

# 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 ``1id-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-iquard-netapp-backup-05.txt >

Expires: March 1998

Document Version: 2.1.7

Last Update: 7/20/98 9:34 AM

<b>1. OVERVIEW.....</b>	<b>4</b>
1.1 MOTIVATION.....	4
1.2 AUDIENCE.....	4
1.3 TERMINOLOGY .....	4
<b>2. ARCHITECTURE .....</b>	<b>5</b>
2.1 ARCHITECTURAL MODEL.....	5
2.2 COMPARISON ARCHITECTURES.....	7
2.3 STATE DESCRIPTION .....	9
2.3.1 <i>The Data State Diagram</i> .....	9
2.3.2 <i>The Mover State Diagram</i> .....	10
2.4 PROTOCOL INTERFACES.....	13
2.4.1 <i>Messages from NDMP Client to NDMP Server</i> .....	13
2.4.2 <i>Messages from NDMP Server to NDMP Client</i> .....	13
2.5 MESSAGING PROTOCOL .....	14
2.6 HEADER .....	14
2.7 ERROR CODES.....	16
2.8 MESSAGE NUMBERS .....	18
2.9 MESSAGE DEFINITIONS .....	20
<b>3. NDMP SERVER INTERFACES .....</b>	<b>20</b>
3.1 CONNECT INTERFACE.....	20
3.1.1 <i>Open Connection</i> .....	21
3.1.2 <i>Client Authentication</i> .....	21
3.1.3 <i>Close Connection</i> .....	24
3.1.4 <i>Server Authentication</i> .....	24
3.2 CONFIG INTERFACE.....	25
3.2.1 <i>Get Host Info</i> .....	25
3.2.2 <i>Get Backup Type Attribute</i> .....	26
3.2.3 <i>Get Mover Type</i> .....	27
3.2.4 <i>Get Authentication Type Attribute</i> .....	28
3.3 SCSI INTERFACE.....	29
3.3.1 <i>Open SCSI Device</i> .....	29
3.3.2 <i>Close Device</i> .....	30
3.3.3 <i>Get SCSI State</i> .....	31
3.3.4 <i>Set SCSI Target</i> .....	32
3.3.5 <i>Reset Device</i> .....	33
3.3.6 <i>Reset Bus</i> .....	33
3.3.7 <i>Execute CDB</i> .....	34
3.4 TAPE INTERFACE .....	36
3.4.1 <i>Open Tape Device</i> .....	36
3.4.2 <i>Close Device</i> .....	37
3.4.3 <i>Get Tape State</i> .....	38
3.4.4 <i>MTIO</i> .....	39
3.4.5 <i>Write</i> .....	41
3.4.6 <i>Read</i> .....	42
3.4.7 <i>Execute CDB</i> .....	43
3.5 DATA INTERFACE .....	43
3.5.1 <i>Get Data State</i> .....	43
3.5.2 <i>Backup</i> .....	45
3.5.3 <i>Recover</i> .....	47
3.5.4 <i>Abort</i> .....	50
3.5.5 <i>Stop</i> .....	50

3.5.6	<i>Get Env</i> .....	51
3.6	MOVER INTERFACE .....	51
3.6.1	<i>Get Mover State</i> .....	51
3.6.2	<i>Listen</i> .....	54
3.6.3	<i>Set Record Size</i> .....	55
3.6.4	<i>Set Window</i> .....	56
3.6.5	<i>Continue</i> .....	57
3.6.6	<i>Abort</i> .....	57
3.6.7	<i>Stop</i> .....	58
3.6.8	<i>Read</i> .....	58
3.6.9	<i>Close</i> .....	59
<b>4.</b>	<b>NDMP CLIENT INTERFACES</b> .....	<b>61</b>
4.1	NOTIFY INTERFACE .....	61
4.1.1	<i>Notify Data Halted</i> .....	61
4.1.2	<i>Notify Connected</i> .....	61
4.1.3	<i>Notify Mover Halted</i> .....	62
4.1.4	<i>Notify Mover Paused</i> .....	63
4.1.5	<i>Notify DATA Read</i> .....	64
4.2	LOGGING INTERFACE .....	64
4.2.1	<i>Log</i> .....	64
4.2.2	<i>Debug</i> .....	65
4.2.3	<i>File Recovered</i> .....	66
4.3	FILE HISTORY INTERFACE .....	67
4.3.1	<i>Add Unix Path</i> .....	67
4.3.2	<i>Add Unix Dir</i> .....	68
4.3.3	<i>Add Unix Node</i> .....	69
<b>5.</b>	<b>REFERENCES</b> .....	<b>71</b>
<b>6.</b>	<b>SECURITY</b> .....	<b>71</b>
<b>7.</b>	<b>AUTHORS</b> .....	<b>71</b>
FIGURE 1. SIMPLE CONFIGURATION .....		5
FIGURE 2. TWO DRIVE CONFIGURATION .....		6
FIGURE 3. JUKEBOX CONFIGURATION .....		7
FIGURE 4. BACKING UP NDMP HOST THROUGH THE NETWORK TO ANOTHER NDMP HOST .....		7
FIGURE 5 - DATA STATE DIAGRAM .....		9
FIGURE 6 - MOVER STATE DIAGRAM .....		11

## **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 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.

### **1.3 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 bidirectional 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 most simple configuration, an NDMP client will back up the data from the NDMP host to a backup device connected to the NDMP host.

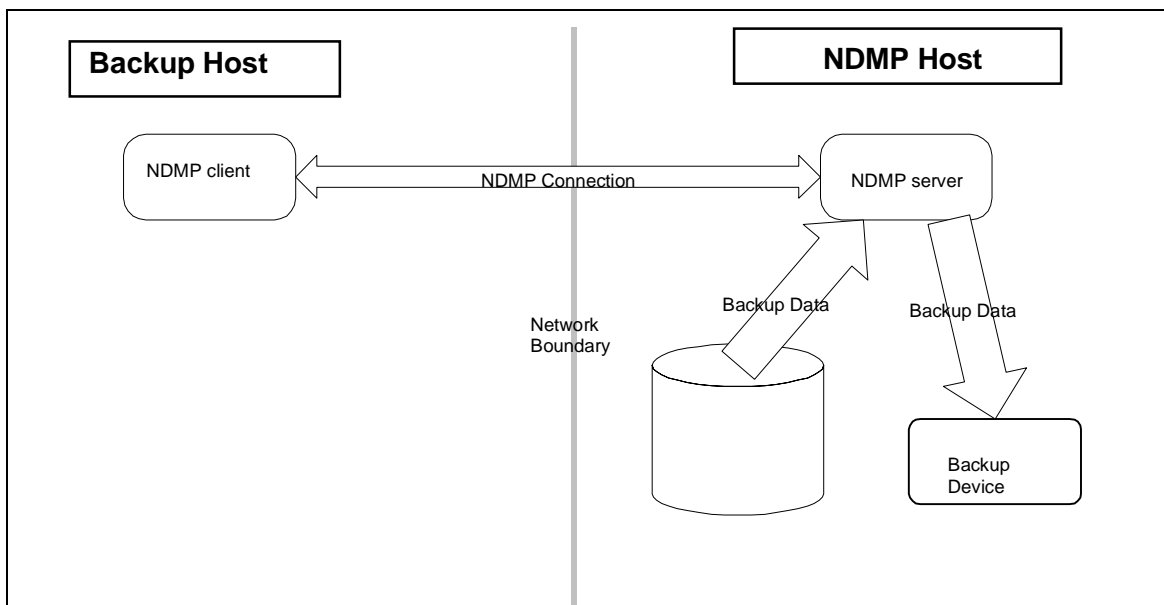
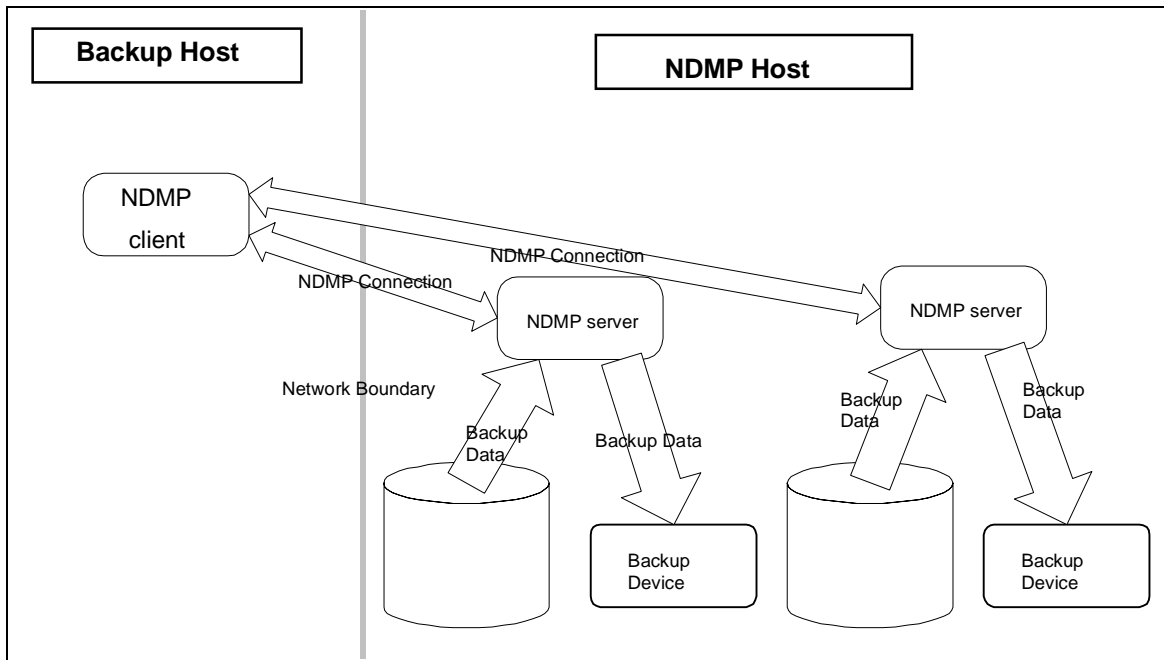


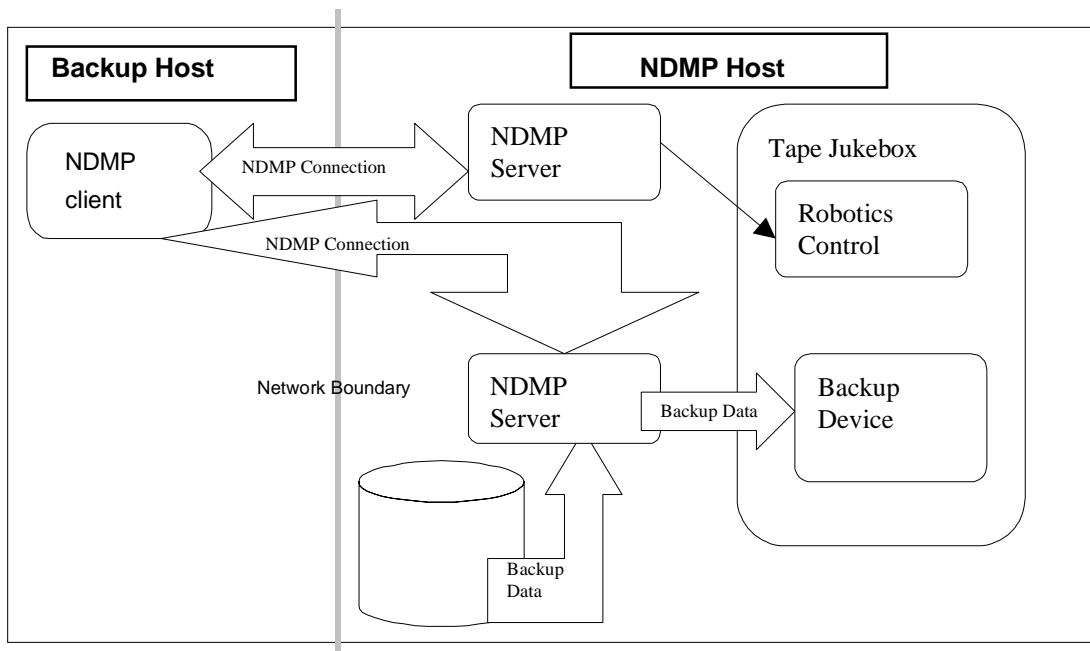
Figure 1. Simple configuration

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



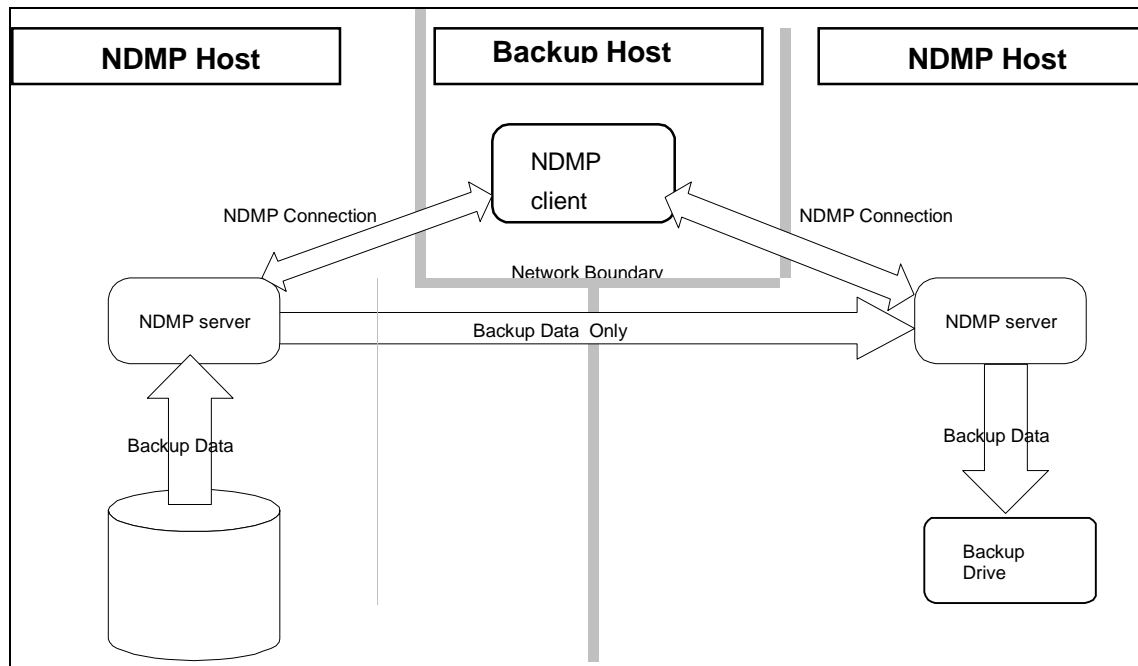
**Figure 2. Two drive configuration**

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**

## 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 demon 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 State Description

Two interfaces have states associated with them: the Data interface and the Mover interface.

### 2.3.1 The Data State Diagram

The following defines the DATA state diagram.

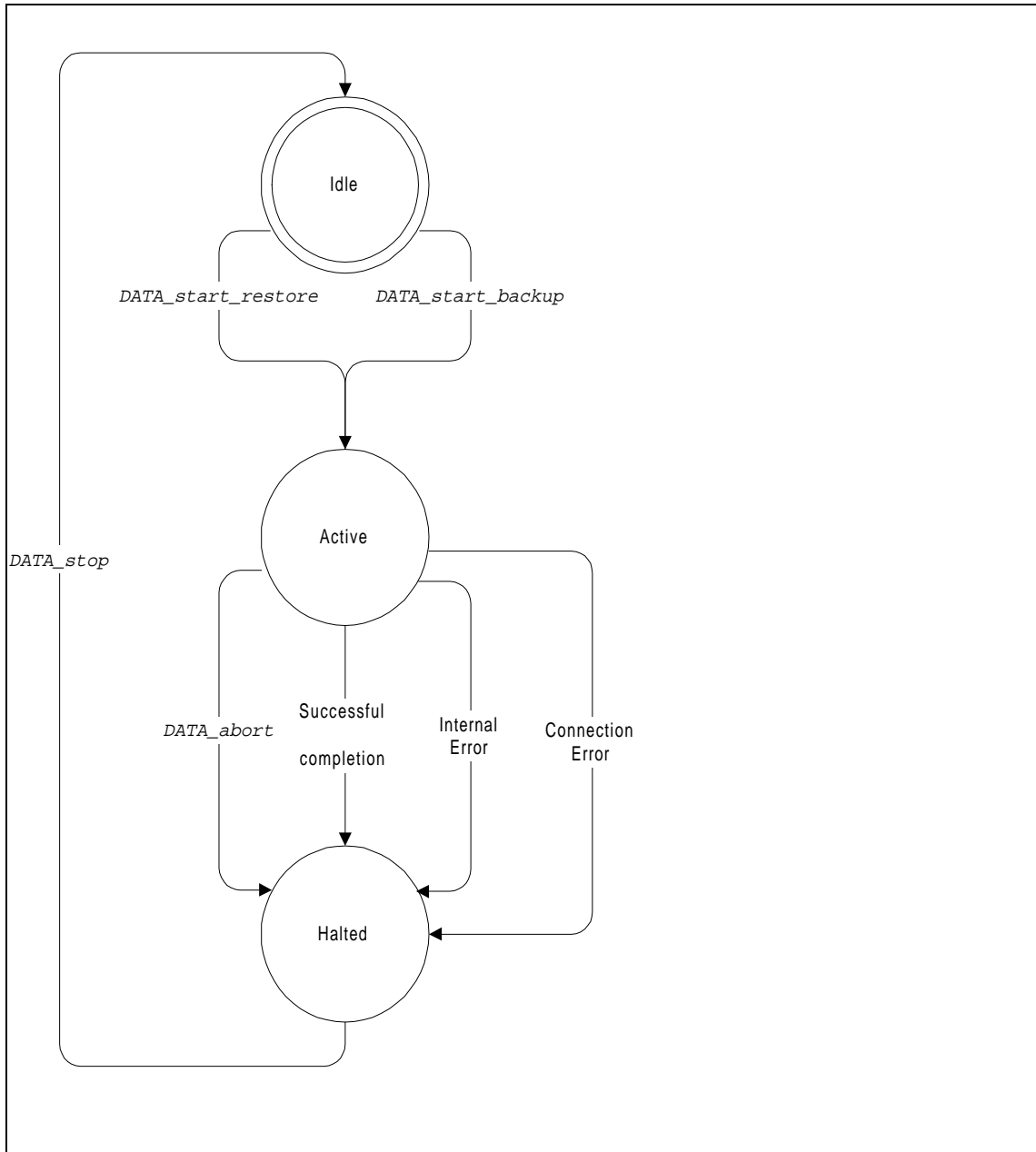


Figure 5 - Data state diagram

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

### **2.3.1.1 Idle State**

This is the start state of the state machine.

- Transition to active state upon receipt of NDMP\_DATA\_START\_BACKUP message.
- Transition to active state upon receipt of NDMP\_DATA\_START\_RECOVER message.

### **2.3.1.2 Active State**

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

- Transition to halted state upon detection of a backup/restore error.
- Transition to halted state upon receipt of NDMP\_DATA\_ABORT message.
- Transition to halted state upon completion of backup/restore.

### **2.3.1.3 Halted State**

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

- Transition to idle state upon receipt of NDMP\_DATA\_STOP message.

## **2.3.2 The Mover State Diagram**

The following defines the state diagram for the Mover interface.

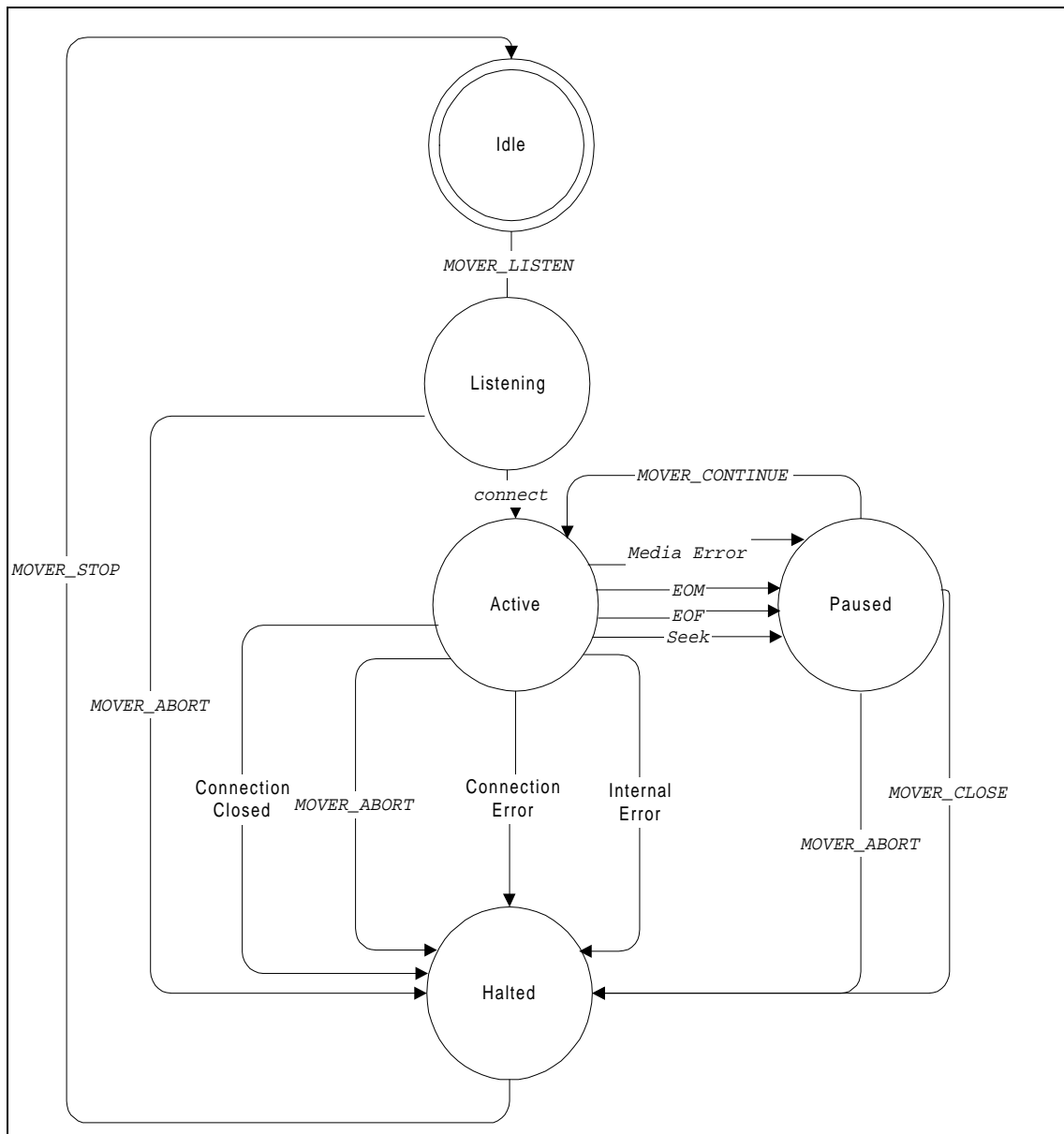


Figure 6 - Mover state diagram

### 2.3.2.1

## Idle State

This is the start state of the state machine.

- Transition to listen state upon receipt of NDMP\_MOVER\_LISTEN message.

### 2.3.2.2 Listen State

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

- Transition to active state upon establishment of connection with NDMP data server.
- Transition to halted state upon receipt of NDMP\_MOVER\_ABORT message.

### 2.3.2.3 Active State

The NDMP server remains in this state while a backup or restore 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.

### 2.3.2.4 Halted State

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

- Transition to idle state upon receipt of NDMP\_MOVER\_STOP message.

### 2.3.2.5 Paused State

The NDMP server remains in this 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.

## 2.4 *Protocol Interfaces*

Messages are grouped together by functionality into several interfaces.

### 2.4.1 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.

### 2.4.2 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.

- LOGGING 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.

## 2.5 Messaging Protocol

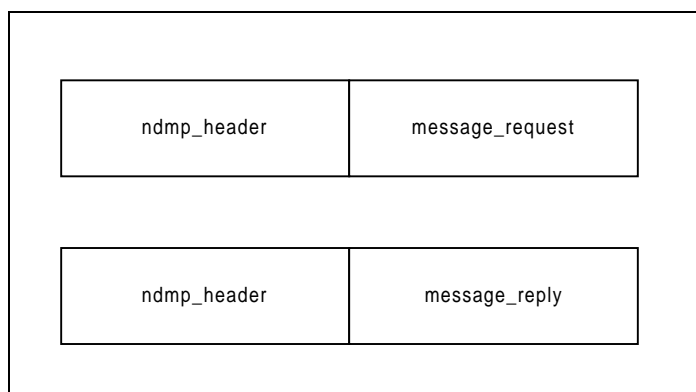
The NDMP protocol is based on XDR encoded messages transmitted over a TCP/IP connection.

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.

## 2.6 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.

## 2.7 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\_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 connect\_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.

## **2.8    *Message Numbers***

The following messages are defined:

```
enum ndmp_message
{
    /* Config Interface */
    NDMP_CONFIG_GET_HOST_INFO      = 0x100,
    NDMP_CONFIG_GET_BUTYPE_ATTR,
    NDMP_CONFIG_GET_MOVER_TYPE,
    NDMP_CONFIG_GET_AUTH_TYPE_ATTR,

    /* SCSI Interface */
    NDMP_SCSI_OPEN                  = 0x200,
    NDMP_SCSI_CLOSE,
    NDMP_SCSI_GET_STATE,
    NDMP_SCSI_SET_TARGET,
    NDMP_SCSI_RESET_DEVICE,
    NDMP_SCSI_RESET_BUS,
    NDMP_SCSI_EXECUTE_CDB,

    /* Tape Interface */
    NDMP_TAPE_OPEN                  = 0x300,
    NDMP_TAPE_CLOSE,
    NDMP_TAPE_GET_STATE,
    NDMP_TAPE_MTIO,
    NDMP_TAPE_WRITE,
    NDMP_TAPE_READ,
    NDMP_TAPE_RESVD1,
    NDMP_TAPE_EXECUTE_CDB,

    /* Data Interface */
    NDMP_DATA_GET_STATE             = 0x400,
    NDMP_DATA_START_BACKUP,
    NDMP_DATA_START_RECOVER,
    NDMP_DATA_ABORT,
    NDMP_DATA_GET_ENV,
    NDMP_DATA_RESVD1,
    NDMP_DATA_RESVD2,
    NDMP_DATA_STOP,

    /* Notify Interface */

    NDMP_NOTIFY_RESVD1              = 0x500,
    NDMP_NOTIFY_HALTED,
    NDMP_NOTIFY_CONNECTED,
    NDMP_NOTIFY_MOVER_HALTED,
    NDMP_NOTIFY_MOVER_PAUSED,
    NDMP_NOTIFY_DATA_READ,

    /* Log Interface */
    NDMP_LOG_LOG                    = 0x600,
    NDMP_LOG_DEBUG,
    NDMP_LOG_FILE,

    /* File History Interface */
    NDMP_FH_ADD_UNIX_PATH           = 0x700,
    NDMP_FH_ADD_UNIX_DIR,
    NDMP_FH_ADD_UNIX_NODE,

    /* Connect Interface */
    NDMP_CONNECT_OPEN               = 0x900,
    NDMP_CONNECT_CLIENT_AUTH,
    NDMP_CONNECT_CLOSE,
    NDMP_CONNECT_SERVER_AUTH,
```

```
/* Mover Interface */
NDMP_MOVER_GET_STATE      = 0xA00,
NDMP_MOVER_LISTEN,
NDMP_MOVER_CONTINUE,
NDMP_MOVER_ABORT,
NDMP_MOVER_STOP,
NDMP_MOVER_SET_WINDOW,
NDMP_MOVER_READ,
NDMP_MOVER_CLOSE,
NDMP_MOVER_SET_RECORD_SIZE,

/* Reserved for prototyping */
/* 0xFF00 through 0xFFFF */
NDMP_RESERVED            = 0xFF00
};
```

## 2.9 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.

## 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 to 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

Used to negotiate the protocol version to be used between the NDMP client and server. 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.
------------------	--

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.

### 3.1.2 Client Authentication

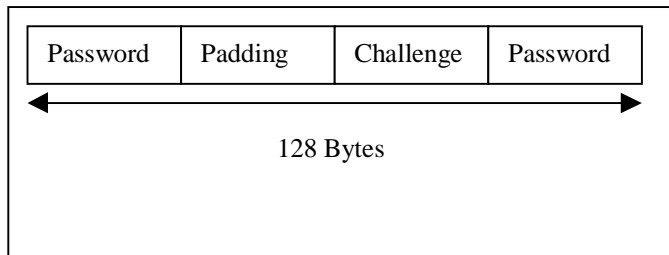
This message is used to authenticate the NDMP client to the NDMP server. Only the request messages within the CONFIG and CONNECT interfaces may be processed on a connection that has not yet been authenticated. A reply message containing an NDMP\_NOT\_AUTHORIZED\_ERR error will be returned in response 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 user name 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      user <>;
    string      password<>;
};

struct ndmp_auth_md5
{
    string      user <>;
    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:

- NONE: no authentication required.
- TEXT: connection is authenticated using a user name and non-encrypted password.
- MD5: connection is authenticated using user name and MD5 digest of the server challenge and the user 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 auth_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.
auth_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"> <li>• NONE: no authentication returned.</li> <li>• TEXT: connection is authenticated using a user name and clear text password. The user name and password authenticates the user on the NDMP client host.</li> <li>• MD5: connection is authenticated using user name and MD5 digest of the challenge and the user password.</li> </ul>

#### 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 information about the NDMP server.

#### 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<>;
    ndmp_auth_type  auth_type<>;
};

```

## 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.
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.

### 3.2.2 Get Backup Type Attribute

This message is used to query the capability of the supported backup type.

## Message XDR definition

```

/* NDMP_CONFIG_GET_BUTYPE_ATTR */
const NDMP_NO_BACKUP_FILELIST =      0x0001;
const NDMP_NO_BACKUP_FHINFO =        0x0002;
const NDMP_NO_RECOVER_FILELIST =      0x0004;
const NDMP_NO_RECOVER_FHINFO =        0x0008;
const NDMP_NO_RECOVER_RESVD =         0x0010;
const NDMP_NO_RECOVER_INC_ONLY =      0x0020;

struct ndmp_config_get_butype_attr_request
{
    string          name <>;
};

struct ndmp_config_get_butype_attr_reply
{
    ndmp_error      error;
    u_long          attrs;
};

```

## Request Arguments

name	Name of backup type for which attributes are being requested (such as dump, tar, cpio). Backup types are NDMP server
------	--

implementation dependent.

#### Reply Arguments

error	Error code.
attrs	Backup attributes bit mask. The following attribute bits are defined:
NDMP_NO_BACKUP_FILELIST	NDMP server doesn't support archiving of selective files as specified by a file list. (only supports dumping the entire specified filesystem/directory.)
NDMP_NO_BACKUP_FILEINFO	NDMP server doesn't support the file history.
NDMP_NO_RECOVER_FILELIST	NDMP server doesn't support restoration of individual files.
NDMP_NO_RECOVER_FHINFO	NDMP server doesn't support direct access restore (positioning to the offset of a backup image and restoring the specified file).
NDMP_NO_RECOVER_RESVD	This value is reserved.
NDMP_NO_RECOVER_INC_ONLY	NDMP server doesn't support incremental-only restoration (a full restore must be done before an incremental restore).

#### Reply Errors

NDMP_NO_ERR	Attributes for specified backup type successfully returned.
NDMP_ILLEGAL_ARGS_ERR	Specified backup type not supported.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.

### 3.2.3 Get Mover Type

This request returns a list of the connection methods which NDMP server supports. LOCAL and TCP are the only types supported in the current version of the protocol.

Message XDR definition

```

/* NDMP_CONFIG_GET_MOVER_TYPE */
/* no request arguments */
enum ndmp_mover_addr_type
{
    NDMP_ADDR_LOCAL,
    NDMP_ADDR_TCP
};

struct ndmp_config_get_mover_type_reply
{
    ndmp_error          error;
    ndmp_mover_addr_type methods<>;
};

```

#### Request Arguments

This request does not have a message body.

#### Reply Arguments

error	Error code.
methods	Array of supported mover methods.

#### Reply Errors

NDMP_NO_ERR	Returned the supported mover 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's 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.

### 3.3 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\_SCISI\_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 the 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_SCISI_OPEN */
struct ndmp_scisi_device
{
    string name<>;
};

struct ndmp_scisi_open_request
{
    ndmp_scisi_device    device;
};

struct ndmp_scisi_open_reply
{
    ndmp_error    error;
};

```

## Request Arguments

name	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
{
    ndmp_scsi_device    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	No SCSI device currently open by the connection.
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 */
struct ndmp_tape_device
{
    string      name<>;
};

enum ndmp_tape_open_mode
{
    NDMP_TAPE_READ_MODE,
    NDMP_TAPE_WRITE_MODE
};

struct ndmp_tape_open_request
{
    ndmp_tape_device      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.

#### 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;
};

const NDMP_TAPE_NOREWIND =      0x0008;
const NDMP_TAPE_WR_PROT =      0x0010;
const NDMP_TAPE_ERROR =        0x0020;
const NDMP_TAPE_UNLOAD =       0x0040;

struct ndmp_tape_get_state_reply
{
    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;
};

```

## Request Arguments

This message does not have a message body.

## Reply Arguments

error	Error code.
flags	Bitmask of the following tape device mode flags:
NDMP_TAPE_NOREWIND	Upon device close, the tape will not be rewind.

NDMP_TAPE_WR_PROT	The loaded tape is write-protected.
NDMP_TAPE_ERROR	A media error was detected during the previous tape operation. This bit is cleared at the start of each tape operation.
NDMP_TAPE_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.
soft_errors	Total number of soft media errors detected since the device was opened.
block_size	Tape block size in bytes. 0 if the device is in variable block size mode.
blockno	Current tape block number. First tape block is block number 0.
total_space	Total tape capacity in bytes. 0 if this feature not supported by the NDMP server implementation.
space_remain	Total remaining tape capacity in bytes. 0 if this feature not supported by the NDMP server implementation.

#### Reply Errors

NDMP_NO_ERR	Tape state successfully returned.
NDMP_DEV_NOT_OPEN_ERR	No tape 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.
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,
};

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,
};

struct ndmp_data_get_state_reply
{
    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_mover_addr       mover;
    ndmp_u_quad           read_offset;
    ndmp_u_quad           read_length;
};

```

#### Request Arguments

This message does not have a message body.

#### Reply Arguments

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.
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. Can be set to 0 to indicate that this feature is not supported by the NDMP server.
est_time_remain	Estimated number of seconds until the data operation to completes. Can be set to 0 to indicate that this feature is not supported by the NDMP server.
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 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 `NDMP_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
{
    ndmp_mover_addr    mover;
    string             bu_type<>;
    ndmp_pval          env<>;
};

struct ndmp_data_start_backup_reply
{
    ndmp_error          error;
};

```

#### Request Arguments

bu_type	The name of the backup method. Backup methods are NDMP-server implementation dependent.
env	List of parameter names and values for configuring the backup method. Backup method parameters are NDMP server implementation dependent. See below for example parameters.
mover	The mover to receive data.

#### 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 is allowed to be executing at a time.
NDMP_NOT_SUPPORTED_ERR	The request is not supported for this implementation.
NDMP_ILLEGAL_ARGS_ERR	Invalid backup method, invalid backup method parameter, or invalid backup method parameter value specified.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

The following are examples of generic environment variables that can be defined by the NDMP client.

Variable Name	Meaning	Value
<b>PREFIX</b>	prefix path for the request	path name

<b>TYPE</b>	the data type	(dump,tar,cpio)
<b>USER</b>	user id to run backup	user name
<b>HIST</b>	a flag to maintain file history	y/n

The following are examples of environment variables that can be defined by dump type.

Variable Name	Meaning	Value
<b>FILESYSTEM</b>	device or file system name to be backed up	file system or device name (/dev/rsd0a)
<b>LEVEL</b>	dump level	0 - 9
<b>EXTRACT</b>	“y” specifies to use -x option for the extraction, or -r option for the extraction.	y/n
<b>UPDATE</b>	update the dumpdates file	TRUE/FALSE

The following are examples of environment variables that can be defined by tar type.

Variable Name	Meaning	Value
<b>FILES</b>	list of files to be backed up	for example, ./* ./*.c ./*h

The following are examples of environment variables that can be defined by cpio type.

Variable Name	Meaning	Value
<b>CMD</b>	command to generate the file list for cpio.	for example, find . -print

### 3.5.3 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.

Message XDR definition

```
/* NDMP_DATA_START_RECOVER */
struct ndmp_name
{
    string          name<>;
    string          dest<>;
    u_short         resvd;
    ndmp_u_quad     fh_info;
};

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

struct ndmp_data_start_recover_reply
{
    ndmp_error       error;
};
```



### Request Arguments

mover	The mover from which to receive data.
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:
name	Name of a file/directory to be recovered. The name is the original backup path name and is relative to the backup root directory.
dest	Full destination pathname to be used when recovering the file.
resvd	Reserved.
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.

### 3.5.4 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_ILLEGAL_STATE_ERR	No data operation in progress.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

### 3.5.5 Stop

This request sends a message to inform the NDMP server that the current backup is complete. The NDMP server will change to idle state and be ready to process another request.

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.

### 3.5.6 Get Env

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

### 3.6.1 Get Mover State

This request returns the state of the mover.

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
};

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;
};

```

### 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	Awaiting 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.
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 mover.
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 request.
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.
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 mover should begin listening for a data connection from a data server. The mover returns an address that may be used by a data server to connect to the mover.

Message XDR definition

```

/* NDMP_MOVER_LISTEN */
enum ndmp_mover_mode
{
    NDMP_MOVER_MODE_READ,
    NDMP_MOVER_MODE_WRITE
};

struct ndmp_mover_tcp_addr
{
    u_long      ip_addr;
    u_short     port;
};

union ndmp_mover_addr switch (ndmp_mover_addr_type addr_type)
{
    case NDMP_ADDR_LOCAL:
        void;
    case NDMP_ADDR_TCP:
        ndmp_mover_tcp_addr addr;
};

struct ndmp_mover_listen_request
{
    ndmp_mover_mode      mode;
    ndmp_mover_addr_type addr_type;
};

struct ndmp_mover_listen_reply
{
    ndmp_error      error;
    ndmp_mover_addr mover;
};

```

Request Arguments

mode	One of the following:
NDMP_MOVER_MODE_READ	The mover 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 mover should read data from tape and write the data to the data connection. This mode is used for restore operations.
addr_type	One of the following:
NDMP_ADDR_LOCAL	Mover should listen for a connection from a data server that is colocated with the mover. This means that the data server and mover are controlled via the same NDMP client connection. The communication mechanism is

implementation dependent.

NDMP\_ADDR\_TCP

Mover should listen for a connection from a remote data server using a TCP/IP port.

#### Reply Arguments

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

#### Reply Errors

NDMP_NO_ERR	Listen successful.
NDMP_ILLEGAL_STATE_ERR	Mover 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.

### 3.6.3 Set Record Size

This request sets the record size used by the mover for all tape reads and writes. When writing to tape, the mover 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.

## 3.6.4 Set Window

This request defines the valid data window. 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 mover 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.
length	Number of bytes in the window.

## 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.

### 3.6.5 Continue

This request notifies the mover 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                                Mover successfully continued.

NDMP\_NOT\_SUPPORTED\_ERR    The request is not supported for this implementation.

NDMP\_ILLEGAL\_STATE\_ERR    Illegal state. Mover not currently in the paused state.

NDMP\_NOT\_AUTHORIZED\_ERR    Connection not authorized.

### 3.6.6 Abort

This request aborts the mover. The mover 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	Mover successfully aborted.
NDMP_ILLEGAL_STATE_ERR	Illegal state. Mover not currently in the listen, active, or paused state.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

## 3.6.7 Stop

This request frees any resources associated with the mover and returns the mover 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	Mover successfully stopped.
NDMP_ILLEGAL_STATE_ERR	Illegal state. Mover not currently in the halted state.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

## 3.6.8 Read

This request notifies the mover to begin reading backup data from the tape drive and write the data to the data connection. The mover 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 mover will notify the NDMP client and then enter the paused state. While fulfilling this request, the mover 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 mover 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 mover 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. Mover not currently in the paused state.
NDMP_NOT_AUTHORIZED_ERR	Connection not authorized.

### 3.6.9 Close

This request notifies the mover 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 code.

**4.**

Data connection successfully closed.

Illegal state. Data connection not open.

Connection not authorized.

## NDMP Client Interfaces

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

### 4.1 *NOTIFY Interface*

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

#### 4.1.1 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.
NDMP_HALT_RESVD1	Reserved.
text_reason	Diagnostic error message. NDMP server implementation dependent.

Reply Arguments

This message does not have a message body.

#### 4.1.2 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.

## 4.1.3 Notify Mover Halted

This message is used to notify the NDMP client that the NDMP mover 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 mover 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 mover.
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.

### 4.1.4 Notify Mover Paused

This message is used to notify the NDMP client that the NDMP mover 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 mover 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.
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 mover.

#### Reply Arguments

This message does not have a message body.

### 4.1.5 Notify DATA Read

This message is used to notify the NDMP client that the NDMP server wants to read data from a remote mover. The NDMP server must send at least one NOTIFY\_DATA\_READ message to the NDMP client if the mover is remote. In response to this message, the NDMP client will send an NDMP\_MOVER\_READ message to the remote mover.

#### 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 mover should read from tape and sent to the data connection.

#### Reply Arguments

This message does not have a message body.

## 4.2 **LOGGING 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.

### 4.2.1 Log

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



## Message XDR definition

```
/* NDMP_LOG_LOG */
struct ndmp_log_log_request
{
    string          entry<>;
};
```

## Request Arguments

entry                                      Text message.

## Reply Arguments

This message does not have a message body.

## 4.2.2 Debug

This request sends a diagnostic message to the NDMP client. This message is typically used to diagnose NDMP server problems. The mechanism used to enable/disable diagnostic messages is NDMP server dependent. This feature is primarily intended to be used during software development and when troubleshooting.

## Message XDR definition

```
/* NDMP_LOG_DEBUG */
enum ndmp_debug_level
{
    NDMP_DBG_USER_INFO,
    NDMP_DBG_USER_SUMMARY,
    NDMP_DBG_USER_DETAIL,
    NDMP_DBG_DIAG_INFO,
    NDMP_DBG_DIAG_SUMMARY,
    NDMP_DBG_DIAG_DETAIL,
    NDMP_DBG_PROG_INFO,
    NDMP_DBG_PROG_SUMMARY,
    NDMP_DBG_PROG_DETAIL
};

struct ndmp_log_debug_request
{
    ndmp_debug_level    level;
    string              message<>;
};
```

## Request Arguments

level                                      The level is divided into two components. The first component is the intended audience. The audience can be the end user (user), the technical support personnel (diag), or the development engineer (prog). The second component is the level of detail requested. The level of detail is specified as info, summary, and detail. There are no specific guidelines on the use of level of detail, but a message that typically is encountered less than 10 times during a backup should be an info level; a message that is encountered more than 100 times should be at a detail level.

message	Diagnostic text message
---------	-------------------------

#### Reply Arguments

This message does not have a message body.

### 4.2.3 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<>;
    u_short         reserved;
    ndmp_error      error;
};
```

#### Request Arguments

name	File name.
reserved	Reserved.
error	Error code.
NDMP_NO_ERR	File successfully recovered.
NDMP_PERMISSION_ERR	Some sort of permission problem.
NDMP_FILE_NOT_FOUND_ERR	File not found during restore.

#### Reply Arguments

This message does not have a message body.

### 4.3 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 a UNIX filesystem compatible format. There are two sets of messages for sending file history data. The first set consists of the add path message. The first set is for use by filename based backup methods (such as the UNIX 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.

#### 4.3.1 Add Unix Path

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

Message XDR definition

```
/* NDMP_FH_ADD_UNIX_PATH */
typedef string      ndmp_unix_path<>;
enum ndmp_unix_file_type
{
    NDMP_FILE_DIR,
    NDMP_FILE_FIFO,
    NDMP_FILE_CSPEC,
    NDMP_FILE_BSPEC,
    NDMP_FILE_REG,
    NDMP_FILE_SLINK,
    NDMP_FILE SOCK
};

struct ndmp_unix_file_stat
{
    ndmp_unix_file_type    ftype;
    u_long                 mtime;
    u_long                 atime;
    u_long                 ctime;
    u_long                 uid;
    u_long                 gid;
    u_long                 mode;
    ndmp_u_quad            size;
    ndmp_u_quad            fh_info;
};

struct ndmp_fh_unix_path
{
    ndmp_unix_path          name;
    ndmp_unix_file_stat     fstat;
};

struct ndmp_fh_add_unix_path_request
{
    ndmp_fh_unix_path        paths<>;
};
```

Request Arguments

paths	Array of file history path entries. Each entry contains:
name	The full pathname of the backed up file, or relative to the backup root directory.
fstat	File attribute data consisting of:
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).
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.
uid	File owner identifier.
gid	File group identifier.
mode	File mode flags.
size	File size.
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.

#### 4.3.2 Add Unix Dir

This message is used to support directory/inode types of backup formats. The node number can be any unique number that matches a corresponding fh\_add\_unix\_node message.

Message XDR definition

```

/* NDMP_FH_ADD_UNIX_DIR */
struct ndmp_fh_unix_dir
{
    ndmp_unix_path    name;
    u_long            node;
    u_long            parent;
};

struct ndmp_fh_add_unix_dir_request
{
    ndmp_fh_unix_dir    dirs<>;
};

```

#### Request Arguments

dirs	Array of directory entries. Each entry contains:
name	Node file name. 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.

### 4.3.3 Add Unix 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\_unix\_dir message.

#### Message XDR definition

```

/* NDMP_FH_ADD_UNIX_NODE */
struct ndmp_fh_unix_node
{
    ndmp_unix_file_stat    fstat;
    u_long                node;
};

struct ndmp_fh_add_unix_node_request
{
    ndmp_fh_unix_node    nodes<>;
};

```

#### Request Arguments

nodes	Array of file history node entries. Each entry contains:
-------	--

fstat	File attribute data.
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.

#### Reply Arguments

This message does not have a message body.

## 5.

## 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

## 6. 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 filesystem security.

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

## 7. 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  
111C Lindbergh Ave  
Livermore, CA 94550  
USA  
Tel: 510-449-6881  
Fax: 415-428-5151  
email: rstager@pdc.com  
<http://www.pdc.com>

Expires: March 1998