



NDMP

Network Data Management Protocol

Network Working Group
Internet Draft
Category: Informational

R. Stager, Intelliguard Software
D. Hitz, Network Appliance
September 1997

Network Data Management Protocol

Status of this Memo

This document is an Internet-Draft. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups can also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

To learn the current status of any Internet-Draft, please check the "lid-abstracts.txt" listing in the Internet-Drafts Shadow Directories on ftp.is.co.za (Africa), nic.nordu.net (Europe), munnari.oz.au (Pacific Rim), ds.internic.net (US East Coast), or ftp.isi.edu (US West Coast).

Abstract

The Network Data Management Protocol (NDMP) is an open protocol for enterprise-wide network based backup. The NDMP architecture allows network attached storage vendors to ship NDMP compliant file servers which can be used by any NDMP-compliant backup administration application. This same architecture is also used for network-attached backup devices, such as tape drives and tape libraries.

Filename: <draft-stager-iguard-netapp-backup-05.txt >

Expires: September 1998

Document Version: 3.1.3

Last Update: 7/20/98 10:04:03 AM

Revision Log:

Document Version	Update Date	Change Log
3.1.0	11/15/1997	The initial draft copy of the NDMP v3 protocol document V3 features include: Discovery, Data to data copy and mover to mover copy, NT name space support and LOG message enhancement.
3.1.1	12/17/1998	<ul style="list-style-type: none"> 1• Add the revision log section. 2• Add the section 2.3 Character and Role to describe the data server and tape server. 3• Add the figure for the data to data copy function. 4• Add the figure for the tape to tape copy function. 5• Data state diagram and mover state diagram uses the consistent typographical conventions. 6• NDMP_MOVER_WINDOW message will be used to limit the number of bytes of data written to each tape file.
3.1.2	2/10/1998	<ul style="list-style-type: none"> 1• Re-sync the protocol document and protocol file.
3.1.3	4/17/98	<ul style="list-style-type: none"> 1. Add 1.2 Scope section and 2.11 Message Authentication section. 2. Modify the NDMP_FH_ADD_FILE, NDMP_FH_ADD_DIR and NDMP_FH_ADD_NODE messages for the new structure. 3. Modify the NDMP_DATA_START_RECOVER message to use the new ndmp_name structure.

1. OVERVIEW.....	5
1.1 MOTIVATION	5
1.2 SCOPE.....	5
1.3 AUDIENCE.....	5
1.4 TERMINOLOGY	5
2. ARCHITECTURE.....	6
2.1 ARCHITECTURAL MODEL.....	6
2.2 COMPARISON ARCHITECTURES	10
2.3 CHARACTER AND ROLE	11
2.4 STATE DESCRIPTION.....	11
2.4.1 The Data State Diagram	11
2.4.2 The Mover State Diagram.....	13
2.5 PROTOCOL INTERFACES.....	15
2.5.1 Messages from NDMP Client to NDMP Server.....	15
2.5.2 Messages from NDMP Server to NDMP Client.....	16
2.6 MESSAGING PROTOCOL.....	17
2.7 HEADER.....	17
2.8 ERROR CODES.....	18
2.9 MESSAGE NUMBERS.....	20
2.10 MESSAGE DEFINITIONS.....	22
2.11 MESSAGE AUTHENTICATION.....	22
3. NDMP SERVER INTERFACES.....	23
3.1 CONNECT INTERFACE.....	23
3.1.1 Open Connection	23
3.1.2 Client Authentication	24
3.1.3 Close Connection.....	26
3.1.4 Server Authentication.....	26
3.2 CONFIG INTERFACE.....	27
3.2.1 Get Host Info	27
3.2.2 Get Server Info	28
3.2.3 Get Connection Type.....	29
3.2.4 Get Authentication Type Attribute	30
3.2.5 Get Backup Type Information.....	30
3.2.6 Get File System Information.....	33
3.2.7 Get Tape Information.....	35
3.2.8 Get SCSI Information.....	37
3.3 SCSI INTERFACE.....	38
3.3.1 Open SCSI Device.....	38
3.3.2 Close Device.....	39
3.3.3 Get SCSI State	40
3.3.4 Set SCSI Target.....	40
3.3.5 Reset Device	41
3.3.6 Reset Bus.....	42
3.3.7 Execute CDB	43
3.4 TAPE INTERFACE	44
3.4.1 Open Tape Device.....	44
3.4.2 Close Device.....	46
3.4.3 Get Tape State	46
3.4.4 MTIO.....	49
3.4.5 Write	50
3.4.6 Read.....	51
3.4.7 Execute CDB	52
3.5 DATA INTERFACE	52

3.5.1	<i>Get Data State</i>	52
3.5.2	<i>Listen</i>	55
3.5.3	<i>Connect</i>	57
3.5.4	<i>Backup</i>	57
3.5.5	<i>Recover</i>	58
3.5.6	<i>Abort</i>	61
3.5.7	<i>Stop</i>	61
3.5.8	<i>Get Env</i>	62
3.6	MOVER INTERFACE	63
3.6.1	<i>Get Mover State</i>	63
3.6.2	<i>Listen</i>	65
3.6.3	<i>Connect</i>	67
3.6.4	<i>Set Record Size</i>	67
3.6.5	<i>Set Window</i>	68
3.6.6	<i>Continue</i>	69
3.6.7	<i>Abort</i>	70
3.6.8	<i>Stop</i>	70
3.6.9	<i>Read</i>	71
3.6.10	<i>Close</i>	72
5.	NDMP CLIENT INTERFACES	73
5.1	NOTIFY INTERFACE.....	73
5.1.1	<i>Notify Data Halted</i>	73
5.1.2	<i>Notify Connected</i>	73
5.1.3	<i>Notify Mover Halted</i>	74
5.1.4	<i>Notify Mover Paused</i>	75
5.1.5	<i>Notify DATA Read</i>	76
5.2	LOG INTERFACE.....	76
5.2.1	<i>Log Message</i>	77
5.2.2	<i>File Recovered</i>	77
5.3	FILE HISTORY INTERFACE	79
5.3.1	<i>Add File</i>	79
5.3.2	<i>Add Directory</i>	82
5.3.3	<i>Add Node</i>	83
6.	REFERENCES	84
7.	SECURITY	84
8.	AUTHORS	84
9.	ACKNOWLEDGEMENT	84
	FIGURE 1. SIMPLE CONFIGURATION	6
	FIGURE 2. TWO DRIVE CONFIGURATION	7
	FIGURE 3. JUKEBOX CONFIGURATION	8
	FIGURE 4. BACKING UP NDMP HOST THROUGH THE NETWORK TO ANOTHER NDMP HOST.....	8
	FIGURE 5 TAPE TO TAPE COPY	9
	FIGURE 6 DATA TO DATA COPY	9
	FIGURE 7 DATA STATE DIAGRAM	12
	FIGURE 8 - MOVER STATE DIAGRAM.....	14

1. Overview

1.1 Motivation

The purpose of this protocol is to allow a network backup application to control the backup and retrieval of an NDMP-compliant server without installing third-party software on the server.

The control and data transfer components of the backup/restore are separated. The separation allows complete interoperability at a network level. The file system vendors need only be concerned with maintaining compatibility with one, well-defined protocol. The backup vendors can place their primary focus on the sophisticated central backup administration software.

The NDMP protocol is targeted towards the process of backup and restore. There are extensive references to these tasks. The protocol is specifically intended to support tape drives. However, the protocol can be used for other applications and support other media in the future.

1.2 Scope

This document is the specification for Network Data Management Protocol version 3.

1.3 Audience

This document is intended for use by software developers to implement Network Data Management Protocol. The reader is assumed to be familiar with network protocol specifications and with the general operation of backup software. The user is not expected to have knowledge of internal backup software behavior.

1.4 Terminology

NDMP

Network Data Management Protocol. An open protocol for enterprise-wide network-based backup.

NDMP client

The application that controls the NDMP server.

NDMP host

The host that executes the NDMP server application. Data is backed up from the NDMP host to either a local tape drive or to a backup device on a remote NDMP host.

NDMP server

The virtual state machine on the NDMP host that is controlled using the NDMP protocol. There is one of these for each connection to the NDMP host. This term is used independent of implementation.

2. Architecture

2.1 Architectural Model

The architecture is a client server model and backup management software is considered a client to the NDMP server. For every connection between the client on the backup management software host and the NDMP host, there is a virtual state machine on the NDMP host that is controlled using NDMP. This virtual state machine is referred to as the NDMP server. Each state machine controls at most one device used to run backups. The protocol is a set of XDR-encoded messages that are exchanged over a bi-directional TCP/IP connection and are used to control and monitor the state of the NDMP server and to collect detailed information about the data that is backed up

In the simplest configuration, an NDMP client will backup the data from the NDMP host to a backup device connected to the NDMP host.

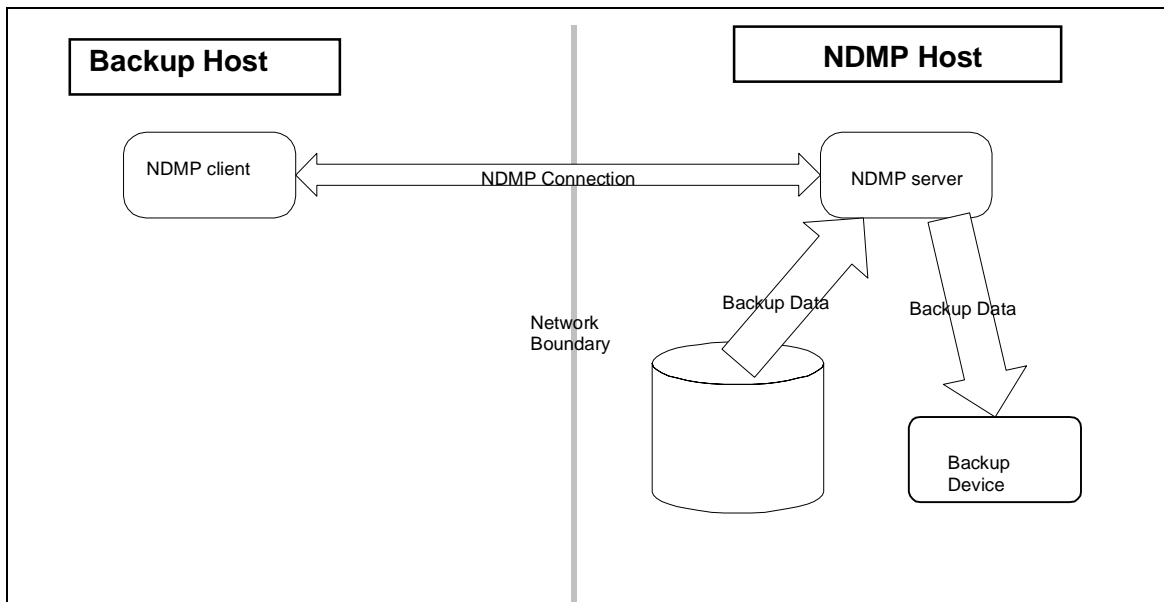


Figure 1. Simple configuration

It is also possible to use NDMP to simultaneously back up to multiple backup devices physically attached to the NDMP host. In this configuration, there are two instances of the NDMP server on the NDMP host.

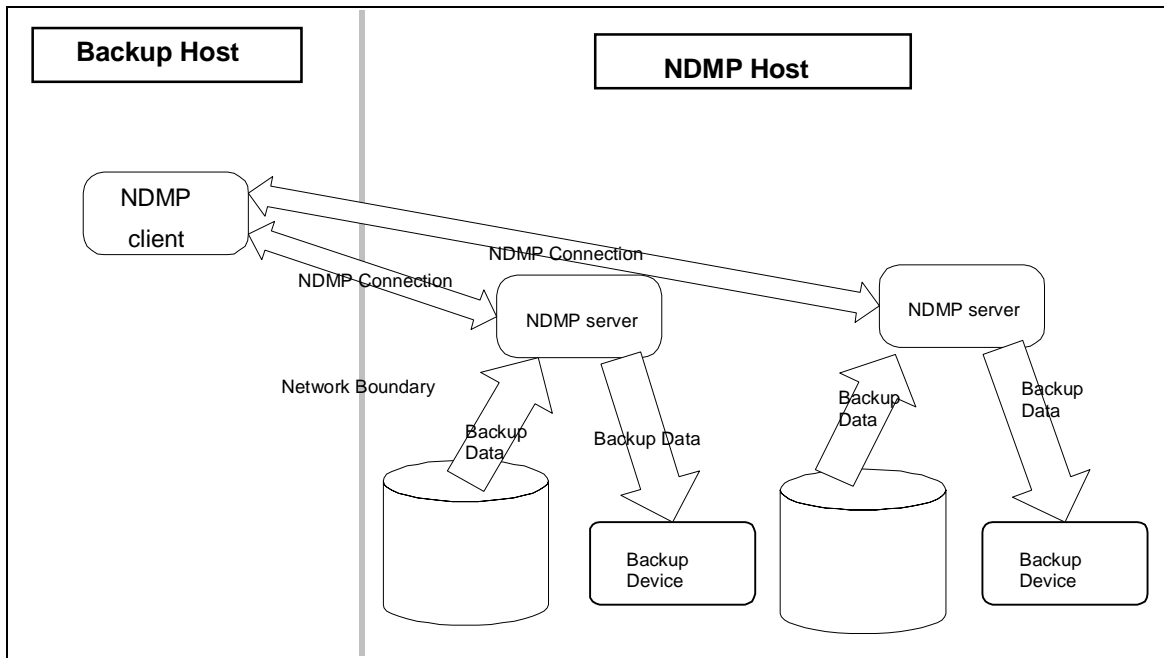


Figure 2. Two drive configuration

NDMP can be used to back up data to a backup device in a jukebox that is physically attached to the NDMP host. In this configuration, there is a separate instance of the NDMP server to control the robotics within the jukebox.

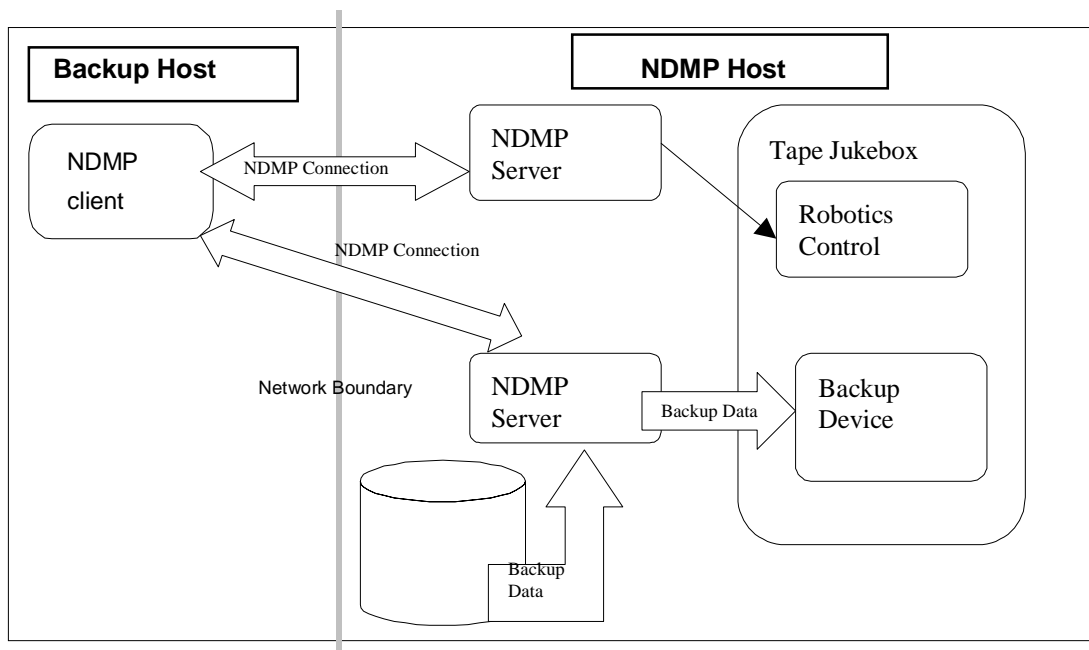
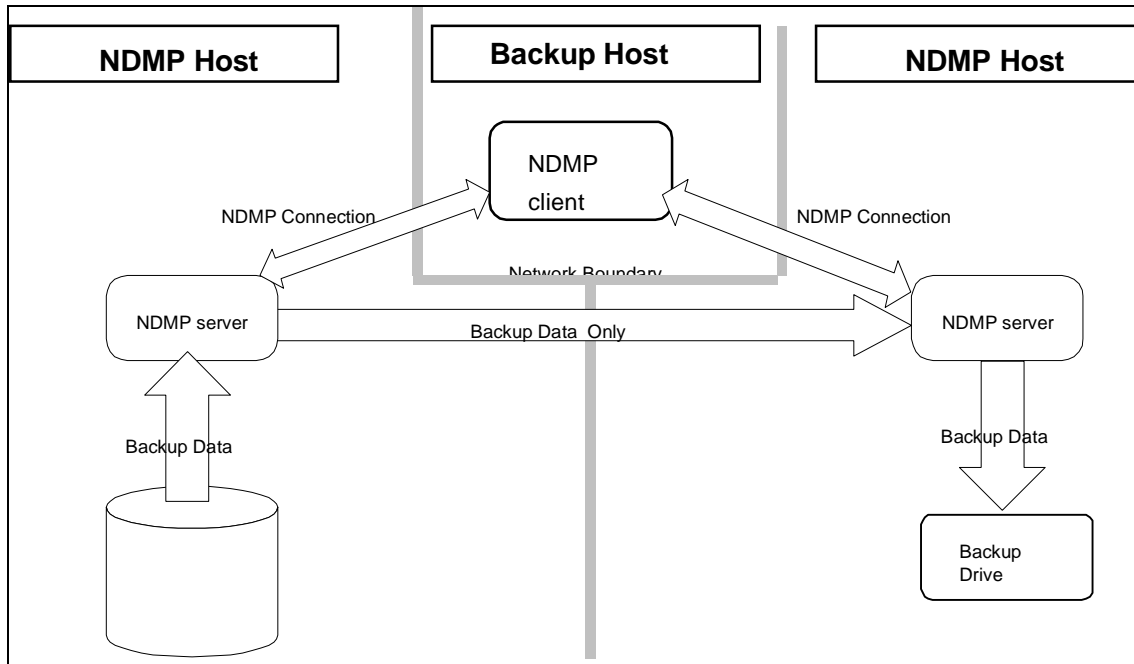


Figure 3. Jukebox configuration

It is possible to back up a host that supports NDMP but does not have a locally attached backup device by sending the data through a raw TCP/IP connection to another NDMP host.

**Figure 4. Backing up NDMP host through the network to another NDMP host**

In addition to the backup/retrieval function, NDMP supports the tape to tape or data to data copy from one NDMP server instance to another NDMP server instance.

Tape to tape copy function could be used to duplicate the backup tape for the offsite storage.

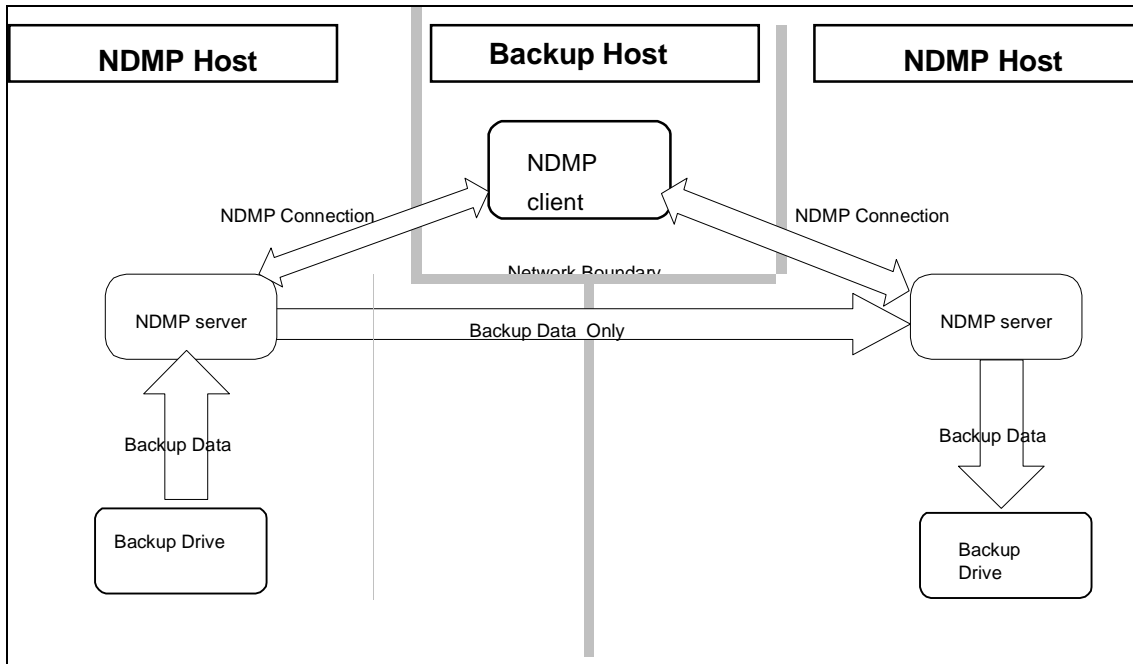


Figure 5 Tape to tape copy

Data to data copy is used to restore the entire data from one disk to the other disk.

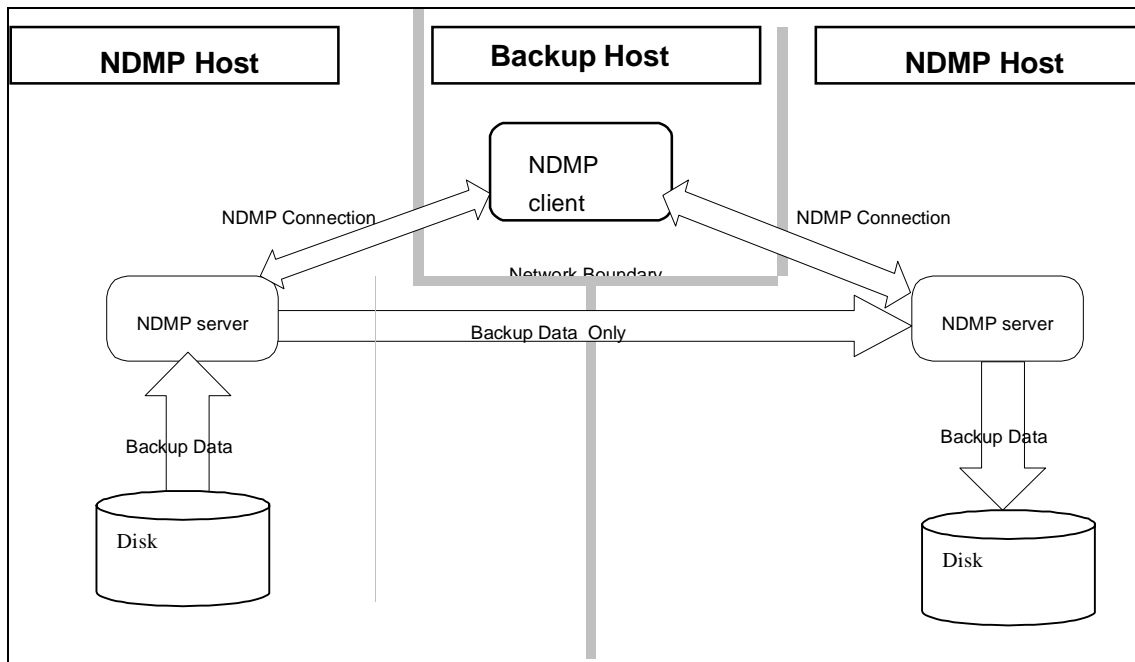


Figure 6 Data to data copy

2.2 Comparison Architectures

It is useful to compare the NDMP architecture to other architectures and note the similarities and differences.

rmt

The NDMP architecture is similar to the rmt architecture in that connection is made to a generic server and the server is instructed to open a specific tape device. NDMP differs in that it uses a TCP/IP connection to a dedicated port whereas rmt uses the rsh daemon to launch a server.

X11

The NDMP architecture is similar to the X11 architecture in that it uses a single connection to a TCP/IP port. NDMP differs in that the NDMP server is not assigned to a device until the client opens a device and that there is only one client per NDMP server, whereas X11 is assigned to a display device before the first client connects and accepts connections from many clients.

RPC

The NDMP architecture is similar to the RPC architecture in that it uses XDR encoding. NDMP differs in that it is only defined for a TCP/IP connection and that it is not a call-return model, but rather a bi-directional asynchronous messaging model.

2.3 Character and Role

An NDMP server provides two services: A data server and a tape server.

During the backup, data server reads the data from disk, generates an NDMP data stream using a specified backup format, and send the file history information, if requested, back to the NDMP client. For the retrieval, the data server will read the NDMP data stream and restore it back to the disk. The data server should not be aware of any backup device or medium issues, e.g. tape size, block size, end of medium and so on. The data server will work the same way when writing the backup data to an unlimited-size backup tape or reading the backup data from that.

The tape server either reads an NDMP data stream and writes it to tape or reads from tape and writes to the NDMP data stream, depending upon whether a backup or restore is taking place. The tape server should not be aware of the backup format, e.g. dump, tar and so on. All tape handling functions, such as split-image issues should be dealt with by this service.

2.4 State Description

Each server has a separate state diagram that dictates its behavior. The NDMP client can use the messages from the NDMP DATA interface to control the data server and trigger the data state transition. The NDMP client also can use the messages from the NDMP MOVER interface to control the tape server and change the mover state as well.

The following state diagram uses several typographical conventions:

- **This font** identifies a state.
- *This font* indicates an NDMP message.
- This font indicates the reason of a state transition.

2.4.1 The Data State Diagram

The following defines the DATA state diagram.

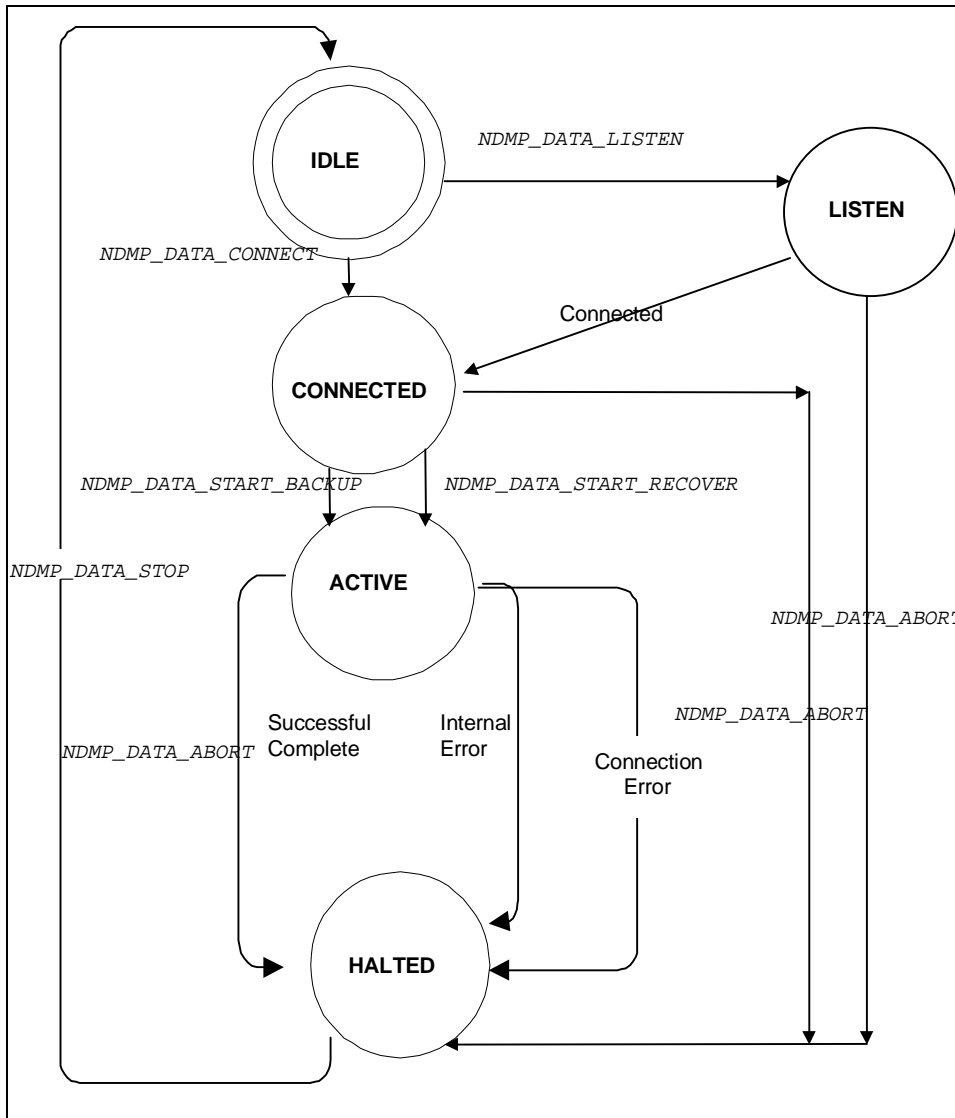


Figure 7 Data state diagram

The following defines the state machine for the data interface and the rules for transitions between states.

2.4.1.1 Idle State

Idle is the start state of the state machine.

- Transition to Listen state upon receipt of NDMP_DATA_LISTEN message.
- Transition to Connected state upon receipt of NDMP_DATA_CONNECT message.

2.4.1.2 Listen State

The NDMP server remains in Listen state while waiting for a connection from either a local or remote NDMP server.

- Transition to Connected state upon establishment of connection with an NDMP server.
- Transition to Halted state upon receipt of NDMP_DATA_ABORT message.

2.4.1.3 Connected State

Once the data connection is established, the NDMP server is in the Connected state.

- Transition to Active state upon receipt of NDMP_DATA_START_BACKUP message.
- Transition to Active state upon receipt of NDMP_DATA_START_RECOVER message.
- Transition to Halted state upon receipt of NDMP_DATA_ABORT message.

2.4.1.4 Active State

The NDMP server remains in Active state while a backup or restore is active.

- Transition to Halted state upon detection of a backup/restore error.
- Transition to Halted state upon detection of a connection error.
- Transition to Halted state upon receipt of NDMP_DATA_ABORT message.
- Transition to Halted state upon completion of backup/restore.

2.4.1.5 Halted State

The NDMP server enters Halted state after a backup/restore has either completed or been aborted.

- Transition to Idle state upon receipt of NDMP_DATA_STOP message.

2.4.2 The Mover State Diagram

The following defines the state diagram for the Mover interface.

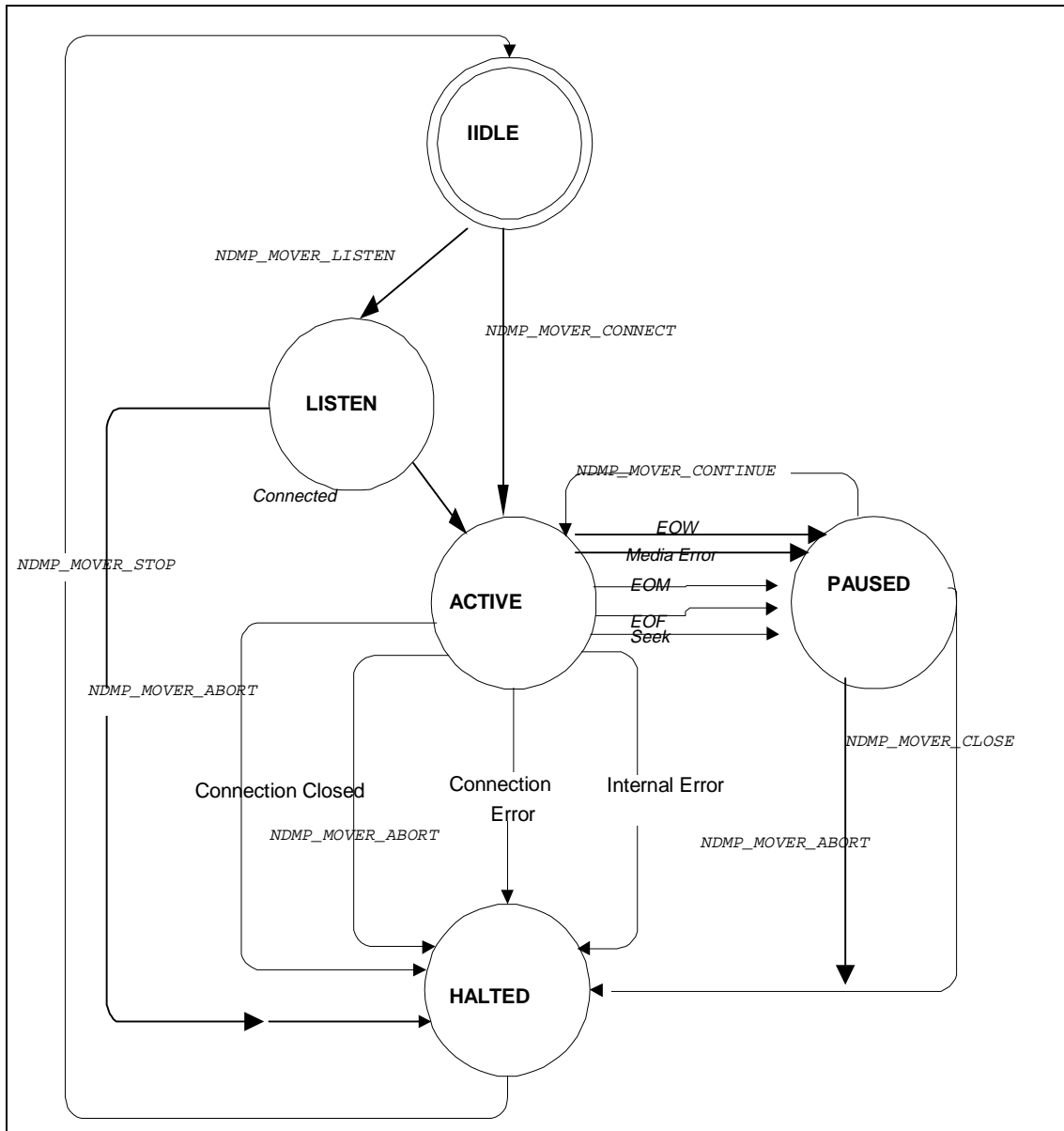


Figure 8 - Mover state diagram

2.4.2.1 Idle State

Idle is the start state of the state machine.

- Transition to Listen state upon receipt of NDMP_MOVER_LISTEN message.
- Transition to Active state upon establishment of connection with another NDMP server by an NDMP_MOVER_CONNECTED message.

2.4.2.2 Listen State

The NDMP server remains in Listen state while waiting for a connection from either a local or remote NDMP data server.

- Transition to Active state upon establishment of connection with an NDMP server.
- Transition to Halted state upon receipt of NDMP_MOVER_ABORT message.

2.4.2.3 Active State

The NDMP server remains in Active state while the data connection is active.

- Transition to Halted state upon detection of an internal error.
- Transition to Halted state upon receipt of NDMP_MOVER_ABORT message.
- Transition to Halted state upon detection of a connection error.
- Transition to Halted state upon detection of connection close.
- Transition to Paused state upon detection of EOM.
- Transition to Paused state upon detection of EOF.
- Transition to Paused state upon detection of media error.
- Transition to Paused state upon reaching end of data window.

2.4.2.4 Halted State

The NDMP server enters Halted state after a backup/restore has either completed or been aborted.

- Transition to Idle state upon receipt of NDMP_MOVER_STOP message.

2.4.2.5 Paused State

The NDMP server remains in Paused state while waiting for a tape to be changed.

- Transition to Active state upon receipt of NDMP_MOVER_CONTINUE message.
- Transition to Halted state upon receipt of NDMP_MOVER_ABORT message.
- Transition to Halted state upon receipt of NDMP_MOVER_CLOSE message.

2.5 Protocol Interfaces

Messages are grouped together by functionality into several interfaces.

2.5.1 Messages from NDMP Client to NDMP Server

The NDMP server must implement the following interfaces:

- CONNECT interface

This interface will be used after a client first establishes a connection to an NDMP server. The CONNECT interface allows the NDMP server to authenticate the client and negotiate the version of protocol used.

- CONFIG interface

This interface allows an NDMP client to discover the configuration of the NDMP server. The CONFIG interface can be used to discover NDMP server configuration and attributes.

- SCSI interface

This interface is used to pass SCSI CDBs through to a SCSI device and retrieve the resulting SCSI status. The NDMP client will use the SCSI interface to control a locally attached jukebox. Software on the NDMP client will construct SCSI CDBs and will interpret the returned status and data. The SCSI interface can also be used to exploit special features of SCSI backup devices.

- TAPE interface

This interface will support both tape positioning and tape read/write operations. The NDMP client will typically use the TAPE interface to write tape volume header and trailer files. The NDMP client will also use the TAPE interface to position the tape during backups and restores.

- DATA interface

This is the interface that actually deals with the format of the backup data. The NDMP client will initiate backups and restores using the DATA interface. The NDMP client provides all of the parameters that may affect the backup or restore using the DATA interface. The NDMP client does not place any constraints on the format of the backup data other than it must be a stream of data that can be written to the tape device.

- MOVER interface

This interface is used to control the reading/writing of backup data from/to a tape device. During a backup the MOVER reads data from the data connection, buffers the data into tape records, and writes the data to the tape device. During a restore the MOVER reads data from the tape device and writes the data to the data connection. The MOVER is responsible for handling tape exceptions and notifying the NDMP client.

2.5.2 Messages from NDMP Server to NDMP Client

The NDMP server's implementation might send the following messages to the NDMP client. All the messages that the NDMP client accepts are asynchronous. None of these messages will generate a reply message.

- NOTIFY interface

The NDMP server uses this message to notify the NDMP client that the NDMP server requires attention.

- FILE HISTORY interface

These messages allow the NDMP server to make entries in the file history for the current backup. The file history will be used by the NDMP client to select files for retrieval.

- LOG interface

These messages allow the NDMP server to make entries in the backup log. The operator uses the backup log to monitor the progress and completion status of the backup. The log is also used to diagnose problems.

2.6 Messaging Protocol

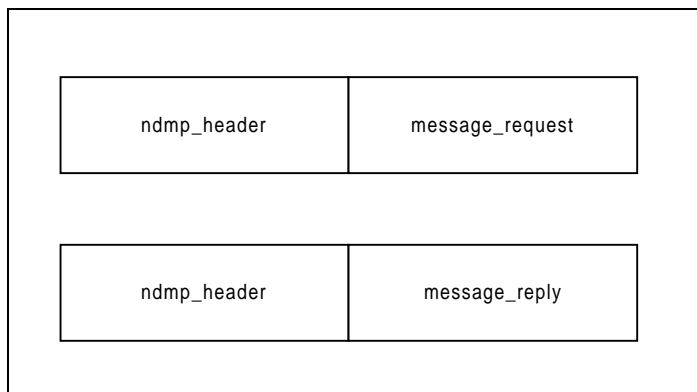
The NDMP protocol is based on XDR encoded messages transmitted over a TCP/IP connection.

NDMP messages are asynchronous. Not all request messages have an associated reply message. An NDMP message consists of a message header optionally followed by a message body. Each message is identified by a message number that is sent as part of the message header. Each message (message header plus message body) will be XDR encoded and sent within a single XDR record.

Messages that cannot be parsed or have invalid sequence information can be logged on the receiving host but no response is returned to the sender.

2.7 Header

Each message is preceded by a message header. The header is used to identify the message and defines how to deserialize the arguments and dispatch the message.



The message headers are defined by the following XDR block

```

enum ndmp_header_message_type
{
    NDMP_MESSAGE_REQUEST,
    NDMP_MESSAGE_REPLY
};
struct ndmp_header
{
    u_long          sequence;
    u_long          time_stamp;
    ndmp_header_message_type message_type;
    enum ndmp_message message;
    u_long          reply_sequence;
    ndmp_error      error;
};
  
```

Message header data definitions:

sequence	The sequence number is a connection local counter that starts at one and increases by one for every message sent. The client and the server both start with one and increase independently.
time_stamp	The time_stamp identifies the time, in seconds since 00:00:00 GMT, Jan 1, 1970, that the message was sent.
message_type	The message_type enum identifies the message as a request or a reply message.
message	The message field identifies the message.
reply_sequence	The reply_sequence field is 0 in a request message. In reply messages, the reply_sequence is the sequence number from the request message to which the reply is associated.
error	The error field is 0 in request messages. In reply messages, the error field identifies any problem that occurred receiving or decoding the message. If the error value is nonzero, no message body will follow the message header. The complete list of error codes is in the next section.

2.8 Error Codes

The following error codes are defined:

```
enum ndmp_error
{
    NDMP_NO_ERR,
    NDMP_NOT_SUPPORTED_ERR,
    NDMP_DEVICE_BUSY_ERR,
    NDMP_DEVICE_OPENED_ERR,
    NDMP_NOT_AUTHORIZED_ERR,
    NDMP_PERMISSION_ERR,
    NDMP_DEV_NOT_OPEN_ERR,
    NDMP_IO_ERR,
    NDMP_TIMEOUT_ERR,
    NDMP_ILLEGAL_ARGS_ERR,
    NDMP_NO_TAPE_LOADED_ERR,
    NDMP_WRITE_PROTECT_ERR,
    NDMP_EOF_ERR,
    NDMP_EOM_ERR,
    NDMP_FILE_NOT_FOUND_ERR,
    NDMP_BAD_FILE_ERR,
    NDMP_NO_DEVICE_ERR,
    NDMP_NO_BUS_ERR,
    NDMP_XDR_DECODE_ERR,
    NDMP_ILLEGAL_STATE_ERR,
    NDMP_UNDEFINED_ERR,
    NDMP_XDR_ENCODE_ERR,
    NDMP_NO_MEM_ERR,
    NDMP_CONNECT_ERR
};
```

NDMP_NO_ERR

No error.

NDMP_NOT_SUPPORTED_ERR

Specified message not supported. Some NDMP implementations might only support a subset of the NDMP protocol.

NDMP_DEVICE_BUSY_ERR

Specified device is in use. This error will be returned if an attempt is made to open a tape or SCSI device that is already in use.

NDMP_DEVICE_OPENED_ERR

A device is already open. NDMP connections are limited to having a single device opened at a time.

NDMP_NOT_AUTHORIZED_ERR

NDMP connection not yet authenticated. Prior to issuing most requests, the NDMP connection must first be authenticated via the NDMP_CONNECT_CLIENT_AUTH message. This error is returned if a message requiring connection authentication is received when the connection has not yet been authenticated.

NDMP_PERMISSION_ERR

The user that was used to authenticate the connection does not have the access permissions to execute this message.

NDMP_DEV_NOT_OPEN_ERR

Device not open. An attempt was made to access a device that was not open.

NDMP_IO_ERR

Device I/O error.

NDMP_TIMEOUT_ERR

Command timeout error.

NDMP_ILLEGAL_ARGS_ERR

Message received containing one or more invalid arguments.

NDMP_NO_TAPE_LOADED_ERR

Tape device could not be opened because no tape was loaded.

NDMP_WRITE_PROTECT_ERR

Tape device could not be opened in write mode because the tape is write protected.

NDMP_EOF_ERR

The tape command failed because end-of-file was encountered.

NDMP_EOM_ERR

The tape command failed because the end of media mark was encountered.

NDMP_FILE_NOT_FOUND_ERR

During a recover operation, a specified file was not found.

NDMP_BAD_FILE_ERR

Error due to invalid file descriptor.

NDMP_NO_DEVICE_ERR

Specified device does not exist.

NDMP_NO_BUS_ERR

Specified SCSI controller does not exist.

NDMP_XDR_DECODE_ERR

Error decoding message.

NDMP_ILLEGAL_STATE_ERR

Message cannot be processed in the current state.

NDMP_UNDEFINED_ERR

Undefined error.

NDMP_XDR_ENCODE_ERR

Error encoding reply message.

NDMP_NO_MEM_ERR

Memory allocation error.

NDMP_CONNECT_ERR

Error connecting to another NDMP server.

2.9 Message Numbers

The following messages are defined:

```

enum ndmp_message
{
    /* Connect Interface */
    NDMP_CONNECT_OPEN                = 0x900,
    NDMP_CONNECT_CLIENT_AUTH         = 0x901,
    NDMP_CONNECT_CLOSE               = 0x902,
    NDMP_CONNECT_SERVER_AUTH         = 0x903,

    /* Config Interface */
    NDMP_CONFIG_GET_HOST_INFO        = 0x100,
    NDMP_CONFIG_GET_CONNECTION_TYPE  = 0x102,
    NDMP_CONFIG_GET_AUTH_ATTR        = 0x103,
    NDMP_CONFIG_GET_BUTYPE_INFO      = 0x104,
    NDMP_CONFIG_GET_FS_INFO          = 0x105,
    NDMP_CONFIG_GET_TAPE_INFO        = 0x106,
    NDMP_CONFIG_GET_SCSI_INFO        = 0x107,
    NDMP_CONFIG_GET_SERVER_INFO      = 0x108,

    /* SCSI Interface */
    NDMP_SCSI_OPEN                   = 0x200,
    NDMP_SCSI_CLOSE                   = 0x201,
    NDMP_SCSI_GET_STATE               = 0x202,
    NDMP_SCSI_SET_TARGET              = 0x203,
    NDMP_SCSI_RESET_DEVICE            = 0x204,
    NDMP_SCSI_RESET_BUS               = 0x205,
    NDMP_SCSI_EXECUTE_CDB             = 0x206,

    /* Tape Interface */
    NDMP_TAPE_OPEN                   = 0x300,
    NDMP_TAPE_CLOSE                   = 0x301,
    NDMP_TAPE_GET_STATE               = 0x302,
    NDMP_TAPE_MTIO                    = 0x303,
    NDMP_TAPE_WRITE                   = 0x304,
    NDMP_TAPE_READ                    = 0x305,
    NDMP_TAPE_EXECUTE_CDB             = 0x307,

    /* Data Interface */
    NDMP_DATA_GET_STATE               = 0x400,
    NDMP_DATA_START_BACKUP             = 0x401,
    NDMP_DATA_START_RECOVER            = 0x402,
    NDMP_DATA_ABORT                   = 0x403,
    NDMP_DATA_GET_ENV                 = 0x404,
    NDMP_DATA_STOP                    = 0x407,
    NDMP_DATA_LISTEN                  = 0x409,
    NDMP_DATA_CONNECT                 = 0x40a,

    /* Notify Interface */
    NDMP_NOTIFY_DATA_HALTED           = 0x501,
    NDMP_NOTIFY_CONNECTED              = 0x502,
    NDMP_NOTIFY_MOVER_HALTED           = 0x503,
    NDMP_NOTIFY_MOVER_PAUSED           = 0x504,
    NDMP_NOTIFY_DATA_READ              = 0x505,

    /* Log Interface */
    NDMP_LOG_FILES                    = 0x602,
    NDMP_LOG_MESSAGE                   = 0x603,

    /* File history interface */
    NDMP_FH_ADD_FILE                   = 0x703,
    NDMP_FH_ADD_DIR                    = 0x704,
    NDMP_FH_ADD_NODE                   = 0x705,

    /* Mover Interface */
    NDMP_MOVER_GET_STATE               = 0xa00,

```

```

NDMP_MOVER_LISTEN           = 0xa01,
NDMP_MOVER_CONTINUE         = 0xa02,
NDMP_MOVER_ABORT            = 0xa03,
NDMP_MOVER_STOP             = 0xa04,
NDMP_MOVER_SET_WINDOW       = 0xa05,
NDMP_MOVER_READ             = 0xa06,
NDMP_MOVER_CLOSE            = 0xa07,
NDMP_MOVER_SET_RECORD_SIZE  = 0xa08,
NDMP_MOVER_CONNECT          = 0xa09,

/* Reserved for the vendor specific usage */
/* from 0xf000 to 0xffff */
NDMP_VENDORS_BASE           = 0xf000,

/* Reserved for prototyping */
/* from 0xff00 to 0xffff */
NDMP_RESERVED_BASE         = 0xff00
};

```

2.10 Message Definitions

Each message is described using a block of XDR specification in the following format:

```

struct message_name_request
{
    type    request_argument1;
    ...
    type    request_argumentN;
};

struct message_name_reply
{
    enum    ndmp_error error;
    type    reply_argument1;
    ...
    type    reply_argumentN;
};

```

Each XDR specification conforms to rpcgen format. No XDR specification is provided for the request message if the request message does not contain any arguments. No XDR specification is provided for the reply message if the reply message does not contain any argument or if no reply message is defined. Not all request messages have an associated reply message. Following the XDR specification is a description of each request and reply argument. Each reply message contains an error code. If an error code is returned that is not equal to NDMP_NO_ERR, some of the reply arguments might be meaningless. A list of errors that typically might be returned in the reply is provided for each message. Note that this is not an exhaustive list. Generic errors, such as NDMP_NO_MEM_ERR, are not listed.

2.11 Message Authentication

Each NDMP message can be processed only if the NDMP server has been authenticated except CONNECT interface and NDMP_CONFIG_GET_SERVER_INFO in the CONFIG interface.

3. NDMP Server Interfaces

This section defines the protocol interfaces implemented by the NDMP server.

3.1 *CONNECT Interface*

This interface is used to authenticate the client and negotiate the version of protocol which will be used.

The NDMP client first connects to a well known port (10,000). The NDMP server accepts the connection and sends an NDMP_NOTIFY_CONNECTED message. The NDMP client then sends an NDMP_CONNECT_OPEN message. The NDMP client will be authenticated by the NDMP server using an NDMP_CONNECT_CLIENT_AUTH message. Optionally, the NDMP client may use an NDMP_CONNECT_SERVER_AUTH message to authenticate the NDMP server as well.

3.1.1 Open Connection

It is used to negotiate the protocol version to be used between the NDMP client and server. This message is optional if the NDMP client agrees on the protocol version specified in the NDMP_NOTIFY_CONNECTED message, or it is the first message sent by the NDMP client. If the suggested protocol version is not supported on the NDMP server, an NDMP_ILLEGAL_ARGS_ERR is returned. The NDMP client could try the same request with the different protocol version supported until the negotiation is complete. Once the protocol version has been successfully negotiated, it remains until the end of the NDMP session.

Message XDR definition

```
/* NDMP_CONNECT_OPEN */
struct ndmp_connect_open_request
{
    u_short    protocol_version;
};

struct ndmp_connect_open_reply
{
    ndmp_error error;
};
```

Request Arguments

protocol_version	Protocol version suggested by the NDMP client. The valid protocol_version is 1, 2 or 3.
------------------	---

Reply Errors

NDMP_NO_ERR	Protocol version suggested by the client is supported by the server.
NDMP_ILLEGAL_ARGS_ERR	Protocol version suggested by the client is not supported by the server. The client should retry the request with a lower protocol version number.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_ILLEGAL_STATE_ERR	The protocol has been negotiated.

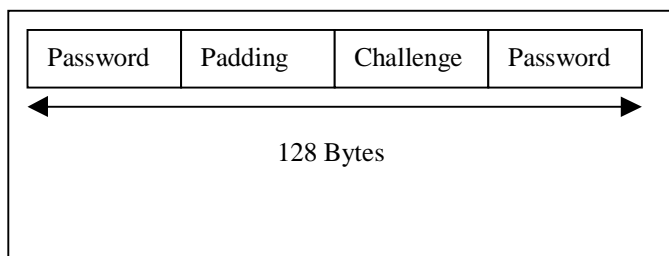
3.1.2 Client Authentication

This message is used to authenticate the NDMP client to the NDMP server. Only some of the request messages within the CONFIG and CONNECT interfaces may be processed on a connection that has not yet been authenticated. A reply message containing an NDMP_NOT_AUTHORIZED_ERR error will be returned in response to any other request messages received when the connection has not yet been authenticated.

NDMP servers must support at least one of the following authentication methods.

- NONE: no authentication required.
- TEXT: connection is authenticated using a auth id and clear text password.
- MD5: connection is authenticated using an MD5 algorithm.

The MD5 method uses the MD5 message-digest algorithm described in RFC1321 to authenticate the client and/or the server using a shared secret (password). The message used to compute the MD5 digest is a concatenation of the password, null padding, the 64 byte fixed length challenge and a repeat of the password. The length of the null padding is chosen to result in a 128 byte fixed length message. The length of the padding can be computed as $64 - 2 * (\text{length of the password})$. The client digest is computed using the server challenge from the NDMP_CONFIG_GET_AUTH_ATTR reply.



Message XDR definition


```

/* NDMP_CONNECT_CLIENT_AUTH */
enum ndmp_auth_type
{
    NDMP_AUTH_NONE,
    NDMP_AUTH_TEXT,
    NDMP_AUTH_MD5
};

struct ndmp_auth_text
{
    string      auth_id<>;
    string      auth_password<>;
};

struct ndmp_auth_md5
{
    string      auth_id <>;
    opaque      auth_digest[16];
};

union ndmp_auth_data switch (enum ndmp_auth_type auth_type)
{
    case NDMP_AUTH_NONE:
        void;
    case NDMP_AUTH_TEXT:
        struct ndmp_auth_text      auth_text;
    case NDMP_AUTH_MD5:
        struct ndmp_auth_md5      auth_md5;
};

struct ndmp_connect_client_auth_request
{
    ndmp_auth_data      auth_data;
};

struct ndmp_connect_client_auth_reply
{
    ndmp_error          error;
};

```

Request Arguments

auth_data	Authentication data. NDMP servers must support at least one of the following authentication methods:
	<ul style="list-style-type: none"> • NONE: no authentication required. • TEXT: connection is authenticated using an auth id and non-encrypted password. • MD5: connection is authenticated using auth id and MD5 digest of the server challenge and the client password.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Connection successfully authenticated.
NDMP_NOT_AUTHORIZED_ERR	Incorrect authentication data.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.
NDMP_ILLEGAL_ARGS_ERR	Specified authentication method not supported.

3.1.3 Close Connection

This message is used when the client wants to close the NDMP connection. This message should be sent by the NDMP client before shutting down the TCP/IP connection.

Message XDR definition

```
/* NDMP_CONNECT_CLOSE */
/* no request arguments */
/* no reply message */
```

3.1.4 Server Authentication

This message is used to authenticate the NDMP server to the NDMP client. This request message is optional and can only be processed after the NDMP client has been authenticated by the server.

The same client authentication methods are supported for the server authentication. Please refer to section 3.1.2 for usage of the authentication methods. If the NDMP_AUTH_MD5 authentication method is applied, the server digest will be computed using the client challenge from the request.

Message XDR definition

```
/* NDMP_CONNECT_SERVER_AUTH */
union ndmp_auth_attr
{
    switch (enum ndmp_auth_type auth_type){
        case NDMP_AUTH_NONE:
            void;
        case NDMP_AUTH_TEXT:
            void;
        case NDMP_AUTH_MD5:
            opaque challenge[64];
    };
};

struct ndmp_connect_server_auth_request
{
    ndmp_auth_attr client_attr;
};

struct ndmp_connect_server_auth_reply
{
    ndmp_error error;
    ndmp_auth_data server_result;
};
```

Request Arguments

client_attr

The following attribute is defined:

challenge For NDMP_AUTH_MD5 the NDMP client will include a per session challenge.

Reply Arguments

error	Error code.
server_result	<p>Authentication result. NDMP servers may return information to the NDMP client to authenticate the server to the client.</p> <ul style="list-style-type: none"> • NONE: no authentication returned. • TEXT: connection is authenticated using an auth id and clear text password. The auth id and password authenticate the auth id on NDMP client host. • MD5: connection is authenticated using user name and MD5 digest of the challenge and the user password.

Reply Errors

NDMP_NO_ERR	Connection successfully authenticated.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.
NDMP_ILLEGAL_ARGS_ERR	Specified authentication method not supported.

3.2 CONFIG Interface

This interface allows the NDMP client to discover the configuration of the NDMP server.

3.2.1 Get Host Info

This request is used to get NDMP host information.

Message XDR definition

```
/* NDMP_CONFIG_GET_HOST_INFO */
/* no request arguments */
struct ndmp_config_get_host_info_reply
{
    ndmp_error      error;
    string          hostname<>;
    string          os_type<>;
    string          os_vers<>;
    string          hostid<>;
};
```

Request Arguments

This request does not have a message body.

Reply Arguments

error	Error code.
hostname	Host name of the NDMP server
os_type	Name of NDMP server operating system (for example, Solaris).
os_vers	Version of NDMP server operating system (for example, 2.5).
hostid	NDMP server host identifier.

Reply Errors

NDMP_NO_ERR	Request successfully processed.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.

3.2.2 Get Server Info

This request is used to get information about the NDMP server.

Message XDR definition

```

/* NDMP_CONFIG_GET_SERVER_INFO */
/* no request arguments */
struct ndmp_config_get_server_info_reply
{
    ndmp_error      error;
    string          vendor_name<>;
    string          product_name<>;
    string          revision_number<>;
    ndmp_auth_type  auth_type<>;
};

```

Request Arguments

This request does not have a message body.

Reply Arguments

error	Error code.
vendor_name	The name of the vendor that implements the NDMP server.
product_name	The product name of the NDMP server provided by the vendor.
revision_number	The current revision number of NDMP server provided by the vendor.
auth_types	Connection authentication types supported by the NDMP server.

Reply Errors

NDMP_NO_ERR	Request successfully processed.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.

3.2.3 Get Connection Type

This request returns a list of the data connection types which NDMP server supports.

Message XDR definition

```

/* NDMP_CONFIG_GET_CONNECTION_TYPE */
/* no request arguments */
enum ndmp_addr_type
{
    NDMP_ADDR_LOCAL,
    NDMP_ADDR_TCP,
    NDMP_ADDR_FC,
    NDMP_ADDR_IPC
};

struct ndmp_config_get_connection_type_reply
{
    ndmp_error      error;
    ndmp_addr_type  addr_types<>;
};

```

Request Arguments

This request does not have a message body.

Reply Arguments

error	Error code.
addr_types	Array of supported connection types.
NDMP_ADDR_LOCAL	One tape server should listen for a connection from the data server that is collocated with the tape. This means that the data server and the tape server are controlled via the same NDMP client connection. The communication mechanism is implementation dependent.
NDMP_ADDR_TCP	One NDMP server should listen for a connection from a remote NDMP server using a TCP/IP port.
NDMP_ADDR_FC	The connection between one NDMP server and the other NDMP server is built upon the Fibre Channel.
NDMP_ADDR_IPC	Two NDMP servers are on the same host, but controlled by the separate NDMP client connection.

Reply Errors

NDMP_NO_ERR	Returned the supported connection type successfully.
-------------	--

NDMP_NOT_SUPPORTED_ERR The request sequence is not supported for this implementation.

3.2.4 Get Authentication Type Attribute

This message is used to query the attributes of the supported authentication methods. If the connection will be authenticated using the MD5 method, the client should use this message to get the server challenge before sending the NDMP_CONNECT_CLIENT_AUTH message.

Message XDR definition

```
/* NDMP_CONFIG_GET_AUTH_ATTR */

struct ndmp_config_get_auth_attr_request
{
    ndmp_auth_type auth_type;
};

struct ndmp_config_get_auth_attr_reply
{
    ndmp_error      error;
    ndmp_auth_attr server_attr;
};
```

Request Arguments

auth_type The specific authentication method is used to authenticate the NDMP client to the NDMP server.

Reply Arguments

error Error code.

server_attr Returned the attributes required for a specific authentication scheme:

The following attribute is defined:

challenge For NDMP_AUTH_MD5, the NDMP server will return a per session challenge.

Reply Errors

NDMP_NO_ERR Returned the specific authentication type attributes successfully.

NDMP_NOT_SUPPORTED_ERR The request sequence is not supported for this implementation.

NDMP_ILLEGAL_ARGS_ERR Specified authentication method not supported.

3.2.5 Get Backup Type Information

This message is used to query the backup types supported by the NDMP server and the capability of each supported backup type.

Message XDR definition

```

/* NDMP_CONFIG_GET_BUTYPE_INFO */
/* No request arguments */

/* backup type attributes */
const NDMP_BTYPE_BACKUP_FILE_HISTORY =      0x0001;
const NDMP_BTYPE_BACKUP_FILELIST =          0x0002;
const NDMP_BTYPE_RECOVER_FILELIST =         0x0004;
const NDMP_BTYPE_BACKUP_DIRECT =            0x0008;
const NDMP_BTYPE_RECOVER_DIRECT =           0x0010;
const NDMP_BTYPE_BACKUP_INCREMENTAL =       0x0020;
const NDMP_BTYPE_RECOVER_INCREMENTAL =      0x0040;
const NDMP_BTYPE_BACKUP_UTF8 =              0x0080;
const NDMP_BTYPE_RECOVER_UTF8 =             0x0100;

struct ndmp_butype_info
{
    string      butype_name <>;
    ndmp_pval   default_env <>;
    u_long      attrs;
};

struct ndmp_config_get_butype_attr_reply
{
    ndmp_error      error;
    ndmp_butype_info butype_info <>;
};

```

Request Arguments

This request does not have a message body.

Reply Arguments

error	Error code.
butype_info	Returned the information of the backup types supported by the NDMP server. Backup types are NDMP server implementation dependent.

The following information is provided:

butype_name	Name of backup (such as dump, tar, cpio).
default_env	The default value of the environment variables specific to the backup type.

The following are examples of the environment variables that can be defined by the NDMP server and the corresponding default values.

Variable Name	Meaning	Value	The Default Value
PREFIX	prefix path for the request	path name	(No default value)

TYPE	the data type	Backup type, e.g. dump, tar, and cpio	(No default value)
USER	user id to run backup	user name	(No default value)
DIRECT	Direct access retrieval	y/n	n
HIST	a flag to maintain file history	y/n	n

The following are examples of the environment variables that can be defined by dump type and the corresponding default values.

Variable Name	Meaning	Value	The Default Value
FILESYSTEM	device or file system name to be backed up	file system or device name, e.g. /dev/rsd0a	(No default value)
LEVEL	dump level	0 - 9	0
EXTRACT	“y” specifies the -x option for the extraction, or -r option for the extraction.	y/n	y
UPDATE	update the dumptypes file	TRUE/FALSE	TRUE

The following are examples of environment variables that can be defined by tar type and the corresponding default values.

Variable Name	Meaning	Value	The Default Value
FILES	list of files to be backed up	e.g. /* /*.c /*.h	(No default value)

The following are examples of environment variables that can be defined by cpio type and the corresponding default values.

Variable Name	Meaning	Value	The Default Value
CMD	command to generate the file list for cpio.	e.g. find . -name - print	(No default value)

attrs

Backup attributes bit mask. The following attribute bits are defined:

NDMP_BUTYPE_BACKUP_FILE_HISTORY	The backup type supports the file history.
NDMP_BUTYPE_BACKUP_FILELIST	The backup type supports archiving of selective files as specified by a file list.
NDMP_BUTYPE_RECOVER_FILELIST	The backup type supports restoration of individual files.
NDMP_BUTYPE_BACKUP_DIRECT	The backup type provides the file history information for the direct access restore.
NDMP_BUTYPE_RECOVER_DIRECT	The backup type supports direct access restore (positioning to the offset of a backup image and restoring the specified file).
NDMP_BUTYPE_BACKUP_INCREMENTAL	The backup type supports the incremental backup.
NDMP_BUTYPE_RECOVER_INCREMENTAL	The backup type supports incremental-only restoration (a full restore must be done before an incremental restore).
NDMP_BUTYPE_BACKUP_UTF8	The backup type supports UTF8 format in the file history.
NDMP_BUTYPE_RECOVER_UTF8	The backup type supports UTF8 format in the restored file list.

Reply Errors

NDMP_NO_ERR	The backup type information successfully returned.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.

3.2.6 Get File System Information

This message is used to query the information of the file systems on the NDMP server host.

Message XDR definition

```

/* NDMP_CONFIG_GET_FS_INFO */

/* No request arguments */
/* invalid bit */
const NDMP_FS_INFO_TOTAL_SIZE_INVALID = 0x00000001;
const NDMP_FS_INFO_USED_SIZE_INVALID = 0x00000002;
const NDMP_FS_INFO_AVAIL_SIZE_INVALID = 0x00000004;
const NDMP_FS_INFO_TOTOAL_INODES_INVALID = 0x00000008;
const NDMP_FS_INFO_USED_INNODES_INVALID = 0x00000010;

struct ndmp_fs_info
{
    u_long        invalid;
    string        fs_type <>;
    string        fs_logical_device <>;
    string        fs_physical_device <>;
    ndmp_u_quad   total_size;
    ndmp_u_quad   used_size;
    ndmp_u_quad   avail_size;
    ndmp_u_quad   total_inodes;
    ndmp_u_quad   used_inodes;
    ndmp_pval     fs_env <>;
    string        fs_status<>;
};

struct ndmp_config_get_fs_info_reply
{
    ndmp_error error;
    ndmp_fs_info fs_info <>;
};

```

Request Arguments

This request does not have a message body.

Reply Arguments

error	Error code.
fs_info	Returned the information of the file system.

The following attributes are defined:

invalid	The <code>invalid</code> flag is used to identify the unsupported argument in the message.
fs_type	The type of the file system.
fs_logical_device	The mount point or share name of the file system.
fs_physical_device	The physical device name of the file system, e.g. <code>/dev/rsd0c</code>
total_size	The total size of the file system in bytes. NDMP_FS_INFO_TOTAL_SIZE_INVALID is set if this argument is not supported.

used_size	The used size of the file system in bytes. NDMP_FS_INFO_USED_SIZE_INVALID is set if this argument is not supported.
avail_size	The available size of the file system in bytes. NDMP_FS_INFO_AVAIL_SIZE_INVALID is set if this argument is not supported.
total_inodes	The total number of inodes within the file system. NDMP_FS_INFO_TOTAL_INODES_INVALID is set if this argument is not supported.
used_inodes	The number of the inodes being used within the file system. NDMP_FS_INFO_USED_INODES_INVALID is set if this argument is not supported.
fs_env	The environment variables are defined for the file system.
fs_status	The current status of the file system.

Reply Errors

NDMP_NO_ERR	Return the file system information successfully.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.

3.2.7 Get Tape Information

This message is used to query the information of the tape devices on the NDMP server host.

Message XDR definition

```

/* NDMP_CONFIG_GET_TAPE_INFO */

/* tape attributes */
const NDMP_TAPE_ATTR_REWIND      = 0x00000001;
const NDMP_TAPE_ATTR_UNLOAD     = 0x00000002;
/* No request arguments */
struct ndmp_device_capability
{
    string          device <>;
    u_long          attr;
    ndmp_pval       capability <>;
};
struct ndmp_device_info
{
    string          model <>;
    ndmp_device_capability caplist <>;
};

struct ndmp_config_get_tape_info_reply
{
    ndmp_error       error;
    ndmp_device_info tape_info <>;
};

```

Request Arguments

This request does not have a message body.

Reply Arguments

error	Error code.
tape_info	Returned the information of the tape device.

The following attributes are defined:

model	The manufacture name of the tape device. For example, EXB-8500 and DLT4000.
-------	---

caplist	The attributes of the tape device. One physical tape device can have more than one device name with the different capabilities.
---------	---

device	The device name of the tape device. For example, /dev/rmt/0mn.
--------	--

attr	The bit mask value for the tape attributes.
------	---

NDMP_TAPE_ATTR_REWIND	The tape will be rewound when the device is closed.
-----------------------	---

NDMP_TAPE_ATTR_UNLOAD	The tape will be unloaded when the device is closed
-----------------------	---

capability	The capability of the tape drive device.
------------	--

The following are examples of the environment variables that can be defined for a tape device.

Variable Name	Meaning	Value	The Default Value
COMPRESSION	Compression ratio	integer	1 (no compression)

Reply Errors

NDMP_NO_ERR	Returned the tape information successfully.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.

3.2.8 Get SCSI Information

This message is used to query the information of the SCSI jukebox devices on the NDMP server host.

Message XDR definition

```

/* NDMP_CONFIG_GET SCSI_INFO */

/* No request arguments */
/* no SCSI attributes */

struct ndmp_config_get_scsi_info_reply
{
    ndmp_error          error;
    ndmp_device_info    scsi_info <>;
};

```

Request Arguments

This request does not have a message body.

Reply Arguments

error	Error code.
scsi_info	Returned the information of the SCSI jukebox device.

The following attributes are defined:

model	The manufacture name of the SCSI jukebox device. For example, DLT4700.
caplist	The attributes of the SCSI jukebox device.
device	The device name of the SCSI jukebox device. For example, /dev/scsi0.

attr	The bit mask for the well-defined SCSI attributes. (It is not defined yet.)
capability	The capability of the SCSI jukebox device.

Reply Errors

NDMP_NO_ERR	Returned the SCSI jukebox device information successfully.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.

3.3 SCSI Interface

The SCSI interface allows low level control of SCSI jukebox devices.

3.3.1 Open SCSI Device

Opens the specified SCSI device. This operation is required before any other SCSI requests can be executed. For security reasons, NDMP SCSI_OPEN should be supported only for the jukebox devices.

The open must be an exclusive open. The NDMP server can open only one jukebox device (via the NDMP SCSI interface) or tape device (via NDMP TAPE interface) at a time. An NDMP_DEVICE_BUSY_ERR is returned if the NDMP server already has a tape or jukebox device opened.

Message XDR definition

```
/* NDMP SCSI_OPEN */
struct ndmp_scsi_open_request
{
    string      device<>;
};

struct ndmp_scsi_open_reply
{
    ndmp_error  error;
};
```

Request Arguments

device	Name of SCSI interface device to open. The usage of this argument is NDMP-server implementation dependent. This argument can be used to specify the name of an actual SCSI device but more typically the argument will be used to specify the name of a SCSI pass-through driver pseudo device. The specific device to be controlled is selected through the set SCSI target request.
--------	---

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	SCSI interface device successfully opened.
NDMP_DEVICE_OPENED_ERR	The connection already has a tape device or SCSI device open.
NDMP_DEVICE_BUSY_ERR	Another NDMP connection currently has the specified device open.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized or device is not a tape or jukebox.
NDMP_NO_DEVICE_ERR	Invalid device specified.
NDMP_IO_ERR	IO error while opening SCSI device.

3.3.2 Close Device

This request closes the currently open SCSI interface device. No further requests can be made until another open request is successfully executed.

Message XDR definition

```
/* NDMP SCSI CLOSE */
/* no request arguments */
struct ndmp_scsi_close_reply
{
    ndmp_error      error;
};
```

Request Arguments

This request does not have a message body.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Device successfully closed.
NDMP_DEV_NOT_OPEN_ERR	No device currently open by the connection.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.3.3 Get SCSI State

This request returns the current state of the SCSI interface. The target information provides information about which SCSI device is controlled by this interface.

Message XDR definition

```

/* NDMP SCSI_GET_STATE */
/* no request arguments */
struct ndmp_scsi_get_state_reply
{
    ndmp_error      error;
    short           target_controller;
    short           target_id;
    short           target_lun;
};

```

Request Arguments

This request does not have a message body.

Reply Arguments

error	Error code.
target_controller	Identifier of the SCSI controller to which the currently targeted SCSI device is attached. -1 if no device currently is targeted.
target_id	SCSI target identifier. Specifies the SCSI bus address of the targeted device. -1 if no device currently is targeted.
target_lun	Logic unit number of the targeted device. -1 if no device currently is targeted.

Reply Errors

NDMP_NO_ERR	Target device information successfully returned.
NDMP_DEV_NOT_OPEN_ERR	No SCSI device currently open by the connection.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.3.4 Set SCSI Target

This request selects or changes the SCSI target. When the SCSI interface is opened, it is not known if the NDMP server has opened a device file that can pass commands to a single SCSI target or to multiple SCSI targets. This request is used to pass the information describing the SCSI target to which to send commands. Additionally, if the target can talk to multiple targets, this allows “scanning” the SCSI bus on the NDMP host for diagnostics or for jukebox discovery.

For security reasons this message should only be supported for tape or jukebox devices.

Message XDR definition

```

/* NDMP SCSI SET TARGET */
struct ndmp_scsi_set_target_request
{
    string          device<>;
    u_short         target_controller;
    u_short         target_id;
    u_short         target_lun;
};

struct ndmp_scsi_set_target_reply
{
    ndmp_error      error;
};

```

Request Arguments

device	SCSI device name. This argument is NDMP server implementation dependent. Some implementations might support the targeting of a device through a logical device name. If this argument is used, the following arguments might be ignored. If this argument is not specified or supported, then the following arguments must be specified.
target_controller	Identifier of the SCSI controller to which the targeted SCSI device is attached.
target_id	SCSI target identifier. Specifies the SCSI bus address of the targeted device.
target_lun	Logic unit number of the targeted device.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Specified SCSI device successfully targeted.
NDMP_NO_BUS_ERR	The specified SCSI controller is not found.
NDMP_NO_DEVICE_ERR	No device at this specified target.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized or device is not a tape or jukebox.

3.3.5 Reset Device

This request sends a SCSI device reset message to the currently targeted SCSI device.

Message XDR definition

```
/* NDMP SCSI RESET_DEVICE */
/* no request arguments */
struct ndmp_scsi_reset_device_reply
{
    ndmp_error      error;
};
```

Request Arguments

This request does not have a message body.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	SCSI device successfully reset.
NDMP_DEV_NOT_OPEN_ERR	No SCSI device currently open by the connection.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.3.6 Reset Bus

This request asserts a SCSI bus reset on the SCSI bus to which the SCSI device is attached.

Message XDR definition

```
/* NDMP SCSI RESET_BUS */
/* no request arguments */
struct ndmp_scsi_reset_bus_reply
{
    ndmp_error      error;
};
```

Request Arguments

This request does not have a message body.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	SCSI device successfully reset.
NDMP_DEV_NOT_OPEN_ERR	No SCSI device currently open by the connection.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.3.7 Execute CDB

This request sends a SCSI Control Data Block to a SCSI device. If a check condition is generated, then the extended sense data is also retrieved. Data can be transferred to or from the SCSI device as part of the command.

The server selects the SCSI target. The cdb is sent to the SCSI device in command mode. If DATA_OUT is set in the flags, then the dataout is sent to the SCSI device in data mode. Sometimes the host will disconnect from the target and wait for a reselect. If timeout is zero, the host will wait indefinitely for the target to reselect. If timeout is not zero, the host will wait timeout milliseconds for the target to reselect. If the reselect does not occur, an NDMP_TIMEOUT_ERR is returned. If the target reselects and the status is CHECK CONDITION, then the server executes a REQUEST SENSE cdb. If the DATA_IN flag is set, the server reads data from the target in data mode. The SCSI status, the DATA_IN, and the extended sense data are returned.

Message XDR definition

```

/* NDMP SCSI EXECUTE_CDB */
const NDMP SCSI_DATA_IN  = 0x00000001;
const NDMP SCSI_DATA_OUT = 0x00000002;

struct ndmp_execute_cdb_request
{
    u_long      flags;
    u_long      timeout;
    u_long      datain_len;
    opaque      cdb<>;
    opaque      dataout<>;
};

struct      ndmp_execute_cdb_reply
{
    ndmp_error      error;
    u_char          status;
    u_long          dataout_len;
    opaque          datain<>;
    opaque          ext_sense<>;
};

```

Request Arguments

flags	Specifies the data transfer (if any) direction. DATA_IN and DATA_OUT reference the host. They refer to data from the SCSI device into the host and data out of the host to the SCSI device.
timeout	Number of milliseconds to wait if a reselect occurs. If timeout is zero, then the host will wait indefinitely for the target to reselect.
datain_len	If the data transfer direction is DATA_IN, the expected number of data bytes to read.
cdb	The SCSI command data block.
dataout	If the data transfer direction is DATA_OUT, the data to be transferred to the SCSI device.

Reply Arguments

error	Error code.
status	SCSI status byte.
dataout	If the data transfer direction is DATA_OUT, the number of data bytes transferred to the device.
datain	If the data transfer direction is DATA_IN, the data transferred from the SCSI device.
ext_sense	Extended SCSI sense data.

Reply Errors

NDMP_NO_ERR	Message successfully processed. Does not necessarily indicate that the SCSI command was successfully executed. The returned SCSI status byte must be used to determine if the command was successful.
NDMP_DEV_NOT_OPEN_ERR	No SCSI device currently open by the connection.
NDMP_IO_ERR	I/O error.
NDMP_ILLEGAL_ARGS	Invalid argument in request message.
NDMP_TIMEOUT_ERR	The SCSI command timed out.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.4 TAPE Interface

The TAPE interface provides complete control of a tape drive. If the tape drive is a SCSI tape drive, then the TAPE interface also provides low level CDB access to the tape drive. This interface is analogous to the rmt protocol. The physical device is assigned when the server is started.

3.4.1 Open Tape Device

This request opens the tape device in the specified mode. This operation is required before any other tape requests can be executed. The device is opened exclusively; no other NDMP server can concurrently open the device. Each tape device can be opened only by one NDMP server at a time. Each NDMP server can have only one tape or SCSI device open at a time. If the drive does not have a tape loaded, an error is returned. If the loaded media is write-protected, then the device can be opened only in read-only mode.

Message XDR definition

```

/* NDMP_TAPE_OPEN */
enum ndmp_tape_open_mode
{
    NDMP_TAPE_READ_MODE,
    NDMP_TAPE_RDWR_MODE
};

struct ndmp_tape_open_request
{
    string          device <>;
    ndmp_tape_open_mode mode;
};

struct ndmp_tape_open_reply
{
    ndmp_error      error;
};

```

Request Arguments

device	Name of tape device to open.
mode	Tape open mode. One of the following mode can be specified when open the tape device: NDMP_TAPE_READ_MODE: open the tape device for read only. NDMP_TAPE_RDWR _MODE: open the tape device for read/write.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Tape device successfully opened.
NDMP_DEVICE_OPENED_ERR	The NDMP server already has a SCSI device or tape device open.
NDMP_NO_DEVICE_ERR	The specified device does not exist.
NDMP_DEVICE_BUSY_ERR	The device is already open by another NDMP server or system process.
NDMP_IO_ERR	Device I/O error.
NDMP_WRITE_PROTECT_ERR	Device cannot be opened in write mode because the tape is write protected.
NDMP_NO_TAPE_LOADED_ERR	No tape loaded in the tape device.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.4.2 Close Device

This request closes the tape drive. No further tape requests can be processed until another tape open request is successfully executed.

Message XDR definition

```
/* NDMP_TAPE_CLOSE */
/* no request arguments */
struct ndmp_tape_close_reply
{
    ndmp_error      error;
};
```

Request Arguments

This message does not have a message body.

Reply Arguments

error Error code.

Reply Errors

NDMP_NO_ERR Tape device successfully closed.

NDMP_DEV_NOT_OPEN_ERR No tape device currently open by the connection.

NDMP_IO_ERR Device I/O error.

NDMP_NOT_SUPPORTED_ERR The request is not supported for this implementation.

NDMP_NOT_AUTHORIZED_ERR Connection not authorized.

3.4.3 Get Tape State

This request returns the state of the tape drive interface.

Message XDR definition

```

/* NDMP_TAPE_GET_STATE */
/* no request arguments */

struct ndmp_u_quad
{
    u_long    high;
    u_long    low;
};
/* invalid bit */
const NDMP_TAPE_STATE_FILE_NUM_INVALID = 0x00000001;
const NDMP_TAPE_STATE_SOFT_ERRORS_INVALID = 0x00000002;
const NDMP_TAPE_STATE_BLOCK_SIZE_INVALID = 0x00000004;
const NDMP_TAPE_STATE_BLOCKNO_INVALID = 0x00000008;
const NDMP_TAPE_STATE_TOTAL_SPACE_INVALID = 0x00000010;
const NDMP_TAPE_STATE_SPACE_REMAIN_INVALID = 0x00000020;
const NDMP_TAPE_STATE_PARTITION_INVALID = 0x00000040;

/* flags */
const NDMP_TAPE_STATE_NOREWIND = 0x0008;
const NDMP_TAPE_STATE_WR_PROT = 0x0010;
const NDMP_TAPE_STATE_ERROR = 0x0020;
const NDMP_TAPE_STATE_UNLOAD = 0x0040;

struct ndmp_tape_get_state_reply
{
    u_long    invalid;
    ndmp_error error;
    u_long    flags;
    u_long    file_num;
    u_long    soft_errors;
    u_long    block_size;
    u_long    blockno;
    ndmp_u_quad total_space;
    ndmp_u_quad space_remain;
    u_long    partition;
};

```

Request Arguments

This message does not have a message body.

Reply Arguments

invalid	The invalid flag is used to identify the unsupported argument in the message.
error	Error code.
flags	Bitmask of the following tape device mode flags:
NDMP_TAPE_STATE_NOREWIND	Upon device close, the tape will not be rewound.
NDMP_TAPE_STATE_WR_PROT	The loaded tape is write-protected.

NDMP_TAPE_STATE_ERROR	A media error was detected during the previous tape operation. This bit is cleared at the start of each tape operation.
NDMP_TAPE_STATE_UNLOAD	The currently loaded tape will be unloaded automatically when the device is closed. Only applies to media changer devices such as tape stackers and jukeboxes.
file_num	Current file position. First file on the tape is file number 0. NDMP_TAPE_STATE_FILE_NUM_INVALID is set if this argument is not supported.
soft_errors	Total number of soft media errors detected since the device was opened. NDMP_TAPE_STATE_SOFT_ERRORS_INVALID is set if this argument is not supported.
block_size	Tape block size in bytes. 0 if the device is in variable block size mode. NDMP_TAPE_STATE_BLOCK_SIZE_INVALID is set if this argument is not supported.
blockno	Current tape block number. First tape block is block number 0. NDMP_TAPE_STATE_BLOCKNO_INVALID is set if this argument is not supported.
total_space	Total tape capacity in bytes. NDMP_TAPE_STATE_TOTAL_SPACE_INVALID is set if this argument is not supported.
space_remain	Total remaining tape capacity in bytes. NDMP_TAPE_STATE_SPACE_REMAIN_INVALID is set if this argument is not supported.
partition	Tape partition number. First partition on the tape is partition number 0. NDMP_TAPE_STATE_PARTITION_INVALID is set if this argument is not supported.
Reply Errors	
NDMP_NO_ERR	Tape state successfully returned.
NDMP_DEV_NOT_OPEN_ERR	No tape device currently opened by the connection.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.
NDMP_IO_ERR	Device I/O error.

3.4.4 MTIO

This request provides access to the standard magnetic tape I/O operations. When spacing forward over a record, the tape head is positioned in the tape gap between the record just skipped and the next record. When spacing forward over file marks, the tape head is positioned in the tape gap between the next file mark and the record that follows it. When spacing backward over a record data, the tape head is positioned in the tape gap immediately preceding the tape record where the tape head is currently positioned. When spacing backward over file marks, the tape head is positioned in the tape gap preceding the file mark. The next read would fetch the EOF.

Message XDR definition

```

/* NDMP_TAPE_MTIO */
enum ndmp_tape_mtio_op
{
    NDMP_MTIO_FSF,
    NDMP_MTIO_BSF,
    NDMP_MTIO_FSR,
    NDMP_MTIO_BSR,
    NDMP_MTIO_REW,
    NDMP_MTIO_EOF,
    NDMP_MTIO_OFF
};

struct ndmp_tape_mtio_request
{
    ndmp_tape_mtio_op    tape_op;
    u_long               count;
};

struct ndmp_tape_mtio_reply
{
    ndmp_error           error;
    u_long               resid_count;
};

```

Request Arguments

tape_op	One of the following tape operations:
NDMP_MTIO_FSF	Forward space over file marks. The tape head is positioned in the tape gap between the file mark and the record that follows the file mark.
NDMP_MTIO_BSF	Backward space over file marks. The tape head is positioned in the tape gap preceding the file mark such that the next read encounters EOF.
NDMP_MTIO_FSR	Forward space over tape records. The tape head is positioned in the tape gap between the record just skipped and the next record.
NDMP_MTIO_BSR	Backward space over tape records. The tape head is positioned in the tape gap preceding the tape record just skipped.
NDMP_MTIO_REW	Rewind the tape.
NDMP_MTIO_EOF	Write end of file marks.

NDMP_MTIO_OFF	Eject the tape from the device.
count	Number of operations to run.
Reply Arguments	
error	Error code.
resid_count	Residual operation count. Represents the number of operations that could not be done due to encountering beginning of tape, end of tape, end of written media, or a tape error.

Reply Errors

NDMP_NO_ERR	Tape operation successfully completed.
NDMP_DEV_NOT_OPEN_ERR	No tape device currently open by the connection.
NDMP_IO_ERR	Device I/O error.
NDMP_ILLEGAL_ARGS_ERR	Invalid tape operation specified.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.
NDMP_WRITE_PROTECT_ERR	Tape is write protected.

3.4.5 Write

Writes data to the tape device. The NDMP server writes the specified data as a single record. It is the responsibility of the NDMP client to ensure that the length of the data is a multiple of the tape device block size if the device is a fixed block device. The NDMP server does not buffer or pad the data. . This request is typically used by the NDMP client to write tape header and trailer file data.

Message XDR definition

```

/* NDMP_TAPE_WRITE */
struct ndmp_tape_write_request
{
    opaque          data_out<>;
};

struct ndmp_tape_write_reply
{
    ndmp_error      error;
    u_long          count;
};

```

Request Arguments

data_out	The data to be written to the tape device.
----------	--

Reply Arguments

error	Error code.
-------	-------------

count	Number of data bytes written.
-------	-------------------------------

Reply Errors

NDMP_NO_ERR	All data successfully written to the tape device.
NDMP_DEV_NOT_OPEN_ERR	No tape device currently open by the connection.
NDMP_IO_ERR	Device I/O error.
NDMP_EOM_ERR	End of tape was encountered while writing.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.
NDMP_WRITE_PROTECT_ERR	Tape is write protected.

3.4.6 Read

This request reads the requested amount of data from the tape drive. The NDMP server always reads a complete record. If the specified number of bytes to read is not a multiple of the tape record size, then the NDMP server discards the bytes from the end of the record. The next read will return bytes starting from the beginning of the next record. To do contiguous reads, the number of bytes read must be a multiple of the tape record size. The client is responsible for ensuring that the data length is a multiple of the tape record size if the tape device is in fixed block size mode.

Message XDR definition

```

/* NDMP_TAPE_READ */
struct ndmp_tape_read_request
{
    u_long          count;
};

struct ndmp_tape_read_reply
{
    ndmp_error      error;
    opaque          data_in<>;
};

```

Request Arguments

count	Number of bytes to read.
-------	--------------------------

Reply Arguments

error	Error code.
data_in	The data read from the tape drive.

Reply Errors

NDMP_NO_ERR	Requested number of bytes successfully read from the tape device.
-------------	---

NDMP_DEV_NOT_OPEN_ERR	No tape device currently open by the connection.
NDMP_IO_ERR	Device I/O error during read.
NDMP_EOF_ERR	End of file was encountered while reading. The number of returned data bytes can be less than the number of bytes requested.
NDMP_NOT_SUPPORTED_ERR	The request sequence is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.4.7 Execute CDB

This message behaves in exactly the same way as the SCSI_EXECUTE_CDB request except that it sends the cdb to the tape device. This request should not be used to change the state of the tape device (such as tape positioning).

Message XDR definition

```
/* NDMP_TAPE_EXECUTE_CDB */
typedef ndmp_scsi_execute_cdb_request ndmp_tape_execute_cdb_request;
typedef ndmp_scsi_execute_cdb_reply ndmp_tape_execute_cdb_reply;
```

3.5 DATA Interface

This interface controls backup and recover operations.

3.5.1 Get Data State

This request returns data state information that can be used to monitor the progress of the current data operation.

Message XDR definition

```

/* NDMP_DATA_GET_STATE */
/* no request arguments */
enum ndmp_data_operation
{
    NDMP_DATA_OP_NOACTION,
    NDMP_DATA_OP_BACKUP,
    NDMP_DATA_OP_RESTORE
};

enum ndmp_data_state
{
    NDMP_DATA_STATE_IDLE,
    NDMP_DATA_STATE_ACTIVE,
    NDMP_DATA_STATE_HALTED,
    NDMP_DATA_STATE_LISTEN,
    NDMP_DATA_STATE_CONNECTED
};

enum ndmp_data_halt_reason
{
    NDMP_DATA_HALT_NA,
    NDMP_DATA_HALT_SUCCESSFUL,
    NDMP_DATA_HALT_ABORTED,
    NDMP_DATA_HALT_INTERNAL_ERROR,
    NDMP_DATA_HALT_CONNECT_ERROR,
};

/* ndmp_addr */
struct ndmp_tcp_addr
{
    u_long    ip_addr;
    u_short   port;
};

struct ndmp_fc_addr
{
    u_long    loop_id;
};

struct ndmp_ipc_addr
{
    opaque    comm_data<>;
};

union ndmp_addr switch (ndmp_addr_type addr_type)
{
    case NDMP_ADDR_LOCAL:
        void;
    case NDMP_ADDR_TCP:
        ndmp_tcp_addr    tcp_addr;
    case NDMP_ADDR_FC:
        ndmp_fc_addr      fc_addr;
    case NDMP_ADDR_IPC:
        ndmp_ipc_addr      ipc_addr;
};

/* invalid flag */
const    NDMP_DATA_STATE_EST_BYTES_REMAIN = 0x00000001;
const    NDMP_DATA_STATE_EST_TIME_REMAIN  = 0x00000002;

struct ndmp_data_get_state_reply
{

```

```

    u_long          invalid;
    ndmp_error      error;
    ndmp_data_operation operation;
    ndmp_data_state state;
    ndmp_data_halt_reason halt_reason;
    ndmp_u_quad     bytes_processed;
    ndmp_u_quad     est_bytes_remain;
    u_long          est_time_remain;
    ndmp_addr       data_connection_addr;
    ndmp_u_quad     read_offset;
    ndmp_u_quad     read_length;
};

```

Request Arguments

This message does not have a message body.

Reply Arguments

invalid	The <code>invalid</code> flag is used to identify the unsupported argument in the message.
error	Error code.
operation	Data operation currently in progress.
NDMP_DATA_OP_NOACTION	No data operation currently in progress.
NDMP_DATA_OP_BACKUP	Backup operation currently in progress.
NDMP_DATA_OP_RESTORE	Restore operation currently in progress.
state	Current state of the NDMP server.
NDMP_DATA_STATE_IDLE	No active data operation.
NDMP_DATA_STATE_ACTIVE	Data operation in progress.
NDMP_DATA_STATE_HALTED	Data operation completed.
NDMP_DATA_STATE_LISTEN	Wait for the establishment of the connection.
NDMP_DATA_STATE_CONNECTED	Data connection is established.
halt_reason	Reason the data operation is halted.
NDMP_DATA_HALT_NA	Data operation not in progress or not in the halt state.
NDMP_DATA_HALT_SUCCESSFUL	Data operation completed successfully.
NDMP_DATA_HALT_ABORTED	Data operation aborted by the NDMP client.
NDMP_DATA_HALT_INTERNAL_ERROR	

Data operation halted due to unrecoverable error incurred by the NDMP server data backup/recover software.

NDMP_DATA_HALT_CONNECT_ERROR

Error establishing connection to tape server.

bytes_processed	Total number of bytes processed by the data operation.
est_bytes_remain	Estimated number of bytes processed remaining to be processed by the data operation. NDMP_DATA_STATE_EST_BYTES_REMAIN_INVALID is set if this feature is not supported.
est_time_remain	Estimated number of seconds until the data operation to completes. NDMP_DATA_STATE_EST_TIME_REMAIN_INVALID is set if this feature is not supported.
data_connection_addr	When the data server listens a data connection: In the LISTEN state, data_connection_addr is the address to listen the data connection. While the state transition to CONNECTED or ACTIVE State, data_connection_addr is the address of client, which connects to this data server. In the IDLE state, the field is ignored. When the data server connects to the other NDMP server: In the ACTIVE state, data_connection_addr is the address of server, which listens the data connection. In the IDLE state, the field is ignored.
read_offset	Offset value specified in last NDMP_NOTIFY_DATA_READ request.
read_len	Length value specified in last NDMP_NOTIFY_DATA_READ request.

Reply Errors

NDMP_NO_ERR	Data state successfully returned.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.5.2 Listen

The data server could begin listening a data connection from the local or remote NDMP server. The NDMP_DATA_LISTEN message returns an address that is used to make the data connection.

Message XDR definition

```

/* NDMP_DATA_LISTEN */
struct ndmp_tcp_addr
{
    u_long    ip_addr;
    u_short   port;
};

struct ndmp_fc_addr
{
    u_long    loop_id;
};

struct ndmp_ipc_addr
{
    opaque    comm_data<>;
};

union ndmp_addr switch (ndmp_addr_type addr_type)
{
    case NDMP_ADDR_LOCAL:
        void;
    case NDMP_ADDR_TCP:
        ndmp_tcp_addr    tcp_addr;
    case NDMP_ADDR_FC:
        ndmp_fc_addr      fc_addr;
    case NDMP_ADDR_IPC:
        ndmp_ipc_addr      ipc_addr;
};

struct ndmp_data_listen_request
{
    ndmp_addr_type    addr_type;
};

struct ndmp_data_listen_reply
{
    ndmp_error        error;
    ndmp_addr         connect_addr;
};

```

Request Arguments

addr_type The specific type of the data connection.

Reply Arguments

error Error code.

connect_addr Address on which the data server is listening for a connection.

Reply Errors

NDMP_NO_ERR Listen successful.

NDMP_ILLEGAL_STATE_ERR The data server not currently in idle state.

NDMP_ILLEGAL_ARGS_ERR Invalid mode or address type specified.

NDMP_NOT_SUPPORTED_ERR The request is not supported for this implementation.

NDMP_NOT_AUTHORIZED_ERR Connection not authorized.

3.5.3 Connect

The data server could connect to a specific address specified by the local or remote NDMP server. If failed to connect to the specific address, an *NDMP_CONNECT_ERR* is returned to the NDMP client application.

Message XDR definition

```
/* NDMP_DATA_CONNECT */
struct ndmp_data_connect_request
{
    ndmp_addr      addr;
};

struct ndmp_data_connect_reply
{
    ndmp_error      error;
};
```

Request Arguments

addr	The address where the server is listening a data connection.
------	--

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Listen successful.
NDMP_ILLEGAL_STATE_ERR	The data server not currently in idle state.
NDMP_ILLEGAL_ARGS_ERR	Invalid mode or address type specified.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.
NDMP_CONNECT_ERR	Failed to establish the data connection.

3.5.4 Backup

This request begins a backup. The *bu_type* is the name of the backup type. The type of backup is implementation dependent. The *env* is a list of parameters that might affect the behavior of the backup. The *env* returned by the *DATA_GET_ENV* will be saved and made available to the retrieval process.

Message XDR definition

```

/* NDMP_DATA_START_BACKUP */

struct ndmp_data_start_backup_request
{
    string          bu_type<>;
    ndmp_pval      env<>;
};

struct ndmp_data_start_backup_reply
{
    ndmp_error      error;
};

```

Request Arguments

bu_type	The name of the backup type. Backup types are NDMP-server implementation dependent.
env	List of parameter names and values for configuring the backup type. Backup type parameters are NDMP server implementation dependent.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Backup operation successfully started.
NDMP_ILLEGAL_STATE_ERR	A data operation is already in progress. Only one data operation per connection can execute at a time.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_ILLEGAL_ARGS_ERR	Invalid backup type, invalid backup type parameter, or invalid backup type parameter value specified.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.5.5 Recover

This request recovers the files specified in `nlist` from the backup. The `env` is the list of parameters and values saved at the end of the backup. The `bu_type` is the name of the backup type.

Message XDR definition

```
/* NDMP_DATA_START_RECOVER */
struct ndmp_name
{
    string          original_path<>;
    string          destination_dir<>;
    string          new_name <>;
    string          other_name <>;
    ndmp_u_quad     node;
    ndmp_u_quad     fh_info;
};

struct ndmp_data_start_recover_request
{
    ndmp_pval       env<>;
    ndmp_name       nlist<>;
    string          bu_type<>;
};

struct ndmp_data_start_recover_reply
{
    ndmp_error       error;
};
```

Request Arguments

env	The backup environment that was returned from a data get environment request made prior to notifying the NDMP server that the backup was complete through a data stop message.
nlist	List of files to be recovered and the location to which each file is to be recovered. Definition of list entry:
original_path	Path of a file/directory to be recovered. The <code>original_path</code> is the original backup path name and is relative to the backup root directory. If the <code>original_path</code> is not specified, the entire backup image will be restored.
destination_dir	Destination directory to be used when recovering the file. If <code>destination_dir</code> is not specified, the files will be restored to the original directory.
new_name	This argument is used for the direct access retrieval only. If <code>new_name</code> is specified, the restored file is renamed to <code>new_name</code> in the <code>destination_dir</code> directory.
other_name	This argument is used for the direct access retrieval only and supported by some file systems, that support multiple file names for a single file. For example, NT File Systems support both NT file names and dos file names.
node	This argument is used for the direct access retrieval only. <code>node</code> is used to verify the retrieval if the backup method supports inode type of backup, e.g. dump. The argument is set to 0 if not supported.
fh_info	File history tape positioning data recorded when the file was backed up. This data may be used by the restore method to position tape for direct access data retrieval. The positioning data is NDMP-server dependent. Typically it will be the byte or record offset from the beginning of the tape of the file to be recovered. This field is ignored by data method implementations that do not support this feature.
bu_type	Name of the recover method. Recover methods are NDMP server implementation dependent.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Recover operation successfully started.
NDMP_ILLEGAL_STATE_ERR	A data operation is already in progress. Only one data operation per connection is allowed to execute at a time.

NDMP_ILLEGAL_ARGS_ERR	Invalid recover method, invalid recover method parameter, invalid recover method parameter value, or invalid name list entry specified.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.5.6 Abort

This request sends a message to abort the current backup or restore operation. The operation should be terminated as soon as possible.

Message XDR definition

```
/* NDMP_DATA_ABORT */
/* no request arguments */
struct ndmp_data_abort_reply
{
    ndmp_error          error;
};
```

Request Arguments

This message does not have a request body.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Data operation successfully terminated.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.5.7 Stop

This request sends a message to inform the NDMP server that the current backup is complete. The NDMP server will change to the idle state and be ready to process another request. The environment variable defined from the previous backup will be cleared as well.

Message XDR definition

```
/* NDMP_DATA_STOP */
/* no request arguments */
struct ndmp_data_stop_reply
{
    ndmp_error          error;
};
```

Request Arguments

This message does not have a message body.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Message successfully processed.
NDMP_ILLEGAL_STATE_ERR	The request cannot be run in the current state.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.5.8 Get Env

This request gets the backup environment. Returns the environment included in the NDMP_DATA_START_BACKUP request along with any additional parameters added or modified by the backup method. The returned environment should be saved and passed in the NDMP_DATA_START_RECOVER request whenever data from the backup is to be recovered.

Message XDR definition

```
/* NDMP_DATA_GET_ENV */
/* no request arguments */
struct ndmp_data_get_env_reply
{
    ndmp_error      error;
    ndmp_pval       env<>;
};
```

Request Arguments

This message does not have a message body.

Reply Arguments

error	Error code.
env	The backup method environment parameters and values.

Reply Errors

NDMP_NO_ERR	Environment successfully returned.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_ILLEGAL_STATE_ERR	Illegal state. A data operation is not currently in progress.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.6 *MOVER Interface*

During a backup, the tape server accepts data from the data connection and writes the data to tape using fixed size records. During restores, the tape server accepts requests to read portions of the data from tape. If the requested data is not a multiple of the record size, the tape server will do a full record read and only return the requested amount of data. If the tape server encounters end-of-file (EOF), media error, or reaches the end-of-the-data window while reading, it pauses and notifies the NDMP client.

3.6.1 Get Mover State

This request returns the state of the tape server.

Message XDR definition

```

/* NDMP_MOVER_GET_STATE */
enum ndmp_mover_state
{
    NDMP_MOVER_STATE_IDLE,
    NDMP_MOVER_STATE_LISTEN,
    NDMP_MOVER_STATE_ACTIVE,
    NDMP_MOVER_STATE_PAUSED,
    NDMP_MOVER_STATE_HALTED
};

enum ndmp_mover_pause_reason
{
    NDMP_MOVER_PAUSE_NA,
    NDMP_MOVER_PAUSE_EOM,
    NDMP_MOVER_PAUSE_EOF,
    NDMP_MOVER_PAUSE_SEEK,
    NDMP_MOVER_PAUSE_MEDIA_ERROR,
    NDMP_MOVER_PAUSED_EOW
};

enum ndmp_mover_halt_reason
{
    NDMP_MOVER_HALT_NA,
    NDMP_MOVER_HALT_CONNECT_CLOSED,
    NDMP_MOVER_HALT_ABORTED,
    NDMP_MOVER_HALT_INTERNAL_ERROR,
    NDMP_MOVER_HALT_CONNECT_ERROR
};

/* no request arguments */
struct ndmp_mover_get_state_reply
{
    ndmp_error          error;
    ndmp_mover_state    state;
    ndmp_mover_pause_reason pause_reason;
    ndmp_mover_halt_reason halt_reason;
    u_long              record_size;
    u_long              record_num;
    ndmp_u_quad          data_written;
    ndmp_u_quad          seek_position;
    ndmp_u_quad          bytes_left_to_read;
    ndmp_u_quad          window_offset;
    ndmp_u_quad          window_length;
    ndm_addr             data_connection_addr;
};

```

Request Arguments

This message does not have a message body.

Reply Arguments

error	Error code.
state	Current state of the NDMP server.
NDMP_MOVER_STATE_IDLE	No active data operation.
NDMP_MOVER_STATE_LISTEN	Waiting connection for backup or restore.
NDMP_MOVER_STATE_ACTIVE	Data operation in progress.
NDMP_MOVER_STATE_PAUSED	Operation paused awaiting operator attention.
NDMP_MOVER_STATE_HALTED	Operation completed.
pause_reason	Reason the operation is paused.
NDMP_MOVER_PAUSE_NA	Operation not in progress or not in the pause state.
NDMP_MOVER_PAUSE_EOM	Operation encountered end of media. NDMP client attention required.
NDMP_MOVER_PAUSE_EOF	Operation encountered end of file. NDMP client attention required.
NDMP_MOVER_PAUSE_SEEK	Data operation requested a seek that requires NDMP client intervention.
NDMP_MOVER_PAUSE_MEDIA_ERROR	Error while reading/writing tape.
NDMP_MOVER_PAUSE_EOW	Operation encountered end of mover window. NDMP client attention required.
halt_reason	Reason the operation is halted.
NDMP_MOVER_HALT_NA	Operation not in progress or not in the halt state.
NDMP_MOVER_HALT_CONNECTION_CLOSED	Connection to data server close detected.
NDMP_MOVER_HALT_ABORTED	Operation aborted by the NDMP client.
NDMP_MOVER_HALT_INTERNAL_ERROR	Operation halted due to unrecoverable error incurred by the tape server.
NDMP_MOVER_HALT_CONNECT_ERROR	Error establishing connection to data server.
record_size	Size of tape data record.
record_num	Current tape record number.
data_written	Total number of bytes written to tape.

seek_position	Offset value from last NDMP_MOVER_READ message.
bytes_left_to_read	Number of bytes remaining to be read to satisfy the last NDMP_MOVER_READ request.
window_offset	Offset value from last NDMP_MOVER_SET_WINDOW request.
window_length	Length value from last NDMP_MOVER_SET_WINDOW request.
data_connection_addr	<p>When the tape server listens a data connection:</p> <p>In the LISTEN state, <code>data_connection_addr</code> is the address to listen the data connection. While the state transition to CONNECTED or ACTIVE State, <code>data_connection_addr</code> is the address of client, which connects to this data server. In the IDLE state, the field is ignored.</p> <p>When the tape server connects to the other NDMP server:</p> <p>In the ACTIVE state, <code>data_connection_addr</code> is the address of server, which listens the data connection. In the IDLE state, the field is ignored.</p>

Reply Errors

NDMP_NO_ERR	Data state successfully returned.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.6.2 Listen

The tape server could begin listening for a data connection from a data server or the other tape server. The tape server returns an address that may be used by the other server to make a connection. One server should provide the address and listen for the data connection. The other server will connect to that specified address.

Message XDR definition

```

/* NDMP_MOVER_LISTEN */

struct ndmp_mover_listen_request
{
    ndmp_mode      mode;
    ndmp_addr_type addr_type;
};

struct ndmp_mover_listen_reply
{
    ndmp_error      error;
    ndmp_addr       connect_addr;
};

```

Request Arguments

mode	One of the following:
NDMP_MOVER_MODE_READ	The tape server should read data from the data connection and write the data to tape. This mode is used for backup operations.
NDMP_MOVER_MODE_WRITE	The tape server should read data from tape and write the data to the data connection. This mode is used for restore operations.
addr_type	The specific type of the connection.

Reply Arguments

error	Error code.
connect_addr	Address on which the tape server is listening for a connection. If the address type is TCP, then the returned address contains the IP address and port number that the tape server is listening on.

Reply Errors

NDMP_NO_ERR	Listen successful.
NDMP_ILLEGAL_STATE_ERR	The tape server not currently in idle state.
NDMP_ILLEGAL_ARGS_ERR	Invalid mode or address type specified.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.
NDMP_DEV_NOT_OPEN_ERR	No device currently open.

3.6.3 Connect

The tape server could connect to a specific address to a data server or the other tape server. If failed to connect to the specific address, an *NDMP_CONNECT_ERR* is returned to the NDMP client application.

Message XDR definition

```

/* NDMP_MOVER_CONNECT */
struct ndmp_mover_connect_request
{
    ndmp_mode      mode;
    ndmp_addr      addr;
};

struct ndmp_mover_connect_reply
{
    ndmp_error      error;
};

```

Request Arguments

mode	The direction of the data flow.
addr	The address where the data server is listening for a data connection.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Listen successful.
NDMP_ILLEGAL_STATE_ERR	The tape server not currently in idle state.
NDMP_ILLEGAL_ARGS_ERR	Invalid mode or address type specified.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.
NDMP_CONNECT_ERR	Failed to establish the data connection.

3.6.4 Set Record Size

This request sets the record size used by the tape server for all tape reads and writes. When writing to tape, the tape server will buffer data until a full record has been buffered before writing the record to tape. The client is responsible for setting the record size to a multiple of the tape block size if the tape device being used is a fixed block size device.

Message XDR definition

```

/* NDMP_MOVER_SET_RECORD_SIZE */
struct ndmp_mover_set_record_size_request
{
    u_long          len;
};

struct ndmp_mover_set_record_size_reply
{
    ndmp_error      error;
};

```

Request Arguments

len	Record size in bytes.
-----	-----------------------

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Record size successfully set.
NDMP_ILLEGAL_ARGS_ERR	Invalid record size specified. The maximum record size is implementation dependent
NDMP_ILLEGAL_STATE_ERR	Illegal state. The record size may only be set when in the idle state.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.6.5 Set Window

This request defines the valid data window. This message is used at both backup and retrieval time. During the backup, the backup client can send this message to limit the number of bytes of data will be written to each tape file. This message is optional. If it is not specified, the NDMP server will keep write the data until the data connection is closed or reach the end of the tape. The `offset` field is ignored at the backup time.

For the retrieval, the backup client sends this message to indicate the current tape file `offset` to the beginning of the entire backup image and the `length` of the current tape file. This message is used for direct access retrieval only. The window begins at the first record of the current tape file and extends for the specified number of bytes. After reading all data specified by the window, the tape server will notify the NDMP client that a tape change/seek is required.

Message XDR definition

```

/* NDMP_MOVER_SET_WINDOW */
struct ndmp_mover_set_window_request
{
    ndmp_u_quad      offset;
    ndmp_u_quad      length;
};

struct ndmp_mover_set_window_reply
{
    ndmp_error        error;
};

```

Request Arguments

offset	The data stream byte offset of the first byte in the window. This field is ignored if the tape server is writing the data to the backup device.
length	Number of bytes in the window during the direct access retrieval. The maximum number of bytes of data could be written to each tape file if the tape server is writing the data to the backup device.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Window successfully set.
NDMP_ILLEGAL_ARGS_ERR	Invalid window specified.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_ILLEGAL_STATE_ERR	Illegal state. A window can be set only when in the listen or paused state.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.6.6 Continue

This request notifies the tape server to continue reading/writing tape data. This request is sent after the NDMP client has completed a tape change or tape positioning.

Message XDR definition

```

/* NDMP_MOVER_CONTINUE */
/* no request arguments */
struct ndmp_mover_continue_reply
{
    ndmp_error        error;
};

```

Request Arguments

This message does not have a message body.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Tape server successfully continued.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_ILLEGAL_STATE_ERR	Illegal state. Tape server not currently in the paused state.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.6.7 Abort

This request aborts the tape server. The tape server stops reading or writing data from/to tape and closes the data connection.

Message XDR definition

```
/* NDMP_MOVER_ABORT */
/* no request arguments */
struct ndmp_mover_abort_reply
{
    ndmp_error          error;
};
```

Request Arguments

This message does not have a message body.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Tape server successfully aborted.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.6.8 Stop

This request frees any resources associated with the tape server and returns the tape server to the idle state.

Message XDR definition

```

/* NDMP_MOVER_STOP */
/* no request arguments */
struct ndmp_mover_stop_reply
{
    ndmp_error          error;
};

```

Request Arguments

This message does not have a message body.

Reply Arguments

error Error code.

Reply Errors

NDMP_NO_ERR Tape server successfully stopped.

NDMP_ILLEGAL_STATE_ERR Illegal state. Tape server not currently in the halted state.

NDMP_NOT_AUTHORIZED_ERR Connection not authorized.

3.6.9 Read

This request notifies the tape server to begin reading backup data from the tape drive and write the data to the data connection. The tape server will continue to write data to the data connection until the requested number of bytes have been read from tape and written to the data connection. If EOF or the end-of-the-data window is encountered, the tape server will notify the NDMP client and then enter the paused state. While fulfilling this request, the tape server should continue to accept messages from the NDMP client. It is invalid to issue another read request while the current request is in progress.

Message XDR definition

```

/* NDMP_MOVER_READ */
struct ndmp_mover_read_request
{
    ndmp_u_quad          offset;
    ndmp_u_quad          length;
};

struct ndmp_mover_read_reply
{
    ndmp_error          error;
};

```

Request Arguments

offset Offset within the data stream of the first byte to be sent to the data connection. The tape server should seek the tape to the record containing the requested offset and then read and discard data until the offset has been reached. If the offset is outside of the currently set data window, the tape server should notify the NDMP client that a seek is required.

length	Number of data bytes to be read and send to the data connection.
--------	--

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Read successfully started.
NDMP_ILLEGAL_STATE_ERR	Illegal state. Tape server not currently in the paused state.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

3.6.10 Close

This request notifies the tape server to close the data connection. The NDMP client will send this request after a backup operation has completed or after all data for a restore operation has been read.

Message XDR definition

```
/* NDMP_MOVER_CLOSE */
/* no request arguments */
struct ndmp_mover_close_reply
{
    ndmp_error      error;
};
```

Request Arguments

This message does not have a message body.

Reply Arguments

error	Error code.
-------	-------------

Reply Errors

NDMP_NO_ERR	Data connection successfully closed.
NDMP_ILLEGAL_STATE_ERR	Illegal state. Data connection not open.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

5. NDMP Client Interfaces

This section defines the protocol interfaces implemented by the NDMP client.

5.1 *NOTIFY* Interface

This interface is used by the NDMP server to let the NDMP client know that the NDMP server requires attention.

5.1.1 Notify Data Halted

This message is used to notify the NDMP client that the NDMP data server has halted

Message XDR definition

```
/* NDMP_NOTIFY_DATA_HALTED */
struct ndmp_notify_data_halted_request
{
    ndmp_data_halt_reason    reason;
    string                   text_reason<>;
};
```

Request Arguments

reason	Reason the data operation halted.
NDMP_HALT_NA	Data operation not in progress or not in the halt state.
NDMP_HALT_SUCCESSFUL	Data operation completed successfully.
NDMP_HALT_ABORTED	Data operation aborted by the NDMP client.
NDMP_HALT_MEDIA_ERROR	Data operation halted due to unrecoverable media error.
NDMP_HALT_INTERNAL_ERROR	Data operation halted due to unrecoverable error incurred by the NDMP server or the data backup/recover method.
text_reason	Diagnostic error message. NDMP server implementation dependent.

Reply Arguments

This message does not have a message body.

5.1.2 Notify Connected

This message is sent in response to a connection establishment attempt. This message is always the first message sent on a new connection. It is also used prior to NDMP server shutdown to inform the client that the server is shutting down.

Message XDR definition

```

/* NDMP_NOTIFY_CONNECTED */
enum ndmp_connect_reason
{
    NDMP_CONNECTED, /* Connect sucessfully */
    NDMP_SHUTDOWN, /* Connection shutdown */
    NDMP_REFUSED /* reach the maximum number of connections */
};
struct ndmp_notify_connected_request
{
    ndmp_connect_reason reason;
    u_short protocol_version;
    string text_reason<>;
};

```

Request Arguments

reason	Reason code describing the current connection state.
NDMP_CONNECTED	NDMP connection successfully established. This code will be returned in a message sent immediately after successful connection establishment.
NDMP_SHUTDOWN	The NDMP server is shutting down the NDMP connection. Will typically used when shutting down the NDMP host to gracefully close down the NDMP connection.
NDMP_REFUSED	NDMP connection refused by the NDMP server. This code will be returned in a message sent immediately after a connection establishment attempt to notify the NDMP client that the NDMP server is not able to accept the connection at the current time. This will typically be used if the NDMP-server implementation limits the total number of concurrent NDMP connections, when NDMP services on the NDMP host are disabled, or when the NDMP host is in the process of shutting down.
protocol_version	Version of protocol being used.
text_reason	NDMP-server implementation dependent message indicating why the connection is being shutdown or refused.

Reply Arguments

This message does not have a message body.

5.1.3 Notify Mover Halted

This message is used to notify the NDMP client that the NDMP tape server has entered the halted state.

Message XDR definition

```

/* NDMP_NOTIFY_MOVER_HALTED */
struct ndmp_notify_mover_halted_request
{
    ndmp_mover_halt_reason  reason;
    string                  text_reason<>;
};

```

Request Arguments

reason	Reason the tape server halted.
NDMP_MOVER_HALT_NA	Operation not in progress or not in the halt state.
NDMP_MOVER_HALT_CONNECTION_CLOSED	Connection to data server close detected.
NDMP_MOVER_HALT_ABORTED	Operation aborted by the NDMP client.
NDMP_MOVER_HALT_INTERNAL_ERROR	Operation halted due to unrecoverable error incurred by the tape server.
NDMP_MOVER_HALT_CONNECT_ERROR	Error establishing connection to data server.
text reason	Message providing additional diagnostic information. NDMP server implementation dependent.

Reply Arguments

This message does not have a message body.

5.1.4 Notify Mover Paused

This message is used to notify the NDMP client that the NDMP tape server has paused.

Message XDR definition

```

/* NDMP_NOTIFY_MOVER_PAUSED */
struct ndmp_notify_mover_paused_request
{
    ndmp_mover_pause_reason reason;
    ndmp_u_quad             seek_position;
};

```

Request Arguments

reason	Reason the tape server paused.
NDMP_MOVER_PAUSE_NA	Operation not in progress or not in the pause state.
NDMP_MOVER_PAUSE_EOM	Operation encountered end of media. NDMP client attention required.
NDMP_MOVER_PAUSE_EOF	Operation encountered end of file. NDMP client attention required.

NDMP_MOVER_PAUSE_SEEK	Data operation requested a seek that is outside of the current data window. NDMP client attention required.
NDMP_MOVER_PAUSE_MEDIA_ERROR	Error while reading/writing tape.
NDMP_MOVER_PAUSE_EOW	Operation encountered end of mover window. NDMP client attention required.
seek_position	If reason is NDMP_MOVER_PAUSE_SEEK, indicates the desired data stream seek position. The NDMP client should load the tape containing the requested seek_position, position the tape appropriately, set a new data window, and then continue the tape server.

Reply Arguments

This message does not have a message body.

5.1.5 Notify DATA Read

This message is used to notify the NDMP client that the NDMP server wants to read data from a remote tape server. The NDMP server must send at least one NDMP_NOTIFY_DATA_READ message to the NDMP client if the tape server is remote. In response to this message, the NDMP client will send an NDMP_MOVER_READ message to the remote tape server.

Message XDR definition

```
/* NDMP_NOTIFY_DATA_READ */
struct ndmp_notify_data_read_request
{
    ndmp_u_quad      offset;
    ndmp_u_quad      length;
};
```

Request Arguments

offset	Data stream offset of first byte that should be sent to the data connection.
length	Number of data bytes the tape server should read from tape and sent to the data connection.

Reply Arguments

This message does not have a message body.

5.2 LOG Interface

This interface is used by the NDMP server to send informational and diagnostic data to the NDMP client. This data is used by the client to monitor the progress of the currently running data operation and to diagnose problems.

5.2.1 Log Message

This request sends an informational message to the NDMP client. It is typically used to send log and diagnostic messages generated by the backup or recover method.

Message XDR definition

```

/* NDMP_LOG_MESSAGE */
enum ndmp_log_type
{
    NDMP_LOG_NORMAL,
    NDMP_LOG_DEBUG,
    NDMP_LOG_ERROR,
    NDMP_LOG_WARNING
};

struct ndmp_log_message_request
{
    ndmp_log_type    log_type;
    u_long           message_id;
    string            entry<>;
};

```

Request Arguments

log_type

one of the following:

NDMP_LOG_NORMAL: The message doesn't require immediate attention. This kind of message could be used to report the status of the backup/retrieval process.

NDMP_LOG_DEBUG: This kind of message is used for the diagnostic purpose. This feature is primarily intended to be used during software development and when troubleshooting.

NDMP_LOG_ERROR: This message reports an error condition on the server site. User should pay immediate attention to this message.

NDMP_LOG_WARNING: This message reports a warning condition on the server site. User should pay attention to this message.

message_id

message_id is NDMP server dependent.

entry

Text message.

Reply Arguments

This message does not have a message body.

5.2.2 File Recovered

This request sends a file recover message to the NDMP client. Used during recovery to notify the NDMP client that a file from the recovery list sent in the NDMP_DATA_START_RECOVER request has or has not been recovered. This message should not be sent for every recovered or failed file, just files having a name that matches a name in the recovery list.

Message XDR definition

```
/* NDMP_LOG_FILE */
struct ndmp_log_file_request
{
    string          name<>;
    ndmp_error      error;
};
```

Request Arguments

name	File name.
error	Error code.
NDMP_NO_ERR	File successfully recovered.
NDMP_PERMISSION_ERR	Permission problem.
NDMP_FILE_NOT_FOUND_ERR	File not found during restore.

Reply Arguments

This message does not have a message body.

5.3 *FILE HISTORY Interface*

The NDMP server uses this interface to send file history entries to the NDMP client. The file history entries provide a file by file record of every file backed up by the backup method. The file history data is defined using UNIX file system or NT file system compatible format. Backup method can generate either UNIX or NT, or both, file system compatible format file history for each single file. There are two sets of messages for sending file history data. The first set consists of the add path messages. The first set is for use by filename based backup methods (such as the tar and cpio commands) for which the full pathname and file attributes are available at the time each file is backed up. The second set consists of the add directory and add node messages. The second set is for use by inode based backup methods (such as the UNIX dump command) for which the full pathname is not necessarily available at the time each file is backed up. Some backup methods might not support the sending of file history data.

5.3.1 Add File

This request adds a list of file paths with the corresponding attribute entries to the file history. The file path could be the full pathname or the relative pathname to the backup root directory.

Message XDR definition

```

/* NDMP_FH_ADD_FILE */
enum ndmp_fs_type
{
    NDMP_FS_UNIX,
    NDMP_FS_NT,
    NDMP_FS_OTHER
};

typedef string          ndmp_path<>;

struct ndmp_nt_path
{
    ndmp_path    nt_path;
    ndmp_path    dos_path;
};

union ndmp_file_name switch (ndmp_fs_type fs_type)
{
    case NDMP_FS_UNIX:
        ndmp_path    unix_name;
    case NDMP_FS_NT:
        ndmp_path    nt_name;
    default:
        ndmp_path    other_name;
};

/* file type */
enum ndmp_file_type
{
    NDMP_FILE_DIR,
    NDMP_FILE_FIFO,
    NDMP_FILE_CSPEC,
    NDMP_FILE_BSPEC,
    NDMP_FILE_REG,
    NDMP_FILE_SLINK,
    NDMP_FILE_SOCKET,
    NDMP_FILE_REGISTRY,
    NDMP_FILE_OTHER
};

/* file stat */
/* invalid bit */
const NDMP_FILE_STAT_ETIME_INVALID    = 0x00000001;
const NDMP_FILE_STAT_CTIME_INVALID    = 0x00000002;
const NDMP_FILE_STAT_GROUP_INVALID    = 0x00000004;

struct ndmp_file_stat
{
    u_long          invalid;
    ndmp_fs_type    fs_type;
    ndmp_file_type  ftype;
    u_long          mtime;
    u_long          atime;
    u_long          ctime;
    u_long          owner; /* uid for UNIX, owner for NT */
    u_long          group; /* gid for UNIX, none for NT */
    u_long          fattr; /* mode for UNIX, fattr for NT */
    ndmp_u_quad     size;
    u_long          links;
};

```



```

struct ndmp_file
{
    ndmp_file_name      name<>;
    ndmp_file_stat      stat<>;
    ndmp_u_quad         node;
    ndmp_u_quad         fh_info;
};

struct ndmp_fh_add_file_request
{
    ndmp_fh_file        files<>;
};

/* no reply */

```

Request Arguments

files Array of file history files. Each file contains:

 name Array of the file names for a single file.

 The ndmp_file_name is presented by UNIX or NT, or both file path name formats.

 unix_name The full path name of the backed up file, or relative to the backup root directory. The sub-directory is separated by the UNIX path separator character, i.e. /.

 nt_name The full path name of the backed up file, or relative to the backup root directory. The sub-directory is separated by the NT path separator character, i.e. \. NT file name could contain both nt_path and dos_path names.

stat Array of the file attributes for a single file.

 The following attributes are defined in the ndmp_file_stat structure:

 invalid The invalid flag is used to identify the unsupported argument in the message.

 ftype file type.

 mtime Time the file was last modified (in seconds since 00:00:00 GMT, Jan 1, 1970).

 atime Time the file was last accessed (in seconds since 00:00:00 GMT, Jan 1, 1970). NDMP_FILE_STAT_ETIME_INVALID is set if this feature is not supported.

 ctime Time the file status was last modified (in seconds since 00:00:00 GMT, Jan 1, 1970). Indicates the last time that either the file data or the file attributes were modified.

	NDMP_FILE_STAT_CTIME_INVALID is set if this feature is not supported.
owner	File owner identifier. uid is used for UNIX file system type and owner id is used for NT file system type.
group	File group identifier. gid is used for UNIX file system type. No group information is defined in NT file system.
fattr	System file attribute. UNIX file mode is used for UNIX file system type and NT fattr is used for NT file system type.
size	File size in bytes.
links	The number of the links.
node	inode number. It is used in UNIX-like file system only. 0 is not supported.
fh_info	File history tape positioning data representing the tape position at the time the file was written to tape. This data may be used by the restore method to perform tape positioning for direct access file retrieval. The positioning data is NDMP server dependent. Typically it will be the byte or record offset from the beginning of the tape of the file to be recovered. This field is ignored by data method implementations that do not support this feature.

Reply Arguments

This message does not have a message body.

5.3.2 Add Directory

This message is used to support directory/inode type of backup formats. The `node` number can be any unique number that matches a corresponding `NDMP_FH_ADD_NODE` message.

Message XDR definition

```

/* NDMP_FH_ADD_DIR */
struct ndmp_dir
{
    ndmp_file_name    name<>;
    ndmp_u_quad       node;
    ndmp_u_quad       parent;
};

struct ndmp_fh_add_dir_request
{
    ndmp_dir          dirs<>;
};

```

Request Arguments

dirs	Array of directory entries. Each entry contains:
name	Array of the file name for a single node. This is not a full pathname; just the basename relative to the node's parent directory.
node	Node identifier that matches a node in a corresponding add node message. NDMP-server implementation dependent but will typically be the inode number of the file.
parent	Node identifier of the node's parent directory. NDMP-server implementation dependent but will typically be the inode number of the file.

Reply Arguments

This message does not have a message body.

5.3.3 Add Node

This request adds a list of file attribute entries to the file history. These entries must match a corresponding node number from a previous add directory message. For each file, this message must be sent after the corresponding *NDMP_FH_ADD_DIR* message.

Message XDR definition

```

/* NDMP_FH_ADD_NODE */
struct ndmp_node
{
    ndmp_file_stat    stats<>;
    ndmp_u_quad       node;
    ndmp_u_quad       fh_info;
};

struct ndmp_fh_add_node_request
{
    ndmp_node         nodes<>;
};

```

Request Arguments

nodes	Array of file history node entries. Each entry contains:
stats	Array of file attribute data for a single file.
node	Node identifier that matches a node in a corresponding add directory message. NDMP-server implementation dependent but will typically be the inode number of the file.
fh_info	File history tape positioning data representing the tape position at the time the file was written to tape. This data may be used by the restore method to perform tape positioning for direct access file retrieval. The positioning data is NDMP server dependent. Typically it will be the byte or record offset

from the beginning of the tape of the file to be recovered. This field is ignored by data method implementations that do not support this feature.

Reply Arguments

This message does not have a message body.

6. References

[1] RFC 1832 , "XDR: External Data Representation Standard", R. Srinivasan, Sun Microsystems, August 1995

[2] RFC 1321 , "The MD5 Message-Digest Algorithm", R. Rivest, MIT Laboratory for Computer Science and RSA Data Security, Inc. , April 1992

[3] RFC 2044, "UTF-8 a transformation format of Unicode and ISO 10646", F. Yergeau, Alis Technologies, October 1996

7. Security

The NDMP client normally is authenticated by the NDMP server using a secure MD5 digest. However, the NDMP server optionally can authenticate using a clear text password or even permit access without authentication. Once authenticated, privileges are not specified by the NDMP protocol, but it is expected that NDMP-server implementations will permit data to be transferred to and from tape using the protocol.

The NDMP SCSI_OPEN permits low level access to SCSI jukebox devices. The NDMP server should prevent access to other SCSI devices (such as disk drives) to prevent the NDMP client from bypassing file system security.

File history information is transferred to the NDMP client through a TCP/IP connection.

8. Authors

D. Hitz
Network Appliance
2770 San Tomas Expressway.
Santa Clara, CA 95051
USA
Tel: 408-367-3106
Fax: 408-367-3151
email: hitz@netapp.com
<http://www.netapp.com>

R. Stager
Intelliguard Software
6200 Village Parkway
Dublin, CA 94568
USA
Tel: 510-556-4100
Fax: 510-556-4148
email: rstager@iguard.com
<http://www.iguard.com>

9. Acknowledgement

[need more work]

Expires: June 1998