page 0x80 serial number)

drive count (hidden, read-only)
This attribute is the number of drives that the library reports that it has.

drive base (hidden, read-only)
This attribute is the lowest SCSI element address for aany of the drives in the library.

drive addrs (hidden, read-only)
This attribute is a list of the actual element addresses for any drives that are present on the library. They are usually contiguous, but that is no longer a requirement for use with NetWorker.

drive ids (hidden, read-only)
This attribute is a list of the tape drives' identifiers returned to us by the library. These are used to match up the drives with the various access paths (i.e. tape drive device files) on all of the storage nodes that can see the drives in this library.

to be configured devices (hidden, read-only)
This attribute is a working list of devices that will be configured during the autoconfig process. It will change during the configuration process so the contents are not terribly meaningful.

unconfigured devices (hidden, read-only)
This attribute is a list of devices that are not yet matched to device access paths by the autoconfig process. As with "to be configured devices" this will change during the autoconfig process and is therefore not terribly meaningful.

slot count (hidden, read-only)
This attribute is the number of slots (SCSI storage elements) that the library reports that it contains.

slot base (hidden, read-only)
This attribute is the lowest SCSI element address for the slots in this library.

ie count (hidden, read-only)
This attribute is the number if import/export elements that the library reports that it contains.

ie base (hidden, read-only)
This attribute is the lowest SCSI element address for the import/export elements in this library.

media transport count (hidden, read-only)
This attribute is the number of medium transports (a.k.a. robot, gripper, the thing that moves the tapes around) that the library reports. This is usually only one.

media transport base (hidden, read-only)
This attribute is the lowest SCSI element address for the medium transport element(s) in this library.

feature list (hidden, read-only)
This attribute is a list of NetWorker's internal "features" that are applicable for this library. Possible values include:

"elements_status", "no_init_elem_sts", "no_start_init_elem_sts", "volume_tags", "barcode", "autoeject", "two_sided", "doorlock", "init_inlet_rqd", "no_trnsport_source_dest", "has_range", "no_ies_range", "need_align", "ies_no_barcode", "internal_ports", "display", "auto_cc_unload", "cc_eject", "recheck", "stacker_import", "stacker_export", "read_elem_unload_fail", "auto_inventory", "auto_allocate", "trust_compliant_jbox", "multi_stack_import", "multi_stack_export",

EXAMPLE

A complete example follows:

```
        type: NSR unconfigured library;
        name: EXABYTE Magnum20        WWNN=200000D0803FF06F;
 library types: scsi, scsi;
  silo library: ;
reference names: "rd=nash:EXABYTE@110.8.0", EXABYTE@4.7.0;
 storage nodes: nash, hal.devlab.emc.com;
  descriptions: <EXABYTE Magnum20 2.6 at SCSI Bus 110 Target 8 LUN 0>,
                <EXABYTE Magnum20 2.6 at SCSI Bus 4 Target 7 LUN 0>;
 control ports: "rd=nash:scsidev@110.8.0", scsidev@4.7.0;
       model: Exabyte Jukebox;
 serial number: WWNN=200000D0803FF06F;
   drive count: 2;
    drive base: 851;
   drive addrs: 851, 852;
     drive ids: "Serial Numbers:ATNN=IBM ULTRIUM-TD2 1110081463",
                "Serial Numbers:ATNN=IBM ULTRIUM-TD2 1110081756";
to be configured devices: "hal.devlab.emc.com:/dev/rmt4.1:000851",
                "hal.devlab.emc.com:/dev/rmt5.1:000852",
```

```
                    "nash:/dev/rmt/c111t4d0BESTnb:000851",
                    "nash:/dev/rmt/c112t2d0BESTnb:000852";
 unconfigured devices: ;
          slot count: 143;
           slot base: 1;
            ie count: 5;
             ie base: 801;
media transport count: 1;
 media transport base: 901;
        feature list: volume_tags, elements_status,
                    read_elem_unload_fail, barcode,
                    no_trnsport_source_dest;
            hostname: hal.devlab.emc.com;
       administrator: "user=root,host=hal.devlab.emc.com",
                    "user=administrator,host=hal.devlab.emc.com",
                    "user=system,host=hal.devlab.emc.com";
  ONC program number: 390109;
  ONC version number: 2;
      ONC transport: [TCP]   UDP ;
```

FILES */nsr/res/nsrdb* − files in this directory should never be edited directly. Use **nsrmm**(8), **nsradmin**(8), or **NetWorker Management Console** instead.

SEE ALSO **nsr_resource**(5), **nsradmin**(8), **nsr_storage_node_resource**(5)

NAME nsr_usergroup − NetWorker resource type ''NSR usergroup''

SYNOPSIS **type: NSR usergroup**

DESCRIPTION Each NSR user group is described by a single resource of type **NSR usergroup** (see **nsr_resource**(5)). To edit the NSR usergroup resources for a NetWorker server, type:

```
nsradmin -c "type:NSR usergroup"
```

or use **NetWorker Management Console.** See the **nsradmin**(8) manual page for more information on using the NetWorker administration program.

This resource describes groups of NetWorker users and their privileges.

ATTRIBUTES The following attributes are defined for resource type **nsr_usergroup**. The information in parentheses describes how the attribute values are accessed. **Read-only** indicates that the value cannot be changed by an administrator. **Read/write** means the value can be set as well as read. **Choice** means that the value of the attribute can only be one from a list specific to that attribute. For example, privileges can be 'Backup local data', or 'Operate NetWorker'. Several additional attributes (for example, name) are common to all resources, and are described in **nsr_resource**(5).

comment (read/write)
This attribute is provided for the administrator to keep any explanatory remarks or supplementary information about the user group.

users (read/write, list of strings)
This attribute specifies the list of users that are members of the user group. Each line specifies a user or a group of users, using one of these formats: *user/host@domain* , *group/host@domain* , *user@host* , *user@domain* , *group@host* , *group@domain* , *&netgroup* (only available on platforms that support netgroups) , *user_attribute=value[, ...]*.

where *user* is a user name; *host* is a host name; *group* is a user group name; *domain* is a domain name; *user_attribute* can be **user**, **group**, **host**, **nwinstname**, **nwinstancename**, **domain**, or **domaintype** (type of the domain, **NIS** or **WINDOMAIN**).

The user attributes: **nwinstname** and **nwinstancename** are used to indicate a NetWorker instance name. The value that should be entered for either of these attributes is the value in the "name" field in the NSRLA resource for the machine where a matched user is connecting from.

value can be any string delimited by white space. If the value has space in it, then it can be quoted with double quotes. The value may contain wild cards, "*". Entering just a user name allows that user to administer NetWorker from any host (equivalent to *user*@* or **/user* or **user=***user*). Netgroup names are always preceded by an "&".

The format: *user_attribute=value[, ...]* is more secure because the format is not over-**loaded. For example, if** *test@test.acme.com* **is entered, then any users in the** *test* **group or users named** *test* **and that are in the domain;** *test.acme.com* **or from the host;** *test.acme.com* **will match this entry.**

privileges (read/write, choice, null ok)
This attribute specifies the privileges members of this user group have. This attribute may have zero or more of the following privileges: *Change Security Settings*, *Remote Access All Clients*, *Configure NetWorker*, *Monitor NetWorker*, *Operate NetWorker*, *Operate Devices and Jukeboxes*, *Backup Local Data*, *Recover*

Local Data

Change Security Settings grants the permission to change security settings such as updating a NSR usergroups resource or changing remote access attribute in the NSR client resource.

Remote Access All Clients grants the permission to access other clients data.

Configure NetWorker grants the permission to configure NetWorker, such as creating new clients or devices.

Operate NetWorker grants the permission to perform maintenance operations on NetWorker, such as managing volumes or controlling savegroups.

Monitor NetWorker grants the permission to monitor the activities and status of NetWorker.

Operate Devices and Jukeboxes grants the permission to operate devices and jukeboxes, such as mounting, unmounting, and labeling of volumes.

Backup Local Data grants the permission to backup local data to NetWorker.

Recover Local Data grants the permission to recover local data from NetWorker.

This attribute can be any combination of the privileges described above. The only exception is some privileges require other privileges. For example, *Change Security Settings* privilege requires *Configure NetWorker* privilege, *Configure NetWorker* privilege must be set if *Change Security Settings* is set.

EXAMPLES The usergroup resource named *Users* is shown below. (Hidden options are not shown.) This is the default setup with the exception of the comment field. Users on any machine in any domain are members of this user group. Members in this group have the privilege to *Recover local data*, *Backup local data*, and *Monitor NetWorker*.

```
      type: NSR usergroup;
      name: Users;
  comments: Users can backup/recover data and monitor NetWorker;
     users: *@*;
privileges: Monitor NetWorker,
            Recover local data,
            Backup local data;
```

Another example of how to setup the usergroup resource named *Users* is shown below. Any users logged in to the domain *engineering.acme.com* are members of this user group. Members in this group has the privilege to *Backup local data*.

```
      type: NSR usergroup;
      name: Users;
  comments: Members of this group can backup data to NetWorker;
     users: domain=engineering.acme.com;
privileges: Backup local data;
```

SEE ALSO **nsradmin**(8), **nsr**(8)

NAME nwinstcreate − create a file containing NetWorker instance information

SYNOPSIS **nwinstcreate**
[−**i**] [−**x**] [−**f** *output-file*] [−**n** *NetWorker-instance-name*] [−**d** *NetWorker-instance-ID*] [−**p** *private-key-file* [−**c** *certificate-file*]]

DESCRIPTION The **nwinstcreate** command is used to create NetWorker instance information. This data is essential for using GSS EMC v1 authentication. In most cases, the user should let NetWorker create this information. The **nwinstcreate** program can be used to override the NetWorker defaults in the few cases where it is necessary.

The NetWorker instance information includes the NetWorker instance name, NetWorker instance ID, private key, and certificate. This information will be used on every NetWorker server, client and storage node. The NetWorker instance ID is the identity of the instance of NetWorker. The NetWorker instance name is a shorthand for the NetWorker instance ID and will be what you enter in various access control lists. The private key and certificate are used to verify the identity of a NetWorker instance.

It is best to run nwinstcreate before starting NetWorker for the first time. Then if the default *output-file* name was used, then NetWorker will pick up the information from the file without any further action from the user. Otherwise, the user must import the information after running **nwinstcreate** (see **nsr_la(5)**).

OPTIONS

−**i** Use interactive mode. The user will be asked to provide information for non-expert mode items whose values are not specified on the command line. The non-expert mode items include: the NetWorker instance name (which can be specified via the −**n** option), and the output file name (which can be specified via the −**f** option).

−**x** Use expert mode. Use with caution. The user will be asked to provide information on the expert mode items as well as the non-expert mode items whose values are not specified on the command line. The expert mode items are: the NetWorker instance ID (which can be specified via the −**d** option), the private key file (which can be specified via the −**p** option), and the certificate file (which can be specified via the −**c** option).

−**f** *output-file*
Specify the file in which the NetWorker instance information should be stored. The default value is the file: IDinitialize and it will be created in the /nsr/res directory (or the res directory under the base NetWorker install directory in Windows). If the default file name is used and the file was created before NetWorker was first started up, then NetWorker will automatically import in the information from this file and delete the file.

−**n** *NetWorker-instance-name*
Specify the NetWorker instance name that should be used for the NetWorker installation. This name is a shorthand for the NetWorker instance ID. The NetWorker instance name is the value that should be entered in access control lists if one wishes to limit access to users who are using GSS EMC v1 authentication and are from a certain instance of NetWorker. The NetWorker instance name should be unique through out the data zone.

−**d** *NetWorker-instance-ID*
Specify the NetWorker instance ID that should be used for the NetWorker installation. This ID will identify the NetWorker installation. It will be used whenever two NetWorker programs want to talk to each other. It should be

unique through out the data zone. It is highly recommended to let NetWorker choose this value unless there is a reason that the default is not valid.

−**p** *private-key-file*
Specify a file containing the private key that will be used by the NetWorker instance. The file is expected to be in PEM format.

−**c** *certificate-file*
Specify a file containing the certificate that will be used by the NetWorker instance. The certificate must correspond to the private key given with the −**p** option. That is, the certificate must contain the public key which corresponds to the private key. The file is expected to be in PEM format. The −**c** option is not allowed if the −**p** option is not also given.

SEE ALSO

nsr_la(5), **nsr_service**(5)

DIAGNOSTICS

The following exit status values are meaningful:

0 The program exited normally.

1 There was a usage or other error when attempting to create the output file.

10 There was a problem with the command line arguments.

NAME nwrecover − NetWorker graphical recover interface

SYNOPSIS **nwrecover** [−**s** *server*] [−**c** *client*] [−**x** *index namespace*] [−**T** *browse time* | "l locale_date"] [<*X-args*>] [*path*]

DESCRIPTION **nwrecover** is an X Window System application. It is used to recover lost files that have been saved with NetWorker. If you are running in a non-X11 environment, **recover**(1m) can be used to recover files.

The server's name can be specified with the −**s** *server* argument. When no server is specified, **nwrecover** uses the server selection rules found in **nsr**(1m). When multiple NetWorker servers are accessible, they can be selected from within the **nwrecover** command. If *path* is specified, **nwrecover** will attempt to initialize the current selection to the given path. The default attempted selection if *path* is not specified is the current working directory.

If you are recovering files that were saved with Access Control Lists (ACLs), you need to be root or the file owner to recover the file. Files with an ACL have a trailing '+' (e.g., -rw-r--r--+) after the mode bits when viewing file details. See **recover**(1m) for more information about ACLs.

There are three basic steps to recover a lost file: (1) Browse NetWorker's index in the Recover window to find the lost file, (2) Mark the file for recovery by selecting its checkbox and (3) Start the recovery. In addition, there are recover commands for relocating recovered files (Relocate), finding past versions of a file (Versions), changing the browse time (Change Browse Time), and overwriting or renaming recovered files that are in conflict with existing files (Conflict Resolution).

Opening the Recover window connects the client to its file indexes maintained on the server. The entries in the index represent previously saved files and are organized exactly like the filesystem. The file index is created in the *backup* index namespace when files are saved with **save**(1m). If files are saved into an index-storing archive pool using **nsrarchive**(1m), the file index is created in the *archive* index namespace. **nwrecover** offers command for changing the index namespace being browsed (Recover Archived Files). To browse the index for another filesystem, enter the pathname in the Location field.

To browse the index: The tree view of entries shown in the Recover window allows you to browse through your files and directories. You may use the mouse to open a directory and display its contents.

To mark files: After you have located your files by browsing the index, mark the files you want to recover by selecting their checkboxes. You can also highlight a file and use the Mark command from the Selected menu to mark files.

To start the recovery: Select the Start recover command from the File menu. The Recover Options dialog box appears, where you indicate to NetWorker what to do when a conflict occurs between a recovered file and an existing file. You select whether to be prompted for each individual conflict or to select one global resolution for all conflicts. Then, indicate to NetWorker whether you want to: Rename the recover file with a .R extension to preserve both files, Discard the recover file and preserve the existing file, or Overwrite the existing file to preserve the recover file as the only copy of the file.

Before starting the recovery, you have the option of relocating the recover files with the Relocate command. Enter the pathname of a new or existing directory in which to place your recovered files.

After you press OK in the Recover Options dialog box, the recover continues, and NetWorker automatically determines the media needed to complete the recovery, prompts the operator to mount the media, and executes the recovery. You can monitor the status of the recovery in the Recover Command window.

The Recover window also offers two commands for browsing the index in the past. Versions shows you the entire backup history for a file. Changing the Browse Time allows you to change the time at which you are viewing the online index.

A complete explanation of the **nwrecover** command can be found in the NetWorker Administrator's Guide.

OPTIONS

−**s** *server*
Set the current NetWorker server to *server.*

−**c** *client*
Set the current NetWorker client index to browse to *client.*

−**x** *index namespace*
Set the current file index namespace to *index namespace.* By default the *backup* namespace is used. The other recognized index namespace is: *archive.* This field is case sensitive.

−**T** **[browse time** | "l locale_date" **]**
Set the current index browse time to *browse time* (in **nsr_getdate**(3) format), or locale_date format when "l locale_date" is specified. Using this option is equivalent to using the **change browse time** dialog within **nwrecover**. See **recover**(1m) for more details on locale_date format supported.

<**X-args**>
Since it is an X-Windows application, nwrecover may also be invoked with various arguments that are not specific to nwrecover, but are generic X-Windows arguments. For instance, you may use the *-d* option in order to specify which display the nwrecover application should use. See "man X" or your system's X-Windows documentation for more details.

FILES

/usr/lib/X11/app-defaults/Networker
The X11 resources for **nwrecover**.

SEE ALSO

nsr_getdate(3), **nsr**(1m), **nsradmin**(1m), **recover**(1m)

The NetWorker Administrator's Guide

NAME nwretrieve − NetWorker graphical retrieve interface

SYNOPSIS **nwretrieve** −**s** *server* [<*X-args*>]

DESCRIPTION **nwretrieve** is an X Window System application. It is used to retrieve files that have been archived with NetWorker. If you are running in a non-X11 environment, **nsrretrieve**(1m) may be used to retrieve files.

The server's name may be specified with the −**s** *server* argument. When no server is specified, **nwretrieve** uses the server selection rules found in **nsr**(1m). When multiple NetWorker servers are accessible, they may be selected from within the **nwretrieve** command.

If you are retrieving files that were archived with Access Control Lists (ACLs), you need to be in group operator or the file owner to retrieve the file. See the *NetWorker Administrator's Guide* for more information about file permissions issues associate with archiving and retrieval.

There are three basic steps to retrieve a lost file: (1) Browse NetWorker's list of Archives in the nwretrieve browser window. (2) Select the Archive you wish to retrieve. (3) Start the retrieve.

To start the retrieve: Select the Start retrieve command from the File menu. The Recover Options dialog box appears, and you may enter a path to relocate to and select if you want to overwrite existing files.

After you press OK in the Recover Options dialog box, the retieve will begin recovering the selected archives and status will be displayed in the retrieve command dialog. NetWorker will then automatically determine the media needed to complete the retrieve, prompt the operator to mount the media, and execute the retrieve.

A complete explanation of the nwretrieve command may be found in the NetWorker Administrator's Guide.

OPTIONS −**s** *server*

Set the current NetWorker server to *server.*

<**X-args**>

Since it is an X-Windows application, nwretrieve may also be invoked with various arguments that are not specific to nwretrieve, but are generic X-Windows arguments. For instance, you may use the *-d* option in order to specify which display the nwretrieve application should use. See "man X" or your system's X-Windows documentation for more details.

/usr/lib/X11/app-defaults/Networker

The X11 resources for **nwretrieve**.

SEE ALSO **nsr**(1m), **nsradmin**(1m), **nsrarchive**(1m), **nsrretrieve**(1m)

NAME pathownerignore − ignore path-ownership rules during scheduled saves

SYNOPSIS <nsr_bin>/pathownerignore

DESCRIPTION In a clustered environment, the NetWorker software must distinguish between filesystems associated with the physical client, and those that are managed by a resource group (a virtual client). These criteria are referred to as the path-ownership rules. These rules determine which client index a save set is written to.

If a filesystem owned by a virtual client is defined in the save set list for a physical client resource, by default the filesystem **will not** be backed up during a scheduled save. The same is true for a filesystem owned by a physical client defined in the save set list for a virtual client resource. In both cases, the filesystem is omitted. This occurs because the NetWorker software views the client (which owns the filesystem) as not having matched the client of the current scheduled save.

To check the NetWorker path-ownership rule:

1. Run the following command on the NetWorker server:
   # savegrp -p -c **client_name**

2. Review which filesystems are owned by client_name. This procedure is part of the normal cluster installation setup. For detailed instructions, refer to the appropriate EMC NetWorker Installation Guide.

To test for the existence of misappropriated save sets, run a test probe with the verbose option set. The command output will warn you to which client indexes a save set will be saved. For example:

savegrp -pv -c **client_name group_name**

To ignore NetWorker default path-ownership rules, you can create the <nsr_bin>/pathownerignore file. This file causes the NetWorker software to back up the filesystem in question; however, the filesystem will be saved under the index of its correct owner. Creating the <nsr_bin>/pathownerignore file is not recommended, but it might be required under special circumstances. The <nsr_bin>/pathownerignore file does not override the default path-ownership rules. It causes the path-ownership rules to be ignored when determining if a filesystem should be backed up during a scheduled save.

SEE ALSO **save**(8), **savegrp**(8), **savefs**(8)

NOTES To override the path-ownership rules and have a save set written to an index other than its default owner, one must use the "save -c **client_name** " command. Refer to **save**(8) for more information.

NAME pmode − print mode sense data

SYNOPSIS **pmode** [**-f** *filename*]

DESCRIPTION The **pmode** program will parse the data output by the **msense**(1m) program and print in technological English. (C-style variables with hexadecimal numbers)

OPTIONS −**f** *filename* Specifies input; otherwise standard input is assumed.

EXAMPLE Sample output might look like:

```
viper# msense -a 0.0.0 -p 0x03 | pmode
Mode Header: mdl=35 mtype=0x0 dparm=0x10 bdlen=8
Block Desc[0]: dens=0x0 nblks=3933040 blklen=512
        Fixed Page, code 0x03 (Format Device):
              tracks_per_zone: 0xf
          alt_sectors_per_zone: 0x22
           alt_tracks_per_zone: 0x0
            alt_tracks_per_vol: 0x0
             sectors_per_track: 0x5e
           data_bytes_per_sect: 0x200
                    interleave: 0x1
             track_skew_factor: 0x8
           cylinder_skew_factor: 0x11
                          SSEC: 0x0
                          HSEC: 0x1
                           RMB: 0x0
                          SURF: 0x0
```

SEE ALSO **msense**(1m)

NAME preclntsave − child process to run pre-processing commands for NetWorker savepnpc

SYNOPSIS **preclntsave** −*s server* −*c client* −*g group* [−D debuglevel]

DESCRIPTION The **preclntsave** process attempts to lock the **/nsr/tmp/***group***.tmp** file. If it cannot, then it exits with status 0 indicating the pre-processing commands had been run. If the lock succeeds, then **preclntsave** invokes all of the pre-processing commands specified in the **/nsr/res/***group***.res** file, and then spawns the **pstclntsave** process, and exits with status 0. The lock on **/nsr/tmp/***group***.tmp** that **preclntsave** acquired is bequeathed to **pstclntsave**.

Note: This is to be invoked by **savepnpc** program only. It is not meant for users to use.

OPTIONS −**s** *server*
Specifies the controlling server.

−**c** *client*
The name of the client where the pre-processing commands will be performed on.

−**g** *group*
Specifies the group name that is being run.

−**D** *debuglevel*
For debugging purpose, the debug level could be 1, 2 or 3.

FILES **/nsr/tmp/***group***.lck**
The lock on this file must be acquired before **preclntsave** performs any other actions. The lock is released only upon process exit. If the file does not exist, it is created. The file is never removed.

/nsr/tmp/*group***.tmp**
Preclntsave attempts to acquire a lock on this file. If it fails, then the command exits with status zero. If it succeeds, then it executes the pre-processing commands. If they succeed, then **preclntsave** executes the **pstclntsave** command and bequeaths the lock to it. If the file does not exist, it is created. This command does not remove the file.

/nsr/res/*group***.res**
The file that contains the actual pre-commands that this command executes.

/nsr/res/*group***.res.lck**
This vestigial file is created and locked before the **/nsr/res/***group***.res** is opened. The file is never removed.

SEE ALSO **savepnpc**(1m), **pstclntsave**(1m), **save**(1m)

NAME pstclntsave − child process of preclntsave to run post-processing commands for NetWorker savepnpc

SYNOPSIS **pstclntsave** −*s server* −*c client* −*g group* [−p pollinterval] [−t timeout] [−D debuglevel]

DESCRIPTION The **pstclntsave** process checks the STATUS and WORKLIST attributes of the GROUP resource from the server every number of seconds specified in the poll interval to see if the group is running and if client is still included in the WORKLIST. Whenever **pstclntsave** is aborted, or the time_out condition occurs, or the group's STATUS is not running, or the client is no longer in the WORKLIST (whichever comes first), **pstclntsave** performs all the post-processing commands specified in **/nsr/res/***group***.res** file, unlinks **/nsr/tmp/***group***.tmp,** then records the results (success or failure) in the **/nsr/logs/savepnpc.log** file.

Note: This is to be invoked by **preclntsave** program only. It is not meant for users to use.

OPTIONS

−**s** *server*
Specifies the controlling server.

−**c** *client*
The name of the client where the pre-processing commands will be performed on.

−**g** *group*
Specifies the group name that is being run.

−**p** *pollinterval*
Specifies how often (in seconds) to poll the server.

−**t** *timeout*
The timeout condition in nsr_getdate() format string to start the post-processing commands. This can also be specified in the **/nsr/res/***group***.res** file.

−**D** *debuglevel*
For debugging purpose, the debug level could be 1, 2 or 3.

FILES

/nsr/tmp/*group***.tmp**
This file must exist. Upon start-up, **pstclntsave** immediately locks this file. Before exiting, **pstclntsave** most likely removes this file.

/nsr/tmp/*group***.lck**
Before exiting, **pstclntsave** attempts to lock this file. If it succeeds, then **/nsr/tmp/***group***.tmp** is removed. If **/nsr/tmp/***group***.lck** does not exist, it is created. The file is never removed.

/nsr/res/*group***.res**
The file that contains the actual post-command that this command executes.

/nsr/res/*group***.res.lck**
This vestigial file is created and locked before the **/nsr/res/***group***.res** is opened. The file is never removed.

/nsr/logs/savepnpc.log
Where this command logs its status.

SEE ALSO **preclntsave**(1m), **savepnpc**(1m), **save**(1m)

NAME read_a_block - read a 32KB block from tape

SYNOPSIS **read_a_block** −**f** *device* [−**v**]

DESCRIPTION The **read_a_block** program reads a single 32KB block from a test tape.

OPERANDS −**f** *device*
Specifies the device on which the tape is mounted.

OPTIONS −**v** Run the program in verbose mode. This option will print out the version number of the CDI library used by the program.

EXAMPLES Sample output including drive status information:

```
% read_a_block -f /dev/rmt/3cbn
read returns buffer full of 564f4c31 (1448037425 decimal)
```

SEE ALSO **libcdi**(1m)

NAME recover − browse and recover NetWorker files

SYNOPSIS

recover [-f] [-n] [-q] [-u] [-i {**nNyYrR**}**] [-d** *destination***] [-c** *client***] [-x** *index-namespace***] [**−**t** <*date* | *"l locale_date*">**] [-s** *server***] [-J** *storage-node***] [-p** *pass-phrase***] [-e** *exclude-file***] [-B] [** *dir***]**
recover [-f] [-n] [-u] [-q] [-i {**nNyYrR**}**] [-I** *input file***] [-d** *destination***] [-c** *client***] [-x** *index-namespace***] [**−**t** <*date* | *"l locale_date*">**] [-s** *server***] [-J** *storage-node***] [-B] [-e** *exclude-file***] [-p** *pass-phrase***] -a** *path*...
recover [-f] [-n] [-u] [-q] [-i {**nNyYrR**}**] [-d** *destination***] -s** *server* **[-B] [-e** *exclude-file***] [-J** *storage-node***]-S** *ssid[/cloneid]* **[-S** *ssid[/cloneid]***]... [** *path***]...**
recover [-f] [-q] -i {**NYR**} **-R** *recover-target* **[-c** *client]* **[-d** *destination***] [-x** *index-namespace***] [**−**t** <*date* | *"l locale_date*">**] [-s** *server***] [-J** *storage-node***] [-p** *pass-phrase***] [** *dir*]
recover [-f] [-n] [-U] [-q] [-i {**nNyYrR**}**] [**−**t** <*date* | *"l locale_date*">**] [-s** *server***] [-J** *storage-node***] [-p** *pass-phrase***] [-N** *system save set***]**

DESCRIPTION **recover** browses the saved file index and recovers selected files from the NetWorker system. The file index is created in the *backup* index namespace when files are saved with **save**(1m). If files are saved into an index-storing archive pool using **nsrarchive**(1m), the file index is created in the *archive* index namespace. When in interactive mode (the default), the user is presented with a view of the index similar to a UNIX filesystem, and may move through the index to select and recover files or entire directories. In automatic mode (−**a** option), the files specified on the command line are recovered immediately without browsing. While in save set recover mode (−**S** option), the save set(s) specified are retrieved directly without browsing the NetWorker file index. Use of save set recover mode is restricted to root.

When using recover without the −S option, users who are root may recover any file. The remaining permission checking rules described in the paragraph apply to users who are not root. For files that don't have an Access Control List (ACL), the normal UNIX mode bits must allow you to read the file in order to recover it. Files with an ACL can only be recovered by their owner or by root.

If path argument is used with the save (1m) command and one of the directories in the path is a symbolic link, then the target path of the symbolic link must be specified with −a option to recover the files. Recovering the files by specifying the symbolic link in the path with −a option will result in "<path> not in index" message. Alternatively, −S option may be specified to recover these files.

Concurrent recoveries can be performed from an advanced file type device (adv_file), either by using multiple -S options to identify multiple save sets, or executing multiple recover commands concurrently.

OPTIONS

−**a** Specifies automatic file recovery with no interactive browsing. *Path* specifies one or more files or directories to be recovered. Symbolic links are not followed, though the link file itself will be recovered. Mount points are also not followed unless the most recent **save**(1m) was performed with the '-x' option.

−**B** Specifies a BMR recovery that automatically uses the standard BMR exclude file, **exclude.NETWORKER.** All the file paths listed in the standard exclude file are excluded from the recovery.

−**c** *client*
client is the name of the machine that saved the files. When browsing a directory that was saved by another client, the pathnames will reflect the file tree of the client that saved the files. By default **save** and **recover** determine the client name from the filesystem table. This option might be necessary if the −**L** option was used on the **save** command. This option cannot be used in

conjunction with the −**S** *ssid* option (save set recover mode).

−**d** *destination*
Specifies the destination directory to relocate recovered files. Using this option is equivalent to using the **relocate** command when in interactive mode (see usage). Relative paths are interpreted relative to the current working directory.

−**e** *exclude file*
Specifies an exclude file that contains a list of all the paths to exclude from the recovery. The exclude file can be an absolute or relative path for browsable or save-set based recovery. The exclude file must be an absolute path for directed recovery. If the exclude file lists a valid directory path to exclude but does not list file entries from the directory, then the directory is recovered along with the files in it. The recovery total in the exclude file report will account for this directory exclusion.

The following wildcards are supported for pattern matching in the list of files to exclude. Except for the ** wildcard, any of the wildcards can appear anywhere in the file path.

** **-** Specifies a directory and all the files and subdirectories within it.

For example, the pattern **C:\globe**** (on Windows) or **/usr/globe/**** (on UNIX) in the exclude file specifies the **globe** directory and all its subdirectories and files for exclusion from the recovery. The ** wildcard must appear at the end of the file path.

* **-** Matches any string of characters in a path.

For example, the pattern **C:\globe*** (on Windows) or **/usr/globe/*** (on UNIX) in the exclude file specifies all the subdirectories and files within the **globe** directory.

Similarly, the pattern **/a*/globe/*** specifies any directory that starts with the letter **a** in the root directory and contains the **globe** directory within it. All the subdirectories and files within the **globe** directory are also excluded from the recovery.

[..] - Matches any single character within a range, where the range is specified with a dash **(-).**

For example, the pattern **globe[0-9]** specifies **globe,** followed by a number between 0 and 9, inclusive.

The pattern **globe[adrs]** specifies **globe,** followed by any single character within the brackets.

? - Matches any single character in a path.

For example, the pattern **C:/gl?be/** (on Windows) or **/usr/gl?be/** (on UNIX) in the exclude file specifies all the files and directories with the name **gl?be,** where **?** is any single character.

−**f** Forces recovered files to overwrite any existing files whenever a name conflict occurs. This is the same as specifying −**iY**.

−i {**nNyYrR**}
Specifies the initial default overwrite response to use when recovering existing files. Only one letter may be specified. This option is the same as the **uasm −i** option when running in recover mode. See the **uasm**(1m) man page for a detailed explanation of this option. For directed recovers (see the -R option), only 'N', 'Y', and 'R' are valid values.

−I *input file*
In addition to the paths specified on the command line, this option will flag paths from the named file to be recovered as well. The paths in the file must be listed one per line. If no paths are specified on the command line, then only those paths specified in the file will be recovered. To be used in conjunction with -a option.

−J *storage-node*
Specifies which host to use as the storage node for the recovery (see **nsr_storage_node**(5)).

−n Does not write or create any files or directories when recovering.

−N *system save set*
Used to recover the following system save sets: SYSTEM DB, SYSTEM FILES, or SYSTEM STATE. (Windows Only)

−p *pass-phrase*
Specifies an additional pass phrase to use when attempting to recover files backed up using the *aes* directive. By default the current datazone encryption key is tried as well as the key generated from the default pass phrase. Using this option causes recover to generate an encryption key from the pass phrase and try it if the default and datazone pass phrase keys do not work. This option can be specified multiple times.

−q Turns off the verbose output. The **recover** command normally runs with verbose output.

−R *recover-target*
Specifies the name of the remote machine to direct the recovery. This is used in conjunction with the -c option to specify browsing of another client's index. When the -R option is used, either the -f or the -i option must also be specified in order to instruct the recover target what to do when it is recovering existing files. Note that the values 'N', 'Y', and 'R' are the only valid ones to use with the -i option for directed recovers. Note also that the -a option is not supported with the -R option.

−s *server*
Selects which NetWorker server to use.

−S *ssid[/cloneid]*
Specifies save set recover mode and can only be used by root. This mode can be used to implement fast batch file recovery without requiring the NetWorker file index entries. The save set id may be for either a backup save set or an archive save set. *ssid* specifies the save set id's for the save set(s) to be recovered. When there are multiple clone instances for a save set, the *cloneid* can also be specified to select the particular clone instance to be recovered from. When no *path* arguments are specified, the entire save set contents will be recovered. One or more *path*s can be specified to limit which directories and files are actually recovered. If *path*s are supplied, then the beginning of each path name as it exists in the save set must exactly match one of the *path*s before it will be recovered. Shell like file name matching using meta characters like '*', '**?**', and '**[...]**' is not done. You can use a *path* that ends in with a slash

('/') to force a directory only match (e.g., use a *path* of **/etc/fs/** instead of **/etc/fs** to prevent files like **/etc/fsck** from being recovered as well).

−**t** <*date* | "*l locale_date*">
Display/recover files as of the specified *date.* The date specified can be in **nsr_getdate**(3) format, or locale_date format when "l locale_date" is specified. Note that the surrounding quotes and a blank space after "l" are required for locale_date format. Using this option is equivalent to using the **changetime [-l]** command with the given *date* or *locale_date* when in interactive mode (see usage). This option cannot be used in conjunction with the −**S** *ssid* option (save set recover mode).

For example, -t *date* specifies date/time in nsr_getdate (3) format. -t "l <locale_date>" specifies date/time in the locale format.

See **changetime** for more information on the locale_date format supported.

−**U** Authoritative restore of writers that support such functionality is to be performed via the recover command line using this new flag. Only one writer can be specified per recover session. Examples:

recover [-s *server***]** -N "VSS USER DATA:\DFS Replication service writer;"

recover [-s *server***]** -N "VSS SYSTEM SERVICES:Cluster Database"

A VSS saveset itself is not supported:

recover [-s *server***]** -U -N "VSS SYSTEM SERVICES:\"

−**u** Stops when an error occurs during recovery. Normally, recover treats errors as warnings and tries to continue to recover the rest of the files requested. However, when this option is used, recover will stop recovering on the first error it encounters. This option is not valid for directed recovers.

−**x** *index-namespace*
Browse/recover files in the specified file index namespace. By default the *backup* namespace is used. The other recognized index namespace is: *archive.* This field is case sensitive.

USAGE When using recover in the interactive mode, an image of the filesystem at a particular time is presented. Using commands similar to the shell, you can change the view and traverse the filesystem. Files may be selected for recovering, and the actual recover command issued.

The following commands manipulate the view of the filesystem and build the list of files to recover. In all of the commands that take a *name* argument pattern matching characters can be used. The pattern matching characters and regular expression format are the same as for the UNIX shell **sh**(1).

ls [*options*] [*name ...*]
List information about the given files and directories. When no *name* arguments are given, **ls** lists the contents of the current directory. When a *name* is given and *name* is a directory, its contents are displayed. If *name* is a file, then just that file is displayed. The current directory is represented by a '**.**' (period). The options to this command correspond to those of the UNIX command, **ls**(1). An additional recover specific −**S** option can be used to select the save time instead of the last modified time for sorting (with the −**t** option) and/or printing (with the −**l** option). Files that have been added to the

recover list are preceded by a '**+**'. Files that have an ACL have a trailing '+' (e.g. -rw-r--r--+) after the mode bits when viewing file details.

lf [*name ...*]
is the same as **ls** −**F**. Directories are marked with a trailing '**/**', symbolic links with a trailing '@', sockets with a trailing '**=**', FIFO special files with a trailing '|', and executable files with a trailing '*'.

ll [*name ...*]
is the same as **ls** −**lgsF**. Generates a long format listing of files and directories. This command can be used to find the value of a symbolic link.

cd [*directory*]
Change the current working directory to [*directory*]. The default directory is the directory **recover** was executed in. If *directory* is a simple symbolic link, **cd** will follow the symbolic link. However, if *directory* is a path containing symbolic links anywhere but at the end of the path, the **cd** command will fail; you should **cd** a component of the path at a time instead.

pwd Print the full pathname of the current working directory.

add [*name ...*]
Add the current directory, or the named file(s) or directory(s) to the recover list. If a directory is specified, it and all of its descendent files are added to the recover list. Symbolic links are not followed, though the link file itself will be recovered. Mount points are also not followed unless the most recent **save**(1m) was performed with the '-x' option.

debug [*level*]
Turn on or turn off debugging. Level must be a number. If level is 0, debugging is off. As the debug level goes higher, the recover command prints out more messages. By default, debugging is off.

delete [*name ...*]
Delete the current directory, or the named file(s) or directory(s) from the recover list. If a directory is specified, that directory and all its descendents are deleted from the list. The most expedient way to recover a majority of files from a directory is to add the directory to the recover list, and then delete the unwanted files.

dir [**/w**] [**filename...**]
This command is similar to the "ll" command with the following differences. The dir command uses the display format used by "dir" command in DOS command prompt. Also this command does not add a + to the files selected for recovery. With /w option, the names of the files or directories only are displayed.

list [−**l**] | [−**c**]
Display the files on the recover list. With no arguments the recover list is displayed as a list of full path names, one per line, followed but a total count of the files to be recovered. The **-c** argument prints just the total count of files to be recovered. The **-l** argument prints the files in the same format as the **ll** command with the −**dS** options.

volumes
Prints a list of the volumes needed to recover the current set of files on the recover list. If all volumes are near-line (near-line volumes are available volumes that are not mounted), this command will note that all volumes needed are near-line. If all volumes are on-line (on-line volumes are those that are available and mounted) or if some volumes are on-line and some are near-line, this command will note that all volumes are on-line. Both near-line and

on-line volumes do not require manual intervention.

recover
Recover all of the files on the recover list from the NetWorker server. Upon completion the recover list is empty.

verbose
Toggle the status of the ''verbose'' option. When verbose mode is on, **recover** displays information about each file as it is recovered. When verbose mode is off, **recover** only prints information when a problem occurs. The default is verbose mode on.

force If name conflicts exist, overwrite any existing files with recovered files.

noforce
Cancel the **force** option. When in 'noforce' mode, a prompt is issued each time a naming conflict arises between a file being recovered and an existing file. At each prompt, six choices are presented: 'y', 'Y', 'n', 'N', 'r' and 'R'. To overwrite the existing file, select 'y'. To rename the file to an automatically generated alternative name, select 'r'. Selecting 'n' causes the recovered file to be discarded. The capital letters invoke the same action for all subsequent conflicts without further prompting. Hence, selecting 'Y' will cause all existing conflicting files to be overwritten, 'N' will cause all conflicting recovered files to be discarded, and 'R' will automatically rename all conflicting recovered files (except when an external ASM has a conflicting file name that already ends in the rename suffix).

relocate [*directory*]
Change the target recover location to *directory*. If *directory* is not specified then the user will be prompted for a destination directory. Relative paths are interpreted relative to the current working directory within the **recover** program. The recovered files will be placed into this directory, which will be created if necessary. When files from multiple directories are being recovered, they will be placed below this directory with a path relative to the first common parent of all the files to be recovered. For example, if */usr/include/sys/errno.h* and */usr/include/stdio.h* are being recovered, and the relocation directory is set to */tmp*, then the first common parent of these two files is *include*, so the recovered files will be named */tmp/sys/errno.h*, and */tmp/stdio.h*.

destination
Print destination location for recovered file.

exit Immediately exit from **recover**.

help Display a summary of the available commands.

? Same as **help.**

quit Immediately exit from **recover**. Files on the recover list are not recovered.

changetime [*time* | **−l locale_date [time]**]
Display the filesystem as it existed at a different time. If no *time* is specified the 'current' time is displayed, and a prompt is issued for a 'new' time. The new time is given in **nsr_getdate**(3) format by default, or <locale date [time]> format if -l is specified.

For example, changetime <date> specifies date/time in nsr_getdate (3) format, whereas changetime -l <locale date [time]> specifies date/time in the locale format, where [time] is optional.

This nsr_getdate format is very flexible. It accepts absolute dates, such as

March 17, 1997, and relative dates, such as *last Tuesday*. Absolute dates can be given in two formats: *MM/DD[/YY]*, and *Month DD[, YYYY]*. Times can also be specified as either absolute or relative, with absolute times in the format: *HH[[:MM][:SS]] [am|pm] [time zone]*. For example, 12:30 am, 14:21, and 10 pm PST. The current time is used to calculate unspecified parts of a relative date (e.g. 2 days ago means 2 days ago at the current time), and the end of the day is assumed for unspecified times on an absolute date (e.g. July 2 means July 2 at 11:59:59 PM). By default, the present is used as the current time.

The resolution of the filesystem image at a time in the past depends on how often **save** was run and how far back the NetWorker file index information goes.

On UNIX, <locale date [time]> is supported in the date/time format display of "dir" and "ls -l" command of recover (where time is optional) and date +%c (1) command format of a locale. Except for locale date/time format of date +%c (1) command, the optional time format is supported in 24 hour format if specified.

The locale date/time specified are parsed in with strptime (3C) C library function. Hence, the support for the locale date/time input may be limited by the platform implementation of strptime C library function.

The supported locale date [time] formats are:

<locale_abbreviated_month> dd [yyyy [HH:MM[:SS]]]
: <locale_abbreviated_month> is the locale abbreviated month listed in "ls -l" of recover command. dd is the 2 digit number for the day of month (without locale "day" symbol). yyyy is the optional year and defaulted to current year if not specified. HH:MM[:SS] is the optional time in 24 hour format, defaulted to 23:59:59 if not specified, whereas both hour and minute are required when specified.

<date_of_dir_in_recover> [HH:MM[:SS]]
: <date_of_dir_in_recover> is the locale date format of "dir" in recover command. HH:MM[:SS] is optional time in 24 hour format, defaulted to 23:59:59 if not specified, whereas both hour and minute are required when specified.

date +%c (1) command format for non-English locale
: Locale date/time format of UNIX "date +%c" (1) command.

For example, in English (United States) locale, following locale date/time input can be specified for July 11, 2006 [13:24:30]:

07/11/06
: "dir" format of recover command (where time is defaulted to 23:59:59)

07/11/06 13:24:30
: "dir" format of recover command

Jul 11 2006 13:24:30
: "ls -l" format of recover command

On Windows, <locale date [time]> is supported in the date/time format display of "dir" or "ls -l" command of recover:

<locale date> specified in mm-dd-yyyy, dd-mm-yyyy, or yyyy-mm-dd depending on user locale and must be specified with the user locale's default date separator as displayed in "dir" command of recover

[time] optional, both hour and minute are required if specified. Time is defaulted to 23:59:59 (end of day) if not specified, or if there is any error in the locale time (format or value) input.

The supported locale time formats are:

HH:MM[:SS]
24 hour format if user locale is in 24 hour format or if locale AM/PM designator string is not specified

<locale AM|PM string> hh:MM[:SS]
12 hour format with locale AM/PM designator string as prefix if user locale and system locale are in the same 12 hour locale

hh:MM[:SS] <locale AM|PM string>
12 hour format with AM/PM designator string as suffix if user locale is in 12 hour format and the system locale is the same as user locale

hh:MM[:SS]<a|p>
12 hour format with 'a' (for AM) or 'p' (for PM) if the user locale is in 12 hour format and system locale is different than user locale

For example, in English (United States) locale, where system locale is the same as user locale, date/time is in 12 hour time format with AM/PM designator string as suffix. Following locale date/time input can be specified for April 05, 2006 [13:24:30]:

04/05/2006

04/05/2006 13:24:30

04/05/2006 1:24:30 PM

Note that first example without the time specification will result in 23:59:59.

versions [*name*]
All instances of the current directory, if *name* is not specified, or the named file or directory, found in the NetWorker file index are listed. For each instance, three lines of data are displayed. The first line is similar to the **ll** output. The second line lists the instance's save time. The third line specifies which tape(s) this instance may be recovered from. With appropriate use of the **changetime** command, any one of the entries may be added to the recover list. As with **ls**, **lf**, and **ll**, files that have been added to the recover list are preceded by a '**+**'.

SEE ALSO

ls(1), **date**(1), **nsr_getdate**(3), **strptime**(3C), **nsr_service**(5), **nsr_device**(5), **nsr**(1m), **nsrd**(1m), **nsrindexd**(1m), **nwrecover**(1m), **save**(1m)

DIAGNOSTICS

Recover complains about bad option characters by printing a ''usage'' message describing the available options.

Message from server: other clones exist for failed save set
The request failed on a save set that had multiple clones. The server automatically picks a different clone on each attempt. Recover automatically re-submits its recover request to the server, if any files remain to be recovered.

Path *name* **is within** *machine*:*export-point*
An informative message that lets you know that the given path name is mounted from a network file server and that the recovery will use the index for the named file server. If the *machine* is not a NetWorker client, then the **−c** option may be necessary.

Browsing *machine*'**s on-line file index**
An informative message that explicitly states which NetWorker client's index is being browsed for interactive recovers that resolve to another machine.

Using *server* **as server for** *client*
An informative message that lets you know which NetWorker server was selected for client's index.

Cannot open recover session with *server*
This message indicates that some problem was encountered connecting to the NetWorker server on the named machine.

error, *name* **is not on client list**
This message indicates that the client invoking the **recover** command is not in the server's client list. See **nsr_service**(5) for details.

path: **Permission denied**
The file *name* cannot be recovered because you are not root, and you don't have read permission for the file.

path: **Permission denied (has acl)**
The file *name* cannot be recovered because you are not root, the file has an ACL (Access Control List), and you are not the owner of the file.

NAME recoverpsm − recover the NetWorker Management Console database from long-term storage with NetWorker

SYNOPSIS **recoverpsm** [−**s** *server*] [−**c** *client-name*] [−**d** *destination*] [−**p** *pass-phrase*] [−**t** *time*] [−**S** *NetWorker Management Console database server*] [**-hfO**]

DESCRIPTION **recoverpsm** recovers the contents of a NetWorker Management Console database from a NetWorker server. The NetWorker Management Console database credential file (gstd_db.conf) is automatically recovered after the NetWorker Management Console database is recovered. The GST service needs to be stopped before **recoverpsm** can be used.

The LD_LIBRARY_PATH or equivalent environment variable of the shell from which **recoverpsm** is started must contain <product install directory>/sybasa/lib64 (Solaris/AIX/HP-UX) or <product install directory>/sybasa/lib (linux) as one of the components. If NetWorker Management Console is not installed in the default /opt/LGTOnmc directory on Solaris, you may also need to add <product install directory>/bin to LD_LIBRARY_PATH.

recoverpsm assumes that the database directory has not changed since the last backup. If this is not the case, use the -O option to skip the recovery of database credential file and recover the file later using the NetWorker recover (1m) command.

OPTIONS

−**c** *client-name*
Use client-name as the name of the host that saved the database. See recover(1m) for more details.

−**d** *destination*
Use "destination" as the directory to which the database should be recovered to.

−**f** Force recovered database file to overwrite any existing database file when a name conflict occurs.

−**h** Display the usage of the **recoverpsm** binary.

−**O** Use this option to omit the recovery of the database credential file. This could be used in case the location of the database directory has changed since last backup (for e.g. the NetWorker Management Console is being moved to a different machine). In this scenario, run the recover command after the **recoverpsm** has completed sucessfully to recover the credential file.

−**p** *pass-phrase*
Specifies an additional pass phrase to use when attempting to recover the database credential file backed up using the aes directive. See recover(1m) for more details.

−**S** *NMC database server*
Use "NMC database server" as the name of the NMC database server. The name should be of the form "gst_on_<host name>" (where host name is the short name of the machine running the NetWorker Management Console server).

−**s** *server* Use "server" as the NetWorker server.

−**t** *time* Recover database as of the specified date (in nsr_getdate(3) format).

EXIT STATUS **recoverpsm** exits with a non-zero exit code if an error occurs and with zero on success.

SEE ALSO **recover**(1m), **savepsm**(1m)

NAME relem − read element status

SYNOPSIS **relem** [**-a** *b.t.l*] [**-fvtb**] [**-m** {**0**|**1**|**2**}] [**-r** *eladdr.nel*] [**-l**]

DESCRIPTION The **relem** program will send a READ ELEMENT STATUS command to all changers, or (with the **-a** option) to the named device.

SIMPLE OPTIONS

−**a** *b.t.l* Selects a specific ordinal SCSI address, where **b** is the logical SCSI bus, **t** is the SCSI target, and **l** is the SCSI logical unit number (LUN) on that target. See **libscsi**(1m).

−**f** generates full output (somewhat verbose).

−**v** generates very verbose output.

−**t** causes volume tags, if present, to be printed.

−**b** causes the returned element status data to be dumped as hexadecimal codes rather than interpreted.

−**l** Performs a complete LUN search for all SCSI adapters in the system when performing Autodetection. This argument is accepted on all systems, but does not have any effect on HP-UX systems. Due to the method used to scan for available devices on HP-UX systems, all accessible devices are shown, and the −**l** option has no additional effect. On all other systems, checking starts at LUN 0 for SCSI devices. The first empty LUN found will end the search for a given target ID. With the −**l** option, LUNS on all target IDs for all SCSI busses in the system are checked for jukeboxes. This can take a **very long time** and should only be used when necessary. For example, a Fibre Channel adapter can support 126 target IDs, each of which may have 80 or more LUNs. Checking all LUNs on this single adapter may take over 10 minutes.

METHOD OPTIONS

-m 0 all element status data is fetched in one call

-m 1 element status data is fetched per element type (e.g., all drive elements are read at once, then all slot elements, etc.)

-m 2 element data is fetched per element (which is the default method)

The SCSI specification allows each of the previous methods of fetching element data.

Note: some changers have defects with respect to fetching element data. For example, one changer may be accurate when reporting all elements of a particular type (using method −**m 1**), but return only zeros if asked for all elements at once (using method −**m 2**).

RANGE OPTIONS

−**r** *eladdr.nel* is used to read a range of addresses, where *eladdr* is the starting decimal address (in the particular changer's numbering scheme) of the element to start from, and *nel* is the number of elements of status to read. Use **changers** to get the particular addresses used by your changer.

SEE ALSO libscsi (1m)

NAME resource descriptors − RAP resource file format

SYNOPSIS

resource ::= *attribute list <blank line>*
attribute list ::= *attribute* [**;** *attribute*]*
attribute ::= *name* [**:** *value* [**,** *value*]*]
name, value ::= *<printable string>*

DESCRIPTION Files with the **.res** suffix use a common format to describe *resources.* Generally, a resource represents something that a system administrator might want to manage (for example, devices, backup schedules, file systems), or that a user might want to locate. The encoding of the information describing a resource is called the *resource descriptor*. Resource description files are are accessed by applications and services that use the Resource Administration Platform (RAP), but they can also be viewed with a normal text editor.

Each resource descriptor is made up of a list of attributes, and ends in a blank line. Each attribute in the attribute list has a name and an optional list of values. The attribute name is separated from the attribute values by a colon (**:**), attribute values are separated by commas (**,**), and attributes are separated by semicolons (**;**). A comma at the end of a line continues the line, as does a back-slash (\) character. The back-slash character can also be used to escape the special meaning of a single character (such as comma, semicolon, double quote, and back-slash), or the string can be included in quotes. A line beginning with a pound-sign (#) is a comment and the rest of the line is ignored. The end of a resource attribute list is marked with a blank line.

The attribute name and values can contain any printable character. Upper and lower case is ignored on comparisons, and extra white space is ignored on both ends but not in the middle of names and values. For example,

```
Name: testing 1 2;
```

will match

```
name   :   Testing 1 2 ;
```

but is different than

```
Name: testing 1     2;
```

EXAMPLE Here is an example that includes two resources. The first resource has eight attributes: **type**, **name**, **server**, **schedule**, **directive**, **group**, **save set**, and **remote access**. The **group** attribute has two values: **marketing** and **sales**. The **remote access** attribute has no value. The second example includes an attribute that needs quotes because it contains a colon.

```
          type: NSR client;
          name: venus;
        server: mars;
      schedule: Default;
     directive: custom;
         group: marketing, sales;
      save set: /, /usr;
remote access: ;

          type: NSR group;
          name: engineering servers;
     autostart: Enabled;
    start time: "3:33";
```

Each resource includes the special attribute **type**. The **type** attribute defines which other attributes a resource can contain. For example, a resource with type printer might include an attribute **paper size**, while in a resource of type **NFS filesystem** this attribute makes no sense. The type attribute is case sensitive and must be used exectly as described. For example, a type "NSR group" is different from "nsr group".

The **name** attribute is a descriptive name of the object that a resource represents. In the example above, the name of the second resource is **engineering servers**, which describes a group of machines to be saved together.

The **administrator** attribute is the list of users that have permission to modify this resource. This attribute is inherited from the server resource when a new resource is created. The administrator in the server resource also controls who has permission to create new resources and delete old ones.

The **resource identifier** is set and used internally by the RAP system. It provides a unique identification of each resource, and although it is sometimes printed like an attribute, it is stored differently. When new resources are created the **resource identifier** attribute should be left off. This signals the system that this is a new resource and a new identifier will be assigned.

TYPES There are special resources that define the attributes found in a given type. They are called resource type descriptors. Type descriptors have the same syntax as other resources except that they have a type attribute with the value **type** and a **type name** attribute with the value of the type they describe. For example, the resource type descriptor for type NFS filesystem would have, among its other attributes:

type:type; type name:NFS filesystem

Type descriptors are used internally, and should normally never be stored in files or seen by administrators. For each of the other attributes in a type descriptor, there are three or more values. The first value gives the base type, the second value gives a list of flags separated by spaces, the third value is a string for on-line help, and any subsequent strings are default values. This type information is used by system administration tools to improve the user interface.

FILES ***.res** Files that contain resource descriptors.

SEE ALSO **nsradmin**(1m)

NAME

save − save files to long-term storage with NetWorker

savepnpc − save files to long-term storage with NetWorker and run pre- and post-processing commands on a NetWorker client

SYNOPSIS

save [−**BEiKLnquSVvx**] [−**s** *server*] [−**J** *storage-node*] [−**c** *client-name*] [−**N** *name*] [−**e** *expiration*] [−**f** *dirfile*] [−**o** *save_operations*] [−**b** *pool*] [−**F** *file*] [−**I** *input_file*] [−**g** *group*] [−**k** *checkpoint_id*] [−**l** *level*] [−**t** *date*] [−**m** *masquerade*] [−**w** *browse_time*] [−**y** *retention_time*] [−**W** *width*] [*path* ...]

savepnpc

-s server -g group [−**BEiKLnquSVvx**] [−**J** *storage-node*] [−**c** *client-name*] [−**N** *name*] [−**e** *expiration*] [−**f** *dirfile*] [−**b** *pool*] [−**F** *file*] [−**I** *input_file*] [−**l** *level*] [−**t** *date*] [−**m** *masquerade*] [−**w** *browse_time*] [−**y** *retention_time*] [−**W** *width*] [*path* ...]

DESCRIPTION

save saves files, including directories or entire filesystems, to the NetWorker server (see **nsr**(1m)). The progress of a save can be monitored using the Java based **NetWorker Management Console** program or the **curses**(3X) based **nsrwatch**(1m) program for other terminal types.

The user of this command may retain root privileges if the command's modes are properly set as described in **nsr**(1m).

If no *path* arguments are specified on the command line or via the −**I** option, the current directory will be saved. **save** will save a directory by saving all the files and subdirectories it contains, but it will not cross mount points, or follow symbolic links. If the paths to be saved are mounted from a network file server, **save** instructs the user to run the save on the remote machine or use the **-L** option.

If the path argument is specified on the command line and one of the directories in the path is a symbolic link, the target path of the symbolic link will be saved, rather than the symbolic link path itself. Therefore, the target path of the symbolic link must be specified when using the recover (1m) command or the nwrecover (8) program to recover the files.

The directive files (see **nsr**(5)) encountered in each directory are read by default, and they contain special instructions directing how particular files are to be saved (i.e. compressed, skipped, etc.). These files are named '**.nsr**' on UNIX or 'nsr.dir' on Windows.

Each file in the subdirectory structures specified by the *path* arguments is encapsulated in a NetWorker save stream. This stream of data is sent to a receiving process (see **nsrd**(1m)) on the NetWorker server, which processes the data, adding entries to the on-line index (see **nsrindexd**(1m)) for each file in the stream, with the data finally ending up on a long term storage media (see **nsrmmd**(1m)). By default, these on-line index entries are stored in the "backup" index namespace.

Details about handling media are discussed in **nsrmm**(1m) and **nsr_device**(5).

savepnpc consists of the same command options as **save but requires the -g group to** run. Apart from running the actual save, it also performs the pre and post processing commands, if any. Prior to the actual save of the first saveset on a NetWorker client, **savepnpc** performs pre-processing commands if any exists in the /nsr/res/<grpname>.res file, and at the end of the save of the last save set on the client, the post-processing commands (if any) will be invoked. It is possible to setup multiple clients in a savegroup such that each client can run different pre and post commands. The <grpname>.res file resides on the client machine and is unique to that host. In the condition of failure to run the pre-processing commands, **savepnpc**

aborts itself. All results are logged in /nsr/logs/savepnpc.log file on the client. A timeout condition can be set by the user to indicate at which point in time the post-processing commands need to be run without waiting for all the save sets to be backed up. This timeout attribute resides in the /nsr/res/<grpname>.res file. The timeout should be specified in double quotes, in such a format that **nsr_getdate()** can understand (see **nsr_getdate**(3)). Also, abort precmd with group attribute exists in the /nsr/res/<grpname>.res file. This can be set to Yes or No. If set to Yes, the precmd will terminate if the particular savegrp is aborted. If it is set to No, the precmd will run to completion even after the abnormal exit of the savegrp session.

An example of /nsr/res/<grpname>.res can be described as:

```
type: savepnpc;
precmd: /bin/true;
pstcmd: /bin/true, "/bin/sleep 5";
timeout: "12:00pm";
abort precmd with group: No;
```

The **precmd** field can be manually modified to contain any number of commands that are needed to be run at the beginning of the save of the 1st save set. The **pstcmd** is to hold any commands that are needed to be run at the end of the save of the last save set. The post-processing commands are run after the save of the last save set or the timeout condition, whichever comes first. Note that on Windows Networker Clients, the shell should be set to "cmd.exe"and the Shell flag should be set to "/c", for running the precmd and postcmd. This will force the OS to close all the opened File Descriptors and other resources in a timely manner, after the execution of the commands. Also, for both the precmd and the pstcmd, on all Networker Clients, its best to redirect the output(stdout and stderr) to another file, to avoid unclosed File Descriptors, after the commands have completed executing.

An example of precmd and pstcmd for Windows Clients is shown below:
precmd: cmd.exe /c start_pre_cmd > pre_result.txt 2>&1
pstcmd: cmd.exe /c start_post_cmd > post_result.txt 2>&1

OPTIONS

−**b** *pool* Specifies a particular destination pool for the save.

−**c** *client-name*

Specifies the client name for starting the save session. This is useful on clients with multiple network interfaces, and multiple host names. It can be used to create multiple index databases for the same physical client with multiple network interfaces.

This does not specify the network interface to use. This is specified in the **server network interface** attribute of the client resource (see **nsr_client**(5)). This option can also be used on a cluster when performing manual saves, or in specifying a non-default **backup command** for scheduled saves. This option directs NetWorker to override the cluster path-ownership rules, saving the path argument(s) as belonging to *client-name* and making index entries in the index for *client-name* instead of using the name of the physical host or virtual host which owns the path, according to the cluster management software. Refer to **pathownerignore**(5) for more information about path-ownership rules.

−**e** *expiration*

Set the date (in **nsr_getdate**(3) format) when the saved data will expire. When a save set has an explicit expiration date, the save set remains both browsable and non-recyclable until it expires Thus, the explicitly provided expiration overrides the existing browse and retention times specified in the client policy and the browse and retention times get changed to the expiry time. The "-e

exp_time" option cannot be used in conjunction with "-w browse_time" or "-y reten_time". By default, no explicit expiration date is used and the client's longest browse and longest retention policy are used.

−**f** *dirfile*
The file from which to read prototype default directives (see **nsr**(5)). A *dirfile* of **-** causes the default directives to be read from standard input.

−**o** *save_operations*
Save Operations of the form "KEYWORD:TOKEN=STATE[;KEYWORD:TOKEN=STATE;...]".

The following form is used to configure VSS saves on Windows 2003. Examples:

"vss:*=off"

Turn off VSS.

"vss:Microsoft Exchange Writer=off"

Disable a writer.

"vss:C:=off"

Disable VSS for a drive.

Checkpoint restart related options:

"CHECKPOINT_BY_FILE:checkpoint_by_file=on"

Enables checkpointing after each file that is saved. Turning this option on introduces extra overhead and will slow down the save process. The default is to checkpoint after each directory.

"CHECKPOINT_BY_FILE:checkpoint_by_file=off"

Enables checkpointing after each directory. This is the default behaviour and it is not necessary to explicitly specify this option.

The following forms are specified by savegrp program during scheduled backups. Manual specifications of these forms are not supported.

Backup of renamed directories, in the form of:

"RENAMED_DIRECTORIES:index_lookup=on"

Enables support for the backup of renamed directories. The save program performs a lookup in the client file index to determine whether a directory has been renamed. If a directory has been renamed, all of the files and subdirectories under the directory will be backed up.

"RENAMED_DIRECTORIES:index_lookup=off"

Disables the backup of renamed directories.

Backup of non-ASCII save set names.

See **nsr_client (5)** and the Admin Guide for more information on the forms for non-ASCII names and how to enable or disable support for renamed directories.

−**g** *group*
This option is used by **savegrp**(1m) and **savefs**(1m) to denote the group of the save (see **nsr_client**(5) and **nsr_group**(5)) and is used by the NetWorker server to select the specific media pool.

−**i** Ignores any .nsr (UNIX) or nsr.dir (Windows) directive files as they are encountered in the subdirectory structures being saved.

−**J** *storage-node-name*
Specifies which host to use as the storage node for the backup (see nsr_storage_node(5)).

−**k** *checkpoint_id*
Checkpoint id for a corresponding checkpoint restart enabled save. This option is used by **savegrp**(1m) to enable checkpoint restart functionality for clients that have the 'Checkpoint enabled' option set to 'Enabled'. The **checkpoint_id** can contain only decimal digits. It should be set to 0 to start a new checkpoint restart save, or be equal to the **checkpoint_id** of a save that you wish to continue. For more details about how to obtain the **checkpoint_id** of a save, see the **mminfo**(1m) man page. See the -o option for more checkpoint restart related options.

−**l** *level* The level of the save. This option is used by **savegrp**(1m) and **savefs**(1m) to specify a particular level, for a scheduled save, to be stored in media database. This option is ignored by manual save command and backup is performed at level adhoc. The level information is not used by save to determine whether a file should be backed up during scheduled backup, but parameter on **-t** option is used instead.

−**m** *masquerade*
Specifies the tag to precede the summary line. This option is used by **savegrp**(1m) and **savefs**(1m) to aid in savegrp summary notifications. **savepnpc**(1m) also uses this tag to identify client operations on the savegrp's work list that should complete before **pstclntsave**(1m) will trigger its post-processing.

−**n** No save. Estimate the amount of data which will be generated by the save, but do not perform the actual save.

−**q** Quiet. Displays only summary information and error messages.

−**s** *server*
Specifies which machine to use as the NetWorker server.

−**t** *date* The date (in **nsr_getdate**(3) format) by which files must have been modified for them to be saved. This option is used by **savegrp**(1m) and **savefs**(1m) to perform scheduled saves by consulting with the media database to determine the appropriate time value based on the previous saves for the save set and the level of the scheduled save.

On Windows, file modification/change time refers to Last Written time, Creation time and the Archive file attribute of a file. All of these are used to determine whether a file needs to be backed up. If the Archive file attribute is set,

the file will always be backed up, since some older filesystems may not have the proper file creation time, unless NSR_AVOID_ARCHIVE environment variable is set (to a value other than "no").

−**u** Stop the save if an error occurs. The **save** program normally treats errors as warnings and continues to save the rest of the files in the backup. When this option is set, errors will cause **save** to exit and abort the save. This option is not recommended for general use, although it can be useful when a group of files needs to be backed up as a set.

−**v** Verbose. Causes the **save** program to provide great detail about the **save** as it proceeds.

−**y** *retention*
Sets the date (in **nsr_getdate**(3) format) when the saved data will become recyclable. The special value **forever** is used to indicate that a volume that never expires (i.e. an archive volume) must be used. By default, the server determines this date for the save set based on the retention policies in effect. This option allows overriding the existing policies on a save by save basis.

−**w** *browse_time*
Sets the date (in **nsr_getdate**(3) format) after which this save set will no longer be browsable. By default, the server determines the browse date for the save set based on the browse policies in effect. This option allows overriding the existing policies on a save by save basis.

−**x** Cross mount points. This option is only applicable to the manual save command. When the **-x** option is specified with a (save set) path, the mount point under this path is crossed and files in the mounted file system are backed up. When the **-x** option is not specified, only the local files and directories of the path are backed up.

This option has no effect and is ignored when a mount point (or file/subdirectory path under it) is specified as the (local) path to be backed up for the manual save command, which generally requires the −**L** option.

For example, if /tmp_mnt/pumbaa is a mount point for pumbaa:/space, the following three commands behave differently with the specification of -x option:

save -x /tmp_mnt follows mount points, backs up the local files and directories of /tmp_mnt along with files and directories of /tmp_mnt/pumbaa

save /tmp_mnt does not cross mount points, only local files and directories of /tmp_mnt are backed up

save [-x] /tmp_mnt/pumbaa[/...] when mount point is specified as the (save set) path, "-x" option is ignored and an error message is displayed to indicate "-L" option is required on most platforms

For Windows, mount point refers to an alternative path for a mounted volume on the same machine. For example, H:\tmp_mnt\d_drive may be defined as a mount point to the local volume (or drive) D:.

−**B** Force save of all connecting directory information from root (''/'') down to the point of invocation.

−**E** Estimate the amount of data which will be generated by the save, then perform the actual save. Note that the estimate is generated from the inode information; thus, the data is only read once.

−**F** *file* Only save files whose change time is newer than the file modification date of *file*.

For Windows, see *-t* option for more information on ***file change time*.**

−**I** *input_file*
In addition to taking the paths to save from the command line, read paths to save from the named file. The paths must be listed one per line. If no paths are specified on the command line, then only those paths specified in the file will be saved.

−**K** Does not build connecting directory index entries.

−**L** Local. Saves will be performed from the local NetWorker client, even when files are from a network file server. To recover these files, run **recover**(1m) with the −**c** *client* arguments, where *client* is the name of the NetWorker client that did the **save**.

−**LL** In addition to treating the backup as a local backup, causes an extra line to be printed at the end of the completion output of the form ''complete savetime=number'', where number is the savetime of the save set created by this backup. This option is meant to be used by the **savegrp**(1m) command in performing automatic cloning.

−**N** *name*
The symbolic name of this save set. By default, the most common prefix of the *path* arguments is used as the save set name. If the -N option is used when saving any of the SYSTEM save sets (SYSTEM STATE, SYSTEM FILES, and SYSTEM DB), the path must also be specified and must match the name value assigned with the -N option.

−**S** Allows only save set recovery. This performs the save without creating any index entries. This means that the save set will not be browsable, although save set recovery may be used to recover the data.

−**V** Prevent the OFC mechanism from creating a point-in-time copy of the source volume. (Included for compatibility with NT NetWorker servers.)

−**W** *width*
The width used when formatting the summary information output. Valid values for width are integer values from 1 to 10000. If the supplied width is too small for the summary to fit in, the width will be silently adjusted upwards as necessary. If the supplied width is larger than the minimum needed, then spaces will be used to pad the summary to the correct width. Note that if no -W argument is supplied then there is no fixed width used, and the summary simply expands to whatever minimum width is necessary.

SEE ALSO **curses**(3X), **nsr_getdate**(3), **nsr**(5), **nsr**(1m), **nsr_client**(5), **nsr_device**(5), **nsr_group**(5), **nsr_service**(5), **nsrd**(1m), **nsrim**(1m), **nsrindexd**(1m), **nsrmm**(1m), **nsrmmd**(1m), **nsrwatch**(1m), **recover**(1m), **savefs**(1m), **savegrp**(1m), **pathownerignore**(5)

DIAGNOSTICS

Exit Codes:

0 Normal exit. This means that a save set was correctly created on the server. Messages about individual file backup failures are *warnings*, and do not cause abnormal exit.

<>**0** Abnormal exit. A save set was not correctly created on the server.

Messages:

host: *saveset* **level=***level*, *size time count* **files.**
This message (with the appropriate client host name, saveset name, level, total save set size, elapsed time, and file count) is printed whenever **save** is run by **savegrp**(1m) and exits normally.

host: *filename*: *warning*
Messages of this form are warnings about difficulties backing up individual files. Such messages do not normally cause the **save** to fail, and therefore may appear in the **save** output found in the Successful section of the ''Savegroup Completion'' message.

path: **File index could not be obtained due to** <*reason*>**.**
Contents of this directory may not be properly backed up.

There was an error in retrieving an on-line file index record for the path with renamed directory support. Run nsrck (1m) and turn off the "Backup renamed directories" attribute in Client resource (see **nsr_client**(5)) to re-run the group if the problem persists.

NAME savefs − save filesystem to a NetWorker server

SYNOPSIS

savefs
[*options*] *filesystem*

savefs
−p [*options*] [*filesystem ...*]
[**−BEFnpqRv**] [**−s** *server*] [**−N** *name*] [**−g** *group*] [**−c** *client*] [**−l** *level* | **−C** *schedule*] [**−e** *expiration*] [**−w** *browse*] [**−y** *retention*] [**−f** *filename*] [**−o** *save_operations*] [**−W** *width*] [**−t** *date*]

DESCRIPTION

The **savefs** command saves a filesystem (using **save**(1m)) to a NetWorker server. Mount points are not crossed, and symbolic links are not followed. **NOTE**: running **savefs** directly is not recommended; use **savegrp**(1m) instead.

A *level*-based system (similar to **dump**(1m)) is used to save only those files which have been modified since some previous save (a *partial* save).

The **nsr_schedule**(5) for the local NetWorker client is examined to determine the proper level of save for the current date.

The set of files saved depends on when, and at what level, previous saves have been performed, in addition to the effects of the default directives (see **nsr_directive**(5)), and the various directive files (see **nsr**(5)) which are encountered while processing the filesystem.

FILESYSTEM PROBES

The **savefs** command may also be used to probe a client for its filesystems and recent save times. When probing, **savefs** does not save data, but instead produces a machine-parsable report describing the layout of the client's filesystems. When used with the **−p** probe option, the local NetWorker client's **nsr_client**(5) resources are examined, and the filesystems listed in the *save set* attribute are probed (if no filesystems are listed on the command line). If the save set list consists of the keyword *All*, then the **/etc/fstab** file (**/etc/vfstab** on Solaris, **/etc/mnttab** on SCO, and a kernel table on AIX) are examined to determine which filesystems should be saved, making sure to save only local, mounted filesystems.

Note that metadevices within the Sun Solaris Online DiskSuite and Logical Volumes within the HP-UX Logical Volume Manager are treated like independent disks. This approach allows each to be saved in its own session, assuming sufficient parallelism.

Care should be taken when the **NSR client** resource explicitly lists the save sets, for two primary reasons. First, this list must be manually updated when new filesystems are added which need saving. Second, since **savefs** only stops at the end of a path or a mount point, if you list two save sets in the same filesystem, and one is a subdirectory of the other, the subdirectory will be saved twice.

Filesystem arguments can be specified to limit the filesystem saves to only those specified, but the specified filesystems must appear on some **Save Set** list for this client (see the **−F** option).

Probes are also useful when testing how NetWorker will behave in a clustered environment. In this setup ownership of shared filesystems must be determined, and performing a probe with the verbose option set allows one to examine the default ownership rules. Refer to **pathownerignore**(5) for a description of path-ownership rules.

OPTIONS

−B Force save of all connecting directory information from root (''/'') down to the point of invocation. This option is used by **savegrp**(1m), for example, when saving the server's bootstrap information.

−**c** *client*
The name of the client whose filesystem needs to be saved. This option is especially needed in a cluster environment where a physical host can represent its own hostname as well as hostnames of any virtual (also known as "logical") hosts that exist in this physical host. Without this option, the hostname of the physical host is assumed by default. This option is required if a filesystem that belongs to any of the virtual hosts needs to be saved.

−**C** *schedule*
The name of the schedule (see **nsr_schedule**(5)) to use when automatically determining the save level. If this option is not specified, **savefs** uses the schedule named by the NSR client resource for the specified filesystem.

−**e** *expiration*
Set the date (in **nsr_getdate**(3) format) when the saved data will expire. When a save set has an explicit expiration date, the save set remains both browsable and non-recyclable until it expires. After it expires and it has passed its browse time, its state will become non-browsable. If it has expired and it has passed its retention time, the save set will become recyclable. The special value **forever** is used to indicate that a volume that never expires (i.e. an archive volume) must be used. By default, no explicit expiration date is used.

−**w** *browse*
Sets the date (in **nsr_getdate**(3) format) after which this save set will no longer be browsable. By default, the server determines the browse date for the save set based on the browse policies in effect. This option allows overriding the existing policies on a save by save basis.

−**y** *retention*
Sets the date (in **nsr_getdate**(3) format) when the saved data will become recyclable. By default, the server determines this date for the save set based on the retention policies in effect.

−**E** Estimate. Before saving any data, browse the filesystem trees to be saved and accurately estimate the amount of data that will be generated. Without this flag, the estimate size is zero. This flag consumes an amount of time proportional to the number of files in each filesystem. This is because the entire directory is browsed before any saving begins and browsed again when actually saving the directory, but the file data is only read from the disk the last time. In many cases, the overhead for using this flag is small and is well-justified.

−**f** *filename*
The file from which application specific modules (or ASMs) should take their directives (see **nsr**(5)). By default, these are taken from the **NSR directive** resource named by the *directive* attribute in the **NSR client** resource for each client (see **nsr_directive**(5)).

−**F** Force. Save every argument like a filesystem, even if it is not listed in **fstab(5)** or **nsr_client(5).**

−**g** *group*
Restrict the scope of the client to a particular group. If this option is not specified, save sets from all instances of the NSR client resource for this client will be used, regardless of the group. This value is also passed on to **save**(1m), which uses it to select a specific media pool.

−**l** *level* The level of save (see **nsr_schedule**(5)) to perform on ecah client.

−**n** No save. Accurately estimates the amount of data that would be generated (as described for −**E**, but doesn't save any data.

−**N** *name*
The symbolic name this set of saves is to be known by. By default, the first *filesystem* argument is used as the name.

−**p** List the name of the filesystems, the level of save that would be performed, and the time since which files must have been modified to be saved, but don't actually do the save. This information is gleaned from the **/etc/fstab** file (or another operating system specific file, as described above) and the **nsr_schedule**(5).

−**q** Quiet. Display only summary information and error messages.

−**qq** Really quiet. Display only error messages.

−**R** Cause **savefs** to report on its success or failure, by echoing a simple "succeeded" or "failed" message. This is used by **savegrp**(1m) when it is running **savefs**.

−**s** *server*
Specifies which machine to use as the NetWorker server. See **nsr(1m)** for the algorithm NetWorker uses to choose a server when none is specified.

−**t** *date* The date (in **nsr_getdate**(3) format) from which to base schedule level calculations. If not specified, the current time is used.

−**o** *save_operations*
Save Operations of the form KEYWORD:TOKEN=STATE. It is used to configure VSS saves on Windows 2003. Examples:

"vss:*=off" Turn off VSS.

"vss:Microsoft Exchange Writer=off" Disable a writer.

"vss:C:=off" Disable VSS for a drive.

Please see the Admin Guide for more details.

−**v** Verbose.
Causes lots of debugging style output.
This option is also used by
savegrp(1m)
when it is probing for the capabilities of the client's
savefs,
for supporting multiple versions.

−**W** *width*
The width used when formatting output or notification messages.
By default,
this is 80.

RESOURCE TYPES

NSR client
These resources specify the client's save sets, default schedule, and directives to use when saving them.

NSR directive
A resource of this type is named by the *directive* attribute in each **NSR client** resource. These are the directives used for the save sets specified in the associated **NSR client** resource.

NSR schedule
A resource of this type is named by the *schedule* attribute in each **NSR client** resource. This is the schedule used for the save sets specified in the associated

NSR client resource.

FILES

/etc/fstab
If **All** is specified in the *save set* attribute for a **NSR client** resource, then the list of local filesystems is taken from this file.

/etc/vfstab
Solaris only. The same as /etc/fstab on other operating systems.

/etc/mnttab
SCO only. The same as /etc/fstab on other operating systems.

SEE ALSO

nsr_getdate(3), **fstab**(5), **mnttab**(F) (SCO only), **vfstab**(5) (Solaris only), **nsr**(5), **nsr_service**(5), **nsr_schedule**(5), **dump**(1m), **nsr**(1m), **nsrd**(1m), **nsrindexd**(1m), **nsrmmd**(1m), **recover**(1m), **save**(1m), **savegrp**(1m), **pathownerignore**(5)

DIAGNOSTICS

Exit Codes:

0 Normal exit.

255 Abnormal exit.

NAME savegrp − start a group of NetWorker clients saving their filesystems

SYNOPSIS **savegrp**

[*options*] [−**R** | −**G**] [**groupname**] [−**EIOFXgmnpv**] [−**l** *level* | −**C** *schedule*] [−**N** *parallelism*] [−**e** *expiration*] [−**w** *browse*] [−**y** *retention*] [−**t** *date*] [−**r** *retries*] [−**P** *printer*] [−**W** *width*] [−**b** *backup snapshot*] [−**c** *client* [−**c** *client* ...]]

DESCRIPTION The **savegrp** command runs a group of NetWorker clients through the process of saving their filesystems (using **save**(1m)). The group of clients is selected by naming a NetWorker group (see **nsr_group**(5)), from which individual clients can be selected by using one or more −**c** options. If no group name is specified, the NetWorker group **Default** is used. If a NetWorker group is named, clients whose **nsr_client**(5) resources specify the named group in their *group* attribute will be saved. If an explicit client list is also specified, **savegrp** will only back up those clients, with respect to the named group. The **savegrp** command will automatically make a clone of the newly saved data when the appropriate attributes are set on the **NSR group** resource (see below).

The **savegrp** command is normally run automatically by **nsrd**(1m), as specified by each group's **nsr_group**(5) resource.

The **savegrp** command will set up an RPC connection to **nsrjobd**(1m) to request execution of a **save**(1m) job on each client for each filesystem listed in the **nsr_client**(5) resource *save set* attribute. If a save set of *All* is specified for a client, **savegrp** will request from the client a list of filesystems to be saved (this is called the *probe* operation). The probe expands *All* into a list by looking for filesystems that are both local and automatically mounted on that client machine (e.g. NFS mount points and filesystems mounted manually are generally ignored). The exact determination of which filesystems to save varies between different operating systems. See **savefs**(1m) for additional details on the probe operation. To see which filesystems a client saves, run a savegrp preview, *savegrp -c client -p* (assuming the client is in the Default group). Each filesystem saved is called a *save set*.

For NDMP clients, savegrp will run backup command **nsrndmp_save**(1m) given in the client resource on the NetWorker server. If the command line options of nsrndmp_save has -I hostname, then savegrp will run **nsrndmp_save**(1m) command on the given hostname. For more command line options and details of nsrndmp_save, see **nsrndmp_save**(1m).

The **savegrp** command attempts to keep multiple clients busy by queuing all the save jobs immediately for the client's queue maintained by **nsrjobd**(1m).

The **parallelism** attribute in the **nsr_service**(5) resource is the maximum number of save sets that can run simultaneously. Modifications to this parameter will take effect as save sets complete − if the value is reduced, no new save set will be started until the number of active save sets running drops below the new value.

When the savegrp is started from the command line, it does not automatically pick up the level attribute specified in the group resource. However, when -l level otion is explicitly specified, the savegrp command performs the requested level backup.

When all of the save sets are completed on a client for the group, the client's index on the NetWorker server is saved. If the client is scheduled for level=incr index backup is done at "-l 9"(level 9) with "-f -". The default directive causes a backup of all files with extension "rec" under client's index directory (/nsr/index/<clientname>/db6 on a UNIX machine) to happen. The manpage for **save**(1m) has details of "-l" and "-f" options. If the NetWorker server is one of the machines being saved, NetWorker bootstrap is saved after all the other clients are completely done. When the server's

bootstrap is saved, the bootstrap save set information is printed to the default printer (or another specified printer). If **savegrp** detects that the NetWorker server is not listed in any active group (a group with its autostart attribute set), then the server's bootstrap is saved with every group.

If the **savegrp** command detects other active invocations of the same group, then it will exit with an error message. If two different NetWorker groups are running simultaneously, they each will run up to respective group's *parallelism* and not overrun the server and client *parallelism* ; as the *nsrjobd (1m)* will control the *parallelism* and hold these backup jobs in a queue before starting. The NetWorker server also controls the device *parallelism* of these sessions.

The progress of the actively saving clients can be monitored using the Java based **NetWorker Management Console** or the **curses**(3X) based **nsrwatch**(1m) program. The **NetWorker Management Console** or **nsradmin**(1m) browser may also be used to examine the *completion* status and *work list* of each *NSR group* resource, and allow you to track the progress of each savegrp. These two attributes allow you to track the progress of each savegrp. See **nsr_group**(5) for more details.

When **savegrp** starts, it sends a **NSR notification** (see **nsr_notification**(5)) with an event of **savegrp** and priority of **info** to the NSR notification system. This event is normally logged in the *messages* attribute of the **nsr_service**(5) resource, and in the log file specified in the *Log default* NSR notification resource.

If the **NSR group** resource has the **clones** attribute enabled, the save sets are automatically cloned when all the save sets have finished. The client save sets and their indexes are cloned before the bootstrap save set is generated so the bootstrap information can track both the original set of save sets and their clones. The bootstrap save set is also cloned. Clones will be sent to the pool named in the **clone pool** attribute. Changing the values of these attributes while **savegrp** is running has no effect; they must be set before **savegrp** starts. The **nsrclone**(1m) command is used to clone the save sets. **savegrp** uses a heuristic to determine which save sets were generated as part of the group; because of this, it may occasionally clone more save sets than expected, if a client has its filesystems separated into multiple groups that run at the same time. Note that at least two enabled devices are required to clone save sets and/or atleast two enabled ndmp devices are required to clone ndmp save sets.

savegrp, when started for snapshot enabled groups, creates snapshots for each of the clients configured in the group resource (see nsr_group.5). If any of the save sets, for clients configured in this group resource, is non-snapshot capable, then savegrp will report a failure while trying to create the snapshot. If any of the client resources configured with this group has the Keyword *All* (see nsr_client.5), then the non-snapshot capable file systems on the client node will be ignored and no error message is generated for their failures.

NOTE: This option is available with EMC PowerSnap Module only.

When the save sets are all complete and cloned (if cloning is enabled), a **NSR notification** with an event of **savegrp** and priority of **notice** is sent to the NSR notification system. This is generally set up to cause e-mail to be sent to the **root** user specifying the list of clients who failed (if any), and all the output collected from all clients. The format and common error messages included in the savegrp notification are explained in the **SAVEGRP COMPLETION NOTIFICATION MESSAGE** section.

OPTIONS

−E Causes **save**(1m) on each client to estimate the amount of data which will be generated by each save set before performing it. This will result in the filesystem trees being walked twice − once to generate a estimate of how much data would be generated, and again to generate a save stream to the NetWorker

server. Note that the data is only read from the disk on the final filesystem walk, as the estimate is performed by using inode information.

−I Disables the saving of each client's index.

−O Saves only each client's index (the bootstrap is also saved).

−g Skip probing for probe based groups.

−m Disable monitor status reporting, including all **NSR notification** actions. When this option is selected, the progress or completion of a save operation is not reported. The notification of bootstrap information is not affected by the -m option.

−n No save. This causes **save** to perform an estimate as described for **−E**, but not to perform any actual saves. This option also sets the **−m** option.

−p Runs the probe step on each client so you can see which filesystem would be saved and at what level, but does not actually save any data. This option also sets **−m. savegrp** discovers the level to be used as it progresses. For this reason, the output generated by the -p option may show different levels for save sets at different points. This is the expected behavior, and can be useful for debugging. The actual level the savegrp uses is shown the last time each save set is displayed in the output. The media pool the save set would be directed to is also listed in the preview output.

−v Verbose. Prints extra information about what **savegrp** is doing. The **−q** flag is also not passed to the **save** command.

−G Run the group; apply no restart semantics. This is the default mode of operation; the option is provided for compatibility with other versions of **savegrp**.

−R Restart. This option is used to restart a group that was stopped or if savesets failed and they need to be retried. The *restart window* attribute of the group is used to determine if it is too late to be restarted. If the window has elapsed the restart is converted into fresh start.

−l *level* The level of save (see **nsr_schedule**(5)) to perform on each client. This overrides the save level which **savegrp** would normally automatically determine. **−l** and **−C** cannot be specified together.

−C *schedule*
The name of the **NSR schedule** (see **nsr_schedule**(5)) to be used in the automatic save level selection process which **savegrp** normally performs. This overrides the save schedule which **savegrp** would normally use for a given client. **−l** and **−C** cannot be specified together.

−e *expiration*
Set the date (in **nsr_getdate**(3) format) when the saved data will expire. When a save set has an explicit expiration date, the save set remains both browsable and non-recyclable until it expires. After it expires and it has passed its browse time, it will become non-browsable. If it has both expired and passed its retention time, the save set will become recyclable. The special value **forever** is used to indicate that a volume that never expires (i.e. an archive volume) must be used. By default, no explicit expiration date is used. If it involves a saveset being backed up with 'level' as 'Full', the Full saveset will become non-browsable and recyclable after the specified expiration time. If this saveset has dependents with longer browse and retention periods, the dependent savesets will not be browsable after this saveset's expiration date.

−w *browse*
Sets the date (in **nsr_getdate**(3) format) after which the saved data will no longer be browsable. By default, the server determines the browse date for the

save set based on the browse policies in effect. This option allows overriding the existing policies on a save by save basis. If it involves a saveset being backed up with 'level' as 'Full', the Full saveset will become non-browsable after the specified browse time. If this saveset has dependents with longer browse periods, the dependent savesets will not be browsable after this saveset's browse date.

−**y** *retention*
Sets the date (in **nsr_getdate**(3) format) when the saved data will become recyclable. The special value **forever** is used to indicate that a volume that never expires (i.e. an archive volume) must be used. By default, the server determines this date for the save set based on the retention policies in effect. This option allows overriding the existing policies on a save by save basis. If it involves a saveset being backed up with 'level' as 'Full', the Full saveset will become recyclable after the specified retention time. If this saveset has dependents with longer browse and retention periods, the dependent savesets will not be browsable after this saveset's retention date.

−**t** *date* The time to use instead of the current time for determining which level to use for this savegrp (in **nsr_getdate**(3) format). By default, the current time is used.

−**F** Automatically perform a full level backup if save set consolidation fails. This option is ignored if the backup level is not "c".

−**X** Automatically remove the level 1 save set after save set consolidation builds a full level save set. This option is ignored if the backup level is not "c". It is also ignored if the backup level is "c" but the save set consolidation process fails.

−**r** *retries*
The number of times failed clients should be retried before **savegrp** gives up and declares them failed. The default is taken from the group resource. Abandoned saves are not retried, because they may eventually complete. Retries are not attempted if −**p** is specified.

−**P** *printer*
The printer which **savegrp** should use for printing bootstrap information.

−**W** *width*
The width used when formatting output or notification messages. By default, this is 80.

group Specifies the NetWorker group of clients that should be started, rather than the default NSR group (which has the **name** attribute of **default**). See **nsr_group**(5) for more details.

−**b** *backup snapshot*
This option should be used only with snapshot groups. This option will be ignored if used with non-snapshot groups. When passed for a snapshot group, this option will configure the specified snapshots to be backed up to tertiary storage.

−**c** *client*
The name of a client on which to save filesystems. There can be multiple −**c** *client* specifications. When −**c** options are specified, only the named clients from the specified group (which is "Default" if no group is specified) will be run.

−**N** *parallelism*
The *parallelism* value overrides any other parallelism considerations that

savegrp may use to avoid over-utilizing the system's resources.

RESOURCE TYPES

NSR Use the **parallelism** attribute for the maximum number of saves to start simultaneously.

NSR group The attribute *work list* contains values in groups of 3, specifying the client name, level of save, and path to save, for each save set not yet completed. The attribute *completion* contains values in groups of 4, specifying the client name, path saved, status, and the output, for each save set completed. The *success threshold* attribute contains the threshold to determine the success of all savesets within the group.

NSR schedule
Used by the **savegrp** command with each client's **nsr_client**(5) resource to determine which level of save to perform for each specified save set.

NSR client Each client resource names the groups it should be saved by, the names of the save sets which should be saved, the name of the schedule to use (see **nsr_schedule**(5)) and the name of the directives to use (see **nsr_directive**(5)).

NSR notification
Three kinds of notices are sent to the NSR notification system, all with the **event** attribute of *savegrp*. While a savegrp is in progress, status notices are sent with the **priority** of *info*. At the completion of a savegrp, a notice is sent containing the collected output of all saves and the name of clients which had a failed save (if any). This notice will have an **event** type of *savegrp* and a **priority** of *notice*. If **savegrp** is interrupted, a notice stating the group was terminated, with an **event** type of *savegrp* and a **priority** of *alert*, will be sent. These last two will typically result in the notice being encapsulated in a mail message to **root**.

SAVEGROUP COMPLETION NOTIFICATION MESSAGE

The savegroup completion notification message contains 6 parts: the Header, the Never Started Save Sets, the Unsuccessful Save Sets, the Successful Save Sets, the Previously Successful Save Sets and the Cloned Save Sets. Each client in the group will be listed in at least one of the sections categories (multiple sections if some save sets are in one category, and other save sets in another category). The clients are listed in alphanumeric order, with the server listed last.

The header shows the name of the group and lists which clients failed, or were unresolved, disabled, or successful (with warnings). If the group was aborted, the header would include an indicator of this as well. The header also shows the time the group was started (or restarted, if the **−R** option was used), and the time the savegrp completed. The failed clients list in the header shows only those clients for which saves were attempted, not those for which saves never started.

The Never Started Save Sets section is optional and only included if there are some save sets in the group that were never started. This section can be seen when a savegrp is aborted, either by killing the master savegrp daemon or by selecting the *Stop* function in the Monitoring menu of **NetWorker Management Console's** Administration window. Each entry listed in this section shows either the client and save set that was never started or *All* if no save sets were saved for that client. No other error messages should appear in this section.

The Unsuccessful Save Sets section shows all of the saves that were attempted but failed. This section will only be present if at least one save set failed. There are many reasons for a save to fail. The most common are listed below, but more reasons will be listed in the future. It is important to determine why a save failed so that the administrator can quickly determine the cause and fix it.

Each entry in the Unsuccessful Save Sets section lists the client and save set that failed, along with one or more lines of error and information messages. Each client is separated by a blank line, and all the failed save sets for a client are listed together. Typical error or information messages are listed at the end of this section (without the client:saveset prefix) with the necessary action(s) to take to correct the problem.

Each entry in the Successful Save Sets section lists the client and save set that succeeded, along with level of the save, the amount of data saved, the time to run the save set, and the number of files saved. Each entry may also be preceded by one or more warning or informational messages, the most common of which are listed below. These warning or informational messages are usually (but not always) prefixed by ''* ''. A save set's output may include warnings, but these do not necessarily mean the save set was unsuccessful. The *success threshold* attribute is used to determine if the warning(s) effect whether the saveset is reported as successful or failed. See **nsr_group**(1m) for the definitions of *success threshold* and its effect on reporting success/failure of save sets.

Also see **mminfo**(1m) for the definitions of successful and unsuccessful save sets.

The Cloned Save Sets section refers to the save sets cloned, and not the clients that originated those save sets. The output shown in this section is the output of the **nsrclone** command. See the **nsrclone**(1m) man page for information on the output of **nsrclone**.

The Previously Successful Save Sets section is optional and is included only if the group was restarted and there were some savesets completed in previous runs of the savegroup. This section is identical to Successful Save Sets section.

The following is a list of common informational, warning, and error messages found in the completion notification. This list is not complete. The messages you see may vary slightly from those shown here due to differences in the operating system vendor-supplied error messages. Since many messages include client or server names, it is most efficient to look for a keyword in the error message. The messages are listed below in alphabetical order by the first non-variable word in the message. (Note: initial words like "save", "asm", and "savefs" may or may not vary, and initial pathnames are always assumed to vary).

aborted

This informational message only occurs when you abort a running savegrp, generally by selecting *Stop* from the Monitoring menu of **NetWorker Management Console's** Administration window. It means that the specified save set had started saving, but had not completed when the savegrp was aborted. The session (in the Monitoring display of **NetWorker Management Console's** Administration window) for this save set may not disappear immediately, especially if **savegrp**'s attempt to kill the save session fails. The save set will be retried if and when you *Restart* the savegrp (e.g. from the Groups tab of the Monitoring display).

Access violation from *client - insecure port N*

This message, generated by the **save** command on client, means that **save** is not setuid root. Make sure that the **save** command on the client is owned by root and has its setuid bit set. If **save** is on an NFS mounted filesystem, make sure the filesystem was not mounted on that client using the "-nosuid" option.

Access violation − unknown host: *client*

This message is caused when then the client's hostname and IP address are not correctly listed in one or more of /etc/hosts, NIS or DNS on the server. You must change the appropriate host table (depending on which one(s) are in use on your server) to list the client's name as it is known to NetWorker (client's primary name), or you must add the name listed at the end of the error

message to the *aliases* attribute of the client's Client resource(s).

asm: cannot open path: I/O error

This message generally means that there are bad blocks on the disk(s) containing the specified file or directory. You should immediately run a filesystem check on the named client filesystem and check your client's system error log. If there are bad blocks, repair them if possible, or move the filesystem to a different disk.

asm: cannot stat path: Stale NFS file handle

asm: cannot stat path: Missing file or filesystem

These informational messages (or variants of them for other operating systems) mean that the when **save** attempted to test the named directory (to determine if it was a different filesystem from the one currently being saved), the filesystem was NFS mounted, but the mount point was bad. While this message does not affect the saved data, it does mean you have a network or NFS problem between the specified client and one or more of its fileservers. You may need to remount filesystems on the client, or perhaps reboot it to correct the problem.

/path/nsrexecd: Can't make pipe

/path/nsrexecd: Can't fork

fork: No more processes

The specified client-side resource has been exceeded. There are too many other services running on the client while savegrp is running. Inspect the client and determine why it has run out of resources. The client may need to be rebooted. You should also consider re-scheduling any jobs automatically started on the client (e.g. via **cron**(1m)) that run while savegrp is running.

asm: chdir failed path: Permission denied

This message means that while backing up the specified save set, **save** was unable to enter the named directory. This may mean that **save** is not setuid root on the specified client, or that the directory is an NFS mount point for which root is not allowed access. Check the permissions on **save** on the specified client (using **ls**(1)) and make sure that **save** is owned by root and that the setuid bit is set.

connect to address AA.BB.CC.DD: *message*

Trying AA.BB.CC.DD...

These informational messages are displayed only when the **−v** option is used. They mean that the connection to the client failed on the address specified in the first line of the message. If the client has more than one IP address, savegrp has attempted the address listed in the second line. Subsequent lines of the completion mail show if this second address succeeded. You may want to check and change your network routing tables to avoid getting these messages.

Connection refused

This means the client machine is up, but it is not accepting new network connections for **nsrexecd** (or **rshd**). This could mean the client was in the process of booting when the savegrp attempted to connect, or that the client had exceeded some resource limit and was not accepting any new connections. You should attempt to log into the client and verify that it is accepting remote connections. If the client is a non-UNIX machine, you may need to start the NetWorker client there. Refer to the NetWorker Installation Guide for more information.

Connection timed out

This usually means the client has crashed or is hung. Make sure the client has

rebooted, and that **nsrexecd** is running on it (if you are using **nsrexecd**). If the client is a non-UNIX machine, you may need to ensure that the network protocols are loaded, and that the NetWorker client is running on that machine. Refer to the NetWorker Installation Guide for more information.

asm: external ASM 'asm2' exited with code 1

This message generally accompanies another message reporting a specific problem while saving a file or directory on the named save set. The backup will attempt to continue and attempt to save other data. Generally, the backup will not be listed in the failed save sets section of the completion mail if any files on the save set are saved successfully, even if it only saves the top directory of the save set.

save: *path file size changed!*

This informational message is often generated when NetWorker backs up log files. It may also occur for other files. For files that you expect to grow while **savegrp** is running, you can use a directive specifying that the **logasm**(1m) should be used to back up the file. See also **nsr**(5) and **nsr_directive**(5).

asm: getwd failed

This message means that, while backing up the specified save set, an attempt to determine the current directory's name failed. This occurs on clients, generally running older versions of the NetWorker client, on which the **getwd**(3) library call is broken. You may want to contact EMC Tech Support to find out if there is a patch available for your client platform to work around this vendor-specific bug, or contact your operating system vendor to see if a more recent OS version addresses this problem.

Group *groupname aborted, savegroup is already running*

This message is only delivered by itself. It occurs when the named group has already been started or restarted (eg after a reboot, or when requested via the Groups tab of **NetWorker Management Console's** Administration window), either automatically by **nsrd**(1m) or manually, from the command line. You can use **ps**(1) to find out the process id of a running savegrp. The existence of a running group is determined by looking for a file named **/nsr/tmp/sg.***group* which (if it exists and is locked) means a savegrp is running.

Aborting inactive job (id) *client:saveset*

The client has not sent any data to the server for the specified inactivity timeout. **savegrp** will request **nsrjob**(1m) to terminate the backup in progress so that the hung client will not impede other backups or cloning operations.

has been inactive for *N minutes since time.*
client:saveset is being abandoned by savegrp.

A backup of the specified save set started, but after *N* minutes of no activity, **savegrp** gave up on the save set. Generally, this means that the client is hung waiting for an NFS partition. Unfortunately, NetWorker (or any other program) has no way of reliably telling if an NFS partition will hang until after it tries to access the partition. When the partition comes back on line, the save will complete, despite that **savegrp** abandoned it. You should check the client, since you sometimes may need to reboot the client to unhang NFS partitions. Non-UNIX clients also hang for other reasons such as defects in the operating system implementation of their network protocols.

Host is unreachable

The NetWorker server cannot make TCP/IP connections to the client. This generally means the network itself is not configured correctly; most commonly, one or more gateways or routers are down, or the network routes were not set

up correctly. You should verify that the server can connect to the client and, if not, check and reconfigure your routers, gateways, or routing tables (if necessary).

Login incorrect

This message is generated when the *remote user* attribute for the client is not set to a valid login on the client. Verify that the *remote user* attribute for the client is set to the correct login name. You may see this message even when running **nsrexecd** if **nsrexecd** has not been started (or was killed) on the client.

asm: missing hard links not found:

This message is generated when a backed-up file had one or more hard links that were not found. The message is followed by a list of one or more file names which were backed up minus some links. The message means that the files were either created (with multiple hard links) while the backup was occurring, so some of the links were missed due to the order of filesystem tree walking, or the file (or some links) were removed while the backup was occurring. Only those links that were found can be recovered; additional links will have been lost. One can do an additional incremental backup of the affected filesystem if a consistent state for the affected file is essential.

lost connection to server, exiting
save: network error, server may be down

The backup of the named filesystem was begun, but the connection to the NetWorker server closed part way through. This typically means that the server machine rebooted, one or more NetWorker server daemon processes were killed by the system administrator or by the system itself (e.g. due to overwriting the binary or a disk error in swap space), or there was some transport problem that caused the network connection to be dropped by the operating system. Restart the save at a later time.

No save sets with this name were found in the media database;
performing a full backup

This informational message is added by **savegrp** to any save set that is saved at the level *full* instead of the level found in the client's schedule. Due to timing problems, you can occasionally see this message when the clocks on the client and server are out of sync, or when **savegrp** starts before midnight and ends after midnight. You may also get spurious messages of this type from some versions of NetWorker client software backing up a NetWare BINDERY, which ignore the schedule and perform a full save. In both these cases, the client re-checks the level and overrides the server's requested level.

No more processes

See "Can't make pipe" message information.

No 'NSR client' resource for client *clienthostname*
savefs: cannot retrieve client resources

This pair of messages occurs if the the client's hostname changed (in /etc/hosts, NIS or DNS). You may also have deleted the client's *Client* resource while savegrp was running. In the former case, you will need to add the client's new name to the *aliases* attribute of the client (this is a hidden attribute) using **nsradmin**(1m) (selecting the *Hidden* display option) or **NetWorker Management Console** (double clicking on the client entry from the Administration window's *Configuration* display). In the latter case, no additional action is required if this deletion was intentional (the next run of **savegrp** will not attempt to save the client). If it was accidental, and you did not want to delete the client, you should add the client back again and add the client back into the appropriate group(s). The next time **savegrp** runs, it will back up the client, just as if the client had been down the previous day.

no output

The save set completed, but returned no status output. The most common reasons are that the client crashed or lost its network connection (i.e.. a router between the client and server crashed) while the client was being backed up. Another is that the disk on which the client status was being logged filled up (perform a **df /nsr/tmp** to see if this was the case). To determine if the save set was saved, you can use **mminfo**(1m). For example, run **mminfo -v -c** *clientname* **-t '1 day ago'** and look at the *flags* column for the completion status. An 'a' flag means it aborted. Use a more distant time (the −**t** option) to look further back in time.

filesystem: No such file or directory

An explicit save set was named in the *Client* resource for the specified client, and that save set does not exist (or is not currently mounted) on the client. Make sure you spelled the save set name correctly (and that it is capitalized correctly), and log into the client in order to verify that the save set is mounted.

/path/nsrexecd: Couldn't look up address for your host
/path/nsrexecd: Host address mismatch for server

The **nsrexecd** daemon on the client managed to look up the server in the client's host table, but the address listed there did not match the address of the server. Every interface of the server must have a unique name listed in the host table (possibly with non-unique aliases or CNAME's), and each unique name must be listed as a valid server to **nsrexecd**.

/path/nsrexecd: Host server cannot request command execution
/path/nsrexecd: Your host cannot request command execution

The server is not listed in **nsrexecd**'s list of valid servers on the specified client. The list of valid servers is either on the **nsrexecd** command line (with one or more −**s** *server* options to **nsrexecd**), or in a file (with the −**f** *file* option to **nsrexecd**). If neither is specified, **nsrexecd** will look for a file named *servers* in the same directory that contains the *nsrdb* configuration database (e.g. */nsr/res/nsrdb* on a typical UNIX server). Also the server may not be listed in one or more of /etc/hosts, NIS, or DNS, on the client, in which case **nsrexecd** cannot validate the server until the client's host naming configuration is fixed.

/path/nsrexecd: Invalid authenticator
/path/nsrexecd: Invalid command

These two messages should never occur in a savegroup completion message. They mean that **savegrp** did not follow its protocol correctly.

/path/nsrexecd: Permission denied
Permission denied

These similar messages are generated by **nsrexecd** and **rshd**, respectively. In either case, the server does not have permission to execute commands on the client. In the case of the first message, make sure that the server is listed as a valid server on the client (see "Host *server* cannot request command execution", above, for details). In the case of the second message, which does not mention **nsrexecd**, make sure that "*servername*" is listed in the client's */.rhosts* file (or, if you have set the *remote user* attribute for this client, the *.rhosts* file in the home directory for that user on the client).

/path/savegrp: printing bootstrap information failed

See "unknown printer" message information.

reading log file failed

After the specified save set completed, savegrp was unable to read the log file of the output status from the save set. This generally means that someone, or

an automated non-NetWorker administrative program or script, removed the log file. This message can also occur if the filesystem on which the client logs are stored has run out of space (use **df /nsr/tmp** to determine if this is the case). Verify that no scripts remove files from */nsr/tmp* (which is where savegrp stores the save set log files).

request from machine *server rejected*

The server is not listed in the PC (NetWare or DOS) client's list of acceptable servers. See the NetWorker Installation Guide for instructions on adding the server to the client-side list.

N retries attempted
1 retry attempted

One of these informational messages is prepended to a save set's output if **savegrp** is unable to backup the data on the first try and if the *client retries* attribute for the group has a value greater than zero. In this case, the specified number of retries was performed before the backup of the save set succeeded or was finally marked as failed.

RPC error: *details...*
Cannot open save session with '*server***'**

The **save** command generates this message if it is unable to back up data to the NetWorker server. There are several possible *details*. The most likely causes are: resources are exceeded on the server so nsrd cannot accept new save sessions, nsrd actually died since savegrp started (however, this is unlikely, since you cannot normally receive a savegrp completion message after nsrd dies, but you can see this when using the **−p** option), there are numerous network errors occurring and save cannot open a session to save its data (check this by running **netstat -s** and see how many network errors are occurring; you may need to do this several times a few minutes apart to get the change in errors). Save cannot tell which of these three causes are the real cause. If you see these errors frequently, and it looks like a server resource problem, you might consider increasing the value of the *client retries* attribute of the **group** resource having these problems. This won't decrease the resource utilization, but will make **savegrp** more robust. (The trade-off is that increasing *client retries* will increase the load on the server even more).

nsrexecd on *client is unavailable. Using rsh instead.*

This informational message is only displayed when the **−v** flag has been used for verbose information. This message means that **nsrexecd** is not running on the client, and that savegrp is attempting to use the **rshd** service instead for backward compatibility with older versions of **savegrp**.

save: *clientname2 is not on client's access list*

This error occurs when the named client has more than one name, for example, a short name, *client*, and a fully-qualified domain name, *client.EMC.com*. When the client attempts to connect back to the NetWorker server to start a save, that client is calling itself by the name *client*, which matches the client resource name, but when the server looks up the client's network address, it is getting back the name *clientname2*. If this is correct, add the name *clientname2* to the client's **aliases** attribute and re-run the save.

save: path length of *xxxx too long*, *directory not saved*

This message can occur if you have a directory tree that is very deep, or directory names that are very long. This message can also occur if there are bad blocks in the specified filesystem, or if the filesystem is corrupt. NetWorker limits the full pathname to 1024 characters which is the system imposed maximum on most systems. To save such directories, you need to rename or move

the directories so that the full pathname is shorter than 1024 characters. If the filesystem appears to be corrupted (for example, a very long pathname that looks like it has a loop in the name), perform a filesystem check on the specified client.

/path/save: Command not found
/path/savefs: Command not found
/path/save: Not found
/path/savefs: Not found

The **save** or **savefs** command could not be found in the specified path. If you are using **nsrexecd**, this probably means that the **save** or **savefs** command is not in the same directory in which **nsrexecd** is installed (or that **save** or **savefs** was removed). If you are using **rshd** for remote execution, then you need to set the *executable path* attribute in the *Client* resource for this client to be the directory in which the NetWorker executables are installed on the client.

savefs: error starting save of *filesystem*

This informational message accompanies several other **save** or **asm** messages listed here. This message means that savefs has detected the failed save and has marked the save set as failed.

save: unknown host name: *server*
savefs: unknown host name: *server*

The host table on the specified client (either /etc/hosts, NIS or DNS, depending on that client's configuration) does not include the server's name. Add the server's hostname to the specified client's host table. If you use DNS but the server's Client resource name (i.e. the client resource for the server itself) is not fully qualified (i.e. it looks like "server", not "server.dom.ain"), and the server is in a different domain from the client, add the name *server* to the domain table for the domain containing the client. If you use NIS, this error means that either the NIS hosts map does not contain the server, the */etc/hosts* file does not list the server, or the NIS master for the specified client is otherwise misconfigured (the server is a secondary server and there is no **yppush**(1m) from the primary; run **ypwhich -m** on the client to find which NIS server is providing master translation).

savegrp: client rcmd(3) problem for command '*command'*

This error message normally accompanies another, more specific, error message. It is generated when the attempt to run the specified command (usually **save** or **savefs** with several command line parameters) failed on the specified save set. The previous line of error output should include the more specific error message (look for that message elsewhere in this section). Generally, the problem is a bad hosttable configuration, or various permissions denied errors (server not specified when starting nsrexecd, or missing permissions in .rhosts if not using nsrexecd). If not, log into the NetWorker server as root and run the command **savegrp -p -v -c** *clientname groupname* giving the appropriate client for *clientname* and *groupname* . This verbose output should include the necessary additional information needed for fixing the problem.

savegrp: suppressed *N lines of verbose output*

Sometimes a backup will generate a huge amount of output, such as when one runs **savegrp -v**. When **savegrp** updates the group completion attribute in the server, it may suppress some initial lines of of this output, since logging all of the output to the completion attribute can cause **nsrd** to use an unexpectedly large amount of memory. The entire output of **savegrp -v** can be found in the daemon.raw.

savegrp: suppressed *N lines of output* **-**
check daemon.raw for details.

The savegrp completion notification gets truncated if it is 1024 characters or longer. The daemon.raw and **NetWorker Management Console** will have details of the complete backup.

socket: All ports in use

The NetWorker server has run out of socket descriptors. This means that you have exceeded the socket resource limit on your server. To avoid such future messages, you should determine what other network services are running while savegrp is running, and consider re-scheduling either savegrp or the other service(s). You can also reduce the parallelism in the **nsr_service**(5) resource to reduce the resource utilization.

socket: protocol failure in circuit setup.

The client does not seem to support the TCP/IP protocol stack, or has not used a privileged port for setting up its connection. The latter could occur if you use **nsrexecd** but did not start it as root on the specified client. The **nsrexecd** daemon must run as root on each client.

path: This data set is in use and cannot be accessed at this time

This message is generated by save sets on PC clients running DOS or NetWare. The NetWorker client software on these systems cannot back up files open for writing, due to the interface provided by the operating system. This message actually comes from Novell's TSA and is not changeable.

unknown host

The specified client is not listed in the host table on the server (note: a similar "save" or "savefs" specific message is described above). Depending on your host configuration, this means the client is not listed in one (or more) of /etc/hosts, NIS, or the Domain Name Service. If you use fully qualified domain names, you may need to make a new client resource for this client, using that fully qualified domain name (i.e. name the client resource "mars.EMC.com", not "mars").

printer: unknown printer

path/savegrp: printing bootstrap information failed

(reproduced below)

This message, or a similar one, accompanies the bootstrap information when **savegrp** was unable to print the bootstrap on the printer. You need to either specify a different printer in the *printer* attribute for the group, or configure your print server to recognize the printer (by default, your system's default printer is used). The bootstrap information is listed as part of the savegrp completion mail. You should print out this information immediately, in case your server has a disaster and loses a disk, and fix the printer name used by **savegrp**.

Warning – file '*path' changed during save*

This warning message is generated when **save** notices that the file's modification time changed while the file was being backed up. NetWorker does not attempt to lock files before saving them, as doing this would make backups run extremely slowly. You may wish to backup files which generate this message manually (to ensure that a consistent copy is saved). NetWorker does not attempt this automatically, so that it avoids trying forever on the same file.

Warning: '*client' is not in the hosts table!*

This message is generated by a **save** or **savefs** command run on the specified client to save that client's filesystems. The client's hostname is not listed in the host table on the client (either /etc/hosts, NIS or DNS, depending on that client's configuration). This almost always results in a failed save. Fix the client's host table and re-run the save.

asm: path was not successfully saved
This message generally accompanies one or more other more-specific messages for the save set. The specified path within the current save set was not saved successfully, but the backup operation will continue trying to back up other files and directories on the save set.

asm: xdr_op failed for path
This error can be caused by several possible conditions (for example: out of memory, defective networking software in the operating system, an external ASM unexpectedly exiting, or a lost network connection). If it was due to a lost network connection, then the NetWorker server most likely exited (due to **nsr_shutdown**). After restarting the server, re-run the group. If due to an ASM exiting unexpectedly (in this case, the message should be accompanied by a message describing which ASM exited unexpectedly), you may have found a bad block on the disk or, perhaps, a defect. Check if the client ran out of memory (there may be console messages) and verify that there are no bad blocks on the save set's disk. If there were network errors, there may also have been messages logged by other programs on the system console (client or server) or to system log files.

FILES

/nsr/tmp/sg.*group*	A lock file to keep multiple savegrps of the same group from running simultaneously.
/nsr/tmp/sg.*group***.***client***.***	Temporary files used to log the output of individual save sets for the named group and client.
/nsr/tmp/g*group*∗	On filesystems with short names (less than 64 characters), the temporary files used to log the output of individual save sets for the named group.

SEE ALSO

ls(1), **ps**(1), **nsr_getdate**(3), **rcmd**(3), **fstab**(5), **nsr**(5), **nsr_client**(5), **nsr_directive**(5), **nsr_notification**(5), **nsr_service**(5), **nsr_group**(5), **nsr_schedule**(5), **nsr_resource**(5), **nsr_render_log**(1m), **mminfo**(1m), **nsrssc**(1m), **netstat**(1m), **nsr**(1m), **nsradmin**(1m), **nsrjobd**(1m), **nsrexecd**(1m), **nsrwatch**(1m), **rshd**(1m), **save**(1m), **savefs**(1m), **pathownerignore**(5), **yppush**(1m)

NAME

save − save files to long-term storage with NetWorker

savepnpc − save files to long-term storage with NetWorker and run pre- and post-processing commands on a NetWorker client

SYNOPSIS

save [−**BEiKLnquSVvx**] [−**s** *server*] [−**J** *storage-node*] [−**c** *client-name*] [−**N** *name*] [−**e** *expiration*] [−**f** *dirfile*] [−**o** *save_operations*] [−**b** *pool*] [−**F** *file*] [−**I** *input_file*] [−**g** *group*] [−**k** *checkpoint_id*] [−**l** *level*] [−**t** *date*] [−**m** *masquerade*] [−**w** *browse_time*] [−**y** *retention_time*] [−**W** *width*] [*path* ...]

savepnpc

-s server -g group [−**BEiKLnquSVvx**] [−**J** *storage-node*] [−**c** *client-name*] [−**N** *name*] [−**e** *expiration*] [−**f** *dirfile*] [−**b** *pool*] [−**F** *file*] [−**I** *input_file*] [−**l** *level*] [−**t** *date*] [−**m** *masquerade*] [−**w** *browse_time*] [−**y** *retention_time*] [−**W** *width*] [*path* ...]

DESCRIPTION

save saves files, including directories or entire filesystems, to the NetWorker server (see **nsr**(1m)). The progress of a save can be monitored using the Java based **NetWorker Management Console** program or the **curses**(3X) based **nsrwatch**(1m) program for other terminal types.

The user of this command may retain root privileges if the command's modes are properly set as described in **nsr**(1m).

If no *path* arguments are specified on the command line or via the −**I** option, the current directory will be saved. **save** will save a directory by saving all the files and subdirectories it contains, but it will not cross mount points, or follow symbolic links. If the paths to be saved are mounted from a network file server, **save** instructs the user to run the save on the remote machine or use the **-L** option.

If the path argument is specified on the command line and one of the directories in the path is a symbolic link, the target path of the symbolic link will be saved, rather than the symbolic link path itself. Therefore, the target path of the symbolic link must be specified when using the recover (1m) command or the nwrecover (8) program to recover the files.

The directive files (see **nsr**(5)) encountered in each directory are read by default, and they contain special instructions directing how particular files are to be saved (i.e. compressed, skipped, etc.). These files are named '**.nsr**' on UNIX or 'nsr.dir' on Windows.

Each file in the subdirectory structures specified by the *path* arguments is encapsulated in a NetWorker save stream. This stream of data is sent to a receiving process (see **nsrd**(1m)) on the NetWorker server, which processes the data, adding entries to the on-line index (see **nsrindexd**(1m)) for each file in the stream, with the data finally ending up on a long term storage media (see **nsrmmd**(1m)). By default, these on-line index entries are stored in the "backup" index namespace.

Details about handling media are discussed in **nsrmm**(1m) and **nsr_device**(5).

savepnpc consists of the same command options as **save but requires the -g group to** run. Apart from running the actual save, it also performs the pre and post processing commands, if any. Prior to the actual save of the first saveset on a NetWorker client, **savepnpc** performs pre-processing commands if any exists in the /nsr/res/<grpname>.res file, and at the end of the save of the last save set on the client, the post-processing commands (if any) will be invoked. It is possible to setup multiple clients in a savegroup such that each client can run different pre and post commands. The <grpname>.res file resides on the client machine and is unique to that host. In the condition of failure to run the pre-processing commands, **savepnpc**

aborts itself. All results are logged in /nsr/logs/savepnpc.log file on the client. A timeout condition can be set by the user to indicate at which point in time the post-processing commands need to be run without waiting for all the save sets to be backed up. This timeout attribute resides in the /nsr/res/<grpname>.res file. The timeout should be specified in double quotes, in such a format that **nsr_getdate()** can understand (see **nsr_getdate**(3)). Also, abort precmd with group attribute exists in the /nsr/res/<grpname>.res file. This can be set to Yes or No. If set to Yes, the precmd will terminate if the particular savegrp is aborted. If it is set to No, the precmd will run to completion even after the abnormal exit of the savegrp session.

An example of /nsr/res/<grpname>.res can be described as:

```
type: savepnpc;
precmd: /bin/true;
pstcmd: /bin/true, "/bin/sleep 5";
timeout: "12:00pm";
abort precmd with group: No;
```

The **precmd** field can be manually modified to contain any number of commands that are needed to be run at the beginning of the save of the 1st save set. The **pstcmd** is to hold any commands that are needed to be run at the end of the save of the last save set. The post-processing commands are run after the save of the last save set or the timeout condition, whichever comes first. Note that on Windows Networker Clients, the shell should be set to "cmd.exe"and the Shell flag should be set to "/c", for running the precmd and postcmd. This will force the OS to close all the opened File Descriptors and other resources in a timely manner, after the execution of the commands. Also, for both the precmd and the pstcmd, on all Networker Clients, its best to redirect the output(stdout and stderr) to another file, to avoid unclosed File Descriptors, after the commands have completed executing.

An example of precmd and pstcmd for Windows Clients is shown below:

```
precmd: cmd.exe /c start_pre_cmd > pre_result.txt 2>&1
pstcmd: cmd.exe /c start_post_cmd > post_result.txt 2>&1
```

OPTIONS

−**b** *pool* Specifies a particular destination pool for the save.

−**c** *client-name*

Specifies the client name for starting the save session. This is useful on clients with multiple network interfaces, and multiple host names. It can be used to create multiple index databases for the same physical client with multiple network interfaces.

This does not specify the network interface to use. This is specified in the **server network interface** attribute of the client resource (see **nsr_client**(5)). This option can also be used on a cluster when performing manual saves, or in specifying a non-default **backup command** for scheduled saves. This option directs NetWorker to override the cluster path-ownership rules, saving the path argument(s) as belonging to *client-name* and making index entries in the index for *client-name* instead of using the name of the physical host or virtual host which owns the path, according to the cluster management software. Refer to **pathownerignore**(5) for more information about path-ownership rules.

−**e** *expiration*

Set the date (in **nsr_getdate**(3) format) when the saved data will expire. When a save set has an explicit expiration date, the save set remains both browsable and non-recyclable until it expires Thus, the explicitly provided expiration overrides the existing browse and retention times specified in the client policy and the browse and retention times get changed to the expiry time. The "-e

exp_time" option cannot be used in conjunction with "-w browse_time" or "-y reten_time". By default, no explicit expiration date is used and the client's longest browse and longest retention policy are used.

−**f** *dirfile*
The file from which to read prototype default directives (see **nsr**(5)). A *dirfile* of **-** causes the default directives to be read from standard input.

−**o** *save_operations*
Save Operations of the form "KEYWORD:TOKEN=STATE[;KEYWORD:TOKEN=STATE;...]".

The following form is used to configure VSS saves on Windows 2003. Examples:

"vss:*=off"

Turn off VSS.

"vss:Microsoft Exchange Writer=off"

Disable a writer.

"vss:C:=off"

Disable VSS for a drive.

Checkpoint restart related options:

"CHECKPOINT_BY_FILE:checkpoint_by_file=on"

Enables checkpointing after each file that is saved. Turning this option on introduces extra overhead and will slow down the save process. The default is to checkpoint after each directory.

"CHECKPOINT_BY_FILE:checkpoint_by_file=off"

Enables checkpointing after each directory. This is the default behaviour and it is not necessary to explicitly specify this option.

The following forms are specified by savegrp program during scheduled backups. Manual specifications of these forms are not supported.

Backup of renamed directories, in the form of:

"RENAMED_DIRECTORIES:index_lookup=on"

Enables support for the backup of renamed directories. The save program performs a lookup in the client file index to determine whether a directory has been renamed. If a directory has been renamed, all of the files and subdirectories under the directory will be backed up.

"RENAMED_DIRECTORIES:index_lookup=off"

Disables the backup of renamed directories.

Backup of non-ASCII save set names.

See **nsr_client (5)** and the Admin Guide for more information on the forms for non-ASCII names and how to enable or disable support for renamed directories.

−**g** *group*
This option is used by **savegrp**(1m) and **savefs**(1m) to denote the group of the save (see **nsr_client**(5) and **nsr_group**(5)) and is used by the NetWorker server to select the specific media pool.

−**i** Ignores any .nsr (UNIX) or nsr.dir (Windows) directive files as they are encountered in the subdirectory structures being saved.

−**J** *storage-node-name*
Specifies which host to use as the storage node for the backup (see nsr_storage_node(5)).

−**k** *checkpoint_id*
Checkpoint id for a corresponding checkpoint restart enabled save. This option is used by **savegrp**(1m) to enable checkpoint restart functionality for clients that have the 'Checkpoint enabled' option set to 'Enabled'. The **checkpoint_id** can contain only decimal digits. It should be set to 0 to start a new checkpoint restart save, or be equal to the **checkpoint_id** of a save that you wish to continue. For more details about how to obtain the **checkpoint_id** of a save, see the **mminfo**(1m) man page. See the -o option for more checkpoint restart related options.

−**l** *level* The level of the save. This option is used by **savegrp**(1m) and **savefs**(1m) to specify a particular level, for a scheduled save, to be stored in media database. This option is ignored by manual save command and backup is performed at level adhoc. The level information is not used by save to determine whether a file should be backed up during scheduled backup, but parameter on **-t** option is used instead.

−**m** *masquerade*
Specifies the tag to precede the summary line. This option is used by **savegrp**(1m) and **savefs**(1m) to aid in savegrp summary notifications. **savepnpc**(1m) also uses this tag to identify client operations on the savegrp's work list that should complete before **pstclntsave**(1m) will trigger its post-processing.

−**n** No save. Estimate the amount of data which will be generated by the save, but do not perform the actual save.

−**q** Quiet. Displays only summary information and error messages.

−**s** *server*
Specifies which machine to use as the NetWorker server.

−**t** *date* The date (in **nsr_getdate**(3) format) by which files must have been modified for them to be saved. This option is used by **savegrp**(1m) and **savefs**(1m) to perform scheduled saves by consulting with the media database to determine the appropriate time value based on the previous saves for the save set and the level of the scheduled save.

On Windows, file modification/change time refers to Last Written time, Creation time and the Archive file attribute of a file. All of these are used to determine whether a file needs to be backed up. If the Archive file attribute is set,

the file will always be backed up, since some older filesystems may not have the proper file creation time, unless NSR_AVOID_ARCHIVE environment variable is set (to a value other than "no").

−**u** Stop the save if an error occurs. The **save** program normally treats errors as warnings and continues to save the rest of the files in the backup. When this option is set, errors will cause **save** to exit and abort the save. This option is not recommended for general use, although it can be useful when a group of files needs to be backed up as a set.

−**v** Verbose. Causes the **save** program to provide great detail about the **save** as it proceeds.

−**y** *retention*
Sets the date (in **nsr_getdate**(3) format) when the saved data will become recyclable. The special value **forever** is used to indicate that a volume that never expires (i.e. an archive volume) must be used. By default, the server determines this date for the save set based on the retention policies in effect. This option allows overriding the existing policies on a save by save basis.

−**w** *browse_time*
Sets the date (in **nsr_getdate**(3) format) after which this save set will no longer be browsable. By default, the server determines the browse date for the save set based on the browse policies in effect. This option allows overriding the existing policies on a save by save basis.

−**x** Cross mount points. This option is only applicable to the manual save command. When the **-x** option is specified with a (save set) path, the mount point under this path is crossed and files in the mounted file system are backed up. When the **-x** option is not specified, only the local files and directories of the path are backed up.

This option has no effect and is ignored when a mount point (or file/subdirectory path under it) is specified as the (local) path to be backed up for the manual save command, which generally requires the −**L** option.

For example, if /tmp_mnt/pumbaa is a mount point for pumbaa:/space, the following three commands behave differently with the specification of -x option:

save -x /tmp_mnt follows mount points, backs up the local files and directories of /tmp_mnt along with files and directories of /tmp_mnt/pumbaa

save /tmp_mnt does not cross mount points, only local files and directories of /tmp_mnt are backed up

save [-x] /tmp_mnt/pumbaa[/...] when mount point is specified as the (save set) path, "-x" option is ignored and an error message is displayed to indicate "-L" option is required on most platforms

For Windows, mount point refers to an alternative path for a mounted volume on the same machine. For example, H:\tmp_mnt\d_drive may be defined as a mount point to the local volume (or drive) D:.

−**B** Force save of all connecting directory information from root (''/'') down to the point of invocation.

−**E** Estimate the amount of data which will be generated by the save, then perform the actual save. Note that the estimate is generated from the inode information; thus, the data is only read once.

−**F** *file* Only save files whose change time is newer than the file modification date of *file*.

For Windows, see *-t* option for more information on *file change time*.

−**I** *input_file*
In addition to taking the paths to save from the command line, read paths to save from the named file. The paths must be listed one per line. If no paths are specified on the command line, then only those paths specified in the file will be saved.

−**K** Does not build connecting directory index entries.

−**L** Local. Saves will be performed from the local NetWorker client, even when files are from a network file server. To recover these files, run **recover**(1m) with the −**c** *client* arguments, where *client* is the name of the NetWorker client that did the **save**.

−**LL** In addition to treating the backup as a local backup, causes an extra line to be printed at the end of the completion output of the form ''complete savetime=number'', where number is the savetime of the save set created by this backup. This option is meant to be used by the **savegrp**(1m) command in performing automatic cloning.

−**N** *name*
The symbolic name of this save set. By default, the most common prefix of the *path* arguments is used as the save set name. If the -N option is used when saving any of the SYSTEM save sets (SYSTEM STATE, SYSTEM FILES, and SYSTEM DB), the path must also be specified and must match the name value assigned with the -N option.

−**S** Allows only save set recovery. This performs the save without creating any index entries. This means that the save set will not be browsable, although save set recovery may be used to recover the data.

−**V** Prevent the OFC mechanism from creating a point-in-time copy of the source volume. (Included for compatibility with NT NetWorker servers.)

−**W** *width*
The width used when formatting the summary information output. Valid values for width are integer values from 1 to 10000. If the supplied width is too small for the summary to fit in, the width will be silently adjusted upwards as necessary. If the supplied width is larger than the minimum needed, then spaces will be used to pad the summary to the correct width. Note that if no -W argument is supplied then there is no fixed width used, and the summary simply expands to whatever minimum width is necessary.

SEE ALSO **curses**(3X), **nsr_getdate**(3), **nsr**(5), **nsr**(1m), **nsr_client**(5), **nsr_device**(5), **nsr_group**(5), **nsr_service**(5), **nsrd**(1m), **nsrim**(1m), **nsrindexd**(1m), **nsrmm**(1m), **nsrmmd**(1m), **nsrwatch**(1m), **recover**(1m), **savefs**(1m), **savegrp**(1m), **pathownerignore**(5)

DIAGNOSTICS

Exit Codes:

0 Normal exit. This means that a save set was correctly created on the server. Messages about individual file backup failures are *warnings*, and do not cause abnormal exit.

<>**0** Abnormal exit. A save set was not correctly created on the server.

Messages:

host: *saveset* **level=***level*, *size time count* **files.**
This message (with the appropriate client host name, saveset name, level, total save set size, elapsed time, and file count) is printed whenever **save** is run by **savegrp**(1m) and exits normally.

host: *filename*: *warning*
Messages of this form are warnings about difficulties backing up individual files. Such messages do not normally cause the **save** to fail, and therefore may appear in the **save** output found in the Successful section of the ''Savegroup Completion'' message.

path: **File index could not be obtained due to** <*reason*>**.**
Contents of this directory may not be properly backed up.

There was an error in retrieving an on-line file index record for the path with renamed directory support. Run nsrck (1m) and turn off the "Backup renamed directories" attribute in Client resource (see **nsr_client**(5)) to re-run the group if the problem persists.

NAME savepsm − save NetWorker Management Console database to long-term storage with NetWorker

SYNOPSIS **savepsm** { −**I** *NMC Install directory* } -[**STCEh**] [−**s** *server*] [−**c** *client-name*] [−**l** *level*] [−**b** *pool*]

DESCRIPTION **savepsm** saves the contents of the NetWorker Management Console database to a NetWorker server. It can also be used to truncate (delete and restart) the database's transaction log.

The LD_LIBRARY_PATH or equivalent environment variable of the shell from which **savepsm** is started must contain <product install directory>/sybasa/lib64 (Solaris/AIX/HP-UX) or <product install directory>/sybasa/lib (Linux) as one of the components. If NetWorker Management Console is not located in the default /opt/LGTOnmc directory on Solaris, LD_LIBRARY_PATH must also contain <product install directory>/bin.

OPTIONS

−**b** *pool* Use a volume from the specified pool.

−**C** Perform software compression.

−**c** *client-name*
Index the save under client-name. See save(1m) for more details.

−**E** Perform encryption.

−**h** Display the usage of the **savepsm** binary.

−**I** *NMC Install directory*
Use "NMC Install directory" as the package's install directory. NetWorker Management Console is installed in /opt/LGTOnmc by default on Solaris and /opt/lgtonmc on Linux/AIX/HP-UX.

−**l** *level* Use "level" as the backup level. This can be "full", "incr" or "skip".

−**S** Skip backup of authentication configuration files.

−**s** *server* Use "server" as the NetWorker server.

−**T** Truncate the transaction log of the database without performing the backup.

USAGE **savepsm** is used to save a NetWorker Management Console database. The command is used to manually save the database or to truncate the transaction log. The NetWorker Management Console database credential file (gstd_db.conf) located in the database directory and the NetWorker Management Console authentication configuration files are automatically backed up along with the NetWorker Management Console database. The database credential file and the authentication configuration files are saved under the saveset name "CONSOLE_BACKUP_FILES".

The authentication configuration files contain external repository (e.g. LDAP) configuration information if NetWorker Management Console has been configured to use external repository for authentication. NetWorker Management Console by default uses its own user repository for authentication. In order to recover the authentication configuration files, use the NetWorker recover (1m) command.

To perform an automated save of the database through savegrp(1m), the savepsm.sh script is used. **savepsm.sh** is a wrapper script that sets LD_LIBRARY_PATH or equivalent environment variable and then calls **savepsm.** In order to configure the database for an automated save, define a client resource and place the string

"NMCASA:/gst_on_<hostname>/lgto_gst" (where hostname is the short name of the host running the NetWorker Management Console server) in the client's save set attribute. Then place the following command in the 'backup command' attribute:

savepsm.sh

Note that the transaction log is automatically truncated after the backup is completed.

To truncate the database's transaction log without performing a backup, use the -T option.

NOTES The NetWorker Management Console server must be running for **savepsm** to run successfully.

EXIT STATUS **savepsm** exits with a non-zero exit code if an error occurs and with zero on success.

SEE ALSO **save**(1), **savegrp**(1), **recoverpsm**(1m)

NAME scanner − NetWorker media verifier and index rebuilder

SYNOPSIS

scanner [*options*] −**B** −**S** *ssid* [−**im**] [−z] **device**

scanner [*options*] −**i** [−**S** *ssid*] [−**c** *client*] [−**N** *name*] [−**y** *retention time*] **device**

scanner [*options*] −**m** [−**S** *ssid*] [−**y** *retention time*] **device**

scanner [*options*] [−**S** *ssid*] [−**c** *client*] [−**N** *name*] **device** [*command*]

options: [−**npqvk**] [−**b** *pool*] [−**f** *file*] [−**r** *record*]
[−**s** *server*] [−**t** *type*] [−**V** *volume name*]
[−**Z** *datazone-id*]

command: −**x** *command* [**arg** ...]

DESCRIPTION The **scanner** command reads NetWorker media, such as backup tapes or disks, to confirm the contents of a volume, to extract a save set from a volume, or to rebuild the NetWorker online indexes. As installed, only the super-user may run this command. However, the command's modes can be modified such that normal users may run the command while retaining root privileges; see **nsr**(1m) for more details. The *device* must always be specified, and is usually one of the device names used by the NetWorker server. For tape drives, it must be the name of a ''no-rewind on close'' device. For adv_file type device, read-write device name will be used when read-only device name is specified.

When **scanner** is invoked with either no options or −**v**, the volume on the indicated *device* is opened for reading, scanned, and a table of contents is generated. The table of contents contains information about each save set found on the volume. By default, one line of information is written to standard output for each save set found containing the client name, save set name, save time, level, size, files, ssid and a flag. The **client name** is the name of the system that created this save set. The **name** is the label given to this save set by **save**(1m), usually the path name of a file system. The **save time** is the date and time the save set was created. The **level** values are one-letter abbreviated versions of **f**ull, **i**ncremental, levels **0** through **9**, or blank for manual saves. The **size** is the number of bytes in the save set. The **files** labeled by column provide the number of client files contained in the save set. The **ssid** (save set identifier) is an identifier used internally to reference and locate this save set. This same identifier may be specified explicitly with the −**S** option to extract a particular save set.

The table of contents is based on synchronization (sometimes called ''note'') *chunks* (see **mm_data**(5)) interspersed with the actual save set data. There are four types of note chunks: Begin, Continue, Synchronize, and End, symbolized by a *flag* of B, C, S or E respectively. The **B**egin note is used to mark the start of a save set. When a beginning chunk is written, the save set size and number of files are not known. The **C**ontinue note is used to indicate that this save set started on a different volume. The **S**ynchronize note marks locations in the save set where you may resume extracting data in the event of previous media damage (a client file boundary). The **E**nd note marks the end of the save set, and causes the table of contents line to be printed. The other notes are displayed only when the −**v** option is selected.

OPTIONS

−**b** *pool* Specifies which pool the volume should belong to. This option only applies for versions of NetWorker that do not store the pool information on the media. For these versions, you might need to specify the media pool the volume should belong to if the user does not want the volume to be a member of the *Default* pool. For volumes where the pool information is stored on the media, the media must be relabeled (destroying all data on the media) to assign the media to a different pool.

−**B** When used in conjunction with the −**S** option, the save set id specified is

flagged as that of a bootstrap.

−**c** *client*
Process only save sets that come from the specified NetWorker client machine. This option can be used multiple times and in conjunction with the −**N option,** but only in presence of the −**i** or −**x** option.

−**f** *file* Starts the scan at the specific media file number. This option is not useful on media such as optical disks and file device types, for example.

−**i** Rebuilds both the media and the online file indexes from the volumes read. If you specify a single save set with the −**S** *ssid* option, only entries for the specified save set are copied to the online file index. Note that for version 6.0 and later, if you have the tape that contain the index backups that go along with the data backups, the recommended way of restoring your indexes is to run **scanner** −**m** to reload the media database entries for the index and data backups. Once that is done, you should run **nsrck** −**L7** −**t date** <**clientname**> to recover the index for the client as of the time of the backups on the tape. This will roll the index entries for that time back into the index. However, if you have tapes for which there are no index backups, then you will need to use the −**i** option to reconstruct the index entries.
Note:
For NDMP save sets or DSA save sets, this option does not reconstruct the index entries from the volume. However, if you have index backups, use scanner and nsrck as said above. For volumes that have a combination of DSA save sets and regular NetWorker save sets, scanner -i will skip over the DSA save sets with an error.

−**m** Rebuild the media indexes for the volumes read. If you specify a single save set with the −**S** *ssid* option, only entries for the specified save set are copied to the media index, the save set data will be written to standard output which will need to be redirected as needed to prevent any unwanted behavior on the console. The media database will not retain the "scanned-in" status. There is no longer a flag to show that status in the "ssflags" field. The saveset gets a new browse and retention policy depending on the time that it was scanned in and the clock starts ticking for the saveset.

−**n** Checks all media without rebuilding the media or index databases. When used with the −**i** option, this option provides the most complete media checking available, while not modifying the databases at all.

−**N** *name*
Only processes save sets specified by *name* (a literal string only). This option can be used multiple times and in conjunction with the −**c** option, but only in presence of the −**i** or −**x** option.

−**p** Prints out information save set notes as they are processed.

−**q** Displays only errors or important messages.

−**r** *record*
Starts the scan at the specific media record number.

−**s** *server*
Specifies the controlling server when using **scanner** on a storage node. See **nsr_storage_node**(5) for additional detail on storage nodes.

−**y** *retention time*
Specifies the retention time for completed clone instance of save sets in the volume(s) being scanned in. This option is valid only with −i or −m option. If save sets have been specified using −S option, the retention time of the clone

instance of those save sets will be set to the specified value. Note that retention time of clone instances not belonging to the volumes being scanned in will not be modified.

−z This end-silently option will cause the scanner to not prompt for the next volume when the saveset spans onto another volume. It will not wait for user input, but, will just quit when its done reading the first volume.

−S *ssid* Extracts the specified save set(s). When used with the **−i** or **−x** option, this option can be used multiple times and is in addition to any save sets selected using the **−c** and **−N** options. Otherwise, the volume is scanned for save set *ssid*, to be written to the standard output, which will need to be piped as needed to prevent any unwanted behavior on the console. Most often this is piped to a **uasm**(1m) program running in recover mode to process the save set (potentially with a directory list to limit the files to be recovered and potentially using a **−m** argument to map the file location). When using **−S** without **−i** or **−m**, scanner prompts for the volume block size if the volume label is not readable. If the volume information is still in the media database, the user has the option of running recover by save set (see **recover**(1m)). When **−B** is also specified, *ssid* is taken to be that of a bootstrap. Only one ssid is allowed in this case.
Note:
Piping NDMP save set or DSA save set save streams to any recover program such as uasm (1m) is not supported.

−t *type* Specifies the type of media, for example, *optical* for an optical disk, or *8mm 5GB* for an 8mm 5GB tape). Normally the media type is obtained from the NetWorker server, if a known device is being used (see **nsr_device**(5)).

−V *volume name*
Specifies the name of the volume to scan when scanning in cloud volumes. Cloud devices typically contain more than one volume and **scanner** requires the name of the volume to be specified so that the right volume can be scanned in.

−v Displays more verbose messages, such as a log of each note chunk, and a message after every 100 media records. When the **−i** option is used, a line is printed for each client file (an enormous amount of output can be produced).

−k This option should only be used when requested by technical support.

−x *command arg ...*
Specifies an arbitrary UNIX command to process each new selected save set. This argument can only occur once at the end of the argument list (after *device)*. The save stream for each save set is connected to the stdin of a new instance of the command. Most often this command is **uasm**(1m) running in recover mode to process each save set (potentially using a **−m** argument to map the file location). If the volume information is still in the media database, the user has the option of running recover by save set (see **recover**(1m)). Do not attempt console I/O by specified UNIX command. Instead specify conflict resolution parameters as arguments passed to the command (e.g.: scanner -S <ssid> -x uasm -rv -iR). If console interaction is required, pipe scanner output to the desired Unix command instead of invoking the command using the -x option.
Note:
Piping NDMP save set or DSA save set save streams to any recover program such as uasm (1m) is not supported.

−Z *datazone-id*
Specifies the *datazone-id* from which the volume is being scanned in. This

option is used when scanning in cloud volumes from a different datazone than that of the current NetWorker server.

Note that NetWorker's *datazone-id* is automatically created at install time and is unchanged for the life of that installation. This means that a fresh NetWorker install done in times of disaster recovery, to scan in bootstrap data for example, also results in the new installation having a *datazone-id* that is different from the previous one. Unless -Z is specified to point **scanner** to the old *datazone-id,* the volume from the old datazone will not be found by **scanner.** Once a bootstrap is scanned in and recovered, the old *datazone-id* is automatically restored.

To enable the NetWorker server to successfully read a bootstrap volume in times of disaster recovery, the *datazone-id* of the current server is updated to the value specified with −Z. However, this is true *only in disaster recovery mode.* That is, **scanner** updates the NetWorker *datazone-id* only if there are no other volumes in the NetWorker database and there is only one device defined. All other cases where −Z is specified only results in the *datazone-id* being temporarily used by the **scanner** to access the volume for the current operation without updating the server's datazone-id

EXAMPLES *Verifying a tape*:

```
scanner /dev/nrst0
scanner: scanning 8mm tape mars.001 on /dev/nrst0

client name    save set    save time  level          size      files      ssid S
space          /export     10/07/94 12:38 f    100762460      10035    16983 E
space          /usr        10/07/94 13:14 f     27185116       3185    16984 E
space          /nsr        10/07/94 12:40 f     77292280       8436    16980 S
space          /           10/07/94 13:22 f      1693192        518    16985 S
scanner: reached end of 8mm tape mars.001
```

Rebuilding the online file index for a client from a tape:

```
scanner -m /dev/nrst8
scanner: scanning 4mm tape monday.fulls on /dev/nrst8
scanner: ssid 17458697: scan complete
scanner: ssid 17458694: scan complete
scanner: ssid 17458698: scan complete
scanner: ssid 17458693:  NOT complete
scanner: reached end of 4mm tape  monday.fulls

scanner: when next tape is ready, enter device name [/dev/nrst8]?
nsrck -L7 -t  "06/07/99" supernova
nsrck: checking index for 'supernova'
nsrck: The file index for client 'supernova' will be recovered.
nsrck: Recovering index savesets of 'supernova' from 'quasar'
Recover completion time: Fri Jun 16 14:03:16 2000
nsrck: completed recovery of index for client 'supernova'

nsrck: /disk1/nsr/index/supernova contains 85782 records occupying 14 MB
nsrck: Completed checking 1 client(s)
```

Extracting a save set for /usr and relocating to /mnt:

```
scanner -S 637475597 /dev/nrst8 | uasm -rv -m /usr=/mnt
                                                          or
scanner -S 637475597 /dev/nrst8 -x uasm -rv -m /usr=/mnt
```

Extracting all save sets from client mars and relocating to /a:

```
scanner -c mars /dev/nrst8 -x uasm -rv -m/=/a
```

SEE ALSO **mm_data**(5), **mminfo**(1m), **nsrmmdbasm**(1m), **nsr**(1m), **nsrck**(1m), **nsrindexasm**(1m), **nsrmmd**(1m), **nsr_device**(5), **nsr_storage_node**(5), **uasm**(1m).

DIAGNOSTICS

xdr conversion error, fn %d, rn %d, chunk %d out of %d
unexpected file number, wanted %d got %d
unexpected record number, wanted %d got %d

All three preceding messages are indicative of media errors (tape blocks are either lost or damaged). In the case of an xdr conversion error, a non-zero ''chunk'' number means that the block may be partially salvageable. Unexpected file numbers are normal when **scanner** reaches the logical end of the media that has been recycled.

continuation of data in nsrscan.*NNNNN.MMMMMM*

After an XDR decode error (an error denoted by one or more of the messages described above), **scanner** attempts to re-synchronize and send the rest of the stream. However, because programs like **uasm**(1m) are unable to handle decoding streams with parts missing in the middle, scanner sends the remainder of the stream to a file. You can decode this stream manually. For example, if your original command was:

```
scanner -S ssid | uasm -r
```

and a synchronization error occurs, you can decode the rest of the stream with the following command:

```
uasm -r < nsrscan.NNNNN.MMMMMM
```

where the file name you enter corresponds to the name printed in the diagnostic message.

unexpected volume id, wanted *volid1* **got** *volid2*

This message normally appears when running in verbose mode on a tape or disk that has been recycled. It does not indicate an error condition, but details the conditions normally treated as the end of the volume.

ssid %d: finished, but incomplete

Scanner has detected the end of a save stream, but the stream was aborted, and is of dubious value. If online indexes are being rebuilt, the end of the aborted stream may precipitate the next message.

(ssid %d): error decoding save stream

As indexes are being rebuilt, **scanner** detected that the bytes in the save stream are invalid. This is usually caused by processing an aborted save stream. Other causes may include a damaged tape. Once this condition is detected, the process of rebuilding the indexes for the particular save stream exits. This may precipitate the next message.

write failed, Broken pipe

Printed by **scanner** when a process rebuilding a save stream's indexes exits before consuming the entire stream.

You are not authorized to run this command

A normal (non-root) user invoked this command.

could not convert '*arg***' to a file number**

The **−f** and **−r** options require a numeric argument for the starting file or record number of the media.

already exists in the media index

The **−i** or **−m** option was specified and the volume was already listed in the

media database. This message is purely informational, and means that the volume is not being added to the media database because it is already listed there.

fn %d rn 0 read error I/O error
done with *tape_type* **tape volid** *volume_name*

These messages, when occurring together, are a consequence of scanner encountering consecutive filemarks at end of the media. They do not indicate an error condition and can be ignored.

LIMITATIONS

The selected device should be placed in service mode when **scanner** is being run. If this is not done, NetWorker may attempt to unload the volume automatically causing **scanner** to fail.

scanner can run without the NetWorker services (for example, **nsrd**(1m) and **nsrmmdbd**(1m)) when not reconstructing the media or the online file indexes with most device types. For logical and NDMP devices, the NetWorker services have to be running in order to query these device configurations.

File index backups imported from volumes from other NetWorker servers cannot be recovered by **nsrck -L7.** You must use **mmrecov** to recover the Bootstrap of that NetWorker server before the file indexes can be recovered.

When scanning a relabeled optical volume (that is, a re-writable optical volume that had been written once, then re-labeled and used again), **scanner** may read off the end of the new data, and attempt to read the old data from the previous version of the volume, terminating with an ''unexpected volume id'' error. This error occurs after all the good data has been read, and can be ignored.

NAME sjiielm − test the SJI Jukebox Interface SJIIELEM command

SYNOPSIS **sjiielm** *devname* [{ **drive** | **slot** | **inlt** | **mt** } **address nelements**]

DESCRIPTION The **sjiielm** program tests the SJIIELEM command on SJI compliant Jukeboxes (see **libsji**(1m)). This command tests the Initialize Element Status interface for a Jukebox. If a Jukebox doesn't support this feature, an appropriate error message will be printed out.

The *devname* argument should be any device name that can be used to reach a SJI compliant Jukebox driven by the system. Typical usage is in constructing the name that contains the SCSI Bus, the SCSI Target ID and the SCSI Lun on that target, e.g. *scsidev@0.4.0*.

The additional optional arguments are for Jukeboxes that support the initialization of a specific range of elements. In this case, you select one of element types of **drive**, or **slot**, or **inlt** (Import/Export element), or **mt** (Media Transport, or the "Robot Arm"), and give an SJI normalized (i.e., starting from 1) address and number of elements to initialize.

SEE ALSO **libsji**(1m)

NAME sjiinq − test the SJI Jukebox Interface SJIINQ command

SYNOPSIS **sjiinq** *devname*

DESCRIPTION The **sjiinq** program tests the SJIINQ command on SJI compliant Jukeboxes (see **libsji**(1m)). This command returns a string that identifies a Jukebox. If a Jukebox doesn't support this feature, an appropriate error message will be printed out.

The *devname* argument should be any device name that can be used to reach a SJI compliant Jukebox driven by the system. Typical usage is in constructing the name that contains the SCSI Bus, the SCSI Target ID and the SCSI Lun on that target, e.g., *scsidev@0.4.0*.

SEE ALSO **libsji**(1m)

NAME sjimm − test the SJI Jukebox Interface SJIMM command

SYNOPSIS **sjimm** *jukebox* [**drive** | **slot** | **inlt** | **mt**] **src** [**drive** | **slot** | **inlt** | **mt**] **dst**

DESCRIPTION The **sjimm** program tests the SJIMM command on SJI compliant Jukeboxes (see **libsji**(1m)). This command tests the ability to move media around a Jukebox.

The *jukebox* argument should be any control port that can be used to reach a SJI compliant Jukebox driven by the system. Typical usage is in constructing the name that contains the SCSI Bus, the SCSI Target ID and the SCSI Lun on that target, e.g. *scsidev@0.4.0*.

The third and fifth arguments should be one from the set of **drive**,**slot**,**inlt** or **mt** (respectively for Drive Elements, Storage Elements, Inlet/Outlet elements and Media Transport elements). The **dst** and **src** arguments are the SJI ordinal addresses for that type of named device (see **sjirdp**(1m)).

A typical example would be

sjimm scsidev@0.4.0 slot 4 drive 1

which moves a piece of media from the fourth storage element (slot) to the first drive.

NOTE You should note that **sjimm** does **not** have any programmatic connection to the Media elements in a Jukebox. This means that if you want to move a piece of media from a drive element, unless the Jukebox has Autoeject capabilities, you will have use other, platform dependent, means to eject the media from a drive so that it can be moved by the Jukebox.

SEE ALSO **libsji**(1m), **sjirdp**(1m)

NAME sjirdp − test the SJI Jukebox Interface SJIRDP command

SYNOPSIS **sjirdp** *devname*

DESCRIPTION The **sjirdp** program tests the SJIRDP command on SJI compliant Jukeboxes (see **libsji**(1m)). This command reads SJI ordinal device positions from a Jukebox.

Typical output for this command might look like:

```
scsidev@1.0.0 has   2 DATA TRANSPORT Elements starting at address 1
scsidev@1.0.0 has   1 MEDIA TRANSPORT Element starting at address 1
scsidev@1.0.0 has  25 STORAGE Elements starting at address 1
scsidev@1.0.0 has   1 IMPORT/EXPORT Element starting at address 1
```

The *devname* argument should be any device name that can be used to reach a SJI compliant Jukebox driven by the system. Typical usage is in constructing the name that contains the SCSI Bus, the SCSI Target ID and the SCSI Lun on that target, e.g. *scsidev@0.4.0*.

SEE ALSO **libsji**(1m)

NAME sjirdtag − test the SJI Jukebox Interface SJIRTAG command

SYNOPSIS **sjirdtag** *devname*

DESCRIPTION The **sjirdtag** program tests the SJIRTAG command on SJI compliant Jukeboxes (see **libsji**(1m)). This command reads media presence and tag data from a Jukebox.

Example output for this command:

```
Tag Data for 0.2.1, Element Type DATA TRANSPORT:
      Elem[001]: tag_val=0 pres_val=1 med_pres=1 med_side=0
Tag Data for 0.2.1, Element Type STORAGE:
      Elem[001]: tag_val=0 pres_val=1 med_pres=1 med_side=0
      Elem[002]: tag_val=0 pres_val=1 med_pres=1 med_side=0
      Elem[003]: tag_val=0 pres_val=1 med_pres=1 med_side=0
      Elem[004]: tag_val=0 pres_val=1 med_pres=1 med_side=0
      Elem[005]: tag_val=0 pres_val=1 med_pres=0 med_side=0
      Elem[006]: tag_val=0 pres_val=1 med_pres=1 med_side=0
      Elem[007]: tag_val=1 pres_val=1 med_pres=1 med_side=0
            VolumeTag=<00000098>
Tag Data for 0.2.1, Element Type MEDIA TRANSPORT:
      Elem[001]: tag_val=0 pres_val=1 med_pres=0 med_side=0
```

The *devname* argument should be any device name that can be used to reach a SJI compliant Jukebox driven by the system. Typical usage is in constructing the name that contains the SCSI Bus, the SCSI Target ID and the SCSI Lun on that target, e.g. *scsidev@0.4.0*.

The output is sorted by types available, and then by ascending order.

The boolean token **tag_val** states whether the tag data is available (valid). In the example above, only the seventh storage element had valid tag data (which was the bar code numbered '00000098').

The boolean token **med_pres** states whether the Jukebox believes that there is media present at this location.

The boolean token **pres_val** is a subtle and overloaded indicator that states (if set to true, or 1), that the the token **med_pres** should be believed absolutely. If **pres_val** is not true (set to zero), or **med_pres** is true (set to one), it is probable that there *is* media at that location, but that there is some exception associated with it. A possible exception is that a tape has a missing, upsidedown, or unreadable bar code label. Also, a Jukebox cannot determine whether media that existed at one point (for example, prior to a power failure) has been removed from a Data Transport element (tape drive). If the token **pres_val** is not true (set to zero), and **med_pres** is also not true (set to zero), there probably isn't media in that location. This uncertainty is eliminated the next time an INITIALIZE ELEMENT STATUS is done (see **sjiielm**(1m)).

The token **med_size** is for two sided media (e.g. optical disks), where the Jukebox has kept track of which side is up.

SEE ALSO **libsji**(1m), **sjiielm**(1m)

NAME sjirelem − test the SJI Jukebox Interface SJIRELEM command

SYNOPSIS **sjirelem** *devname*

DESCRIPTION The **sjirelem** program tests the SJIRELEM command on SJI compliant Jukeboxes (see **libsji**(1m)). This command reads media presence and origin data from a Jukebox.

Example output for this command:

```
Element Data for 0.2.1, Element Type DATA TRANSPORT:
        Elem[001]: pres_val=1 med_pres=1 med_side=0
                Origin: type STORAGE, address 5
Element Data for 0.2.1, Element Type STORAGE:
        Elem[001]: pres_val=1 med_pres=1 med_side=0
        Elem[002]: pres_val=1 med_pres=1 med_side=0
        Elem[003]: pres_val=1 med_pres=1 med_side=0
        Elem[004]: pres_val=1 med_pres=1 med_side=0
        Elem[005]: pres_val=1 med_pres=0 med_side=0
        Elem[006]: pres_val=1 med_pres=1 med_side=0
        Elem[007]: pres_val=1 med_pres=1 med_side=0
Element Data for 0.2.1, Element Type MEDIA TRANSPORT:
        Elem[001]: pres_val=1 med_pres=0 med_side=0
```

The *devname* argument should be any device name that can be used to reach a SJI compliant Jukebox driven by the system. Typical usage is in constructing the name that contains the SCSI Bus, the SCSI Target ID and the SCSI Lun on that target, for example, *scsidev@0.4.0*.

The output is sorted by types available, and then by ascending order.

The boolean token **med_pres** states whether the Jukebox believes that there is media present at this location.

The boolean token **pres_val** is a subtle and overloaded indicator that states (if set to true, or 1), that the the token **med_pres** should be believed absolutely. If **pres_val** is not true (set to zero), and **med_pres** is true (set to one), it is probable that there *is* media at that location, but that there is some exception associated with it. A possible exception is that a tape has a missing, upside-down, or unreadable bar code label. Also, a jukebox cannot determine whether media that existed at one point (for example, prior to a power failure) has been removed from a DATA TRANSPORT element (tape drive). If the token **pres_val** is not true (set to zero), and **med_pres** is also not true (set to zero), there probably isn't media in that location. This uncertainty is eliminated the next time an INITIALIZE ELEMENT STATUS is done (see **sjiielm**(1m)).

The token **med_size** is for two-sided media (for example, optical disks), where the Jukebox has kept track of which side is up.

When the Jukebox can, it might also say where the last place a piece of media had been prior to its current location, in which case *sjirelem* will print that out.

SEE ALSO **libsji**(1m), **sjiielm**(1m)

NAME sjirjc − test the SJI Jukebox Interface SJIRJC command

SYNOPSIS **sjirjc** *devname*

DESCRIPTION The **sjirjc** program tests the SJIRJC command on SJI compliant Jukeboxes (see **libsji**(1m)). This command reads internal configuration information and options about a Jukebox and prints it out.

Example output for this command might look like:

Device: scsidev@0.2.1

Number of Drives: 1
Number Drive Pairs: 1
Number of Import/Export Elements: 0
Number of Import/Export Pairs: 1
Number of Slots: 7
Number of Slot Pairs: 1
Number of Transport Elements: 1
Number of Transport Pairs: 1

Initialize Element Status Supported
Auto Eject Supported

The *devname* argument should be any device name that can be used to reach a SJI compliant Jukebox driven by the system. Typical usage is in constructing the name that contains the SCSI Bus, the SCSI Target ID and the SCSI Lun on that target, for example, *scsidev@0.4.0*.

Since the SJI specification allows for multiple disjoint sets of elements, you will always see a count of 'Pairs' (but the author has never run across a value other than '1'). The output displays the number of man drives, slots, transports (gripper arm), and input/export elements.

In addition, various internal library options are printed out.

SEE ALSO **libsji**(1m)

NAME sjisn − test the SJI Jukebox Interface SJISN command

SYNOPSIS **sjisn** *jukebox*

DESCRIPTION The **sjisn** program tests the SJISN command on SJI compliant Jukeboxes (see **libsji**(1m)). This command retrieves and display tape drive serialization information that an SJI compliant Jukebox may return. For a description of the formatting of this information, see **inquire**(1m).

Example output for this command for a library that returns only SCSI-3 Identifiers for its drives:

```
Serial Number data for 0.100.0 (HP C7200 ):
        Library:
                Serial Number: US0EG00002
                SCSI-3 Device Identifiers:
                        ATNN:HP      C7200          US0EG00002
        Drive at element address 1:
                SCSI-3 Device Identifiers:
                        ATNN:QUANTUM DLT8000        CXA21P1213
        Drive at element address 2:
                SCSI-3 Device Identifiers:
                        ATNN:QUANTUM DLT8000        CXA18P2046
```

Example output for this command for a library that returns only serial numbers for its drives:

```
Serial Number data for 0.100.0 (HP C7200 ):
        Library:
                Serial Number: US0EG00002
                SCSI-3 Device Identifiers:
                        ATNN:HP      C7200          US0EG00002
        Drive at element address 1:
                Serial Number:  CXA21P1213
        Drive at element address 2:
                Serial Number:  CXA18P2046
```

Example output for this command for a library that returns only SCSI-3 Identifiers for its drives:

```
Serial Number data for 0.100.0 (HP C7200 ):
        Library:
                Serial Number: US0EG00002
                SCSI-3 Device Identifiers:
                        ATNN:HP      C7200          US0EG00002
        Drive at element address 1:
                Serial Number:  CXA21P1213
                SCSI-3 Device Identifiers:
                        ATNN:QUANTUM DLT8000        CXA21P1213
        Drive at element address 2:
                Serial Number:  CXA18P2046
                SCSI-3 Device Identifiers:
                        ATNN:QUANTUM DLT8000        CXA18P2046
```

The *jukebox* argument should be any control port that can be used to reach an SJI compliant jukebox driven by the system. Typical usage is in constructing the name that contains the SCSI bus, the SCSI target ID and the SCSI LUN on that target, for example, *scsidev@0.4.0*.

SEE ALSO **libsji**(1m), **libscsi**(1m), **sn**(1m)

NAME sn − return tape library drive serial numbers

SYNOPSIS **sn -a** *b.t.l*

DESCRIPTION The **sn** program will use several variations of the SCSI READ ELEMENT STATUS command to attempt to retrieve and display any serialization information that the library might have about the drives within it. For a description of the formatting of this information, see inquire(1m).

SIMPLE OPTIONS −**a** *b.t.l* Selects a specific ordinal SCSI address, where **b** is the logical SCSI bus, **t** is the SCSI target, and **l** is the SCSI logical unit number (LUN) on that target. See **libscsi**(1m).

The device selected should be a SCSI medium changer device (such as a tape library or autochanger)

Sample Output

If you specify a device that is not a medium changer:

```
The device at scsidev@4.3.0 does not seem to be a medium changer
```

If you select a library that returns only serial numbers for the drives:

```
<Medium Changer Device HP C7200 G300 at scsidev@0.100.0>
        <Medium Changer S/N>    US0EG00002
        <Medium Changer SCSI-3 Device Identifiers>
                ATNN:HP     C7200       US0EG00002
<Data Transfer Element 1>
        <Serial Number>     CXA21P1213
<Data Transfer Element 2>
        <Serial Number>     CXA18P2046
```

If you select a library that returns only SCSI-3 Device Identifiers for the drives:

```
<Medium Changer Device HP C7200 G300 at scsidev@0.100.0>
        <Medium Changer S/N>    US0EG00002
        <Medium Changer SCSI-3 Device Identifiers>
                ATNN:HP     C7200       US0EG00002
        <Data Transfer Element 1>
                <SCSI-3 Device Identifiers>
                        ATNN:QUANTUM DLT8000        CXA21P1213
        <Data Transfer Element 2>
                <SCSI-3 Device Identifiers>
                        ATNN:QUANTUM DLT8000        CXA18P2046
```

If you select a library that returns both serial numbers and SCSI-3 Device Identifiers for the drives:

```
<Medium Changer Device HP C7200 G300 at scsidev@0.100.0>
        <Medium Changer S/N>    US0EG00002
        <Medium Changer SCSI-3 Device Identifiers>
                ATNN:HP     C7200       US0EG00002
<Data Transfer Element 1>
```

```
                    <Serial Number>         CXA21P1213
                    <SCSI-3 Device Identifiers>
                            ATNN:QUANTUM DLT8000        CXA21P1213
          <Data Transfer Element 2>
                    <Serial Number>         CXA18P2046
                    <SCSI-3 Device Identifiers>
                            ATNN:QUANTUM DLT8000        CXA18P2046
```

SEE ALSO **libscsi**(1m), **inquire**(1m), **sjisn**(1m)

NAME

ssi − StorageTek silo interface module (UNIX only)
mini_el − event logger for use with ssi (UNIX only)
libstlstk − shared library for communication to ssi

SYNOPSIS

ssi [−**A** *ACSLS server*] [−**a** *ACSLS port number*]
[−**S** *SSI port number*] [−**P** *port number*]
[−**r** *retry count*] &

mini_el [−**l** *logfile*] [−**d**] [−**h**] &

libstlstk.so (Solaris)
libstlstk.so.a (AIX)
libstlstk.sl (HPUX)
libstlstk.so.1 (SGI)
libstlstk.so.1 (DYNIX/ptx)
libstlstk.so (DECAXP)
libstlstk.dll (NT i386)

DESCRIPTION

NOTE: in this document, the term "ACSLS server" will be used to indicate the name of the system that is running any one of StorageTek's library manager programs: ACSLS on a Solaris or AIX host, Library Station on an MVS host, or Horizon Library Manager on a system running Windows NT or Windows 2000.

(UNIX only)

The **ssi** command is used indirectly by **nsrjb** to communicate with an ACSLS server. **nsrjb** loads **libstlstk,** which handles the TCP calls to and from **ssi. ssi** then handles all of communication to and from the ACSLS server. Starting with ACSLS version 5.3, it is possible to run NetWorker (either a server or a storage node) on the same host that ACSLS is running on.

ssi and **mini_el** must be running on the system on which jbconfig was run to create the jukebox resource. **ssi** and **mini_el** are almost always run as background processes, and are usually started automatically by the system.

In addition to **ssi** and **mini_el,** a shared library file (usually called **libstlstk.xxx** where xxx is an operating system-dependent extension) is also required. An appropriate version of this library is installed as part of NetWorker.

New in version 2.00 of **ssi:**

ssi now supports communication with the ACSLS server on a specified port number, using the −**a** command line option. This is part of the STK firewall enhancement. The ACSLS server must be running version 7.1 to use this functionality.

While you can still start **ssi** the same way as before - using the environment variable CSI_HOSTNAME to select the ACSLS server to connect to - you can also specify the ACSLS server hostname on the command line using the **-A** option. By using the −**a** option, you may specify the port number that the **ssi** process will use when connecting to the ACSLS server. The ACSLS server must be configured to listen on this port. Using the −**S** option, the **ssi** process can be configured to listen for response messages on a specific port. You may also specify the port number used for communication between NetWorker and that particular instance of **ssi** using the **-P** option. The allowed values for this port number are 50004 (for the first instance), 50011 to 50019, and 50021 to 50099. Note that if you specify a port number that is already being used by an instance of **ssi,** the specified port cannot be used, and the next available port in the allowed range will be selected. If the port number is not specified, each successive instance of **ssi** will take the next available port starting from 50004 and going upwards. If there are no available ports in the range, **ssi** will fail to load and should

display an error message. Note that specifying the port number is not necessary for normal operation. You do not need to insure that a given ACSLS server is always accessed over a given port. NetWorker and **ssi** use the name of the ACSLS server to establish a connection on the fly.

If the −**A** option is not used to specify a hostname on the **ssi** command line, the environment variable **CSI_HOSTNAME** must be set to the name of the library server, before the **ssi** process is started. If this variable is not found, **ssi** will exit with an error message.

mini_el is an event logger used by **ssi** to maintain a log of certain events. It should be started before **ssi.** Multiple instances of **ssi** will share a single instance of **mini_el.** A header consisting of the ACSLS server name and the local TCP port that **ssi** will be listening on is included at the start of any message placed into the log by any instance of **ssi**

(NT only)

On NT, the software equivalent to **ssi** and **mini_el** must be obtained from StorageTek as their product "Library Attach for NT". This package must be installed prior to configuring a Silo in NetWorker.

NOTE: Library Attach version 1.1 includes a portmapper function that will only install properly if the NetWorker services are not running. You should use Control Panel to stop the "NetWorker Backup and Recover Server" and the "NetWorker Remote Exec Service" before installing Library Attach. After Library Attach is installed, you should use Control Panel to start "NetWorker Remote Exec Service" and "NetWorker Backup and Recover Server".

NOTE: Since EMC does not supply "Library Attach for NT", we are unable to add the multiple ACSLS host functionality to our NT version of NetWorker.

NOTE: The firewall enhancements added to the **ssi** and **mini_el** processes are not available on systems running Windows.

(All platforms)

libstlstk.xxx is a shared library that handles the communication between **nsrjb** and **ssi** or Library Attach. **ssi** or Library Attach then handles the communication over the network to the library server (either ACSLS, Library Station or Horizon Library Manager). There are no options, parameters or environment variables that affect the operation of **libstlstk.** The correct path to this file should be entered when an STK silo is configured using **jbconfig.** The default values specified by **jbconfig** match the default locations chosen for the installation program, and in most cases can be accepted.

OPTIONS

mini_el:

−**l** *logfile*
: Specifies the filename of the logfile to be created by **mini_el.** The default value is **/nsr/logs/ssi_event.log.** If present, *logfile* must be the complete path to the logfile. If the file does not exist, it will be created. If the file does exist, it will be appended to. If there is not a −**l** parameter, the default logfile **/nsr/logs/ssi_event.log** will be used.

−**d**
: Sets the debug flag. **mini_el** will output debug information.

−**h**
: Displays usage information for **mini_el.**

ssi:

−**A** *ACSLS server* is required if the CSI_HOSTNAME environment variable has not been set to the name of the system running ACSLS, LibraryStation or Horizon.

−**a** *ACSLS port number* is only required if you need to specify the port number used for communication between the **ssi** process and the ACSLS server. If the ACSLS server is configured to listen on a specific port, this value should be set to that port number.

−**S** *SSI port number* will force the ssi process to listen on the specified port number. This port is used in communications with the ACSLS server.

−**P** *port number* is only required if you need to specify the port to be used for communication between NetWorker and the ssi process. The allowed values for this port number are 50004 (for the first instance), 50011 to 50019, and 50021 to 50099. Note that if you specify a port number that is already being used by an instance of **ssi,** the specified port cannot be used, and the next available port in the allowed range will be selected.

−**r** *retry count* is only required if you need to increase the retry count for communication between ssi and the ACSLS server due to network problems.

These parameters are not position sensitive. The command line option will be parsed accordingly in the **ssi** process.

ENVIRONMENT VARIABLES

ssi:

CSI_HOSTNAME (text, up to 256 chars, there is no default)
If an ACSLS server name is not found on the command line, **ssi** will use the hostname specified by this variable. It is limited to 256 characters, and should simply be the hostname running the library server program that you are trying to connect to. If neither the command line hostname nor this environment variable specify a hostname for **ssi** to use, **ssi** will exit with an error message.

SSI_HOSTNAME (text, up to 256 chars, there is no default)
This variable is intended for use on multi-homed systems. Normally, **ssi** uses the **gethostbyname** system function to determine the name to use for this side of the connection to the ACSLS server. On a system with several network interfaces, the name supplied by that function may not result in the use of the network interface needed to communicate with the ACSLS server. On these systems, you can explicitly specify the exact name of the network interface that **ssi** will use to connect to the ACSLS server. This variable needs to be set before **ssi** is started, and may be different for various instances of **ssi** In all cases, a message will be logged in the event log stating if this environment variable was found, and if not, that **ssi** will be using the hostname returned by **gethostbyname.** This is not an error message.

SSI_BASE_SOCKET (numeric, 0 < x < 64k, no default)
If you need to restrict the socket values that **ssi** communicates on, this variable specifies the starting number for **ssi** to use when it needs to open a socket to talk to the ACSLS server. It appears that **ssi** will only open two sockets if this variable is set. The first, at **SSI_BASE_SOCKET,** will be used to connect to any host. The second, at **SSI_BASE_SOCKET + 1,** will be used for direct communication to the ACSLS server. Note that there will still be the default sockets at 50001 and 50004 used to communicate between **mini_el** and **ssi,** but any communication between this host and the ACSLS server should occur using the two sockets starting at **SSI_BASE_SOCKET.**

NOTE: This environment variable will be ignored if the −**a** option is used with a valid port number.

TIME_FORMAT (time format string,
default = "%m-%d-%y %H:%M:%S")
If you wish to see time values printed in a format other than the default of Month-Day-Year Hour:Minute:Seconds, use this variable.
%m is replaced by the current month
%d is replaced by today's date
%y is replaced by the current year
%H is replaced by the current hour
%M is replaced by the current minute
%S is replaced by the current second

CSI_CONNECT_AGETIME (seconds, 0 < x < 31536000,
default = 600)
This will set the number of seconds for network connect aging purposes.

CSI_RETRY_TIMEOUT (seconds, 0 < x < 4,294,967,295, default = 4)
This will set how long **ssi** will wait before retrying a network request.

CSI_RETRY_TRIES (numeric, 0 < x < 100, default = 5)
This will set the number of times **ssi** will retry sending a network message before reporting an error.

CSI_TCP_RPCSERVICE (boolean, default is TRUE)
This sets whether **ssi** will use TCP sockets to connect with the library server.

CSI_UDP_RPCSERVICE (boolean, default is FALSE)
This sets whether **ssi** will use UDP sockets to connect with the library server. Setting **CSI_UDP_RPCSERVICE** to **TRUE** will allow **ssi** to communicate with a **csi** that is running on the same system.

EXAMPLES

Normal STK silo setup:

```
mini_el &
ssi acsls1 &
```

− or −

```
mini_el &
setenv CSI_HOSTNAME acsls1
ssi &
```

Connect to 3 different ACSLS servers:

```
mini_el &
ssi -A acsls1 &
ssi -A acsls2 &
ssi -A acsls3 &
```

− or −

```
mini_el &
setenv CSI_HOSTNAME acsls1
ssi &
setenv CSI_HOSTNAME acsls2
ssi &
setenv CSI_HOSTNAME acsls3
ssi &
```

Connect to 3 different ACSLS servers over 3 different network interfaces:

```
mini_el &
setenv SSI_HOSTNAME myhost_on_net1
```

```
ssi -A acsls1 &
setenv SSI_HOSTNAME myhost_on_net2
ssi -A acsls2 &
setenv SSI_HOSTNAME myhost_on_net3
ssi -A acsls3 &
```

Connect to ACSLS server configured to accept connections on port 30031

```
mini_el &
ssi -A acsls1 -a 30031 &
```

– or –

```
setenv CSI_HOSTNAME acsls1
mini_el &
ssi -a 30031 &
```

To have

mini_el use /nsr/logs/ssi.log.today as its log file

```
mini_el -l /nsr/logs/ssi.log.today &
ssi -A acsls1 &
```

FILES

/nsr/logs/ssi_event.log
default logfile created/appended to by **mini_el**

SEE ALSO

nsrjb(1m), **jbconfig**(1m), **dasadmin**(1m), **libstlemass**(1m), **libstlibm**(1m)

DIAGNOSTICS

Several startup and shutdown messages along with any errors in communication between the NetWorker server and the ACSLS server will be logged in the logfile /nsr/logs/ssi_event.log (or other logfile as specified on the **mini_el** command line). The messages from any one **ssi** instance will be preceded by the name of the ACSLS server that that instance will be communicating with plus the local TCP port number that will be used between NetWorker and **ssi.**

For example:

```
10-12-00 12:31:44 SSI[0]:
[devlab-acsls/50004] ONC RPC: csi_init(): Initiation Started
source csi_init.c; line 165

10-12-00 12:33:20 SSI[0]:
[acsls2/50011] ONC RPC: csi_init(): Initiation Completed
ONC RPC: csi_init(): ACSLS server acsls2 accessed through port 50011
```

NAME stk_eject − eject tapes from a StorageTek silo

SYNOPSIS **stk_eject**
acsls_hostname **cap_id** *mode* **volser0 ... volsern**

DESCRIPTION The **stk_eject** command allows you to eject one to 42 tapes at a time from a StorageTek silo. By using the ACSLS (through **ssi** on the NetWorker system), this utility originally was intended to avoid the **nsrjb**(1m) command's limitation of ejecting only one tape per command. The limitation in **nsrjb**(1m) was removed in NetWorker version 5.5 Patch 3. However, **stk_eject** is being maintained for compatibility and for use in customer developed scripts.

Starting with version 1.1 of **stk_eject,** you will be able to eject volumes from devices controlled by more than one ACSLS server, as long as you are also using **ssi** version 1.06 or greater. The correct **ssi** to use will be automatically detected by **stk_eject** based on the ACSLS hostname passed in on the command line.

NOTE: stk_eject does **NOT** deallocate volumes for the NetWorker system. To properly maintain the NetWorker software's idea of what is and is not present in the silo, you **MUST** perform the deallocation separately using the **nsrjb -x -Txxx** command.

All parameters are required, and the list of volsers can be from 1 to 42 elements long. Volsers beyond the limit of 42 will be ignored.

acsls_hostname − 1 word host ID of system running ACSLS or Library Station - e.g. **expo1**

cap_id − STK cap name, no internal spaces - e.g. **0,0,0**

mode − should this command wait until the eject is completed?
values are **WAIT** or **NOWAIT** for *QUIET* mode
− or −
WAITV or **NOWAITV** for *VERBOSE* mode

volser − 6 character STK volser list - not checked for size or form - only first 6 characters are used. Volsers should be separated by a single space character.

*** You may eject up to 42 VOLSERS per command ***

NOTE: in **NOWAIT** or **NOWAITV** mode, there is no confirmation of ejection on this system.

If you require confirmation, you should use the **WAIT** or **WAITV** modes. This program will then receive confirmation from **ACSLS,** but it will not return to the caller until all ejected tapes have been removed from the CAP.

You will **not** receive any messages regarding emptying the CAP when it fills, or when the eject is done. You **must** use the ACSLS console to see the CAP status.

This utility uses **ssi** to communicate with **ACSLS,** so **ssi** must be properly configured and running on the system where this command is used. This utility does not depend on any other NetWorker files; it can be run on any system running **ssi** that can communicate with the desired **ACSLS** system.

SEE ALSO **STK_silo**(1m)

NAME

ssi − StorageTek silo interface module (UNIX only)
mini_el − event logger for use with ssi (UNIX only)
libstlstk − shared library for communication to ssi

SYNOPSIS

ssi [−**A***ACSLS server*] [−**a***ACSLS port number*]
[−**S***SSI port number*] [−**P***port number*]
[−**r***retry count*] &

mini_el [−**l***logfile*] [−**d**] [−**h**] &

libstlstk.so (Solaris)
libstlstk.so.a (AIX)
libstlstk.sl (HPUX)
libstlstk.so.1 (SGI)
libstlstk.so.1 (DYNIX/ptx)
libstlstk.so (DECAXP)
libstlstk.dll (NT i386)

DESCRIPTION

NOTE: in this document, the term "ACSLS server" will be used to indicate the name of the system that is running any one of StorageTek's library manager programs: ACSLS on a Solaris or AIX host, Library Station on an MVS host, or Horizon Library Manager on a system running Windows NT or Windows 2000.

(UNIX only)

The **ssi** command is used indirectly by **nsrjb** to communicate with an ACSLS server. **nsrjb** loads **libstlstk,** which handles the TCP calls to and from **ssi. ssi** then handles all of communication to and from the ACSLS server. Starting with ACSLS version 5.3, it is possible to run NetWorker (either a server or a storage node) on the same host that ACSLS is running on.

ssi and **mini_el** must be running on the system on which jbconfig was run to create the jukebox resource. **ssi** and **mini_el** are almost always run as background processes, and are usually started automatically by the system.

In addition to **ssi** and **mini_el,** a shared library file (usually called **libstlstk.xxx** where xxx is an operating system-dependent extension) is also required. An appropriate version of this library is installed as part of NetWorker.

New in version 2.00 of **ssi:**

ssi now supports communication with the ACSLS server on a specified port number, using the −**a** command line option. This is part of the STK firewall enhancement. The ACSLS server must be running version 7.1 to use this functionality.

While you can still start **ssi** the same way as before - using the environment variable CSI_HOSTNAME to select the ACSLS server to connect to - you can also specify the ACSLS server hostname on the command line using the **-A** option. By using the −**a** option, you may specify the port number that the **ssi** process will use when connecting to the ACSLS server. The ACSLS server must be configured to listen on this port. Using the −**S** option, the **ssi** process can be configured to listen for response messages on a specific port. You may also specify the port number used for communication between NetWorker and that particular instance of **ssi** using the **-P** option. The allowed values for this port number are 50004 (for the first instance), 50011 to 50019, and 50021 to 50099. Note that if you specify a port number that is already being used by an instance of **ssi,** the specified port cannot be used, and the next available port in the allowed range will be selected. If the port number is not specified, each successive instance of **ssi** will take the next available port starting from 50004 and going upwards. If there are no available ports in the range, **ssi** will fail to load and should

display an error message. Note that specifying the port number is not necessary for normal operation. You do not need to insure that a given ACSLS server is always accessed over a given port. NetWorker and **ssi** use the name of the ACSLS server to establish a connection on the fly.

If the −**A** option is not used to specify a hostname on the **ssi** command line, the environment variable **CSI_HOSTNAME** must be set to the name of the library server, before the **ssi** process is started. If this variable is not found, **ssi** will exit with an error message.

mini_el is an event logger used by **ssi** to maintain a log of certain events. It should be started before **ssi.** Multiple instances of **ssi** will share a single instance of **mini_el.** A header consisting of the ACSLS server name and the local TCP port that **ssi** will be listening on is included at the start of any message placed into the log by any instance of **ssi**

(NT only)

On NT, the software equivalent to **ssi** and **mini_el** must be obtained from StorageTek as their product "Library Attach for NT". This package must be installed prior to configuring a Silo in NetWorker.

NOTE: Library Attach version 1.1 includes a portmapper function that will only install properly if the NetWorker services are not running. You should use Control Panel to stop the "NetWorker Backup and Recover Server" and the "NetWorker Remote Exec Service" before installing Library Attach. After Library Attach is installed, you should use Control Panel to start "NetWorker Remote Exec Service" and "NetWorker Backup and Recover Server".

NOTE: Since EMC does not supply "Library Attach for NT", we are unable to add the multiple ACSLS host functionality to our NT version of NetWorker.

NOTE: The firewall enhancements added to the **ssi** and **mini_el** processes are not available on systems running Windows.

(All platforms)

libstlstk.xxx is a shared library that handles the communication between **nsrjb** and **ssi** or Library Attach. **ssi** or Library Attach then handles the communication over the network to the library server (either ACSLS, Library Station or Horizon Library Manager). There are no options, parameters or environment variables that affect the operation of **libstlstk.** The correct path to this file should be entered when an STK silo is configured using **jbconfig.** The default values specified by **jbconfig** match the default locations chosen for the installation program, and in most cases can be accepted.

OPTIONS

For mini_el:

−**l** *logfile*
: Specifies the filename of the logfile to be created by **mini_el.** The default value is **/nsr/logs/ssi_event.log.** If present, *logfile* must be the complete path to the logfile. If the file does not exist, it will be created. If the file does exist, it will be appended to. If there is not a −**l** parameter, the default logfile **/nsr/logs/ssi_event.log** will be used.

−**d**
: Sets the debug flag. **mini_el** will output debug information.

−**h**
: Displays usage information for **mini_el.**

For ssi:

−**A** *ACSLS server*

Is required if the CSI_HOSTNAME environment variable has not been set to the name of the system running ACSLS, LibraryStation or Horizon.

−**a** *ACSLS port number*

Is only required if you need to specify the port number used for communication between the **ssi** process and the ACSLS server. If the ACSLS server is configured to listen on a specific port, this value should be set to that port number.

−**S** *SSI port number*

will force the ssi process to listen on the specified port number. This port is used in communications with the ACSLS server.

−**P** *port number*

Is only required if you need to specify the port to be used for communication between NetWorker and the ssi process. The allowed values for this port number are 50004 (for the first instance), 50011 to 50019, and 50021 to 50099. Note that if you specify a port number that is already being used by an instance of **ssi,** the specified port cannot be used, and the next available port in the allowed range will be selected.

−**r** *retry count*

Is only required if you need to increase the retry count for communication between ssi and the ACSLS server due to network problems.

These parameters are not position sensitive. The command line option will be parsed accordingly in the **ssi** process.

ENVIRONMENT VARIABLES

ssi:

CSI_HOSTNAME (text, up to 256 chars, there is no default)

If an ACSLS server name is not found on the command line, **ssi** will use the hostname specified by this variable. It is limited to 256 characters, and should simply be the hostname running the library server program that you are trying to connect to. If neither the command line hostname nor this environment variable specify a hostname for **ssi** to use, **ssi** will exit with an error message.

SSI_HOSTNAME (text, up to 256 chars, there is no default)

This variable is intended for use on multi-homed systems. Normally, **ssi** uses the **gethostbyname** system function to determine the name to use for this side of the connection to the ACSLS server. On a system with several network interfaces, the name supplied by that function may not result in the use of the network interface needed to communicate with the ACSLS server. On these systems, you can explicitly specify the exact name of the network interface that **ssi** will use to connect to the ACSLS server. This variable needs to be set before **ssi** is started, and may be different for various instances of **ssi** In all cases, a message will be logged in the event log stating if this environment variable was found, and if not, that **ssi** will be using the hostname returned by **gethostbyname.** This is not an error message.

SSI_BASE_SOCKET (numeric, 0 < x < 64k, no default)

If you need to restrict the socket values that **ssi** communicates on, this variable specifies the starting number for **ssi** to use when it needs to open a socket to talk to the ACSLS server. It appears that **ssi** will only open two sockets if this variable is set. The first, at **SSI_BASE_SOCKET,** will be used to connect to any host. The second, at **SSI_BASE_SOCKET + 1,** will be used for direct

communication to the ACSLS server. Note that there will still be the default sockets at 50001 and 50004 used to communicate between **mini_el** and **ssi,** but any communication between this host and the ACSLS server should occur using the two sockets starting at **SSI_BASE_SOCKET.**

NOTE: This environment variable will be ignored if the −**a** option is used with a valid port number.

TIME_FORMAT (time format string, default = "%m−%d−%y %H:%M:%S")

If you wish to see time values printed in a format other than the default of Month-Day-Year Hour:Minute:Seconds, use this variable.

%m is replaced by the current month
%d is replaced by today's date
%y is replaced by the current year
%H is replaced by the current hour
%M is replaced by the current minute
%S is replaced by the current second

CSI_CONNECT_AGETIME (seconds, 0 < x < 31536000, default = 600)

This will set the number of seconds for network connect aging purposes.

CSI_RETRY_TIMEOUT (seconds, 0 < x < 4,294,967,295, default = 4)

This will set how long **ssi** will wait before retrying a network request.

CSI_RETRY_TRIES (numeric, 0 < x < 100, default = 5)

This will set the number of times **ssi** will retry sending a network message before reporting an error.

CSI_TCP_RPCSERVICE (boolean, default is TRUE)

This sets whether **ssi** will use TCP sockets to connect with the library server.

CSI_UDP_RPCSERVICE (boolean, default is FALSE)

This sets whether **ssi** will use UDP sockets to connect with the library server. Setting **CSI_UDP_RPCSERVICE** to **TRUE** will allow **ssi** to communicate with a **csi** that is running on the same system.

EXAMPLES

Normal STK silo setup:

```
mini_el &
ssi acsls1 &
```

− or −

```
mini_el &
setenv CSI_HOSTNAME acsls1
ssi &
```

Connect to 3 different ACSLS servers:

```
mini_el &
ssi -A acsls1 &
ssi -A acsls2 &
ssi -A acsls3 &
```

− or −

```
mini_el &
setenv CSI_HOSTNAME acsls1
ssi &
setenv CSI_HOSTNAME acsls2
ssi &
setenv CSI_HOSTNAME acsls3
ssi &
```

Connect to 3 different ACSLS servers over 3 different network interfaces:

```
mini_el &
setenv SSI_HOSTNAME myhost_on_net1
ssi -A acsls1 &
setenv SSI_HOSTNAME myhost_on_net2
ssi -A acsls2 &
setenv SSI_HOSTNAME myhost_on_net3
ssi -A acsls3 &
```

Connect to ACSLS server configured to accept connections on port 30031

```
mini_el &
ssi -A acsls1 -a 30031 &
```

– or –

```
setenv CSI_HOSTNAME acsls1
mini_el &
ssi -a 30031 &
```

To have **mini_el** use /nsr/logs/ssi.log.today as its log file

```
mini_el -l /nsr/logs/ssi.log.today &
ssi -A acsls1 &
```

FILES

/nsr/logs/ssi_event.log
default logfile created/appended to by **mini_el**

SEE ALSO

nsrjb(1m), **jbconfig**(1m), **dasadmin**(1m), **libstlemass**(1m), **libstlibm**(1m)

DIAGNOSTICS

Several startup and shutdown messages along with any errors in communication between the NetWorker server and the ACSLS server will be logged in the logfile /nsr/logs/ssi_event.log (or other logfile as specified on the **mini_el** command line). The messages from any one **ssi** instance will be preceded by the name of the ACSLS server that that instance will be communicating with plus the local TCP port number that will be used between NetWorker and **ssi.**

For example:

```
10-12-00 12:31:44 SSI[0]:
[devlab-acsls/50004] ONC RPC: csi_init(): Initiation Started
source csi_init.c; line 165

10-12-00 12:33:20 SSI[0]:
[acsls2/50011] ONC RPC: csi_init(): Initiation Completed
ONC RPC: csi_init(): ACSLS server acsls2 accessed through port 50011
```

NAME stli − Standard Tape Library Interface

DESCRIPTION STLI is the documented interface for connecting tape libraries to NetWorker. Tape libraries are often referred to as silos. Examples of tape libraries are StorageTek ACSLS silos or the IBM 3494. Tape libraries differ from other jukeboxes supported by NetWorker in two major points:

- o The tape libraries have their own volume and slot management. The volumes are accessed by NetWorker via their barcode labels. The exact location of a given tape is unknown to NetWorker.
- o The interface to the library for issuing requests like mount or unmount of particular volumes differs from that of jukeboxes, which are normally connected via a SCSI or V24 interface. Standard Tape Library commands are transmitted to a Standard Tape Library server, which receives these commands and invokes the appropriate library actions. Usually, the connection between NetWorker and the Library controller is over the network, although serial (RS-232) connections are also used. The actual connection is hidden from NetWorker by the STL library. NetWorker and other applications calling the STL server are called library clients.

To keep applications independent of differing interfaces to various tape library systems, an API called STLI (Standard Tape Library Interface) has been defined, which is used by NetWorker to invoke library requests. The STLI specifies a shared library with well defined functions that is dynamically linked to NetWorker. These STLI interface libraries, which transform the STLI function calls to library-specific calls to the proprietary tape library server, may be provided by EMC or the manufacturers of the tape library.

Not all functions specified in this paper must be implemented in the STLI library. These functions are the minimum necessary for a functional library:

```
stl_open()
stl_close()
stl_mount()
stl_unmount()
```

Implementation of stl_error() is recommended for easier use of the library and better troubleshooting.

If you wish to support dynamic device reservation these are the relevant functions:

```
stl_reserve_dev()
stl_release_dev()
stl_dev_reservation()
```

Optional fucntions for added features:

```
stl_query_volume()
stl_withdraw_volume()
stl_withdraw_volumes()
stl_deposit_volume()
stl_deposit_volumes()
stl_version()
```

stl_close()

Declaration:

```
int stl_close(char* stl);
```

Description:

Close the connection to the tape library.

<stl> is the handle returned by stl_open().

The return value on success is STL_ERR_NOERR. For error return values see Appendix: Return Values.

stl_deposit_volume

Declaration:

int stl_deposit_volume(char *stl, char *volume, char *capname)

Description:

Causes the specified volume to be inserted into the library from the specified cartridge access port (inport/export facility, mailslot....)

<stl> is the handle returned by stl_open().

<volume> is the bar code of the volume to be deposited into the library.

<capname> specifies the cartridge access port/inport-export area/mailslot to be used to insert the volume into the library. It is a character string, understood by the tape library as a name for that device. This argument can be NULL, in which case the default CAP will be used.

This function returns STL_ERR_NOERR if the volume is successfully inserted. STL_ERR_NOVOL is returned if the volume is not present and was not inserted. Other return values are possible if errors occur. See appendix for possible values.

NOTE: on some libraries, this function may not be needed.

- The IBM 3494 will automatically import any tapes placed into its in/out area. There is no 'deposit' function in the 3494's API.
- StorageTek libraries with the CAP set to automatic mode behave the same as the 3494. However, if their CAP is set to manual, then a deposit call is required.
- On EMASS/Grau libraries, a single call to deposit one volume from the EIF will deposit all available volumes. However, subsequent deposit calls will return quickly and the error can be ignored.

stl_deposit_volumes

Declaration:

int stl_deposit_volumes(char *stl, char *volumes, char *capname)

Description:

Causes the specified volumes to be inserted into the library from the specified cartridge access port (inport/export facility, mailslot....). This function will be used instead of stl_deposit_volume() if it is defined and stl_version returns 1.3 or greater. Therefore, it should be capable of functioning with either a single volume specified or with a comma separated list.

<stl> is the handle returned by stl_open().

<volumes> is a comma separated list of bar codes of the volumes to be deposited into the library. There should be no extraneous spaces added between the individual bar codes since the space character (ASCII 32) is a valid bar code character itself.

<capname> specifies the cartridge access port/inport-export area/mailslot to be used to insert the volume into the library. It is a character string, understood

by the tape library as a name for that device. This argument can be NULL, in which case the default CAP will be used.

This function returns STL_ERR_NOERR if the volume is successfully inserted. STL_ERR_NOVOL is returned if the volume is not present and was not inserted. Other return values are possible if errors occur. See Appendix: Return Values.

NOTE: on some libraries, this function may not be needed.

- The IBM 3494 will automatically import any tapes placed into its in/out area. There is no 'deposit' function in the 3494's API.
- StorageTek libraries with the CAP set to automatic mode behave in the same manner as the 3494. However, if their CAP is set to manual, then a deposit call is required.
- On EMASS/Grau libraries, a single call to deposit one volume from the EIF will in fact deposit all available volumes. However, subsequent deposit calls will return quickly and the error can be ignored.

stl_dev_reservation()

Declaration:

int stl_dev_reservation(char *stl, char *device, int *state)

Description:

Get the reservation state of device <device>.

<stl> is the handle returned by stl_open().

<device> specifies from which device to get the reservation state. It is a character string, understood by the tape library as a name for that device.

*<state> returns the reservation state:

STL_DEV_FREE: Free

STL_DEV_RESERVED: Reserved for NetWorker's use

STL_DEV_OCCUPIED: Occupied by another host

The return value on success is STL_ERR_NOERR. For error return values see appendix.

stl_error()

Declaration:

char *stl_error(void)

Description:

Gives a printable error message belonging to the preceding STLI function call. The function returns the address of a buffer which contains a message describing the status of the last STLI function call. These messages can be constant strings, for instance, the messages contained in stl.h for error codes less than 100. But these messages can also be built up with actual parameters, which exactly describe the error situation.

Error and other status information should be maintained in global static variables in the library, as this call is made without any parameters.

The function returns NULL if no message available. See Appendix: Return Values.

stl_mount()

Declaration:

int stl_mount(char* stl, char* volume, char* device);

Description:

Move volume <volume> into drive <drive>.

<stl> is the handle returned by stl_open().

<volume> is the bar code of the volume to be mounted.

<device> specifies the device, on which the volume shall be mounted. It is a character string, understood by the tape library as a name for that device.

This call may not return until the volume is loaded into the drive and the drive is ready. The exact sequence is dependent on the library. It can take several minutes to complete.

The return value on success is STL_ERR_NOERR. See Appendix: Return Values.

stl_open()

Declaration:

int stl_open(char* server, char** stl);

Description:

Connect to the tape library.

<server> is a character string which contains all information necessary to establish a connection to the tape library. The information in this string is proprietary to the special type of tape library. Generally it should be of the form:

[<host>] [<par1>=<val1> [<par2>=<val2>]...]

In most cases the string contains <host>, the nodename of a library server, which receives and serves the STLI requests over the network.

*<stl> is a handle returned for use by the other STLI functions. This handle can be used to store internal information between subsequent function calls. For some libraries, the parameter to this call may or may not be used, as environment variables may be used to hold the required configuration information.

The return value on success is STL_ERR_NOERR. See Appendix: Return Values.

stl_query_volume

Declaration:

int stl_query_volume(char *stl, char *volume)

Description:

Queries a silo to establish the presence of a volume. This function is currently used to verify the presence of a volume before allocating that volume for use with NetWorker.

<stl> is the handle returned by stl_open().

<volume> is the bar code of the volume to be mounted.

This function returns STL_ERR_NOERR if the volume is present, or STL_ERR_NOVOL if the volume is not present.

Other return values are possible if errors occur. See Appendix: Return Values.

stl_release_dev()

Declaration:

int stl_release_dev(char *stl, char *device);

Description:

Release device <device>, which has previously been reserved by stl_reserve_dev().

<stl> is the handle returned by stl_open().

<device> specifies the device to be released. It is a character string, understood by the tape library as a name for that device.

The return value on success is STL_ERR_NOERR. See Appendix: Return Values.

stl_reserve_dev()

Declaration:

int stl_reserve_dev(char *stl, char *device);

Description:

Reserves device <device> for NetWorker's use.

<stl> is the handle returned by stl_open().

<device> specifies the device to be reserved. It is a character string, understood by the tape library as a name for that device.

The return value on success is STL_ERR_NOERR. See Appendix: Return Values.

stl_version()

Declaration:

int stl_version(void)

Description:

Returns STLI version information for the STL library

This function returns the version of the STL library * 10. I.e., it returns a value of 12 for an STL library that supports the functions for STLI version 1.2.

STLI version 1.0 specified the following calls:

stl_close()
stl_dev_reservation()
stl_mount()
stl_open()
stl_release_dev()
stl_reserve_dev()
stl_unmount()

STLI version 1.1 added the following calls:

stl_query_volume()
stl_version()

STLI version 1.2 added the following calls:

stl_deposit_volume()
stl_withdraw_volume()

STLI version 1.3 added the following calls:

stl_deposit_volumes()
stl_withdraw_volumes()

Note that stl_version does not return a value that can be interpreted as an 'STL_' error. Attempting to do so will result in unpredictable results.

stl_unmount()

Declaration:

int stl_unmount(char* stl, char* volume, char* device);

Description:

Remove volume <volume> from drive <drive>.

<stl> is the handle returned by stl_open().

<volume> is the bar code of the volume to be removed.

<device> specifies the device, from which the volume shall be removed. It is a character string, understood by the tape library as a name for that device.

Either <volume> or <device> can be NULL. If both values are specified, they must be consistent.

This call will not return until the volume is ejected from the drive and returned to its slot by the library. It can therefore take several minutes to complete.

The return value on success is STL_ERR_NOERR. See Appendix: Return Values.

stl_withdraw_volume

Declaration:

int stl_withdraw_volume(char *stl, char *volume, char *capname)

Description:

Causes the specified volume to be ejected from the library through the specified cartridge access port (inport/export facility, mailslot....)

<stl> is the handle returned by stl_open().

<volume> is the bar code of the volume to be withdrawn.

<capname> specifies the cartridge access port/inport-export area/mailslot to be used to remove the volume from the silo. It is a character string, understood by the tape library as a name for that device. This argument can be NULL, in which case the default CAP will be used.

This function returns STL_ERR_NOERR if the volume is successfully withdrawn from the library. STL_ERR_NOVOL is returned if the volume is not present. STL_ERR_VOLBUSY is returned if the volume is currently in use and cannot be withdrawn.

Other return values are possible if errors occur. See Appendix: Return Values.

stl_withdraw_volumes

Declaration:

int stl_withdraw_volumes(char *stl, char *volumes, char *capname)

Description:

Causes the specified volume to be ejected from the library through the specified cartridge access port (inport/export facility, mailslot....) This function

will be used instead of stl_withdraw_volume() if it is defined and stl_version returns 1.3 or greater. Therefore, it should be capable of functioning with either a single volume specified or with a comma separated list.

<stl> is the handle returned by stl_open().

<volumes> is a comma separated list of bar codes of the volumes to be deposited into the library. There should be no extraneous spaces added between the individual bar codes since the space character (ASCII 32) is a valid bar code character itself.

<capname> specifies the cartridge access port/inport-export area/mailslot to be used to remove the volume from the silo. It is a character string, understood by the tape library as a name for that device. This argument can be NULL, in which case the default CAP will be used.

This function returns STL_ERR_NOERR if the volume is successfully withdrawn from the library. STL_ERR_NOVOL is returned if the volume is not present. STL_ERR_VOLBUSY is returned if the volume is currently in use and cannot be withdrawn.

Other return values are possible if errors occur. See Appendix: Return Values.

Appendix: Return Values

Return values 0 - 99 are reserved for common, library type independent error codes. The header file stl.h defines the common return values together with a short error message.

Return values greater 100 can be used by each STLI implementation for proprietary error codes.

It is recommended, that a STLI implementation should map all error situations to the common STLI error codes and should provide the function stl_error() for more detailed error messages. This allows NetWorker to react on known error codes, but also to forward the more detailed error messages via the user interface.

No proprietary error codes are allowed in situations, where the common error codes STL_ERR_DEVEMPTY, STL_ERR_DEVFULL or STL_ERR_ALRDYMNTED apply.

The currently defined return codes are:

Error value Meaning

STL_ERR_NOERR
: Successful call return - no error

STL_ERR_UNKNOWN
: Error, no details available

STL_ERR_CONNECT
: Cannot connect to tape library

STL_ERR_BUSY
: Tape library busy, try later

STL_ERR_ACCESS
: Permission denied (to access the library, the requested device, volume or operation)

STL_ERR_NODEV
: Device not known to the tape library or physically not available

STL_ERR_NOVOL

Volume not known to the tape library or physically not available

STL_ERR_DEVFULL
Device already loaded with another volume

STL_ERR_DEVEMPTY
Device empty

STL_ERR_DEVBUSY
Device busy

STL_ERR_ERRNO
Local UNIX error, see errno

STL_ERR_INVALID
Invalid parameter

STL_ERR_VOLBUSY
Volume already loaded in another drive or is otherwise occupied

STL_ERR_LIBRARY
Tape library internal error

STL_ERR_CONFIG
Request doesn't comply with tape library configuration

STL_ERR_DEVOCC
Device reserved by another host

STL_ERR_DEVRES
Device already reserved

STL_ERR_DEVNOTRES
Device not reserved

STL_ERR_NOTINST
STLI-Library is a dummy library

STL_ERR_NOTSUPP
Dummy function return

STL_ERR_ALRDYMNTED
Requested volume already mounted in requested device

SEE ALSO **nsrjb**(1m), **nsr_jukebox**(5), **IBM_silo**(1m), **EMASS_silo**(1m), **STK_silo**(1m)

NAME tapeexercise − exercise a tape drive

SYNOPSIS for UNIX

tapeexercise [−**vBEFP**] *devname*

for Windows

tapeexer [−**vBEFP**] *devname*

DESCRIPTION The **tapeexercise** program writes sample data to a tape, and tests to see if positioning and read operations perform as expected. It needs a write-enabled tape on the indicated (no-rewind) drive.

tapeexercise should be used with extreme caution. The following points apply to testing a tape using the tapeexercise (UNIX) or tapeexer (Windows) command:

− Do not use the tapeexercise or tapeexer command on any tape containing data that you will need later, because the tests overwrite data currently on the tape.

− After testing is completed, re-label any tape that you have used in testing. If the tape is not re-labeled, it cannot be used for subsequent backups or recoveries.

− Using barcode-labeled tapes for testing is not recommended. It is more difficult to re-label a tape with a barcode label, because the barcode label is linked to the volume id, which is in the media database.

Successful completion is indicated by a ''<test name>: test begin,'' ''<test name>: test ok'' pair for each test that is run. An example is as follows:

```
BasicTest: test begin
BasicTest: test ok
```

OPTIONS

−**v** Operate in verbose mode.

−**B** Perform only the basic test.

−**E** Perform only the EOT test.

−**F** Perform only the File Space Forward test.

If none of the options **BEFP** are set, then all of the tests are performed.

devname

The device name of the tape device under test. This should be a non-rewind device, following the local operating system convention.

EXIT STATUS The following are the error numbers with which tapeexercise could exit:

ETAPE (40) : Error in accessing and/or using the tape device
EBASICTEST (41) : Error while running the basic test
EEOTTEST (42) : Error while running the EOT test
EFSFTEST (43) : Error while running the FSF test

Note that if more than one test fails, the exit code will reflect the last test that failed.

For all other errors, tapeexercise exits with a non-zero error, usually 1.

SEE ALSO **nsrmmd**(1m),
Hardware Compatibility Guide

LIMITATIONS The **tapeexercise** program will generally fail for QIC drives, because these devices do not support all of the functionality assumed by **tapeexercise**, most importantly, they do not support back-skip-file. Such devices may work with **nsrmmd**(1m); consult the *Hardware Compatibility Guide* for a complete list of supported devices.

NAME tape_perf_test - test performance versus block size on a tape drive

SYNOPSIS **tape_perf_test** −**f** *device name* [−**t** *total test size*]
[−**x** *max blocksize*] [−**n** *min blocksize*]

DESCRIPTION The **tape_perf_test** program tests the performance of the specified tape drive using various block sizes and types of data. Results are presented on-screen, saved in a text log file and in a .CSV file for easy import into a spreadsheet for analysis. These files are given .log and .csv extensions and are named according to the rule:

vendor-product-rev-hostname-OS-date-time

where **vendor** is the SCSI inquiry data for the device's vendor, **product** is the SCSI inquiry data for product, **rev** is the SCSI inquiry data for firmware revision, **hostname** is the name of the computer the test is being run on, **OS** is the operating system running on that host, and **date** and **time** are the date and time when the test was started.

The default test size if 500MB, and block sizes from 1MB down to 16kB are used for data that is:

random

bigasm-like (pseudo-random)

copies of an executable program

2:1 compressible

3:1 compressible

4:1 compressible

all zeros

On Windows, **tape_perf_test** uses OS calls to determine the maximum block size allowed by the HBA driver, which may be less than the default or specified max size.

The user can adjust this limitation using the registry key:

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\

{SCSI card driver name}\Parameters\Device\MaximumSGList

The value is the number of 4k blocks allowed, with a range of 0 to 255 (0 to ff Hex). 255 (ff) corresponds to 1MB

OPERANDS −**f** *device*
Specifies the device to test. There should be a tape in the drive - all data will be erased.

−**t** *total test size*
Specifies the total number of MB to write and read for each test. 500 MB is sufficient for most tape drives although very high performance devices, such at LTO Ultrium 3 or SDLT 600 will need a larger amount of data to return useful results. This test times the writes and reads to a precision of 1 second, so for reasonable precision each test should take at least 30 seconds.

−**x** *max blocksize*
Specifies the largest blocksize that tape_perf_test will use in this test. This is useful for refining results or for limiting testing when large block sizes are causing errors for some reason. Acceptable values are: 1024, 768, 512, 384, 256, 192, 128, 96, 64, 32, 16

−**n** *min blocksize*

Specifies the smallest blocksize that tape_perf_test will use in this test. This is useful for refining results or for limiting testing when small block sizes are known to be too low in performance. Acceptable values are: 1024, 768, 512, 384, 256, 192, 128, 96, 64, 32, 16

Obviously, setting the max size to less than the min size will result is unexpected things happening.

EXAMPLES Sample output including drive status information:

```
% tape_perf_test -f /dev/rmt/5cbn -t 4000

Block size performance test for device: HP Ultrium 3-SCSI G1CD (/dev/rmt/5cbn)
      from 1024 kB to 16 kB with a data size of 4000 MB
 >>> results being logged to file HP-Ultrium_3-SCSI-G1CD-aurora-09-26-2004-2109.log <<<
            run on system aurora on 09-26-2004-2109
            drive identifiers:
                  serial: HU104380AR
                  atpn: ATPN:HP      Ultrium 3-SCSI  HU104380AR
                  wwnn: WWNN:50060B000029A15E
open_it returns 4
testing 1024kB blocks:
      *** test using random data:
         write returns 54 sec --> 75851 kB/s
         read returns 54 sec --> 75851 kB/s
      *** test using bigasm-like data:
         write returns 41 sec --> 99902 kB/s
         read returns 39 sec --> 105025 kB/s
      *** test using copies of /usr/bin/vi in buffer:
         write returns 39 sec --> 105025 kB/s
         read returns 37 sec --> 110702 kB/s
      *** test using 2:1 compressible data:
         write returns 26 sec --> 157538 kB/s

<<snip>>

            read returns 94 sec --> 43574 kB/s
      *** test using 4:1 compressible data:
         write returns 88 sec --> 46545 kB/s
         read returns 94 sec --> 43574 kB/s
      *** test using buffer full of zeros:
         write returns 88 sec --> 46545 kB/s
         read returns 92 sec --> 44521 kB/s
Results:
```

xfer	random		bigasm-like		executable		2:1		3:1		4:1		zeros	
size	W	R	W	R	W	R	W	R	W	R	W	R	W	R
1024	075851	075851	099902	105025	105025	110702	157538	186181	163840	178086	163840	128000	105025	080313
0768	075851	075851	097523	077283	105025	113777	163840	178086	157538	186181	163840	186181	163840	195047
0512	075851	074472	099902	107789	107789	120470	102400	077283	102400	077283	102400	078769	105025	078769
0384	075851	075851	095255	075851	095255	077283	017504	015814	151703	178086	151703	178086	151703	170666
0256	071859	074472	099902	107789	120470	132129	146285	170666	146285	163840	146285	075851	097523	077283

0192 075851/074472 091022/074472 091022/073142 095255/074472 095255/074472 095255/074472 093090/075851
0128 075851/071859 087148/071859 087148/071859 091022/070620 089043/071859 093090/070620 091022/073142
0096 075851/069423 085333/069423 085333/069423 085333/069423 089043/069423 089043/069423 087148/071859
0064 075851/066064 078769/065015 078769/065015 078769/065015 080313/066064 081920/066064 080313/068266
0032 066064/056109 064000/056109 064000/056888 064000/056888 064000/056888 065015/056888 064000/056888
0016 047627/043574 046022/043574 046545/043574 046545/043574 047080/043574 046545/043574 046545/044521

Contents of the resulting .CSV file:

HP Ultrium 3-SCSI G1CD tested on host aurora at 09-26-2004 @ 21:09
size,Wr-random,Rd-random,Wr-bigasm,Rd-bigasm,Wr-exe,Rd-exe,Wr-2:1,Rd-2:1,Wr-3:1,Rd-3:1,Wr-4:1,Rd-4:1,Wr-zeros,Rd-zeros
1024,075851,075851,099902,105025,105025,110702,157538,186181,163840,178086,163840,128000,105025,080313
0768,075851,075851,097523,077283,105025,113777,163840,178086,157538,186181,163840,186181,163840,195047
0512,075851,074472,099902,107789,107789,120470,102400,077283,102400,077283,102400,078769,105025,078769
0384,075851,075851,095255,075851,095255,077283,017504,015814,151703,178086,151703,178086,151703,170666
0256,071859,074472,099902,107789,120470,132129,146285,170666,146285,163840,146285,075851,097523,077283
0192,075851,074472,091022,074472,091022,073142,095255,074472,095255,074472,095255,074472,093090,075851
0128,075851,071859,087148,071859,087148,071859,091022,070620,089043,071859,093090,070620,091022,073142
0096,075851,069423,085333,069423,085333,069423,085333,069423,089043,069423,089043,069423,087148,071859
0064,075851,066064,078769,065015,078769,065015,078769,065015,080313,066064,081920,066064,080313,068266
0032,066064,056109,064000,056109,064000,056888,064000,056888,064000,056888,065015,056888,064000,056888
0016,047627,043574,046022,043574,046545,043574,046545,043574,047080,043574,046545,043574,046545,044521

NAME tur − test unit ready

SYNOPSIS **tur** [**-a** *b.t.l*] [**-l**]

DESCRIPTION The **tur** program will send a TEST UNIT READY command to all SCSI devices attached to the system.

−**a** *b.t.l* Selects a specific ordinal SCSI address, where **b** is the logical SCSI bus, **t** is the SCSI target, and **l** is the SCSI logical unit number (LUN) on that target. See **libscsi**(1m).

−**l** Performs a complete LUN search for all SCSI adapters in the system. This argument is accepted on all systems, but does not have any effect on HP-UX systems. Due to the method used to scan for available devices on HP-UX systems, all accessible devices are always shown, and the −**l** option has no additional effect. On all other systems, checking starts at LUN 0 for SCSI devices. The first empty LUN found will end the search for a given target ID. With the −**l** option, all LUNS present on all target IDs for all SCSI busses in the system will be checked for devices. This can take a **very long time** and should therefore be used only when necessary. For example, a Fibre Channel adapter can support 126 target IDs, each of which may have 80 or more LUNs. Checking all LUNs on this single adapter may take over 10 minutes.

SEE ALSO **libscsi**(1m)

NAME uasm − NetWorker module for saving and recovering UNIX filesystem data

SYNOPSIS **uasm −s** [**-benouv**] [**-ix**] [**−t** *time*] [**−f** *proto*] [**−p** *path*] *path...*
uasm −r [**−nuv**] [**−i** {*nNyYrR*}] [**−m** <*src*>=<*dst*>] **−z** *suffix*] [*path*] [**−P** *pass-phrase*] ...

DESCRIPTION The **uasm** command is the default filesystem ASM (Application Specific Module). It is built into **save**(1m) and **recover**(1m). **uasm** may also be called directly in a manner similar to **tar**(1). This description of **uasm** applies to all ASMs. For clarity, only **uasm** is mentioned in many of the descriptions in this man page.

uasm has two basic modes: saving and recovering. When saving, **uasm** will browse directory trees and generate a save stream (see **nsr_data**(5)), to the associated *stdout* file representing the file and directory organization. When recovering, **uasm** reads a save stream from the associated *stdin* file and creates the corresponding directories and files.

During backup sessions, the behavior of **uasm** can be controlled by *directives*. Directives control how descendent directories are searched, which files are ignored, how the save stream is generated, and how subsequent directive files are processed. (See **nsr**(5)). When browsing a directory tree, symbolic links are never followed, except in the case of **rawasm**.

ASMs can recover save streams from current or earlier versions of NetWorker. Note: older ASMs may not be able to recover files generated by newer ASMs.

The following list provides a brief description of the ASMs supplied with NetWorker:

aes

The **aes** ASM uses a software encryption algorithm to encrypt file data. **aes** uses a considerable amount of CPU resources so its benefit may be limited on low-powered systems.

always

The **always** ASM always performs a back up of a file, independent of the change time of the file.

atimeasm

The **atimeasm** is used to backup files without changing the access time of the file. This functionality is a subset of **mailasm**. On most systems, **atimeasm** uses the file *mtime* for selection and then resets the file *atime* after the backup (which changes the file *ctime).* On systems that support interfaces for maintaining the file *atime* without changing the file *ctime,* **atimeasm** has no effect, since the file *atime* is normally preserved.

compressasm

The **compressasm** uses a software compression algorithm to compress file data. This ASM does not compress directories. The amount of compression achieved is data-dependent. **compressasm** uses considerable amounts of CPU resources, so its benefits may be limited on low-powered systems.

dmfasm

The **dmfasm** is used to backup and recover files that are managed by the SGI Data Migration Facility (DMF). On backup, offline files not recalled. On recover, offline and dual-state files are recovered as recallable offline files.

holey

The **holey** ASM handles holes or blocks of zeros when backing up files and preserves these holes during recovery. On some filesystems interfaces can be used to find out the location of file hole information. Otherwise, blocks of zeros that are read from the file are skipped. This ASM is normally applied automatically and does not need to be specified.

logasm

The **logasm** enables file changes during backup sessions. **logasm** can be used for "log" files and other similar files where a file changing during a backup operation is not worth noting.

mailasm

The **mailasm** uses mail-style file locking and maintains the access time of a file, preserving "new mail has arrived" flag on most mail handlers.

mtimeasm

The **mtimeasm** is used to backup files using the file *mtime* for file selection instead of the file *ctime.*

nsrindexasm

The **nsrindexasm** is used to recover from NetWorker file index backups performed prior to Version 6. During recovery from these older index backups, **nsrindexasm** is invoked automatically by **nsrck** and **mmrecov**.

nsrmmdbasm

The **nsrmmdbasm** is used to process NetWorker's media index. Normally, **nsrmmdbasm** is invoked automatically by **savegrp** and **mmrecov**, and should not be used in NetWorker directives.

null

The **null** ASM does not back up the specified files and directories, but keeps the file name in the online index of the parent directory. If a file with **null** directive is specified as the save set to be backed up, an empty save set (typically shown in 'mminfo -v' query as '4 B' in size) is created in media index for information but the save set will not contain any recoverable data.

nullasm

nullasm is an alternate name for the **null** ASM, named for backward compatibility with earlier releases where **nullasm** was a separate executable program instead of an *internal* ASM.

posixcrcasm

The **posixcrcasm** is used to calculate a 32-bit CRC for a file during a backup. This CRC is stored along with the file and is verified when the file is restored; no verification occurs during the backup itself. Using this ASM it is possible to validate a file at restore time, but it does not provide a way to correct any detected errors.

rawasm

The **rawasm** is used to back up */dev* entries (for example, block− and character−special files) and their associated raw disk partition data. On some systems, */dev* entries are actually symbolic links to device specific names. Unlike other ASMs, **rawasm** follows symlinks, allowing the shorter */dev* name to be configured. When recovering, **rawasm** requires that the filesystem node for the raw device exist prior to the recovery. This protects against the recovery of a */dev* entry and the overwriting of data on a reconfigured disk. You can create the */dev* entry, having it refer to a different raw partition, and force an overwrite if desired. If you create the */dev* entry as a symbolic link, the data is recovered to the target of the symbolic link. Precautions should be taken when using **rawasm**, see the CAVEATS section.

skip

The **skip** ASM does not back up the specified files and directories, and does not place the filename in the online index of the parent directory. If a file with **skip** directive is specified as the save set to be backed up, an empty save set (typically shown in 'mminfo -v' query as '4 B' in size) is created in media index for information but the save set will not contain any recoverable data.

swapasm

The **swapasm** does not backup actual file data, but recreates a zero-filled file of the correct size on recovery. This ASM is used on systems where the swapping device is a swap file that must be recovered with the correct size, but the contents of the swap file are not important and do not need to be backed up.

xlateasm

The **xlateasm** translates file data so that data backed up is not immediately recognizable.

Internal ASMs are not separate programs, but are contained within all ASMs. *External* ASMs are separate programs, and are invoked as needed. External ASMs provided with NetWorker are **nsrmmdbasm** and **nsrindexasm**. All other ASMs previously listed are internal.

For security reasons, external ASM names must end in *asm* and be located in the *origin* directory, which is the same directory as the originally invoked program (typically **save** or **recover**). In some system architectures, other directories relative to the *origin* will be searched if an ASM cannot be located in the *origin* directory.

Walking ASMs traverse directory trees. The **skip**, **null**, and **nullasm** ASMs do not walk. Note that this does not mean that propagation of the directive can not be applied. The lack of walking means that e.g. all directories that *match* the specified skip pattern will be skipped completely rather than being walked - however, using **+skip** will still cause the skip pattern to be recursively applied to any directories that do *not* match the pattern. As an example, if you have a directory structure of

tmp
source
source/tmp

then a directive of

skip: tmp

will cause *only* directory tmp to be skipped, whereas

+skip: tmp

will cause tmp *and* source/tmp to both be skipped. In other words, the skip directive will still propagate through non-matching subdirectories when + is used, even though skip does not walk through matched directories.

The internal ASMs described here are modes, and a number of different internal ASMs may be applied at the same time. When an external ASM is needed to process a file, the new ASM is invoked and generates the save stream. When a *filtering* ASM is traversing a directory tree and invokes another ASM, that ASMs save stream is processed by the *filtering* ASM. Hence, while using **compressasm** to backup a directory, the **mailasm** can still be used to process the mail files correctly. Note that once different modes are set, the only way to turn them off is to explicitly match an ASM directive for **uasm**.

Auto-applied ASMs are used under certain conditions, and do not need to be specifically mentioned in a directive file. For example, when a large file only has a small number of disk blocks allocated, the **holey** ASM is automatically invoked to process the file. Auto-applied ASMs are not used when a file name matches an explicit directive.

When used in conjunction with **recover**, all standard ASMs support security at recovery time. If a file is saved with an access control list (ACL), then only the owner of the file, root or Administrator may recover the file. For files that do not contain an ACL, the standard mode bits are used to determine who may recover a file. The file's owner, root and Administrator may always recover the file. Note that when ASMs are invoked by hand, these security checking rules do not apply.

OPTIONS

All ASMs accept the options described in this section. These options are generally referred to as the *standard-asm-arguments*. ASMs may also have additional options, which must be capital letters.

Either −**s** (saving) or −**r** (recovering) mode must be specified and must precede any other options. When saving, at least one *path* argument must be specified. *Path* can be either a directory or file name.

The following options are valid for all modes:

−**n** Performs a dry run. When backing up, browse the file system, create the save stream, but do not attempt to open any files. When recovering, consume the input save stream and perform basic sanity checks, but do not create any directories or files when recovering file data.

−**u** This option makes the ASM stop when an error that would normally cause a warning occurs. This can be useful if you are recovering to a file system that may not have enough disk space or you are performing a save and you want any warnings to stop the save. If you use this option with **uasm** on recovery, it will stop if it runs out of disk space. Without this option, **uasm** will continue to try to recover each file until it has processed the entire save stream.

−**v** Turns on verbose mode. The current ASM, its arguments, and the file being processed are displayed. When a filtering ASM operating in filtering mode (processing the save stream of another ASM) modifies the stream, its name, arguments and the current file are displayed within square brackets.

When saving, the following options may also be used:

−**b** Produces a byte count. This option is similar to the −**n** option, but byte count mode will estimate the amount of data that would be produced instead of actually reading file data. (It is faster but less accurate than the −**n** option.) Byte count mode produces three numbers: the number of records (for example, files and directories), the number of bytes of header information, and the approximate number of bytes of file data. Byte count mode does not produce a save stream; its output cannot be used as input to another ASM in recover mode.

−**e** Do not generate the final "end of save stream" boolean string. This flag should only be used when an ASM invokes an external ASM and as an optimization chooses not to consume the generated save stream itself.

−**f** *proto*
Specifies the location of a *.nsr* directive file to interpret before processing any files (see **nsr**(5)). Within the directive file specified by *proto*, <<*path*>> directives must resolve to files within the directory tree being processed, otherwise their subsequent directives will be ignored.

−**i** Ignores all save directives from *.nsr* directive files found in the directory tree.

−**o** Produces a (see **nsr_data**(5)) save stream that can be handled by older NetWorker servers.

−**p** *path*
This string is prepended to the name of each file as it is output. This argument is used internally when one ASM executes another external ASM. *path* must be a properly formatted path that is either the current working directory or a trailing component of the current working directory.

−**t** *date* The date (in **nsr_getdate**(3) format) after which files were modified will be backed up.

−**x** Cross filesystem boundaries. Normally, filesystem boundaries are not crossed during walking. Symbolic links are never followed, except in the case of **rawasm**. 1.0v

When recovering, the following options may also be used:

−**i** {*nNyYrR*}

Specifies the initial default overwrite response. Only one letter can be used. When the name of the file being recovered conflicts with an existing file, the user is prompted for overwrite permission. The default response, selected by pressing [**Return**], is displayed within square brackets. Unless otherwise specified with the −**i** option, "**n**" is the initial default overwrite response. Each time a response other than the default is selected, the new response becomes the default. When either **N**, **R**, or **Y** is specified, there is no prompting (except when auto-renaming files that already end with the rename suffix) and each subsequent conflict is resolved as if the corresponding lower case letter had been selected.

The valid overwrite responses and their meanings are:

n Do not recover the current file.

N Do not recover any files with conflicting names.

y Overwrite the existing file with the recovered file.

Y Overwrite files with conflicting names.

r Rename the conflicting file. A dot, "**.**", and a suffix are appended to the name of the recovered file. If a conflict still exists, the user will be prompted again.

R Automatically renames conflicting files by appending a dot, ("**.**"), and a suffix. If a conflicting file name already ends in a "**.**" suffix, the user is prompted to avoid potential auto rename looping condition.

−**m** *src=dst*

This option maps the file names that are created. Any files that start exactly with *src* will be mapped to have the path of *dst*, replacing the leading *src* component of the path name. This option is useful for the relocation of recovered files that were backed up using absolute pathnames into an alternate directory (for example, −**m /usr/etc=.**).

−**z** *suffix*

Specifies the suffix to append when renaming conflicting files. The default suffix is "**R**". Note, since Windows platforms use the suffix to determine file type and therefore changing the suffix effectively changes the file, this option is ignored on Windows.

−**P** *pass-phrase*

Specifies an additional pass phrase to use when attempting to recover files backed up using the *aes* directive. By default the current datazone encryption key is tried as well as the key generated from the default pass phrase. Using this option causes uasm to generate an encryption key from the pass phrase and try it if the default and datazone pass phrase keys do not work. This option can be given multiple times.

path Used to restrict the files being recovered. Only files with prefixes matching *path* will be recovered. This checking is performed before any potential name mapping is done using the −**m** specification. When *path* is not specified, no checking is done.

CAVEATS

Raw partitions are often used to store active DBMS data. If your raw partition contains data managed and updated by an active DBMS product, **rawasm** alone will *not* give a consistent backup. The database must *not* be updating the data in an uncontrolled fashion while **rawasm** saves or recovers data on the partition. The partition must be offline, the database manager shutdown, or the partition placed in an appropriate state for

backup. EMC has products to assist with online database backup. Similarly if **rawasm** is used to save a partition containing a UNIX filesystem, the filesystem must be unmounted or mounted read-only to obtain a consistent backup.

Ideally, recovery of a raw partition should take place on a system configured with the same disk environment and same size partitions as the system which performed the backup. If the new partition is smaller than the original partition, the recovery will not complete successfully. If the new partition is larger than the original partition, only the amount of data originally saved will be recovered.

If the partition backed up includes the disk label − the label often contains the disk geometry − recovering this partition to a new disk also recovers the label, changing the new disks geometry to match the original disk. Similarly, if a UNIX filesystem partition is backed up using **rawasm**, recovering the partition resets all information on the partition, including timestamps concerning mount times (if applicable).

Since **rawasm does not discover the size** completed, the estimated size reported on recovery is not accurate.

EXAMPLES

Copying files

To copy all of the files in the current directory to *target_dir*, use:

uasm −s . | (cd *target_dir***; uasm −rv)**

This preserves ownership, time, and the other UNIX attributes. Only the data in holey files is copied; the holes are not copied.

Copying a file tree to an archive directory

To copy the file tree under the directory *here* to *archive* and overwrite any files **with conflicting names, use: cd** *here*

uasm −s . | (cd *archive***; uasm −r −iY)**

Change directory (**cd**) to *here* first and give the first **uasm** determining the save a relative path so that the second **uasm** performing the recover will recreate the file tree under *archive*.

Another way to achieve the same result is to use the −m option on the second **uasm performing the recover to explicitly map the path names.**

uasm −s *here* | **uasm −r −iY −m** *here=archive*

FILES

.nsr Save directive files located throughout the filesystem.

SEE ALSO

nsr(5), **nsr_directive**(5), **nsrmmdbasm**(1m), **nsrindexasm**(1m), **nsrck**(1m), **nsr_data**(5), **recover**(1m), **save**(1m), **scanner**(1m), **XDR**(3N)

NAME writebuf − write device buffer

SYNOPSIS **writebuf -a** *b.t.l* [**-m** *mode*] [**-b** *buffer-id*] [**-o** *buffer-offset*] [**-p** *plist-len*] **-f** *filename*

DESCRIPTION The **writebuf** program will send a WRITE BUFFER command to the named SCSI device. It is typically used to download new microcode to SCSI devices.

The required **-a** argument must be used to select a specific ordinal SCSI address (see *libscsi*(1m)).

The required argument set of **-f** *filename* must refer to a file containing the data to write.

OPTIONS −**b** *buffer-id* Specifies the identity of the buffer to be written.

−**o** *buffer-offset* Specifies the offset from the beginning of the buffer to begin writing.

−**p** *plist-len* Specifies the parameter list length.

SEE ALSO The ANSI SCSI-2 specification for more in-depth explanation of the arguments.

BUGS AND WARNINGS Some microcode files are large. Be sure that the platform that you run this command on, and the host adapter that attaches the device to which you are directing this command, can support sending all of the microcode in one command. Loading a fraction of the microcode and failing can be disastrous and can damage a device requiring it to be sent back to the factory. Use the *lusbinfo*(1m) command and find the I/O transfer limit size for these constraining factors. Exercise caution using this command.

SEE ALSO **libuscsi**(1m), **lusbinfo**(1m)

Addendum: Windows Only Commands

This addendum contains commands that are found only on Windows systems.

NetWorker Command (Windows Only): **ossr_director**

NAME

ossr_director -- Windows disaster recovery director program (Windows only)

SYNOPSIS

ossr_director [**-c** *client-name*] [**-s** *server-name*] [**-S** *ssid*[*/cloneid*] [**-t** <*date*>] [**-D** *debuglevel*] [**-v**] [**-p**] [**-h**] [**-?**]

DESCRIPTION

The ossr_director command is used by the Windows Diaster Recovery (DR) Wizard in a WinPE ISO to do the file recovery for system state critical volumes. The command is passed the NetWorker server and the save set ID for the disaster recovery files. With this information and working with the Windows ASR feature, a client machine's critical disks are re-created, reformatted, and the critical volume bootable system state is recovered from the NetWorker system.

OPTIONS

?

Prints the usage message for this command.

-c *client*

[Not Used] client is the name of the machine that saved the disaster recovery files. Although the program accepts this input it is currently not used.

-D *debuglevel*

Sets the debuglevel to show additional debug information in the disaster recovery logs. Levels range from 1-9 with each higher level increasing the debug output.

-h

Prints the usage message for this command.

-p

Format non-critical disks to resolve disk mis-match errors.

By default, the Windows DR recovery skips the formatting of non-critical disks. By using the -p option, any existing partitions are deleted and all disks are reformatted on the restored computer to match the layout of the system image.

However, by Microsoft specification, even if the -p option is selected, a Non-critical volume will not be reformatted if the disk signature has not changed since the backup.

This option may be useful in situations where a system fails to restore because of disk mis-match errors. In this case, the -p option may resolve those errors.

-s *server*

Specifies a NetWorker server with the save sets to restore. See nsr(1m) for a description of server selection. The ossr_director requires a server value.

-S *ssid*[*/cloneid*]

ssid specifies the save set ID for the disaster recovery save set to be recovered. When there are clone instances for a disaster recovery save set, the cloneid can also be specified.

-t *date*

[Not Used] Specifies to recover files as of the specified date. Although the program accepts this input it is not used as the save set ID has precedence.

-v

Run the program in verbose mode. This option will print out messages from the recover.exe command to the ossr_director log file.

SEE ALSO

recover(1m), nsrinfo(1m), nsr(1m), nsr_render_log(1m)

FILES

nsr/logs/recover.log

Log file used by a spawned recover.exe when recovering the Windows disaster recovery save set metadata.

nsr/logs/ossr_director.raw

Default ossr_director log file used to record progress and debug information as directed. The nsr_render_log.exe program is required to output the log data into human-readable form.

nsr/temp/volume_completion_status.txt

File used to record the status of each critical volume after disaster recovery is complete.

DIAGNOSTICS

Exit Codes

0

Normal exit. This means that a save set was correctly used to recover the client target machine's bootable system state. See the log files and system recover status file for more detailed information. Reboot the system and continue with the restore of application data as needed.

<>0

Abnormal exit. The disaster recover did not complete normally and the critical disks and volumes are in an indeterminate state. For more information see the recover logs files.