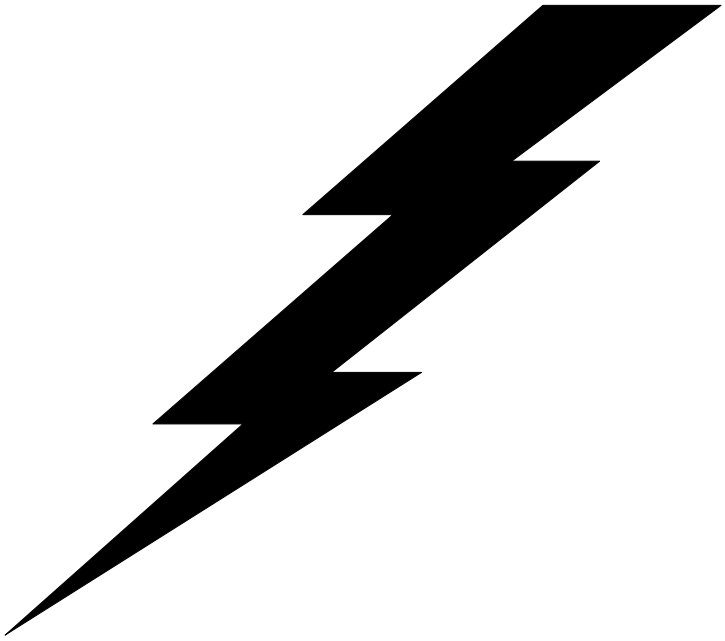


TPS®/NFM

User's guide

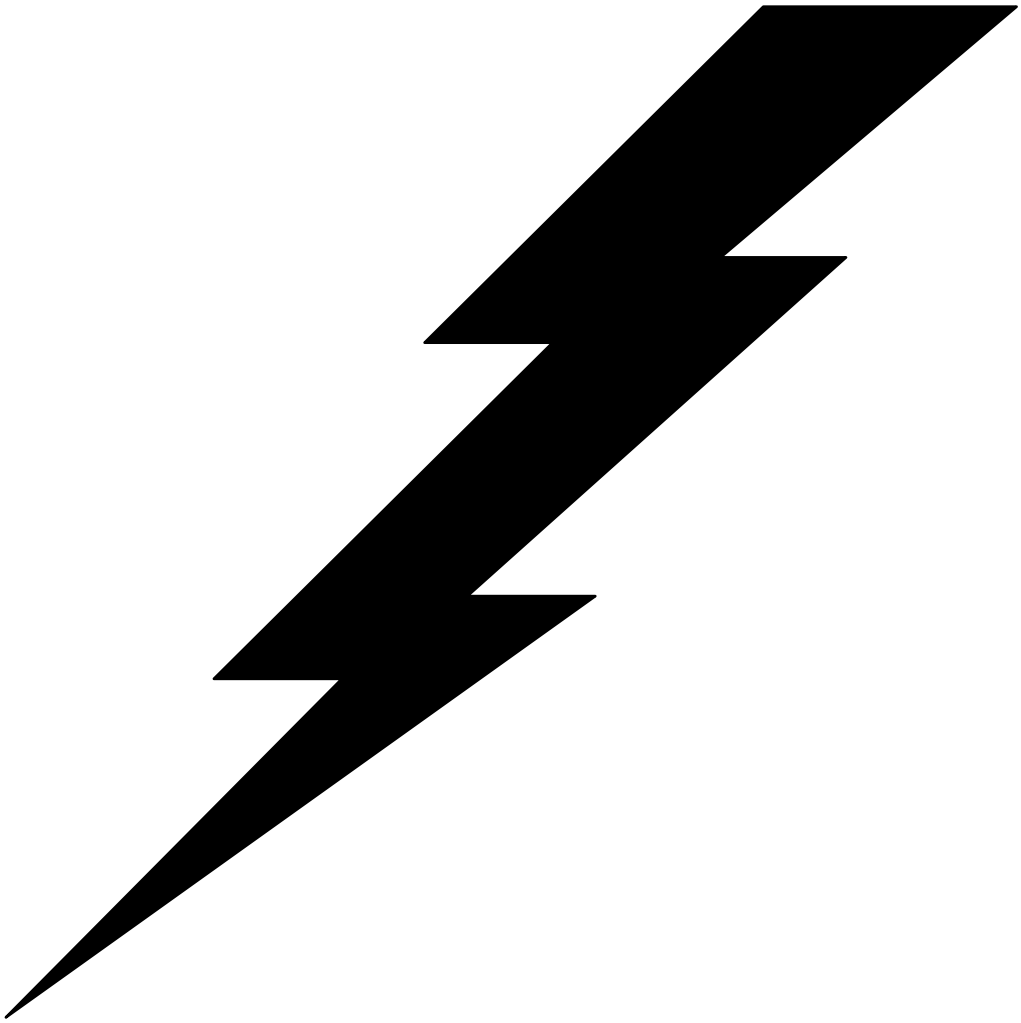
TPS® Systems, Inc.
14100 San Pedro Avenue, suite 600
San Antonio, TX 78232-4399
(210) 496-1984 voice
(210) 490-6805 fax
http://www.tps.com
support@tps.com

Software Copyright © 1999-2016 by TPS® Systems, Inc.
Manual Copyright © 1999-2016 by TPS® Systems, Inc.
All rights reserved.
International protection under the Berne Convention.

TPS® is a registered trademark owned by TPS® Systems, Inc.
Sun®, Solaris® and JAVA® are registered trademarks of Sun Microsystems, Inc.
UNIX® is a registered trademark of UNIX System Laboratories, Inc.
Linux® is a registered trademark of Linus Torvalds.
Windows, Windows 95, Windows NT, Internet Explorer are registered trademarks of Microsoft Corporation.
HP-UX is a registered trademark of Hewlett-Packard Corporation.
IBM, AIX, AS/400, and OS/390 are registered trademarks of International Business Machines Corporation.
MVS, System/36 and System/370, and IBM 4690 OS Version 2 are trademarks of International Business Machines Corporation.
Stratus is a registered trademark of Stratus Technologies International, S.a.r.l.
Android is a registered trademark of Google Inc.
AWS is a registered trademark of Amazon Web Services, Inc.

Software version 2.5.7
Manual revised 08-02-16

Table of Contents



1: Introduction

Overview

What is NFM?

Welcome to the power, flexibility, and versatility of the TPS®/Network File Manager (NFM). NFM provides advanced file transfer capability while maintaining a comfortable level of familiarity for long-time IBM® users. TPS®/NetWork File Manager is a Client/Server system designed to distribute files and other resources throughout a corporation's network. This robust, but easily, manageable file transfer mechanism allows files to be managed through a central server with file distribution taking place from the server to the client or among clients (peer-to-peer).

TPS®/NetWork File Manager Server is responsible for the management of the transmission of data from a source location to a target location. The Network File Manager system allows a user to customize this process through the use of transmission plans. Transmission plans detail the types of transfer, the files to be transferred, the nodes involved, and various other control factors. While remaining extremely flexible, transmission plans allow repetitive tasks to be performed on a regular basis with little or no user intervention. A single server or multiple servers can be configured depending on the complexity and capacity of the network. Authorized personnel using a browser interface or a stand-alone Windows application can easily manipulate the configuration and management of the file transmissions within the network. The TPS®/NetWork File Manager system uses IP sockets to communicate with NFM client enabled nodes and additionally manages FTP file transfers with non-client nodes, including mainframes and Internet FTP sites.

Highlights

- Ability to monitor and configure activity using a browser interface anywhere on the network based on security levels.
- Custom plan builder to control step-by-step file transfers and program execution.
- Sophisticated scheduler that provides automation of plan execution, including calendar based repetition.
- Dynamic monitoring capabilities for viewing plans in progress.
- Detailed error reporting capabilities.
- Remote file browsing capabilities.
- Peer-to-peer file transfers on demand.
- Transfer files with or without client software.
- Monitor node resources.
- SNA and/or TCP/IP file transfer concurrently.
- Extensive logging capabilities.
- Report activities by exporting data to third party databases.
- NetView-like transmission plan terminology.
- System security.

System components

NFM consists of five primary types of components that are available for a combination of multiple operating systems. These include the following:

- NFM Server
- TPS® Command Server
- NFM Client
- NFM Non-client node
- NFM (GUI) User interface

NFM Server The NFM server is the main 'engine' of the NFM system. It contains all configurations for the NFM system in database form at one central location. The NFM server runs as a series of unattended programs that respond to `dialog` from the NFM user interface as well as handle all scheduled and 'on demand' operations involving NFM remote stations (or nodes).

TPS® Command Server This is a separately packaged service module which is utilized by multiple TPS® Systems' software products. This program enables the NFM user interface to communicate with the NFM server. This program must be installed and running on the same computer as the NFM server to enable the user interface to function.

NFM Client This refers to any node that is running the NFM client software. The client is responsible for performing request from the NFM server to send and receive files as well as local program execution. Like the server, the client runs unattended with no local user interface. The NFM client is also responsible for supporting the remote file browsing capability inherent to the system.

The client is capable of performing file transfers directly between itself and another NFM client node. A peer-to-peer transfer of this type is still controlled by the NFM server but the data is sent directly between the clients. This reduces bandwidth requirements at the server while still maintaining centralized control.

The NFM client package also includes a command line utility program that enables the NFM remote server interface feature to be utilized. This feature allows programs or operators at the NFM client site to initiate NFM activity, including uploading files, without the need for the NFM server to continually poll the client to decide when to begin transferring.

Because of the peer-to-peer nature of the NFM system, the NFM server computer must also be defined as an NFM client if it is to be accessed as a node during plan activity. Any use of the server as a file 'repository' is simply done by design in the implementation of user written plans.

NFM Non-client node This refers to any node that is defined to the NFM server but is not running the NFM client software. These nodes rely on a different mechanism to facilitate file transfer and program execution (but not necessarily both). An example would be an FTP site, which could be an Internet site, where the NFM client could not be installed. The available NFM functionality to such nodes will vary based on their type. For example, the NFM file browsing capability also works with an FTP type node but not an HTTP one. An SOCKET node has the capability of executing a program but an FTP one does not.

Additionally, during a file transfer between a source and target node, at least one of the two nodes must be an NFM client node. In order to transfer between two non-clients sites it would be necessary to pull files from the source site to an NFM client and then separately push the files to the target site.

NFM User interface This is the JAVA-based user interface for the NFM system. This program communicates directly to the NFM server to perform and monitor NFM system activities. Any number of users can be logged on simultaneously to the NFM system. The user interface stores no control or configuration information on the local computer. It acts only as a graphics terminal into the NFM system. There are two different methods for configuring an NFM user interface:

Browser interface This option is accessed by simply running a browser and visiting an HTTP (web) server (typically also the NFM Server computer) that hosts the NFM user interface Java application. Although it requires some additional setup, the browser interface has the following advantage over using the stand-alone option:

- It makes the user interface immediately available to any Java enabled web browser that can access the web server on the network, without the need to install additional NFM software on the computer.
- Any future upgrades to the centrally located NFM user interface are automatically available to all browser users.

Note: There may be a noticeable delay on a computer when first accessing the user interface with the browser. Afterwards however the Java application becomes cached and is immediately available.

The necessary NFM software for this option is automatically installed as part of the NFM server. Configuring the NFM browser interface is described in section "Configuring the User Interface" at the beginning of chapter 4 "Configuration".

Stand-alone interface This option requires installing a Windows stand-alone program on the computer that will access the user interface.

The initial screen looks different between these two methods, but once logged on to the browser, it opens a new window at which point the two methods are virtually identical.

Communication between components With the exception of the NFM non-client nodes, all components of the NFM system communicate with one another via TCP/IP sockets. The server, client, and user interface may all be running on different

computers, or all on the same computer. If the NFM server is communicating with an NFM client running on the same computer, it is not even aware of it (it is simply on a loop-back socket). Likewise, if the user interface is running on a computer that is also enabled as an NFM client node, they are not aware of one another. This provides great flexibility in the implementation and load balancing requirements of the system.

Required hardware and software

Software

The following section shows the supported platforms for each of the NFM components along with version requirements and optional software information.

NFM server

The NFM server is currently supported on the following operating systems:

- AIX® 4.1 or higher
- Linux®
- Windows NT® 4.0 or higher
- Windows 2000®

The following software may also be necessary:

- If the NFM server is installed on UNIX (AIX or Linux) and the user interface will operate using a web server, an optional HTML server must be installed as well. The Windows/NT and 2000 systems have this support by default, but they must be enabled.

NFM client

The NFM client is currently supported on the following operating systems:

- AIX® 4.1 or higher
- Linux®
- HP-UX®
- Sun Solaris®
- SCO®
- Windows NT/XP/Vista/2000/2003/2008/7/8®
- IBM 4690 OS Version 2®
- Stratus® VOS version 12.4.0m or higher with OS TCP/IP
- OS/390® MVS™ V2R5 or higher

NFM non-client

Currently, there are four types of NFM non-clients, the necessary hardware and software will vary based on the individual computer types but basically include:

- Any computer that can serve as a standard FTP server. This can include Internet FTP sites.
- Any computer that can serve as a standard HTTP server. This can include Internet HTTP sites.
- The Cloud. The NFM Server can execute AWS CLI S3 commands via an NFM Client. This can include Internet S3 or Intranet S3 sites.

NFM user interface The NFM user interface operates from a browser or from a stand-alone Windows application. The following software is necessary on a system that has TCP/IP access to the NFM server computer:

- If running from a browser, any browser that supports JAVA version 1.5 or higher should work but a Windows system running Firefox or Internet Explorer 4.01 or higher is recommended.
- A stand-alone Windows application can be installed on any Windows system Windows/NT or newer.

Hardware An adapter card may be required for the connection from the RISC System/6000 to the other nodes. Your connection requirements and communication software requirements determine the type of adapter.

Network The network requirements will vary based on the types of communication desired. The standard NFM client using TCP/IP sockets connections requires a typical LAN or virtual LAN network. NFM defaults to using port `8008` to communicate between the NFM server and the NFM clients as well as directly between the NFM clients. This value can be changed on a per client basis by specifying a port number in the communications name field of the node record, as well as a startup parameter for the NFM client program for the corresponding node. This is explained further in the Node Configuration section.

NFM also expects to be able to initiate a sockets connection directly between any of the sockets type nodes, in either direction. This can cause a problem for communicating with nodes outside of a firewall due to the typical firewall setup, which is to keep outside nodes from connecting to nodes inside the firewall. There is an optional field in the node settings called `Answer Mode` that addresses this issue.

Using this book

Organization Chapter 2, "Installation," will help you place the files onto your system and provides a list of those files. Chapter 3, "Theory," is designed to give you the basic knowledge of TPS®/NFM necessary to begin configuring the software.

Conventions The following typographical conventions are used throughout the book:

A `typewriter-style font` is used to indicate keywords, messages, and other characters that are shown as they appear on your screen.

Command syntax obeys the following rules:

- Standard (non-italic) font is used for elements that are to be entered exactly as shown (capitalization, spaces, and punctuation are all significant), as in:

```
tpscserv
```

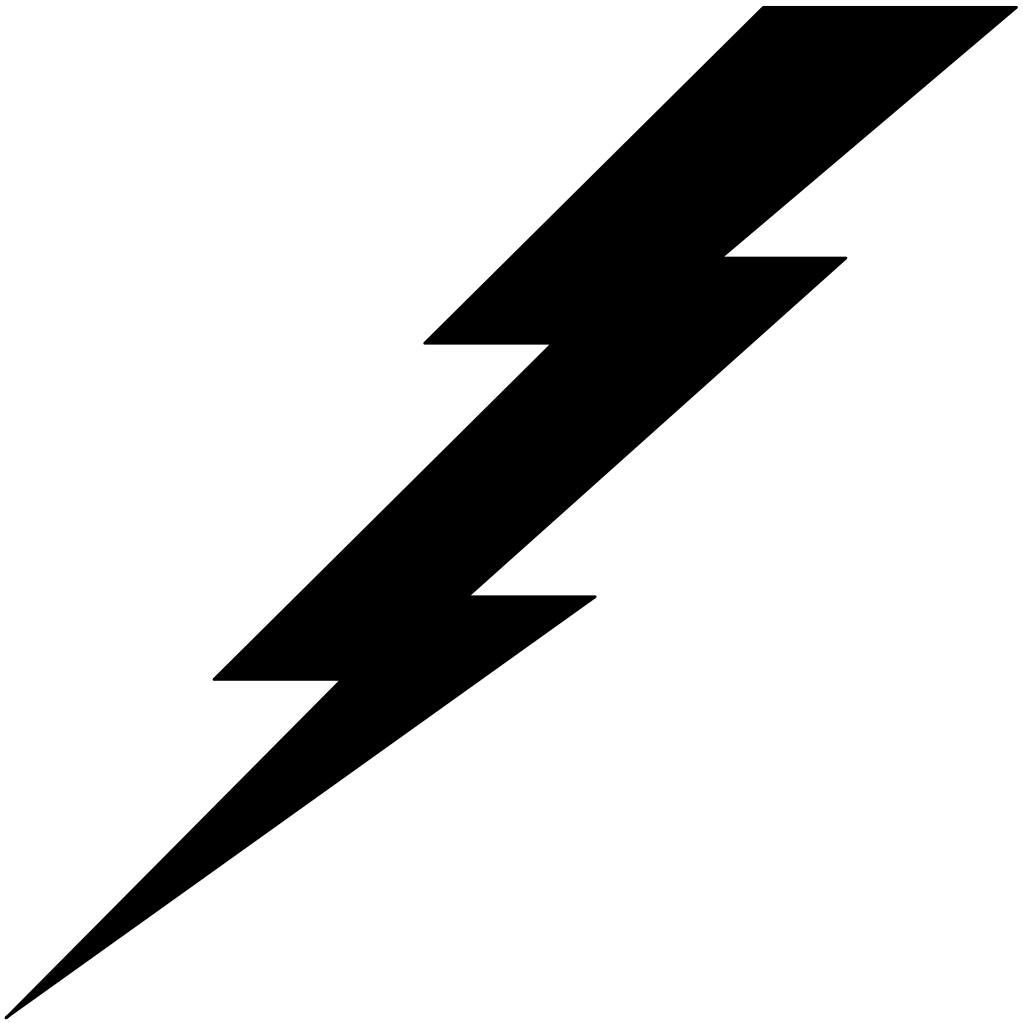
- Italics are used to indicate a parameter for which you are to substitute the value that meets your specific requirements. In the following:

  ```
  tpscserv logs
  ```

  you should substitute the name of a specific configuration file for the italicized expression *logs*.

- Brackets indicate optional elements. In the following example, the `-x` is optional:

  ```
  tpscserv [-x]
  ```



2: Installation

Overview

TPS®/NFM programs are shipped in the appropriate format for your system. Instructions for installation of the various components differ between operating systems. Refer to the appropriate section in the remainder of this chapter for installation procedures on any given platform.

NOTE: If you are updating or reinstalling the product, shutdown the existing version of the product before continuing.

TPS®/NFM programs are shipped in the appropriate format for your system.
TPS®/NFM programs are shipped in the appropriate format for your system. Instructions for installation of the various components differ

Unix Overview

NFM components install and run similarly on most UNIX platforms. Files are installed in locations that follow UNIX conventions, with executables and other static files placed in sub-directories within "`/opt`" or `/usr/lpp` (on AIX) and variable files (database, work files, etc.) placed in sub-directories within `/var`. The installation process will also add some entries to the file `/etc/inittab` to control automatic startup of the necessary programs. Several symbolic links are also created in `/usr/bin` to make common NFM commands available on the path.

AIX installation

Overview

TPS®/NFM programs are installed on your system through `installp` (a utility included with the AIX operating system) or through `smit` (recommended for 4.1). `installp` allows you to install, update, remove, or restore a previous version of a particular software package. See your AIX system documentation or use the `man` command to learn more about `installp`.

Common Steps

Perform the following to install each component on an AIX system:

- Change to `root` user.
- To install from diskettes type the command:

  ```
  installp -acF all_
  ```

  Change diskettes when prompted to do so.

- To install from a file (via FTP) type the command:

```
installp -acF -d <filename> all
```

NFM server The following directories will be created:

```
/usr/lpp/tps/nfm
/var/tps/nfm
/var/tps/dba
```

TPS® Command Server The following directories will be created:

```
/usr/lpp/tps/cserv
/var/tps/cserv
```

NFM client The following directories will be created:

```
/usr/lpp/tps/nfmc
/var/tps/nfmc
```

Linux installation

Overview Components are installed on Linux through the use of an install script. Each component consists of an install script and an install `image.file`. Both files should be placed in a temporary directory on the target computer.

NFM server Files: `installp_tpsnfm` and `tpsnfm.tar.Z`

- Change to `root` user.
- Type the command: ./`installp_tpsnfm`

The following directories will be created:

```
/opt/tps/nfm
/var/tps/nfm
/var/tps/dba
```

TPS® Command Server Files: `installp_tpscserv` and `tpscserv.tar.Z`

- Change to `root` user.
- Type the command: ./`installp_tpscserv`

The following directories will be created:

```
/opt/tps/cserv
/var/tps/cserv
```

NFM client Files: `installp_tpsnfmc` and `tpsnfmc.tar.Z`

- Change to `root` user.
- Type the command: `./installp_tpsnfmc`

The following directories will be created:

```
/opt/tpsnfmc
/var/tpsnfmc
```

Sun Solaris installation

Overview

Components are installed on Solaris using the `pkgadd` utility to install a product image file. This file should be placed in a temporary directory on the target computer.

NFM client

Files: `tpsnfmc`

- Change to `root` user.
- Type the command: `pkgadd -d <filename>`

See your SUN system documentation or use the man command to learn more about `pkgadd`.

The following directories will be created:

```
/opt/tps/nfmc
/var/tps/nfmc
```

HP-UX installation

Overview

Components are installed on HP using the `swinstall` utility to install a product image file. This file should be placed in a temporary directory on the target computer.

NFM client

Files: `tpsnfmc`

- Change to `root` user.
- Type the command: `swinstall -s <filename>`

See your HP system documentation or use the man command to learn more about `swinstall`.

The following directories will be created:

```
/opt/tps/nfmc
/var/tps/nfmc
```

SCO OpenServer 5 installation

Overview Components are installed on SCO OpenServer 5 using the `pkgadd` utility to install a product image file. This file should be placed in a temporary directory on the target computer.

NFM client Files: `tpsnfmc`

- Change to `root` user.
- Type the command: `pkgadd -d <filename>`

See your SCO system documentation or use the man command to learn more about `pkgadd`.

The following directories will be created:

```
/opt/tps/nfmc
/var/tps/nfmc
```

Windows installation

Overview TPS®/NFM programs are installed on the various Windows platforms by way of self-extracting executables. These programs should simply be run from the `Start/Run` menu.

Each component is installed in its own base directory with various sub-directories created within. The location of these directories may be specified during the install procedure, otherwise, they will default to "`/Program Files/TPS Systems/<directory>`". The system registry will be updated with various information as well.

NFM server Executable program will install the server in "`/Program Files/TPS Systems/NFMServer`"

TPS® Command Server Executable program will install the TPS Command Server in "`/Program Files/TPS Systems/CSERV`"

NFM client Executable program will install the client in "`/Program Files/TPS Systems/NFMClient`"

NFM user interface (stand-alone option) Executable program will install the NFM Java Server Interface in "`/Program Files/TPS Systems/Network File Manager`". Also an entry will be added to the start menu in `Programs` →

`TPS Systems` → `Network File Manager` → `NFM Java Server Interface.`

NFM user interface (browser option) This requires no installation of software. The browser will be directed to a specific HTML entry on the NFM server computer.

x64 Notes: Some systems may not have all the required system DLL modules installed. Included in the NFM installations is the Microsoft Visual C++ 2005 Redistributable Package (`vcredist_64.exe`). It is located in the `vcredist` subdirectory. This executable is not installed by default. You must select `Custom` (and select it) or `Complete` installation to get it installed. If you have already installed NFM but not the package you can reinstall NFM and select the `Repair` option to add the package to the installation. You execute the installation from the NFM final installation window or run this program directly to install the required DLL modules. After the package is installed Windows Update should be executed to have the system check for any updates (`KB2538242`) to the package.

4690 installation

NFM client Perform the following steps:

- Create a subdirectory for the product in any desired location.

  Example: `mkdir C:/ADX_UPGM/NFMC.`

- Copy the NFM client modules from the install medium to this directory.

  Example: `copy a:/*.* C:/ADX_UPGM/NFMC.`
  (This will include `NFMC.386` and `NFMI.386` at this time.)

- From the `Controller` configuration menu, create a background application that starts `NFMC.386` automatically from the installed location.

NOTE: The NFM client will automatically create needed sub-directories in the installed location.

Stratus installation

NFM client Perform the following steps:

- Create a subdirectory for the product in any desired location, example:

  `create_directory (master_disk)>system>nfmc`

- Transfer the `nfmclient.pm` and `nfmi.pm` modules to this directory.

- Verify that both `nfmclient.pm` and `nfmi.pm` are fixed files with record length 4096 as required for VOS program modules.

NOTE: `nfmclient.pm` will automatically create needed `>tmp` and `>logs` sub-directories in the current directory at startup.

The NFM client must be started and left executing on Stratus for it to respond to incoming request from the NFM server. In most cases it will be useful to create a command macro (`.cm`) to start `nfmclient.pm` with the appropriate options:

```
!start_process (master_disk)>system>nfmc>nfmclient.pm &+
-output_path (master_disk)>system>nfmc>nfmclient.out &+
-process_name nfmclient &+  -privileged &+
-current_dir (master_disk)>system>nfmc
```

OS/390 installation

Overview

Components are installed on OS/390 using the `TSO RECEIVE` command to install a product image file. This file should be placed in a temporary dataset on the target computer.

NFM client

Files: `PACKAGE.NFMC`

Perform the following steps:

- Transfer the "`PACKAGE.NFMC`" file to the host as an MVS dataset with the parameters - `SEQ`, `FB`, and record size `80`. Make sure this is done as a binary transfer. The product defaults to a top-level qualifier of "`TPS01`".

- From the TSO command line, unpackage the file as follows:

```
RECEIVE INDATASET('TPS01.PACKAGE.NFMC')
```

- After successful execution of the `RECEIVE` command, the following NFM components are installed:

`TPS01.NFMC2.BIN(NFMC)`	The primary NFM client load module.
`TPS01.NFMC2.BIN(NFMI)`	The NFM remote server interface module.

Starting the NFM client:

The NFM client must be started and left executing on OS/390 for it to respond to incoming request from the NFM server. The program module can be executed as a 'started task,' or simply executed as a batch job.

Example JCL for starting the NFM Client for OS/390:

```
//TPS01//TPS01NFM JOB (TPS0),'TPS.SYS',
//              CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//RUNNFMC  EXEC PGM=NFMC,REGION=4096K,PARM='-T'
//STEPLIB  DD DISP=SHR,DSN=TPS01.NFMC.LOAD
//NFMJOB   DD SYSOUT=(A,INTRDR)
//NFMLOG   DD SYSOUT=A
//NFMERR   DD SYSOUT=A
//
```

Starting the NFM remote server interface:

The NFM remote server interface program, or NFMI, is available in `'TPS01.NFMC.LOAD(NFMI)'`. It can be loaded as a batch program using JCL, or as a direct interactive program for TSO. Once started, its user interface is fully documented in Chapter 8.

Example JCL for starting the NFM remote server interface for OS/390:

```
//TPS01NFI JOB (TPS0),'TPS.SYS',
//              CLASS=A,MSGCLASS=X,MSGLEVEL=(1,1)
//RUNNFMI  EXEC PGM=NFMI,REGION=4096K,PARM='CMDFILE=CMD.DATA'
//STEPLIB  DD DISP=SHR,DSN=TPS01.NFMC.LOAD
//NFMLOG   DD SYSOUT=A
//NFMERR   DD SYSOUT=A
//
```

i5 (AS/400) installation

Overview

A component `SAVF` file can be placed on `i5` using the FTP Server. The Restore Object (`RSTOBJ`) command is used to place the product image files in a library. The `SAVF` file should be placed in the target library on the computer.

NFM client

Files: `tpsnfmc-X.X.X.X-AS400-PPC-V5R4M0-64bit.zip`

Perform the following steps:

- Unzip the file to extract the file `NFMC.SAVF`.
- On the i5, create a new `SAVF` file with the `CRTSAVF` command. For example: "`CRTSAVF.CURLIB/NFMC`". Replace `CURLIB` with the name of the existing library to hold the file.
- Transfer the "`NFMC.SAVF`" file to the `i5` as a binary file. FTP it directly into the `CURLIB/NFMC` file.

- From the command line (or menu), restore the files as follows (replace `CURLIB` with your source library name and replace `NEWLIB` with your target library name):

```
  RSTOBJ   OBJ(*ALL)   SAVLIB(TPSSL1)   RSTLIB(NEWLIB)
DEV(*SAVF) SAVF(CURLIB/NFMC)
```

- After successful execution of the RSTOBJ command, the following NFM components are installed in NEWLIB:

NFMC.PGM	The NFM client program.
NFMI.PGM	The NFM remote server interface program.

Starting the NFM client:

The NFM client must be started and left executing on i/5 for it to respond to incoming requests from the NFM server. The NFMC program can be executed at boot time via the QSTRUPPGM, directly or simply executed as a batch job as follows:

Directly:

```
call pgm(NEWLIB/NFMC) parm('-HHOMEDIR')
```

Batch:

```
sbmjob cmd(call pgm(NEWLIB/NFMC) parm('-
  HHOMEDIR')) job(NFMC) jobpty(3) user(USER)
  cpyenvvar(*yes) alwmltthd(*yes)
```

Replace **NEWLIB** with the name of the library that the NFMC program is located. Replace **HOMEDIR** with the path of the home directory (.i.e., /home/TPSDIR/nfmc) for the NFMC client. This directory will be the location for the configuration files and subdirectories (logs, tmp, audit, etc.) used by the client. This directory must exist and should be based on the Root (/) file system. Replace **USER** with the user name you wish the client to execute as. You can also change the priority of the NFM client via the jobpty parameter.

Starting the NFM remote server interface:

The NFM remote server interface program, or NFMI, is available via the NFMI.PGM program. It can be executed directly or as a batch program. Once started, its user interface is fully documented in Chapter 8.

Directly:

```
call pgm(NEWLIB/NFMI) parm('-HHOMEDIR' 'PARM2' 'PARM
   3' . . . 'PARMn')
```

Batch:

```
sbmjob cmd(call pgm(NEWLIB/NFMI) parm('-
   HHOMEDIR''PARM2' 'PARM 3' . . . 'PARMn'))
   job(NFMI) jobpty(3) user(USER) cpyenvvar(*yes)
   alwmltthd(*yes)
```

As before replace **NEWLIB**, **HOMEDIR**, **USER**, and the jobpty parameter with appropriate values. Also, you can have additional parameters besides the '-H**HOMEDIR**' just by adding them to the parameter list as shown with the 'PARMx' entries. The single quotes are required around each parameter and each parameter is separated with at least one space.

NOTES:

1) The programs are multithreaded so the QIBM_MULTI_THREADED environment variable must be set to 'Y'.
2) The programs will only be able to read/write on file systems that are thread-safe.

Android installation and setup

Overview

Components are provided as APK files. They should be placed within a Web servers file system making them available for download from an Android device that visits the appropriate web site with its browser, allowing you to install from a location other than the Android Market.

TPS may publish the application or make it available through another means in the future.

NFM client

File: NFMClientDroid.apk

Once installed the NFM client icon should be available within the applications list.

Starting the NFM client:

Select the NFM client icon to access the NFM Client activity manager. This screen will allow you to configure, start and stop the NFM client service which runs in the background.

Configuration

As with all NFM clients, some setup is required at the NFM server as well as some local configuration prior to being able to use the Android device for file transfers and other NFM activity. The setup at the NFM server for all NFM clients is described in detail in Chapter 4 "Configuration". The next section offers an overview of the items necessary for the Android client and describes some settings that may differ somewhat from other NFM client setups. The section after that details the NFM client settings configured locally on the Android device.

NFM server setup

The following items must be configured at the NFM server prior to using the NFM Client for Android:

- Create a model record representing an Android NFM client, choosing a name as you see fit (like ANDROID_SOCKETS). As usual, required field settings may be done in the model or node record as an override.
- In the "Settings" tab, set the field "OS Name" to the value "Android" in the drop down selection.
- Create a node record specific for this Android device, in the "Node Settings" tab, assign it a unique name, and assign it the model created above.
- In either the model or node "Settings" tab, designate the "Encryption" field to SSL if it is desired that all communications to the Android device be SSL encrypted.
- No other settings are required at this time. Unlike other NFM clients that default to a listen connection, an Android node by default is set to be a "CALL" type call-in connection. It is therefore not necessary to set the "Communications Name" field, nor is it necessary to create a connection entry in the "Connections" tab unless you require using a port other than the default.
- If the NFM server is running behind a firewall, it will be necessary to open up the port used for Nodes such as the android to make connections in to the NFM server. This is port 8028 by default.

Android Client settings

Most NFM clients are configured by editing a text configuration file in the NFM clients install directory. A complete description of this may be found in the section "NFM Client Configuration" near the end of Chapter 4 "Configuration". The NFM Client for Android, however, does not use a configuration file but instead is configured through a settings menu that is available from the NFM Client activity window on the Android. Some of these fields duplicate settings described in the general client configuration mentioned above and a review of that section of the manual is recommended. Other settings are unique to the Android environment.

Once the NFM Client activity window is started, the settings menu is available by pressing the menu button and selecting "settings".

NFM Server location

This must be set to the host DNS name or IP address of the NFM server computer.

Node Record Name

Set this to the node name chosen for the node entry created at the NFM server to represent this Android device.

Use SSL Encryption (recommended)

Check this field if SSL encryption is desired. This must match the encryption type specified in the corresponding node record on the NFM server.

Additional Options: You should be able to connect the NFM client after the above fields are set. Additional optional settings are as follows:

Use Wi-Fi Only

This setting is used to restrict NFM activity to a Wi-Fi network only. This may be desirable if multiple networks are available.

Automatic Startup

Check this setting if you want the NFM client to automatically connect to the NFM server (and be available for file transfers), when the Android device is powered on or restarted. This is described in more detail below in "Operating the NFM client". It is recommended that this setting not be enabled until the user is able to manually connect successfully to the NFM server.

Diagnostics

This setting along with the "Log Directory" are used for problem determination and are intended for use by a systems support person. It recommended that these fields not be changed. This setting causes the NFM client to run in a diagnostic mode and generate diagnostic trace data into files residing on the Android. Use of this setting may substantially impact the performance of the NFM Android client.

Log Directory

When diagnostics is enabled, log files will be written to the directory "/sdcard/nfmc.logs/" by default. This setting may be used to specify an alternate location if desired. Note: This should specify a full path to the directory (beginning with a forward slash).

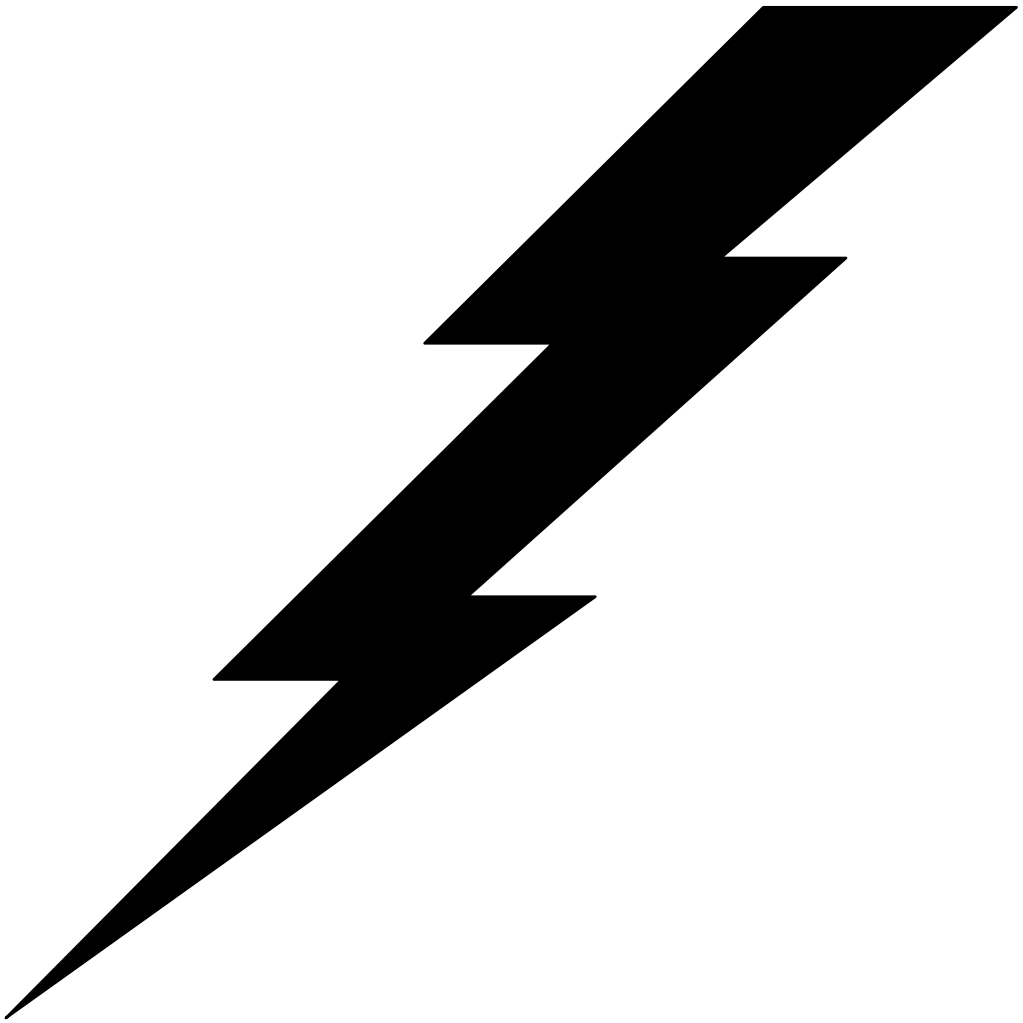
Operating the NFM Client for Android

The NFM client on the Android has two distinct parts to it:

- The NFM application or "activity window" is loaded when you select the icon. It may be unloaded by pressing the "exit" button, and may be closed arbitrarily by the Android when it is not actively being used in the foreground.
- The NFM service represents the actual NFM client piece itself. It runs unattended in the background and is either connected or stopped, regardless of whether or not the activity window is running.

The activity window may be used to manually start or stop the NFM client service. The menu setting "Automatic startup" allows the service to start upon powering on the Android even if the activity windows has never been started.

Once you have started the NFM client application by selecting the icon, the current state of the service (connected, stopped, etc.) should be displayed. Once properly configured, you should be able to manually connect or stop the service using the "Enable" button. Any errors that occur while connecting should be displayed on the screen.



3: Theory

Overview

What next?

Once TPS®/NFM programs have been installed (as detailed in Chapter 2), you are ready to setup the package for your particular needs. In general, this requires several steps:

- setting up the users
- defining your environment (nodes, groups, models and files)
- configuring your transmission plans

This chapter discusses the various features of TPS®/NFM, and summarizes the different configuration components that make up your working environment. This will give you the necessary background for configuring the system.

Detailed instructions for setting up TPS®/NFM are given in Chapter 4.

Configuration database

All configurable items reside in the NFM database located on the NFM server computer. The process of configuration adds or updates these records through the NFM user interface. These components include the following:

- Users
- Usergroups
- Models
- Nodes
- Node groups
- Filesets
- Plans

Users

The NFM system maintains its own set of user records that control access to the various NFM activities. A user must sign on/off the NFM system with a password and individual permissions are maintained that control the users abilities to read/modify different components of the NFM system. There is also the ability to independently restrict users from reading from, or writing to specific nodes.

The user name has no relationship whatsoever to any operating system logon name. The user permissions designated in these records can only be created or changed by a system administrator.

Usergroups

NFM allows the customer to create and maintain separate partitions for most configuration items (plans, nodes, filesets, etc.) referred to as usergroups. A

system administrator may create any number of usergroups. These may represent different departments, users, or any other division the customer cares to make. Each user is assigned access to one or more usergroups, with one of these groups being assigned as their primary usergroup. A user may be designated as a sub-administrator to a usergroup, which implies that they have administrative type authority over other primary users in that usergroup.

When a user logs on to the NFM system, they are automatically working in their primary usergroup. Any items that they list, or create will automatically belong to that usergroup. If the user has access to other usergroups, they may switch to one of these at any time, using a drop down menu that is always present. NFM allows permissions to be specified on a per user / usergroup basis. Additionally, a utility exist to copy/move/delete items among different usergroups.

Nodes & Models

A node record defines a network endpoint that can be specified for NFM activity. Some parameters, such as the communications name (IP address, phone #, etc.), are unique for each node and must be specified in the node record. Other parameters, such as the `Operating System`, or `Encryption` setting, may be specified in the node record or the node's designated model record. The model record allows a common set of definitions, or template, to be created to define characteristics likely to be shared by several nodes. If a particular model, for example, has encryption set to type SSL, then all of the nodes using that model would use SSL by default. Any of these nodes may, however, have the encryption field set differently to override the default specified in the model. Any number of model records may be created to define different sets of default values for various sets of nodes. The node record combined with its designated model record provides the NFM system with all the necessary information for communicating with the node.

There is also a mechanism by which user specified information can be stored on a per node basis with environment variables. Like most of the other fields, environment variables may be specified in the model or the node record (or both), to define a default custom value for a set of nodes, or specific custom values per node. The use of environment variables is discussed in Chapter 6.

Node Groups

A node group record is a simple list of nodes used to enable actions to be taken on a group of nodes simultaneously. Node groups are also used to specify read and write access for given users. A group may contain any combination of nodes and may contain other groups. Nodes may belong to any number of groups.

Filesets

A fileset record in its simplest form is a list of files. A fileset must be defined for any files that are to be manipulated by the NFM system. These files can be data files or directory entries. A source and target name may be specified in the event the files that are being copied need to be renamed or need to be copied into a different directory on the target node. This list can be created by manually typing in the entries or by using the browse facility that allows you to copy the names of the files directly from a directory display of files on any NFM client node.

A fileset may also include wildcards designators to more conveniently specify a larger list of files or files whose names may change on a regular basis. The name may also include an environment variable to specify different file names

based on the node or nodes associated with the files (discussed more in Chapter 6).

Plans

With a general understanding of the base components of NFM, we can now look at plans where everything comes together. A plan contains the instructions for performing various activities amongst various nodes defined to the NFM system. Users can create plans to control any combination of the following functions:

- Copying files or directories between nodes.
- Deleting files or directories on nodes.
- Renaming files on nodes.
- Executing system programs or user applications on nodes (executables can be scripts, bat files, etc.).
- Plans call other plans.
- Delay for a specified interval, or until a specific time.

Functions in a plan are grouped together into phases. Any number of phases may be created and each containing any number of functions. Functions within a phase are always run sequentially, phases however, may be specified to run sequentially or simultaneously or some combination of the two. Various control mechanisms allow for phases to be conditionally run based on the success or failure of other phases. Other controlling parameters affect how and if error recovery is performed when errors occur.

Most of the conditional logic is controlled through the use of a return code. Return codes in this context; indicate the severity level of an error that can occur. The higher the return code, the more severe the errors with zero being considered successful (no error). A return code is associated with each function in a plan as well as each phase. The return code of a phase is set to the highest (or most severe) return code of any function within that phase. Likewise, a return code is associated with an entire run of a plan, which is set to the highest phases' return code within the plan. This 'filtering up' of the highest return codes allows for an operator to know with a glance whether or not errors have occurred.

It is important to understand the distinction between adding and submitting a plan. A plan that is added or updated, simply creates/changes the plan record as described in "Configuration." Submitting a plan will actually queue up the plan to run (as described under "Operations").

Scheduling/Monitoring

Once a plan is created, it can now be scheduled to actually run. The process of scheduling a plan involves first checking the plan for any configuration errors and then creating what is called an '`instance`' of the plan which gets a unique ID assigned to it (this allows for multiple instances of the same plan). The plan is scheduled to run at a particular time and date.

A plan may be scheduled to automatically reoccur at predefined intervals. Through the use of a customized calendar and hour table, even more control over scheduling is possible. For example, a plan could be submitted to run once an hour between 9:00 a.m. and 5:00 p.m. but only Monday through Friday and not on Holidays.

Once scheduled, a monitoring screen allows a user to watch the status of a particular plan instance. The user will see either a countdown, a plan in progress, or a plan finished status along with any errors that occurred. There are also controls that can be accessed from the monitoring screens that allow an operator to hold, release, or exclude individual nodes from plan activity.

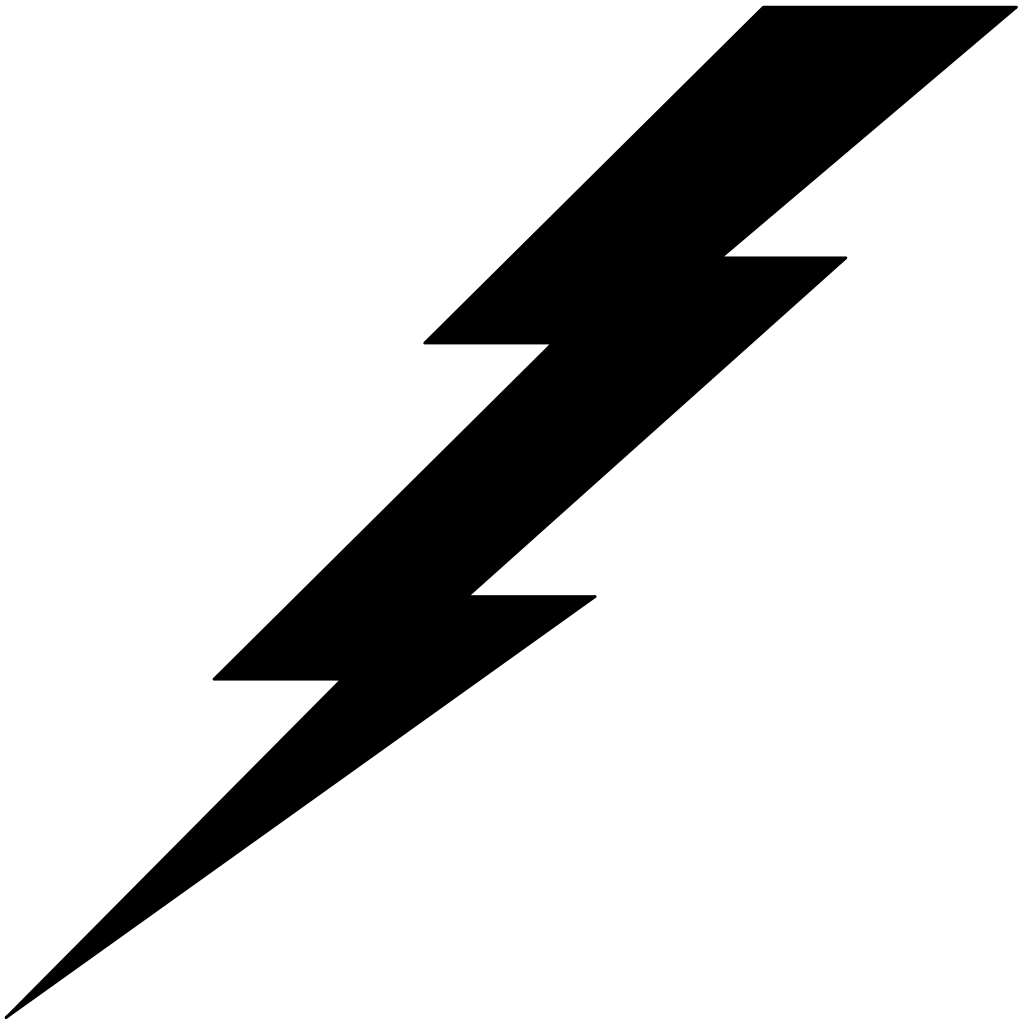
It should also be noted that the plan monitoring can be done by anyone logged in through the NFM user interface, not just by the operator who scheduled the plan.

Auditing

NFM provides detailed logging capabilities in the form of `Audit trail` records. Audit records are generated during a plans execution and may include errors or just informational records. These records include the plan name, instance ID, operation, status, time, and date. The amount of information recorded into audits during a plans execution may be optionally set to different levels. This can range from recording only errors to a detailed list of each operation that takes place.

While primarily used to record plan activity, audit records also reflect maintenance activity and system access (who deleted these nodes?, when did this person log on?, etc.).

Custom audit messages can be created using the NFM command line interface (`PUTAUDIT` subcommand), documented in Chapter 7, "Tools." This can be called from within a plan, or indirectly from a script.



4: Configuration

Overview

The following chapter discusses configuration using the NFM user interface. The first section describes general characteristics of the user interface. The next section describes the configuration items in detail and is logically ordered to provide a guideline for setting up the system. The final section contains a summary of a first time setup.

Configuring the User Interface

If the NFM stand-alone user interface is being implemented, no additional configuration is required. Proceed directly to the next section "Accessing the User Interface".

Configuration of the NFM browser interface requires the following steps:

- Installing an HTTP server.
- Copying the necessary NFM files to the proper location for HTTP server access.

Install HTTP server There are several excellent HTTP (web) servers freely available and any of them should be suitable for hosting the NFM Java user interface. Apache has been widely tested for all NFM server platforms and is recommended.

It is possible to host the NFM user interface on a computer other than the NFM server computer. It requires a more complex setup and is less efficient because it must route all user interface communications between three computers instead of just two. For this reason, we recommend enabling the NFM server computer as an HTTP server when implementing the browser interface.

Follow the HTTP server documentation for installing and testing connectivity from a browser. You should be able to visit the HTTP server's home page prior to setting up or accessing any NFM screens.

Copying GUI files Once the HTTP server is successfully operating on the NFM server, you can copy the appropriate NFM application files in place to allow access to the NFM user interface. These files are in the "gui" subdirectory off of the primary install directory of the NFM server product. There are currently two separate user interfaces available for NFM.

- The current (newest) user interface is packaged in the file "nfmgui.jar" and is referenced by the HTML file "index.html".
- The older, or classic, user interface is packaged in "nfm.jar" and is referenced by the HTML file "nfm.html".

We recommend creating an "NFM" directory off of the HTTP server's default home folder. This is typically "htdocs", the exact location will vary based on the HTTP server so please consult their documentation. Copy the desired files from the "gui" directory to the "NFM" directory.

Note: On Unix systems you may opt to provide a symbolic link from the htdocs directory to the gui directory. An example on AIX using Apache might look like this:

/usr/local/apache/htdocs/NFM → /usr/lpp/tps/nfm/gui

This would be functionally equivalent to copying the contents of directory "gui" to an NFM subdirectory in "htdocs". The advantage of this approach is that if the NFM server is updated, any changes to the user interface are automatically available without having to re-copy the files.

Once the files are in place, choosing which user interface to use is simply a matter of accessing the correct URL from the browser.

New user interface:	<IP address or hostname>/NFM (this defaults the index.html file)
Classis user interface:	<IP address or hostname>/NFM/nfm.html

Accessing the User Interface

The initial logging on to the system will vary slightly between using the browser verses the stand-alone user interface.

Stand-alone option On Windows, the NFM stand-alone interface is available from the following menu:
`START → All Programs → TPS Systems → Network File Manager → NFM Login`

An initial screen will prompt the user for the following information:

`User Name` This is the name of an existing user record that identifies an individual to the NFM system. This name will identify certain attributes and permissions for this user during this session with NFM. See "Users" later in this chapter for more information.

NOTE: NFM is shipped with a single user ID, `root`, on file. No password is required with this ID. Once the Systems Administrator has entered IDs for the users who will actually be on the system, the `root` ID should be deleted.

`Password` The password assigned to this user.

`NFM Server` This field represents the TCP/IP host name or address of the computer running the NFM server software.

Port The default of 8100 should be used unless it was necessary to re-assign this. (See Chapter 2, "Installation.")

After all fields have been entered, click on Login or simply press enter. You should now get a title screen with a pull down menu at the top.

Note: The NFM stand-alone user interface as described, currently ships only for Windows. However the NFM java application file (nfmgui.jar or nfm.jar) may be placed directly on most computers that support Java and loaded directly without installing the NFM stand-alone package. This is true for Windows systems as well.

Browser option The browser should go to a predefined URL that represents the desired HTML page that exists on the NFM server machine (reference the previous section). The log in procedure is then identical to the above procedure for the stand-alone, except that the user will not see the NFM Server or Port fields. These are not required because NFM server is accessed on the same computer as the web server and the port number can be configured from the HTML page itself.

User Interface General characteristics

The user interface has the general look and feel of a browser session. After initially logging on, a pull-down menu at the top of the screen gives you full access to the entire system. The pull-down menu is always present along with the current screen title and current time of day. Most of the configuration screens exist under the Management pull-down menu (nodes, models, etc.). Most configuration items have two screens, summary and detail. The summary screen provides a list of all items of that type. The detail screen allows viewing or configuring any one item.

While working in the detail screen, the first field always represents the item's name (or database primary key) and can be either typed in or more commonly selected from a pop-up list. When adding an entry for the first time the name must be typed in. Items can be copied by displaying an existing item, typing in a new name, and clicking the apply button.

If you are on a particular screen and you have made changes to an item, the system will prompt you to save changes if you are about to leave. However if you do go to another screen and then return, the previous data should still be on that screen.

Most screens do not automatically refresh themselves. If you display an existing node record, and another user on the system makes a change to it, your display will not reflect the change until you have reloaded that item. Other screens like the Plan Activity and Plan Monitor always refresh.

If you are working with a list of items, one of the following types of behavior is generally true:

1. If the list is a selection box, where one item should be selected from the list, a single click on the item should bring it up. This would be true for most fields on configurable item detail screens.

2. If the list is a summary screen, it is necessary to double click an item to view that item in the detail screen.

3. On certain list items, multiple entries may be highlighted prior to performing an action on them. In most cases the following rules apply:

   - A single click on an item will unselect any previously selected items and select that item (highlight it).

   - A single click on an item followed by a shift-click on another item will inclusively select all the items between the two.

   - A control-click on an item will select that item without deselecting any previous ones.

   This behavior can be seen in the following places:

   - The `Members` and `NON-Members` lists on the `Node groups` screen.

   - The `Source files` and `Rename files` list on the `Filesets` screen.

   - The browse window on the `Filesets` screen.

Users

Users may be accessed from the `Management` drop down menu. The user detail screen allows viewing or changing of individual user records. A new user may be typed in, or an existing user selected from the `UserName` drop down list. Once selected, the various parameters associated with a given user are available with most fields being logically divided among five different folders.

NOTE: Only a user designated as an administrator or a sub-administrator may look at a user summary screen and may modify other user's permissions. Non-administrators will have access to only the first two folders of their own user record that will allow them to modify their password and their preferences.

`Inherits Settings from User` This field allows an administrator to tie all of a user's settings to another user. This allows for the creation of a "template" user whose settings are to be shared by any number of other users. The only field that is not inherited is the users password that is always unique for each user. Note: This is different from simply making a copy of a user to get their settings initially, which is done by loading a user, changing the name, and clicking the apply button.

Preferences

The Preferences folder maintains the following information:

Password This field along with the Retype Password is used in conjunction with the Change Password button to change a users password. A user may change his own password, or an administrator may change any persons.

Initial NFM Screen This drop down list allows you to change the NFM screen that is first displayed when logging on. The default is the Welcome screen.

Refresh seconds This value defines the default screen refresh rate, in seconds, used while viewing auto-updating screens. These include screens such as the Plan Activity or Plan Monitor screens.

Initial Size This is the initial size of the NFM window. The Width and Height may be set to numbers representing either pixels or percentage of screen as determined by the choice box directly below.

Initial Position The default initial position of the NFM window.

Audit Defaults These selection boxes configure the default parameters used when viewing audit records in the History drop down menu. (See "History" in "Operations" chapter for more information.)

Font Selection These fields determine the type of font generally displayed throughout the NFM system.

The **Change Password** button can be used only while viewing this folder and changes the password independently of the other fields in this folder. The buttons beneath this folder apply to information in all four folders and is discussed following the description of the other folders.

Permissions

The Permissions folder controls a user's access to the various usergroups in the system. Any number of usergroups may be added to the system using the Usergroup Detail screen described in the next section of the manual. If no usergroups have been added, at least one, [Public], will exist by default. This screen allows for the adding or removing of usergroup entries from a user's access list, which is displayed in the box in the left hand portion of the screen. This screen also allows for individual permissions to be set, controlling a user's activity with regards to any given usergroup and it's corresponding items (nodes, plans, etc.). This screen also allows for the assignment of a Primary Usergroup for each user, which is described in more detail below. The following fields are available on this folder:

Administrator: This designates the user as an administrator, giving them full access to the NFM system. This overrides any remaining check boxes that may be unchecked. This user has full access to information in any other user's record.

Logon Suspended: An administrator may change this checkbox to immediately suspend or activate a user account. A suspended user will

simply not be able to logon to NFM, but their user profile will remain intact. A user account may become suspended automatically by one of the following:

- The user exceeds the maximum (consecutive) failed logon attempts.
- The user exceeds the user inactivity suspension setting (# of days with no logons).

These settings can be found on the "AUTH" folder in the `System Settings` screen (described later in this chapter).

`Account Status:` If a user account becomes suspended, this field should contain a brief description of what caused the suspension.

`Primary Usergroup:` This field designates this user's primary usergroup. A primary usergroup serves two purposes. It is the default working location for a user when they log on to the system. Any configuration items they view or create will be in this usergroup (unless they switch to another group). Also it indicates that this users record may be viewed or modified by another user that is designated as a sub-administrator of this usergroup. Sub-administrators are described in more detail below.

To set a users primary usergroup, click on a usergroup entry from the usergroup access list in the large box to the left of the screen (the usergroup must have first been added to the access list, see below). Once highlighted, click on the Set Primary button to set this field.

`Usergroup access list:` This refers collectively to the large box on the left side of the screen, as well as the drop down field (for usergroups) and the buttons, Add, Remove & Set Primary, directly above. The usergroup access list contains the list of usergroups that this user has access too. A user may only view and modify items (plans, filesets, etc.) in a particular usergroup, if that usergroup is listed here. This is true even for administrators. Once a usergroup is in this list, the individual permissions associated with this user's access may be adjusted as described below. To add an entry to the access list, type the entry in, or select it from the drop down box, then click on the Add button. To remove an entry from the access list, highlight the entry in the access list (large box) and click on the Remove button. To change a user's primary usergroup, highlight the entry in the access list and click on the Set Primary button.

The remaining fields on the right hand side of the screen will always reflect the permissions for this user with regards to the currently highlighted usergroup in the usergroup access list (large box on left hand side of screen).

`Use Default Permissions:` This checkbox, if checked, forces the use of the default permissions assigned to this usergroup for the remaining checkboxes. These defaults are maintained separately in the usergroup detail screen described in the next section. The actual permissions used will be either the Primary or Secondary permissions defined to the usergroup. The Primary permissions will be used if this usergroup is

designated as the user's primary, otherwise the Secondary will be in effect.

Sub-Administrator: This checkbox is used to indicate that this user is a sub-administrator of this usergroup. A sub-administrator is similar to an administrator in the sense that they can access and modify other user's settings, however this ability is confined only to other users in this usergroup (that is, users who have this usergroup set as their primary one). Also, unlike an administrator a sub-administrator is limited to activities designated by the remaining checkboxes.

Sub-administrators may alter permissions for users within the same usergroup and assign access to other usergroups for them, but they may not elevate a users permissions to a level greater than their own. For this reason, a sub-administrator while working with another user's permissions may see checkboxes that are inaccessible or grayed out.

Sub-Admin Extension: This checkbox enables a sub-administrator to view and set permissions for users that are not within their usergroup, but this is limited to the permissions that apply only to this usergroup. This allows them to invite outside users to this usergroup. An outside user would be one who does not have this usergroup designated as their primary one.

Consider the following example:

- Barry is an overall administrator.

- Jim is a sub-administrator of the usergroup SALES. This means that Jim has SALES set as his primary usergroup and has Sub-administrator checked on.

- Joe is a regular user (not administrator or sub-administrator) who is also in SALES.

- Jef is a sub-administrator of the usergroup ACCOUNTING and also has the extension box checked.

Joe does not currently have access to ACCOUNTING. He could be granted access by any one of the following ways:

- Barry could add ACCOUNTING to Joe's access list (he can do anything).

- Because Jim is a sub-administrator over Joe, Jim could add ACCOUNTING to Joe's access list. He could only do this however if he (Jim) already had access to ACCOUNTING himself.

- Jef could add ACCOUNTING to Joe's access list. Because he has the extension flag on, he is allowed to 'invite' others into his usergroup.

Nodes/Models/Modems: The view check box gives authority to view these configuration items only. The modify check box allows adding, update or deleting items.

Node Groups: The view check box gives authority to view node groups. The modify check box allows adding, updating or deleting node groups.

Plans: The view check box gives authority to view plans. The modify check box allows adding, updating or deleting plans.

Filesets: The view check box gives authority to view `filesets`. The modify check box allows adding, updating or deleting `filesets`.

Audits: Audit records are generated automatically and thus a single check box, view, gives a user permission to view audit records.

Plan Activity/Schedules: The view check box gives a user the authority to monitor a submitted plans status. The modify check box gives them the ability to submit plans, change submitted plans (cancel, hold, restart, etc.), and the ability to make changes to `Schedule` tables.

Plan Instances: The delete check box allows a user to delete plan instances from the Plan Activity screen.

Multicast Profiles: The view check box gives authority to view multicast profiles. The modify check box allows adding, updating or deleting multicast profiles.

Quick Transfer: Gives authority to use the `Quick Transfer` function.

Quick Execute: Gives authority to use the `Quick Execute` function.

Quick Submit: Gives authority to use `Quick Plan` type functions from the `Quick List` screen.

Quick Monitor: Gives authority to monitor the plans that are performing the different `Quick` function types.

Quick Multi-Select: Gives authority to select multiple source or target nodes during a `Quick Transfer` or `Quick Execute`.

Performance: The view check box gives authority to view performance extracts and graphs. The modify check box allows adding, updating or deleting performance graphs.

Node Access: The four different checkboxes (read, write, execute & create dir) may be used to grant or deny the implied type of access for the user to all nodes belonging to the currently selected usergroup. If it is necessary to assign different levels of access on a per-node basis, use the `NFM Node Groups`. Folder (described below) instead of these checkboxes.

NFM Node Groups

The `NFM Node Groups` folder makes it possible to give specific users individual access to nodes during plan execution or other node activity such as browsing. This mechanism for assigning node access is more comprehensive

than the similar read/write/exec/dir checkboxes described the previous section. Whereas they grant or deny access to all nodes in the usergroup, this folder allows assigning access on per-node basis. Once again an administrator has unlimited access, but a non-administrator may only submit a plan that works with nodes that are included in the appropriate node groups referenced here. The following four types of access may be individually enabled or disabled:

Read: Users with read access may copy files from these nodes as well as browse their contents.

Write: Users with write access may copy files to these nodes.

Exec: Users with execute access may run commands on these nodes.

Create dir: Users with create directory access may create directories on these nodes.

The Add and Remove buttons allow you to add or remove entries from the users appropriate access list.

Since groups may contain other groups, it should only be necessary to maintain a node group for each type of access / user class combination. In this manner, multiple users of the same class can simply share the same access groups, then making a new node accessible to multiple users would simply involve adding that node to the appropriate group.

Quick folder

The Quick folder configuration section is described in detail in Chapter 9, "Quick Functions," in section "Quick folder".

All user folders

The following buttons are available from the User Detail screen:

Summary	Takes the session directly to the User Summary screen.
Apply	Applies all changes made to any of the User Detail folders. This will cause an add or update to occur, depending on whether or not the user record preexisted. This will not effect a password change from the first folder. As discussed that operation has its own button.
Cancel	Cancels all changes made in any of the folders, since the last time the user record was displayed or applied.
Delete	Deletes the currently displayed user record.

Usergroups

Usergroups may be accessed from the Management drop down menu. The usergroup detail screen allows viewing or changing of individual usergroup records. A new entry may be typed in, or an existing entry selected from the

Usergroup drop down list. Once selected, two different folders are available to set the default permissions for Primary and Secondary users.

Settings The `Settings` folder defines default characteristics for this Usergroup. The following fields may be configured:

`Home Server` This field may be set to specify an alternate (secondary) NFM server computer that is to perform tasks (file transfers, etc.) related to this usergroup. This field may be left blank to default to the current local NFM server. Reference section "NFM multiple server support" in Chapter 6, "Advanced Functions," for more information on this topic.

Primary and Secondary Permissions These folders may be used to define the default permissions for this usergroup. These are the permissions that will be in effect for users while accessing this usergroup, unless a specific set of permissions are specified on a per-user basis to override the defaults.

The default permissions defined here for each usergroup will be available in the following way. If this usergroup is added to a user's access list in the user detail screen, then by default, the secondary permissions here will be in effect for that user's access to this usergroup. If this usergroup is set as that user's primary usergroup, then by default, the primary permissions here will be in effect.

The following buttons are available from the `Usergroup detail` screen:

`Summary` Takes the session directly to the `Usergroup Summary` screen.

`Apply` Applies all changes made. This will cause an add or update to occur, depending on whether or not the usergroup record preexisted.

`Cancel` Cancels all changes made since the last time the usergroup record was displayed or applied.

`Delete` Deletes the currently displayed usergroup record.

Usergroup Utility

The Usergroup utility may be accessed from the `Management` drop down menu, Usergroup sub-menu. The `Usergroup utility` screen is used to copy, move and delete items (plans, nodes, etc.) that exist within the various usergroups. A series of checkboxes at the top of the screen are used to decide which items to view. The source usergroup is then selected and the appropriate items displayed. The user may highlight the desired items and delete them, or select a target usergroup and then copy or move them with the appropriate buttons (move simply copies the item and then deletes it from the source Usergroup).

Nodes

Nodes may be accessed from the Management drop down menu. The `Node detail` screen allows viewing or changing of individual node records. A new entry may be typed in, or an existing entry selected from the Node drop down list. Once selected, six different folders are available to maintain various parameters associated with a given model.

Node Settings

The `Node Settings` folder contains fields that are unique for each individual node and may not be defaulted in a node model. This includes the following:

`Primary Communications Name` The information to be entered here depends on the type of primary transfer method specified in this nodes corresponding model record. See the individual descriptions for each transfer type in the previous section on `Models` to determine what name to use.

`Backup Communications Name` This field is only needed if a backup file transfer method was specified in the corresponding model record. The information to be entered here depends on this file transfer type. See the individual descriptions for each transfer type in the previous section on `Models` to determine what name to use.

`Model` This selection box identifies which model template is to be associated with this node. The model will provide default values for many other fields in the node record that may be left blank. Each node may have its own model defined, but more than likely, several nodes will share the same model. The contents of the model are shown where applicable in other folders on this screen, but cannot be altered here (changes must be made in the `Model detail` screen).

Settings

The `Settings` folder defines general characteristics for the current node. Each of the fields defined here may be left empty (or have the field "default" checked) to allow the corresponding setting in the node's model to be used instead. Any values specified will override the corresponding model's value. The following fields may be configured:

`Primary File Transfer Method` Select from the pop-up list the method used to transfer files to and from this node. The following choices are available:

`SOCKETS` This is the standard choice for an NFM client node that is accessible on a TCP/IP LAN or virtual LAN. A node of this type is assumed to always be connected to the network. The NFM client software must be running on the node before any communications can be established. If this method is chosen, the `communications name` field in the corresponding node record(s) should be set to the IP host name or IP address.

NFM will use port number `8008` by default for communications with the node. If another port number is desired, this number can be appended to the end of the communications name following a colon. Example: `204.62.235.16:8015` to use port `8015` instead. If this is specified, the client on that node must also be started up with a corresponding command line option, `-s8015` with the matching number.

Note: This manner of specifying an alternate port is not the preferred method, but is being supported for backward compatibility. The preferred method is to customize the connections setting in the `Connections` folder, and the configuration file on the client computer as described later in this section.

`PPP` This method is also used for communicating to an NFM client node but in this case the node is not on a network. A dial-up connection must be made over a modem pair before communication takes place. The NFM server handles the dialing and disconnecting automatically to ensure proper sharing of single modem or modem pool when several connections are requested at the same time. If this method is chosen the `communications name` field in the corresponding node record(s) should be set to a phone number. (See "Appendix D" for more information on setting up a `PPP` node.)

`FTP (all types)` These node types represent FTP servers, with or without secure transfer capability. Any transfers to and from these nodes will be in conjunction with a standard NFM client. See "Appendix B: FTP nodes" for a detailed description of the different FTP node types.

`HTTP (all types)` Each of the HTTP node types involves the use of a standard HTTP server. There are multiple ways to configure these depending upon the requirements. See "Appendix D: HTTP nodes" for a detailed description of the different HTTP node types.

`DEMO` This type of node is used for demonstration purposes only. No actual communications is established.

`AWS CLI` The node is enabled when using the Amazon Web Services Command Line Interface (AWS CLI) to transfer files to/from "The Cloud" using the S3 protocol. The AWS CLI must be installed on a system that also has the NFM Client installed. See Appendix F for more information on using AWS CLI.

`Backup File Transfer Method` This optional field can be used to specify a backup form of communications to use in the event the primary method fails. The same choices as `primary` (above) apply. A good example would be to have a dial-up (or PPP) connection available in case the regular IP network becomes unavailable. Use of this option requires that the corresponding node record(s) designate a backup communications name.

OS Name This is the name for the operating system for this node. Again, the pop-up list is available to choose from the operating systems supported by NFM. If this model will represent an FTP site, the entry generic may be chosen.

Answer Mode This field can be checked to put this node in `answer mode`. This basically means that this node will not initiate a socket connection by default. All nodes that may be defined outside of a firewall should have this option on. In this manner, when a file transfer is done between a node inside the firewall and a node outside the firewall, the inside node will be responsible for initiating a socket connection. This satisfies the requirements of most firewall setups that do not allow incoming connection request.

Diagnostics This field turns on `diagnostic` mode for this node. This will cause trace logs to be created during activity involving this node for collecting additional information used in diagnosing problems. This option should be left off generally, as it slows down performance and can generate huge amounts of trace file data on the NFM server and clients involved.

Off Line This field can be checked to disable this node from any plan activity. This can be used to temporarily avoid transferring to a particular node that is part of a node group, without having to remove the node from the group record.

On Hold This field can be checked to cause any plan that references this node, to put the node on hold automatically when the plan is started. The node may then be released from the `Plan Activity` or `Plan Monitor` screen on an individual plan instance.

Encryption

This pop-up list designates which form of encryption (if any) will be used during communications to or from this node. The choices are `Off`, `NFM` and `SSL`. Encryption can only occur between NFM socket type nodes, and SSL encryption is only available between socket type nodes that support this feature.

If a connection is established between two nodes that have this field set differently, then the more secure setting is in effect for both. Example, if one node has encryption set to `NFM` and another has encryption off, a connection between the two will use `NFM` encryption. Likewise, if one node has encryption set to `SSL` and another has encryption off or set to `NFM`, the connection will use `SSL`. For more information on encryption, refer to Chapter 6, "Data encryption and compression."

If `SSL` is selected for a client node, it will also be necessary to make a corresponding change to the configuration file on the client computer. This is discussed in more detail in Chapter 6, "Encryption and connection options."

Timeout This optional field specifies a timeout value (in seconds) to be used in conjunction with most types of node activity during plan functions. If a node is requested to perform an operation and it does not reply within the timeout value, the connection to the node is broken and an error is generated. This is primarily used to avoid certain network conditions that may cause a socket to become non-responsive without breaking the connection.

For a file transfer, this field should be set to similar values on both the source and target nodes. In this case the timeout represents the amount of time in between data block transfers, not the total amount of time needed to transfer a file. This field only applies to socket type transfers, deletes and renames. It does not apply to execute functions and it does not apply to transfers using the `file streaming` option. A timeout for an execute type function may be specified within a plan, see description for field `timeout` in `Plan Functions` later in this chapter.

Retry Count This field specifies how many times a connection to the node will be re-attempted after an initial failure.

Retry Interval This field specifies how long to wait (in seconds) between connection attempts. It is only used if the retry count is non-zero. This field is not used for modem type connections.

Transfer Block Size This field changes the maximum block size (in bytes) used for sockets type file transfers to and from a given node. If left empty or `0`, this value defaults to `32000`. It is usually not necessary to change this from the default unless the following field "Transmit Window Count" is specified as well. While transferring between nodes that have this field set different from one another, the smaller of the two is used.

Transmit Window Count This field along with "`Transfer Block Size`" above, can be set to enable a pacing feature of NFM that can help keep NFM from flooding a network with file transfer data. Normally, NFM transfers files as fast as the network will allow. This can have the adverse effect of choking off other network activity. This is especially visible when transferring to and from a node group.

A window count forces the receiving node to send a reply after receiving this many blocks of data (of size specified above). The sending node must receive this acknowledgment before sending any additional blocks. This simple mechanism keeps NFM from completely filling an outgoing network queue and forcing other traffic to wait behind it.

The optimum numbers used will vary based on any given network, so some experimenting may be required, but the following guidelines should apply:

1. If the window count is to be set, the block size should be set less than or equal to the network maximum packet size (or MTU), typically about `1500` bytes. A lower number may be desirable if data is going across a modem.

2. The window count multiplied by the block size represents how much data is sent before waiting on data acknowledgments. The smaller this total size is, the more NFM is restrained from filling up outbound network queues, which should benefit other network activity. However, as this window count is made smaller, NFM may become slower in performing file transfers. A good approach to setting the count would be to make it as low as possible, while still getting good data transfer rates. A value of `1` may work just fine.

3. If both the sending and receiving node have the window count set to different values, the smaller of the two will be used.

Another configuration parameter that affects NFM's bandwidth usage is the "Maximum Active Node" field in the system settings.

`Home Directory` This field restricts access on a node for use with Quick Functions. See "Home directory" in Chapter 9, "Quick Functions" for a more detailed description.

`Node Password / Retype Password` An optional password may be used to restrict access to an NFM node. The password designated here must match a password that is configured on the NFM client node.

Setting a node password on the NFM client

This task is accomplished using either the `nfmi` or `nfmclient` program at the NFM client node site using the following syntax:

```
nfmi -p
```

[or]

```
nfmi -P password
```

The first method will prompt you to enter and then retype the password. After setting or changing the node password, the NFM client must be stopped and restarted before the change will take effect.

`Account User Name / Account Password` The account user name and password fields are used when the node type requires a login to be supplied when performing activity with the node. This applies only to an FTP type node at this time.

`Nfmi User Name / Nfmi Password` This is the login to use for any remote commands issued from this node using the NFM remote server interface. . (See "NFM remote server interface" in Chapter 8, "Tools" for more information).

`Automatic Plan` Setting this field to a valid plan name will cause this plan to get invoked automatically when the NFM client for the

associated node first starts up. The client must also be configured to perform a check-in request upon startup for this to occur. Refer to the section "NFM Client Configuration" later in this chapter for information on how to enable this option.

`Automatic Command` This field may be set to a console type command (Unix or Windows) to perform on the local NFM server computer automatically when the NFM client for the associated node first starts up. The client must also be configured to perform a check-in request upon startup for this to occur. Refer to the section "NFM Client Configuration" later in this chapter for information on how to enable this option.

Attributes

The `Attributes` folder contains additional node settings that are specific to the transfer method that this node utilizes. The fields available on this tab may change when the current transfer method is changed (in the "Settings" tab, or inherited from a model's settings). Currently only the "FTP Server" and "FTPS Server" method types have formal fields available for this folder (see "Appendix B: FTP nodes" for a description of these settings). This screen also allows for the addition of generic definitions if the need arises.

Notes

The `Notes` folder provides a text window for storing text information specific to this node record.

Environment

The `Environment` folder allows for the assignment of customized information to this node. Environment variable pairs in the form `NAME=Value` may be typed in or modified here. Click on an existing entry, or the first blank line, to highlight it. Once highlighted, you may perform the following tasks:

- Delete the entry with the delete key.
- Hit enter to insert a blank entry after the current one.
- Click on the `Name` or `Value` portion of a highlighted entry to edit it.

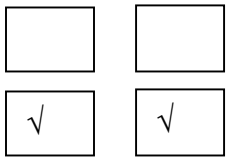
Use of environment variables is explained further in Chapter 6, "Using environment variables."

Paths

The Paths folder maintains a list of path permissions associated with this node. Path permissions allow NFM to restrict or permit file transfer access to the specified directories for any file activity associated with this node. Each entry contains a directory string along with a read and write checkbox. If a directory is enabled or disabled it will affect any files within the directory tree. These entries can overlap to allow setting policy on a directory in one entry, and then changing policy for one of its subdirectories in a subsequent entry (see below). An empty string implies the root directory, and thus, dictates the default access if a filename does not match any subsequent entries.

Example, consider the following paths:

Read	Write	Directory
√	√	



The first entry with the empty directory string implies full access to the file system, which means any files not matching any subsequent entries will have access. The second entry turns off read and write access to `/var`. The final entry turns access on for `/var/nfm/tmp` that would have otherwise been prohibited by the previous entry.

The case sensitivity of the matching of paths to filenames depends upon the operating system involved. The path `/tmp` does not have to be capitalized to match `/TMP` on a Windows or 4690 computer.

Names may begin with drive letters if desired. On a machine with drive letters, a path beginning with / implies that directory on any drive. Also, backslashes (\) may be used in the name, but it is recommended that forward slashes always be used in any reference to a file name within NFM. Therefore, `/tmp` would match `C:\TMP` and `D:\TMP`.

Connections

The `Connections` folder allows the user to customize the manner in which connections are made to and from a sockets type node.

It is not usually necessary to add connection records. NFM uses default connections for sockets type nodes as follows:

- A non-SSL type node listens for a socket connection on port 8008.
- An SSL type node listens for an SSL encrypted socket connection on port 8009.

When custom connection records are specified, the defaults listed above are not active. Any of the following requirements would make it necessary to provide custom connections to function properly:

1. A port other than the default of 8008 (for non-SSL) or 8009 (for SSL) is desired for a listening type client.
2. The NFM client is using a call-in type connection. These are used when the NFM client does not have a directly accessible IP address to connect to, and is configured to call in instead.

To add a connection record, simply fill in the desired fields and click the add button. To update an existing entry, click on the entry, click the remove button and then re-add it. The following fields may be configured:

`Type` Select the desired connection type from the pop-up list The following choices are available:

Listen The node will listen for a socket connection on the specified port.

Call The node will make a connection in to the NFM proxy program, making itself available for connections from the NFM server or from other NFM clients. Connections of this type are usually made when the NFM client program is started.

Port Specifies the port number that the NFM client is listening on (for a "Listen" type connection).

Interval & Duration These fields are not yet applicable.

SSL Indicates an SSL type connection.

GATEWAY This field is used for call-in " type nodes only, and is used to designate the IP address or hostname for another computer that services incoming connections from such nodes. If the NFM server is to receive these connections directly, then this field may be left blank (this is a typical setup). Use of a gateway computer to handle incoming connections is usually done when the customer does not want the NFM server computer to receive incoming socket connections directly for security reasons. If this field is specified, the corresponding NFM client configuration field "GATEWAY" should also be set to the same location. (See "NFM Client Configuration" later in this chapter.)

Chapter 6, "Encryption and connection options" contains a complete description and examples of the different types of node setups with regards to these connection records and the NFM client configuration file.

All model folders The following buttons are available on the `Model Detail` screen:

Summary	Take the session directly to the `Model Summary` screen.
Apply	Applies all changes made on this screen. This will cause an add or update to occur, depending on whether or not the model record preexisted.
Cancel	Cancels all changes made on this screen since the last time this model record was displayed or applied.
Delete	Deletes the currently displayed model record.

Models

Models may be accessed from the `Management` drop down menu. The `Model detail` screen allows viewing or changing of individual model records. A new entry may be typed in, or an existing entry selected from the Model drop down list. Once selected, five different folders are available to maintain various parameters associated with a given model.

A model serves as a node template from which any number of nodes may be configured to inherit their settings from. This provides a convenient way to manage the settings on a large number of similarly configured nodes. Any node that references a model may override any model settings desired on a per-field basis.

Settings The `Settings` folder describes general node characteristics for this model. The description for each field is exactly the same as the ones used to define the corresponding fields in the `Node Detail Settings` folder. Please refer to that section earlier in this chapter for a description of each field.

Attributes The `Attributes` folder describes node characteristics that are specific to the method type used for this model. The description for each field are identical to the ones used to define the corresponding fields in the `Node Detail Attributes` folder. Please refer to that section earlier in this chapter for a description of each field.

Notes The `Notes` folder provides a text window for storing text information for this model.

Environment The `Environment` folder allows for the assignment of customized information to this model. Environment variable pairs in the form `NAME=Value` may be typed in or modified here. Click on an existing entry, or the first blank line, to highlight it. Once highlighted, you may perform the following tasks:

- Delete the entry with the delete key.
- Hit enter to insert a blank entry after the current one.
- Click on the `Name` or `Value` portion of a highlighted entry to edit it.

During plan activity, any node using this model will have all of the environment variables specified here as well as any node specific environment variables configured in the node detail screen, available in a combined list. Any node environment variables that use the same name as a model environment variable will override the model value in the final combined list. Use of environment variables is explained further in Chapter 6, "Using environment variables."

Paths The `Paths` folder allows for the defining of default path permissions for this model. See a description of `Paths` in the previous Nodes section of this chapter.

Connections The `Connections` folder allows for the defining of default custom connection records for this model. See a description of `Connections` in the previous Models section of this chapter.

Buttons The following buttons are available from the `Node Detail` screen:

Summary	Takes the session directly to the `Node Summary` screen.

Apply	Applies all changes made on this screen. This will cause an add or update to occur, depending on whether or not the node record preexisted.
Cancel	Cancels all changes on this screen since the last time this node record was displayed or applied.
Delete	Deletes the currently displayed node record.

Node Access Point (NAP)

A Node Access Point (or NAP) creates a reference point to a node that may be used in place of referring to a node directly in most places where a node name is required. This serves several purposes:

1. Creates node visibility outside of its usergroup. By creating a NAP in one usergroup that references a node in another, users may now be given access to a node without having to add that node's usergroup to the user's access list.

2. Restrict access on a node to a particular directory tree (or set of qualifiers for OS/390).

3. Provide different levels of access to the same node by different users or user groups. This is possible because any number of NAPs may point to the same node. Additionally, a NAP may be defined for common access among members of a usergroup, or a NAP may be created that individually sets the access for a particular user.

4. Allow for an account user name and password to be configured. This will override the user name and password supplied for an FTP type node, allowing for multiple accounts to be accessed without having to create a different node for each one. This account name and password may also be used for user exit authentication purposes.

Node Access Point configuration is available from `Node` menu (under `Management`). The following information can be configured:

`Node Access Point Name` This is the user selected name for the given NAP record. Type in a new entry, or select an existing entry (along with user name if desired). This name combines with the next field `User Name` (which may be blank) to create a unique entry. The name chosen may be the same as the node name to which this NAP refers. Although doing this might seem confusing, it allows a NAP to be transparently introduced to an existing NFM configuration without having to change

all of the plans that reference this node to use the NAP instead. This is described in more detail below.

User Name If this field is set to an existing user name, then this NAP will only be used when the NAP name (above) is chosen or referenced by this particular user. (For a node reference within a plan, this would be the user who submitted the plan). If this field is left blank, then the NAP is considered generic and will be used by default if a user specific NAP by the same name is not found (described in more detail below).

Account User Name / Account Password The account user name and password fields are optional and provide one of two things:

1. The name and password fields may be used to satisfy a login request for an FTP type node. If they are specified here, they override any account name and password fields that may otherwise be specified in the node or model record. This allows for different accounts at the same site to be used with a single node record.

2. The name and password fields may be supplied to a user exit routine (if one exist) for the NFM client type node being accessed.

Impersonate Account User This checkbox forces the NFM client to inherit the account user's identity while performing tasks invoked with this NAP. This allows local security to be enforced. Note, this feature is only available on certain NFM clients, which currently includes only the Windows platform. It requires version 2.3 or higher of the NFM client software to be installed.

Home Directory A home directory may be specified here. A home directory provides either an enforced starting directory for access or a default location, depending upon the setting of the next field (Allow access outside of Home Directory) described below. Note, the use of a home directory here does not circumvent the restrictions of the `Paths` folder settings in the corresponding node or model.

Allow access outside Home Directory If a home directory is specified, this check box indicates whether or not access is allowed outside of it.

If this box is unchecked, any reference to a file name on this node is automatically appended to the home directory. The only exception to this, a file name that exactly matches the prefix (up to the length of the prefix) is not appended, but referred to directly.

If this box is checked, a file name that is not fully qualified (does not start with forward slash or drive letter) is automatically appended to the home directory. A fully qualified name is referred to directly (home directory is ignored). This only applies to non-OS/390 nodes (an OS/390 data set name is always appended to the home directory).

Notes The `Notes` folder provides a text window for storing text information specific to this NAP record.

Environment

The `Environment` folder allows for the assignment of customized information to this NAP.

If an environment variable name is specified here that is the same as one in the node or model record, this one overrides it. Use of environment variables is explained further in Chapter 6, "Using environment variables."

Buttons

The following buttons are available from the `Node Detail` screen:

Summary	Takes the session directly to the `NAP Summary` screen.
Apply	Applies all changes made on this screen. This will cause an add or update to occur, depending on whether or not the NAP record preexisted.
New	Clears all fields for the creation of a new NAP.
Cancel	Cancels all changes on this screen since the last time this NAP record was displayed or applied.
Delete	Deletes the currently displayed NAP record.

More on NAP usage

A NAP name is generally interchangeable with a node name. It may, for example, be the target of a plan transfer function, or a member of a node group, etc. Besides making it easier to introduce its usage into an existing environment, this allows for a simple customization of access based on the user or user groups involved.

When a node name is referenced, say during a transfer, the following lookup is actually done until a match is found:

- The system searches for a NAP record whose name matches the node name requested and whose user name matches the user performing the operation.
- If not found, the system searches for a NAP record whose name matches the node name requested and whose user name is blank.
- If neither of these is found, a simple search for an actual node record with a matching name is done.

This would allow an administrator for example, to restrict one user to accessing only a particular directory on a node by introducing a NAP record with the node and user's name. This would have no affect on other users accessing the same node.

In a similar manner a NAP added to a particular usergroup that referenced a node in a different usergroup, would have no affect on any users outside the usergroup who were accessing the same node.

Node Groups

Node groups may be accessed from the Management drop down menu. The following information can be configured:

Node Group Name This is a user selected name which defines a group of nodes. Type in a new entry, or select an existing entry from the drop down list.

Administrative Group This field assigns the group administrative status. This should be turned on for groups that are used to configure individual users read write access to specific nodes in the User Detail screen. Only an administrator may change an administrative group definition. This does not prohibit the use of this group with plan activity; it merely keeps a non-administrator from changing their own node access permissions.

Maximum Active Nodes A non-zero value limits the number of nodes in this group that may be active, with regards to plan activity at any given time. This field is similar to the overall "Maximum Active Nodes" field in the System settings (described later in this chapter); however, here it can be used to regulate bandwidth usage on a per group basis.

Apply Maximum Globally If this field is checked, the "Maximum Active Nodes" setting (above) is in effect for all the nodes in this group, regardless of what other groups might have been used to initiate plan activity among these nodes. If this option is off, the limit will only take effect if this is the actual group used to specify activity.

Members Lists all the nodes and node groups that are currently a member of the selected group. Groups are displayed with a blue star next to their name.

Non-members Lists all the nodes and node groups that are not currently a member of the selected group.

Clicking on the individual entries can highlight nodes. Then by clicking on the Add or Remove buttons, nodes can be added to or removed from the current group. Nodes may belong to any number of groups, and groups may contain other groups. Uses click and shift-click to highlight a block of nodes. Uses click and control-click to selectively highlight multiple nodes. Changes will not saved until the apply button is pressed.

Buttons

The following buttons are available from the Node groups screen:

Summary	Takes the session directly to the Node Group Summary screen.
Apply	Applies all changes made on this screen. This will cause an add or update to

occur, depending on whether or not the node group pre-existed.

Cancel Cancels all changes on this screen since the last time this node group record was displayed or applied.

New Clears all entries from the `members` list to allow for the creation of a new group.

Notes Displays a pop-up text window for storing notes in.

Delete Deletes the currently displayed node group record.

Filesets

`Filesets` may be accessed from the `Management` drop down menu.

The following fields are maintained on this screen:

Fileset This is a user selected name which defines the specific file entries and their characteristics. Type in a new entry, or select an existing entry from the drop down list.

Use Node Extensions This field is used to enable automatic name extension logic for file transfers. This mechanism is used for avoiding name collisions when transferring same named files to and from a node group. (See "More about `Node group activity`" in Chapter 6, "Advanced Functions.")

Compress Transfer O **on** O **off** O **default**
These check boxes control whether or not compression is used during any transfer of these files. If the default box is checked, then compression will be attempted if the `Compress Transfers` option is checked on in `System settings`. Compression can only occur between NFM enabled client nodes. If compression is attempted for a non-client node, it will not cause a problem, it simply will not be used.

Store External Attributes O **on** O **off** O **default**
These check boxes control whether or not external attribute support is enabled during any transfer of these files. If the default box is checked, the support is enabled based on the checkbox for `External Attributes` in the `System settings`.

This feature, when active, enables NFM to retain attributes of a file that are operating system dependent even if that file is transferred to a different system type. If the file is transferred back to a system representing its original type, these attributes will still be intact. These attributes currently include the following items:

Unix:	Owner name, Group name, and permissions.
Windows:	Attribute byte.
4690	File Distribution Attribute (FDA).

Source Prefix Each file that makes up this `fileset` record must have a full path specified to where it resides. This optional field `Source Prefix` allows any part of the path that is common to all the files to be specified here. If this field is not used than the `Source Files` list must contain a full path to the file (starting with a `/` for UNIX, and a `drive_letter:/` for Windows, see more about drive letters below). Relative sub-directories may still be included in the `Source Files` list. When files are accessed, the `Source Prefix` is combined individually with each filename in the list to form the full path to the file. If the source prefix does not end with a `/`, one is supplied. Using the source prefix accomplishes the following two important tasks:

1. If the files names stay the same during a transfer but the files must be moved from one directory to another, the source and target prefix fields can designate these directories. In this case it will not be necessary to rename each individual file.

2. If changes need to be made to the path, only this field would have to be modified rather than changing each file entry.

Source Files These fields specify the names for individual files relative to the source prefix directory. If the `Source Prefix` is not specified this must be a `full path` to the file (as stated above). If the source prefix is specified, individual filenames may still specify a `full path` to a file that is different from the `Source Prefix`. In this case the name is not combined with the prefix to form the full filename.

Example: The following `fileset` contains these entries:

```
Source Prefix:
  C:/tmp/
Source files:
  file1
  file2
  file3
  C:/docs/file4
```

This implies the following `full source file` specifications:

```
  C:/tmp/file1
  C:/tmp/file2
  C:/tmp/file3
  C:/docs/file4
```

During file transfers directories that make up a filename are not automatically created. To create directories, they must be individually listed in the fileset definition. Files are copied in order, so subdirectories should be listed before any files that are in them as follows:

```
/var/sub1/
/var/sub1/file1
/var/sub1/file2
```

In this example, `var` exists but `sub1` does not. A transfer will create `sub1` before copying files `file1` and `file2`. During a delete operation of this fileset, the files are deleted in reverse order, so `file1` and `file2` are removed making it possible to remove directory `sub1` afterwards. If file entries are being typed in, make sure to include a trailing / to denote a directory.

The source files may also contain wildcard specifications, similar to those used by Windows where, an `asterisks *` substitutes for any part of a filename and a `question mark (?)` substitutes for any given character. Please refer to the section "Using Wildcards" in Chapter 6, "Advanced functions" for a further description of this option.

More about Windows drive letters

The use of a drive letter when specifying a full path is not mandatory but it is recommended. A file name that begins simply with a forward slash will be interpreted to mean the current drive, which is not necessarily known to the user.

A mapped drive letter that references either local or network resources should never be used. This is because they are not global to the system, and are therefore not necessarily available to a services program (in this case the NFM client). The Universal Naming Convention (UNC) may be used instead to reference such a resource.

For more information on this please refer to the following document:

http://msdn2.microsoft.com/en-us/library/ms685143.aspx

Target folder

The `Target folder` on the right side of the screen is almost identical in nature to the `Source fields` discussed above. The `Target prefix` combines with each of the individual `Target File` entries to create a full file specification that becomes the destination name of file on a target node, during a file transfer function. These fields are optional; not using them causes the following defaults conditions to occur:

- If no `target prefix` exist, files are copied to the same path as on the source node (or source prefix).
- If no individual `target file entry` exist, a file keeps the same name specified in its `Source file` entry.

NOTE: The Target folder has no meaning for file transfers to and from an OS/390 type node. In this case the Dataset or PDS name for the file is specified in the OS/390 folder along with other options described further in this chapter.

There are two methods for building the fileset lists. The entries may be entered manually, or they may be added using the browse button.

Editing file entries

To edit file entries, click on a blank or an existing entry underneath Source Files. If the entry is highlighted a solid color, click on it again. The entry should now have a white box with a cursor in it. This is edit mode for the entry, you can now type and backspaces to either enter or modify an entry. Typing enter in this mode will insert a new blank entry after the current entry. If the browse function is used to build this list, the entries can of course still be edited afterward. The Target Files fields are edited in a similar manner. The copy button will allow you to copy the name of any highlighted Source File entries directly into the Target File fields.

Browsing file entries

It is probably more convenient, and certainly less error prone to build the fileset entries by browsing. The browsing function is only useful when the following is true:

- Exact file names will make up the fileset list (no wildcards).
- These file names exist on a node that can be browsed during the fileset list creation. This includes any currently enabled NFM client nodes and any FTP type nodes.

Click on the browse button, to display the browse window. The browse window has the following fields and characteristics:

Client This drop down box allows you to select which node will be used for browsing. After selecting an entry, a connection is made to the node and a directory should be displayed

Current Directory The field directly beneath the client field will always show the current working directory. An attempt is made to initially display the directory currently in the Source Prefix field. If this field is blank the display should be of the root directory.

Directory Entries The directory entries are a simple list of the files in the current working directory. If the Expand tree checkbox is checked the list will be expanded to recursively include all sub-directories of the current directory.

The following actions may be performed from within the browse window:

- Changing directories is accomplished by either double clicking on a sub-directory to change too it, or clicking the Up button to change to the parent directory.

- Once you have reached the desired base directory. Clicking on the Set Prefix button will copy this name into the Source Prefix field. If you browse with this fileset in the future you should be taken directly to this directory.

- To copy file entries from the directory display to the Source Files list, you should first highlight all the desired entries. Use click, shift click or control click to highlight entries (remember double-click is used to change directories). Once the desired entries are highlighted click on the Apply button to copy them.

If the current directory matches the Source Prefix field, the exact file names displayed will be copied. If the Source Prefix field is blank, or if you are browsing outside the tree represented by the Source Prefix field, the entire path and filename are copied. This is consistent with defining the exact information needed by NFM to find the file. When finished browsing, click on OK to remove the browse window and continue with the filesets screen.

Properties folder This folder contains additional information that may be configurable on a per file basis:

File Type O **Text** O **Binary** O **Directory**

This describes the type of file being accessed. The choice of Text or Binary will affect translation of the file during copies between nodes of different types. Binary implies no translation. Text will cause a translation to and from the native text type. This would be the following on the different platforms:

- ASCII with new-line termination on most UNIX computers.
- ASCII with new-line/carriage return termination on Windows and 4690 platforms.
- Fixed or variable record length EBCDIC for OS/390 computers.

This field will designate the default characteristics of using the ascii (for Text) and binary (for Binary) commands when transferring to and from FTP type nodes.

A file may be marked as a directory by choosing the "Directory" option here, or by simply ending the file name with a trailing directory separator '/'.

Recurse Subdirectories

This option enables NFM to include the contents of subdirectories when building the list of files for the fileset.

Do not Create/Delete unspecified directories

This option stops NFM from creating or deleting directories that were not specifically listed in the file set.

Symbolic Links

These checkboxes describe how NFM will handle symbolic links during file transfers, as well as certain other file activity. This only applies for operating systems that support symbolic links (currently Windows and the various Unix platforms).

O **Follow (transfer only)** This option causes NFM to simply copy to or from the file that the symbolic link is pointing to. The symbolic link itself is left untouched. This is the default behavior, and also the behavior that occurs if the NFM clients involved are older than version 2.5.7.

O **Preserve (transfer only)** This option causes NFM to copy the symbolic link itself to the target computer. The file pointed to by the symbolic link is left untouched. The resulting new (or overwritten) symbolic link will only have a valid target if it already existed on the target computer.

O **Skip** This option causes NFM to skip any symbolic links discovered while performing the current file action.

OS/390 folder

This folder contains per file definitions for use with mainframe type nodes. When files are transferred to and from a mainframe, the `Source Prefix` and `Source Files` fields always apply to the non-mainframe node, regardless of the direction of the transfer. If a transfer is being performed between two mainframes, this folder describes the source parameters. Please refer to the "OS/390 2" folder described in the next section when performing a mainframe to mainframe transfer.

MVS Fields

Most of the fields described below are used exclusively for the creation of a dataset when the OS/390 is the target of a file transfer. The following fields are available in this section:

DSNAME Dataset name for the file. This should be in the standard form "`a.b.c.d`" for a dataset and "`a.b.c.d(member)`" for a partition dataset member.

DCB Use the DCB named here to provide default attributes when creating a dataset.

☐ **This is a DD name** This checkbox indicates that the name provided in the previous field (DCB), is actually a DD name that provides default attributes for target dataset creation.

RECFM Specify the record format to use for dataset creation. The following selections are available from the pop-up list:

(blank) Use the default type.
U Undefined.
F Fixed length, unblocked.
FB Fixed-length, blocked.

FS Fixed-length, unblocked, standard.
FBS Fixed-length, blocked, standard.
FA Fixed-length, ASA print-control characters.
FBA Fixed-length, blocked, ASA print-control characters.
FSA Fixed-length, unblocked, standard, ASA print-control characters.
FBSA Fixed-length, blocked, standard, ASA print-control characters.
V Variable-length, unblocked.
VB Variable-length, blocked.
VS Variable-length, unblocked, spanned.
VBS Variable-length, blocked, spanned.
VA Variable-length, ASA print control characters, unblocked.
VBA Variable-length, blocked, ASA print control characters.
VSA Variable-length, spanned, ASA print control characters.
VBSA Variable-length, blocked, spanned, ASA print control characters.

LRECL Logical record length.

BLKSIZE Block size.

LIKE Use the cataloged data set named here to provide default attributes when creating a dataset.

Password Password for a password-protected dataset.

REFDD Use the specified ddname to provide default attributes when creating a dataset.

Space Units Space allocation unit format. Choose one of the following from the pop-up list:

Blocks Tracks Cylinders

☐ **SPACE=RLSE** This checkbox causes unused space to be released after a dataset is created and the necessary space to hold the transferred file is allocated and used.

☐ **SPACE=CONTIG** This checkbox indicates that the space requested for a target dataset be allocated contiguously.

Primary Primary space allocation unit.

Secondary Secondary space allocation unit.

☐ **Allocate PDSE** Allocate a PDSE (extended) rather than a PDS.

Retention Days Number of days before data set expires.

Expiration Date The date the data set expires.

Data Class Data class for new data sets.

Management Class Management class for new data sets.

Storage Class Storage class for new data sets.

VOLUME=SER= Volume serial number.

UNIT Device type if applicable.

VOLUME=REF= Volume information obtained from this named cataloged data set.

Volume Count The number of volumes this data set may span.

☐ **Variable Length Encoding** This checkbox is used to enable support for variable length encoding within files, in order to preserve variable record length information in binary files. This option will only affect files that have a `File Type` of `Binary` and include a `Record Format` of `Variable-length`. Such files have no intrinsic method of denoting the beginning and end of each record when being transferred from a non-OS/390 flat file to or from an OS/390 variable record length file.

For transfers to the NFM/OS390 client, this option states that the non-OS/390 source file contains a specific data format that embeds the variable length record information. Each file begins with a 2-byte length field that contains the length of the first variable length record, followed by the contents of that record. This is followed by another 2-byte length field for record two, and so on. These embedded VLE descriptors can be provided directly by a customer application, and is also provided as an option of the `nfmparse` program in preparing files for transfer. If the VLE option is turned off, the OS/390 target file will have variable record boundaries that are meaningless and undefined.

For transfer from the NFM/OS390 client, this option states that the non-OS/390 target file will be created with VLE descriptors embedded in the data of the target file. If the VLE option is turned off, the non-OS/390 target file will only contain the binary contents of the original file with all record boundaries stripped off.

NOTE: The act of using the VLE option to transfer a binary variable length file from OS/390 to a non-OS/390 platform (followed by transferring the file back to another OS/390 platform) will perfectly preserve the original OS/390 file layout while recreating all the proper variable length record boundaries.

☐ **Allow record truncation** This checkbox is used to modify the handling of text data being written to an OS/390 file that uses a fixed record format. For NFM/OS390 text data transfers, the "newline" characters in the non-OS/390 source file define each record boundary. This makes these files intrinsically variable length in nature. When writing to a fixed length text file, text lines that are shorter than the fixed record size will be padded with spaces on the end of the record until it reaches the fixed record size. By default, if a text line is longer than the fixed record size, a file transfer error occurs; aborting the transfer.

When this option is specified, this condition is treated as a warning that appears in the audit record, and the transfer is allowed to continue.

☐ **NO ASATRANS** Specifies that ASA control characters should not be translated to print control characters during file transfer.

DISP= Specifies the status of a data set. Enter one of the following keywords:

OLD NEW MOD SHR

Optionally include one or two of the following dispositions:

KEEP DELETE CATALOG UNCATALOG

☐ **Preserve Trailing Blanks** Do not automatically truncate trailing blanks when copying a text file from OS/390.

☐ **Wrap Records** Automatically wrap data that exceeds the record size into the next record when copying a text file to OS/390. By default, data that exceeds the record size is truncated.

HFS Fields

This panel contains attributes for files copied to and from an HFS (hierarchical file system) on the OS/390 platform. The following field is available in this section:

HFS Name The file name.

JES Fields

This panel contains attributes used to select data for transfer directly from the JES output queue to standard files on a non-OS/390 system. Each file in the output queue that matches all of the fields that are specified here will be selected. The fields left blank here will not be used as part of the selection process. The following fields are available:

Job Name
User ID
Output Class
Writer Name
Destination

OS/390 2 folder This folder contains the secondary OS/390 file definitions. These settings are used exclusively to define the target parameters for a mainframe to mainframe transfer. Any transfer involving a single mainframe, will only use

the standard OS/390 folder values. Note, this folder only appears if the "Multiple OS/390 Support" feature is enabled.

i5 folder

This folder contains per file definitions for use with `i5` (AS/400) type nodes.

LIB Fields

This panel contains attributes used by NFM when creating a LIB object. See `i5` documentation for details on each one.

Library Type **`*PROD`** | **`*TEST`**

Authority **`*LIBCRTAUT`** | **`*USE`** | **`*CHANGE`** | **`*ALL`** | **`*EXCLUDE`** | **`<authorization-list-name>`**

Object Authority **`*SYSVAL`** | **`*USE`** | **`*CHANGE`** | **`*ALL`** | **`*EXCLUDE`** | **`<authorization-list-name>`**

Audit Value **`*SYSVAL`** | **`*NONE`** | **`*USRPRF`** | **`*CHANGE`** | **`*ALL`**

Description **`'description'`** text of no more than 50 characters.

FILE Fields

This panel contains attributes used by NFM when creating a `FILE` object. See `i5` documentation for details on each one.

Member Name **`*NONE`** | **`*FILE`** | **`<member name>`**

Record Length NFM will create a `FILE` object with a record with a single field with the size specified. A value of `0` will default to `80` bytes.

Maximum Members This will set the maximum number of MBR object allowed in this `FILE` object. A value of `0` will default to `*NOMAX` (no maximum).

CCSID This will be the Character Code Setting ID assigned to this `FILE` object.

Description **`'description'`** text of no more than 50 characters.

Stream/MBR Fields

This panel contains attributes used by NFM when transferring a `STREAM` or `MBR` object.

Character Code Settings Fields

`None` No `CCSID` or `Code Page` is specified for this file. The `CCSID` assigned to the NFMC process will be assigned to any files created.

`CCSID` The values specified in the Conversion ID fields are **C**oded **C**haracter **S**et **Id**entifiers.

Code Page The values specified in the Conversion ID fields are Code Page values.

Conversion ID The specified or derived CCSID is assumed to be the CCSID of the data in the file, when a new file is created. This CCSID is associated with the file during file creation.

Text File Conversion ID The specified or derived CCSID is assumed to be the CCSID of the text data in the file, when a new file is created. This CCSID is associated with the file during file creation.

Record Size If a text file is record oriented then its record size can be specified here if any of the following special processing will be required.

Oversize records When writing a record type text file and the record size detected is different that the one specified the NFM client can:

Wrap: Split any record larger that the record size specified into multiple records by inserting a record separator at the record size specified. Also, if the pad option is specified the new record will be padded with trailing blanks before writing the data to disk.

Truncate (Warning): Truncate any record to the specified record size. This option will also generate a warning audit.

Truncate (Silent): Truncate any record to the specified record size. This option will not generate a warning audit.

Error: This option will stop the transfer and generate an error.

Add/Delete record separators When reading a record type text file that has no record separators the NFM client will add Linefeeds between every record as set by the Record Size specified. When writing a record type text file the NFM client will delete any Linefeeds before writing the data to disk.

Pad/Trim blanks When reading a record type text file the NFM client will remove any trailing blanks off each record based on the record size specified. When writing a record type text file the NFM client will pad each record with trailing blanks to force the record size specified before writing the data to disk.

No NFM Character Translation The NFM client will not do EBCDIC/ASCII translation on inbound or outbound text data. This can be used in conjunction with the CCSID/Code Page setting to allow the i5 to do the translation.

Encryption folder This folder contains options for automatic file encryption. For an overview of the file encryption feature please see section "File Encryption" in Chapter 8 "Tools".

The following fields are maintained in this folder:

Source File options

☐ **Automatically decrypt** This causes NFM to automatically attempt to decrypt any encrypted source files that it encounters during a transfer operation (this is determined by the presence of the ".tef" extension). The following characteristics are in effect for a transfer with this designation:

- No file extension is assumed on the source file, therefore the ".tef" extension must explicitly provided in the source file name, or the full name with extension must match a wildcard specification.
- This setting has no effect on a non encrypted (plain text) file.
- Encrypted source files are not altered. The decrypted data is simply sent to the target.
- This field is only meaningful for a transfer type function, and has no effect on an encrypt or decrypt plan function.

Note: The remaining fields in this section are used by the automatic decryption option specified above, as well as by the "Encrypt on source" or "Decrypt on source" plan functions.

Keyfile Designate a private key file on the source node to use for decryption (leave blank for default). The encrypted source file must have been already encrypted using the corresponding public key file. It is typical to let the key files default to the public/private key pair that is automatically and uniquely created for each node ("public.pem" and private.pem" that reside in a pre-determined location). This field has no effect if the pass-phrase option is chosen instead (below). This field may also be used to specify a public or private key file for use with a source encrypt or decrypt function (described in the "Plans" section).

Keyfile password This field holds a password used to access a private key file that is password protected. This field should not be confused with the next field "Encryption Passphrase" which is used instead of a public/private key pair to encrypt a file.

Encryption Passphrase This field holds a password or phrase that is used to encrypt/decrypt a file. This field should be left blank to designate the use of a public/private key pair operation instead.

Target File options

☐ **Encrypt** This option causes NFM to automatically encrypt the target files of a transfer operation. The following characteristics are in effect for a transfer with this designation:

- The extension ".tef" is automatically added to the target file name, so it is not necessary to designate the extension in the target name field.
- This setting will ALWAYS encrypt the target file, even if it was transmitted as an encrypted file (resulting in a double encrypted file with a ".tef.tef" extension).

- This field is only meaningful for a transfer type function, and has no effect on an encrypt or decrypt plan function.

Note: The remaining fields in this section are used by the encrypt option specified above, as well as by the "Encrypt on target" or "Decrypt on target" plan functions.

Keyfile Designate a public key file on the target node to use for encryption (leave blank for default "public.pem"). This field has no effect if the pass-phrase option is chosen instead (below). This field may also be used to specify a public or private key file for use with a target encrypt or decrypt function (described in the "Plans" section).

Keyfile password This field holds a password used to access a private key file that is password protected. This field should not be confused with the next field "Encryption Passphrase" which is used instead of a public/private key pair to encrypt a file.

Encryption Passphrase This field holds a password or phrase that is used to encrypt/decrypt a file. This field should be left blank to designate the use of a public/private key pair operation instead.

Buttons The following buttons are available from the `filesets` screen:

Apply Applies all changes made on this screen. This will cause an add or an update to occur, depending on whether or not the `fileset` preexisted.

Cancel Cancels all changes on this screen since the last time this `fileset` record was displayed or applied.

New Clears all fields on the screen to allow for the creation of a new `fileset`.

Delete Deletes the currently displayed `fileset` record.

Notes Displays a pop-up text window for storing notes in.

Plans

Overview

Plans may be accessed from the `Management` drop down menu.

There are three different types of components in a plan. There is always a single header that contains general information about the plan. There are one or more phases. Each phase consist of one or more functions. A function performs a

particular action with regards to a node or a node group. If a function works with a node group, a separate process is created for each node in the group and the node processes all run simultaneously. The processes might end up waiting on one another if a modem is required.

Phases allow functions to be grouped together, and control when and if the group of functions will run. Functions in a phase always run sequentially one after the other. The exception to this is when multiple functions work with the same node group, the node processes proceed to the next function, even if other node processes are still on the previous function. This keeps faster nodes from having to wait on slower nodes. If a function works with a node group and is followed by a single node function (like an exec of a shell script), all of the separate node processes must complete before the following function will run.

The Plan Detail screen behaves somewhat differently from most other configuration screens. The top portion of the screen displays all of the elements of a plan in a list form. Clicking on an entry and highlighting it will change the bottom portion of the screen to display the appropriate fields for viewing and changing that entry. Upon first entering the plan screen, an empty plan called New Plan will display with a single phase and single function. Highlight and edit the appropriate entries and use the New Phase and New Function buttons to add entries after the current entry.

Header

When the first line of the plan (the header) is highlighted, the following fields are maintained on the bottom portion of the screen

Plan Name This is the user selected name which defines this plan. Type in a new entry, or select an existing entry from the drop down list.

Source Node/Group This node or node group represents the default source node that will be used in every function when none is specified individually in the functions definition (see below). The source node/group is primarily required for file transfer type functions. It does have certain implications in other functions that will be specified in the function descriptions below.

The following reserved names may be used in this field and in the "Target Node/Group" field described below:

$SRCNODE Use the source node value from the plan header or the plan scheduler.

$TRGNODE Use the target node value from the plan header or the plan scheduler.

$CTLNODE Use the node that triggered this plan instance if applicable. This may be used for watch file plans or plans initiated through the remote command line interface.

Target Node/Group This node or node group represents the default target node that will be used in every function when none is specified individually in the functions definition (see below). A target node/group is specified along with a source node/group

for file transfers. Most other functions work only with the `target node`.

☐ **`One Instance Only`** When checked, this option prevents the system from running more than one instance of this particular plan at any given time. This would keep an operator from accidentally submitting a plan more than once.

☐ **`Enable Plan Retry`** This option enables the plan retry feature for any failed instances of this plan. A plan can be retried from the `Plan Activity` screen.

☐ **`Diagnostic mode`** This option turns on diagnostic mode for this plan. This will cause trace logs to be created for all activity generated by this plan to aid in diagnosing problems. This would be similar to turning on diagnostics for on each node involved in this plan but would only log node activity specifically for this plan.

`Audit Level` This field specifies the default for the level of information recorded into the audit trail during an instance of this plan. Regardless of this setting, a `plan start` and `plan stop` informational message along with all error messages are recorded. This field may be overridden during the submission of this plan. One of the following entries should be checked:

- O **`None`** No additional information is recorded.
- O **`Summary`** Summary information is also recorded including the starting and ending of each phase and the accessing of each individual node.
- O **`Detail`** This records `Audit trail` records for every single file operation.

Phases

When a phase is highlighted the following fields may be maintained on the bottom portion of the screen:

`Phase Name` This name will appear in summary and detail audit records. This name may also be used by another phase within the plan when that phase runs after or with this phase.

☐ **`Independent Nodes`** ☐ ☐

This checkbox has a significant effect on the behavior of a repeating phase with respect to node groups, and a minor effect on non repeating phases.

For a repeating phase where each function works with the same node group, the default behavior is that the phase cannot repeat until each node is finished with the final function in the phase.

With this option checked, each node will continue to repeat the phase independently of the other nodes, causing some nodes to advance into

later passes than others. If the repeating phase is based on a total repeat count, the phase will not completely finish until the slowest node has finished the final pass.

This option also affects the behavior of a delay function when the adjacent functions work with node groups. This happens regardless of whether or not the phase is repeating.

By default, a delay is considered a singular event, so if a previous function works with a node group, the single delay function will not start until each node is finished with the previous function.

With the "Independent Nodes" option checked, if a delay function follows a function working with a node group, each node task will independently wait for the delay amount and then advance into the next function even while other nodes may still be delaying.

☐ **Skip remaining functions if any function completes with rc** [>=] [0]

Check this box if you want the remaining functions within this phase to be skipped if the return code from any of the functions is greater than or equal to the selected value. Note, this is a default termination threshold that may be overridden on a per function basis described in the next section. (See "Appendix A: Return codes," for a description of the individual return codes.)

The next selection box determines when and if the currently selected phase runs. One of the three following options must be selected:

- O **Runs in sequence** Select this entry if you want the phase to run following its proceeding entry listed above. This option is the default value.

- O **Runs after phase name completes with rc > 0** Select this entry if you want the phase to run conditionally after another phase. This phase will only run if the phase specified finishes with a return code that satisfies the expression and value of the two remaining boxes (>, <, >=, <= may be chosen along with a number. (See "Appendix A: Return codes.")

- O **Runs with phase name** Select this entry if you want the phase to run at the same time as another phase. Any number of phases may be grouped together in this manner, with all of them starting at the same time. The additional check boxes should be specified in conjunction with this option:

  - O **synchronously**
    This check box indicates that the phase will not be considered finished until the run with phase is also finished. This would keep a phase that is to run after this one, idle until both phases are finished. If this option is checked, the choice to propagate the return codes can be checked as well. This

forces the return code on both phases to be set to the higher of the two.

O **asynchronously**
This check box indicates that the phase will run independently of the `runs with` phase. Any phases set to run after this phase will start as soon as this one is through.

The combination of these phase options can be used to create a very complex chain of events. (See "More about plan execution flow" in Chapter 6, "Advanced Functions.")

The next set of fields allows for the configuration of a repeating phase. A repeating phase may be used to create a plan that performs a polling type of operation. A repeating phase is usually preferable to a repeating plan for this type of activity because of less overhead and easier manageability. The following fields may be selected:

☐ **Repeat phase every** [#] [interval]

Specify the count and interval for the repeating phase (Example, every 30 seconds). If the phase does not finish the activity specified within the time allotted, the beginning of the next iteration will be delayed.

☐ **maximum repeat count** [0]

Specify a maximum number of times for the phase to repeat. This is one way to stop a phase.

☐ **stop repeating if any function returns** [>=] [#]

Specify an expression and value to terminate a phase. This is another way to stop a phase.

In addition to the previous two methods, a cutoff time may also be used to terminate a repeating phase as described below. Unlike the first two methods, a cutoff time may be used on a non-repeating phase as well which is why it is in a separate box on the screen.

☐ **Cutoff phase**

Use the checkboxes and fields to specify a termination time as either a fixed time of day, or a relative amount of time from when the phase started.

Note about polling plans. As described previously a repeating phase has advantages over a repeating plan, however a combination of the two may offer the best overall solution. Consider the following example:

A plan is required that copies the contents of a directory on a node to a target location every five minutes. This needs to be ongoing indefinitely. The following steps are taken:

- The copying phase is set to repeat every 5 minutes.
- The phase is set to terminate at 11:59 pm.
- The plan is submitted to run at midnight and repeat every 1 day.

This setup is ideal because it creates a single plan instance for each 24 hour period. This keeps the `Plan Activity` screen from getting overloaded with plan ID's. It also keeps the audits grouped nicely, allowing a user to view activity for a particular day by simply clicking on that day's plan ID in the `Plan Activity` screen.

Functions

When a function is highlighted the following fields may be maintained on the bottom portion of the screen:

`Function` This field selection box defines what type of function this is. Depending upon the function chosen, some of the remaining fields may change or have no effect. The remaining descriptions indicate what meaning they have with the different function types. The currently implemented functions types are:

`Transfer` Copies a `fileset` from a `source node` or `node group` to a `target node` or `node group`.

`Delete on Source` Deletes a `fileset` on the specified source node or node group. The `Source Prefix` and `Source Files` fields in the `filesets` record would designate which filenames to delete.

`Delete on Target` Deletes a `fileset` on the specified target node or node group. The `Target Prefix` and `Target Files` fields in the `filesets` record would designate which filenames to delete, unless they were empty in which case the `Source` fields would be used.

`Execute` Executes a command on a specified `target node` or `node group`.

`Rename` Renames the entries of a `fileset` on a target node or node group. The source and target fields of the `fileset` must both be specified for this option..

`Run Plan` Submits a plan for execution. The `source` and `target nodes` specified become the `default source` and `target node` just as if they had been typed in manually from the `Plan Scheduler`. This function will not complete until the submitted plan completes. The return code from the plan instance will become this functions return code.

`Custom FTP` This advanced option, allows the use of a customized FTP transfer whereby the user can add additional commands to the

default ones used for an FTP transfer. The default commands used for FTP exist in template files (these are in an "attachments" subdirectory in NFM). The default files ftppush and ftppull may be copied to other names and placed in the subdirectory "attachments/custom" where they may be modified.

Once this is done these files will be available from the Template pop-up list field in the plan function. It is the users' responsibility to use the correct base file based on the direction of the transfer. The file ftppush is used for sending files from an NFM client to an FTP node, the file ftppull copies in the other direction. Never modify these original files.

Delay This simply allows for a wait to occur before continuing on to the next function or phase. The time can be specified in seconds or as an absolute time of day.

Release This function releases a previously started streaming file transfer on a given node or node group for a given fileset. (See more about this feature under "File streaming" in Chapter 6, "Advanced Functions.")

Update This function works just like the Transfer function except if the target file already exists then NFM will compare the modification date, size, and/or checksum of the two files to determine if the files are exactly the same. If they are the same then the transfer is not necessary and it is skipped. You can select which file information is to be checked to determine if the transfer is necessary.

Synchronize This function will synchronize files and/or directories specified in a fileset. NFM will compare the modification date, size, and/or checksum of the corresponding files to determine if any files need to be synchronized. Directories are synchronized (created) if they don't exist. The direction of synchronization can be "Source to Target", "Target to Source", or "Bidirectional". Missing (at destination) or deleted (at source) files can be synchronized except during the Bidirectional option. Recursion into subdirectories and the creation/deletion of directories is configured via the fileset in the Property Tab.

Encrypt on Source / Decrypt on Source Encrypts or decrypts a fileset on the specified source node or node group. The Source Prefix and Source Files fields in the fileset designate which filenames to act on. Also, the Encryption folder source node options, other than the Automatically decrypt, checkbox may be used to designate pass phrase or key file options to use.

Encrypt on Target / Decrypt on Target Encrypts or Decrypts a fileset on the specified target node or node group. The Target Prefix and Target Files fields in the fileset designate which filenames to act on. Also, the Encryption folder target node options, other than the Encrypt, checkbox may be used to designate pass phrase or key file options to use.

Source Node/Group This node or node group specifies the source node(s) for a file transfer and designates where to delete files for a Delete on Source. This field may also be used to run multiple copies of a command on a single target node. (See more about this under "Program Execution" in Chapter 6 "Advanced Functions.") If the source node(s) field is left blank, the default from the Plan header or the Plan Scheduler is used.

Target Node/Group This node or node group specifies the target node(s) for most functions. If the target node(s) field is left blank, the default from the Plan header or Plan Scheduler screen is used. The source or target node field may specify a node group, but not both.

Fileset This selection box displays for a Transfer, Rename, Delete or Release type function and allows selection of an existing fileset definition.

Multicast Profile This field displays for a Transfer type function and allows selection of an existing multicast profile name. When left blank, this specifies that the transfer will not use IP multicast.

Command This field displays for an Execute type function and allows for a command to be typed in that is to be run on the target node(s).

Plan Name This field displays for a Run Plan type function and allows selection of an existing plan name.

Priority ○ **Low** ○ **Medium** ○ **High**
This defines a priority for a file transfer with regards to other file transfers that are taking place within NFM at the same time. This only applies to socket type transfers. This field is only used if the prioritization option is turned on in the System Settings.

☐ **Skip remaining functions if return code** [>=] [0]

Check this box if you want the remaining functions within this phase to be skipped if the return code from this function is greater than or equal to the selected value. This does NOT cause a break in a repeating phase. That can be accomplished using a different setting (see Phases description in previous section). This field overrides the default termination value specified in the phase for the current function. (See "Appendix A: Return codes," for a description of the individual return codes.)

☐ **Hold on recoverable error**

Check this box if you want this function to become held in case of an error. (See more about this option in "Error recovery" in Chapter 6, "Advanced Functions.")

☐ **Enable checkpoint restart**

Check this box to enable the `checkpoint restart` option. This must be specified to allow a retransmission of a file to transfer only the portion that did not copy. (See more about this option in "Error handling" in Chapter 6, "Advanced Functions.")

☐ **Overwrite allowed**

Check this box to allow transfers or renames to automatically overwrite any existing target file; otherwise, an error will occur if the target file already exist.

☐ **Use temporary names**

Check this box to instruct NFM to use temporary names while transferring files. This will cause the files to be copied to their corresponding destination names followed by an underscore _. After the copy is successful, each file will be renamed to its proper destination name. This option allows other software that is polling for the presence of a specific file, to be assured it has a complete copy of the file.

When used during an append transfer (see '`Append to Target`' description below), this checkbox has special significance. This causes the files to be completely copied (to the temporary names) prior to appending them to the end of the target file(s). This is inherently safer because if there is a communication error, the target file remains untouched and it is more likely that a retry operation will work. For this reason it is highly recommended that temporary names be used when appending files.

☐ **Redirect naming**

Check this box to enable the redirect naming option. This option reverses the meaning of the `source` and `target` fields in the `fileset` definition with regards to this specific function. This allows specifying a reverse copy (source prefix and files become target, and, target prefix and files become source). This allows using an existing fileset definition to copy files in the reverse direction implied by the fileset definition.

☐ **Preserve timestamps**

This option sets the timestamp on a target file equal to that of the source file during a transfer.

☐ **Preserve source attributes**

This option attempts to preserve certain file attributes like permissions, or FDA from the source file to the target file during a transfer. This option is only meaningful if the target file already exists. If the target

file is being created, attributes are automatically preserved from the source file.

☐ **Append to Target**

Causes the source file(s) to be appended to any existing target file(s) rather than simply copied over them. It is recommended that '`Use temporary names`' be specified in conjunction with this option (see description above).

☐ **Clear contents first**

This check box is only applicable if an append operation (previous option), is being done. This causes the target file to be cleared prior to the append being done. At first glance this would appear to negate an append operation, and indeed would do so in a one-to-one transfer without wildcards being used. However, an append operation may also be used to combine files from multiple sources, in the following two scenarios:

1. A many-to-one transfer using a fixed target file name (no node extensions & no imbedded environment variables).
2. A transfer involving wildcard names that specifies a fixed target file.

Either of these may combine multiple files and optionally append them to the end of existing data or not based on this option.

☐ **Delete source file after completion**

This option causes a delete on source operation to occur automatically if the file(s) are successfully transferred. If a delete is desired, this is preferable to specifying a separate function to accomplish this task for two reasons:

1. It guarantees that a delete will not occur if the file(s) are not transferred. This would otherwise require the user to carefully select the correct termination logic to avoid accidentally deleting undelivered files.
2. If wildcards are being used, it avoids the potential timing problem of deleting a file that appeared after the transfer was finished but before the delete was done.

☐ **Streaming Interval 0**

This box should be checked if you want to designate this transfer as a streaming type transfer. If so, also specify a non-zero number in the following field to designate the streaming interval in seconds. A streaming file transfer allows a file to be transferred as another application or command is writing it to. (See more about this feature under "File streaming" in Chapter 6, "Advanced Functions.")

☐ **Asynchronous**

This option shows for an execute type function only. Check this box if you do not want to wait for the program to terminate

☐ **Use Shell**

This option shows for an execute type function only. If this option is checked, NFM will attempt to run the command using the native operating system shell or command line interface. This would be necessary for doing things like redirecting output. Also, by specifying this option, the `Command` field is changed to multiple line text box, called `Script` that allows for a multiple line command to be entered. This multiple line command will now be run as if it were a local batch file on the target nodes, which eliminates the need for separately sending down a batch file and then executing it.

☐ **Enable Change-only Transfer**

This option shows for file transfer type functions only. If this option is checked, NFM will attempt to determine what parts of a file have changed and then transfer only the parts of the source file that are different and apply those differences to the target file to create a new file identical to the original source file. This would reduce the amount of data that is transferred when there are only a few differences between the source and target file. The Block Size can be specified to change the default used to check the file. The Threshold can be specified to set the minimum percentage of the size of differences compared to the source file size. That is, if 90% of the file is going to be transferred anyway because of differences then transfer the whole file. The Minimum Size of the source can also be specified. It may be more efficient to just transfer small files in their entirety than to do the change-only transfer. Note: Files marked, as text files in the Fileset will not use this option if the files are on OSes that do not use the same text file format.

Detect changes by

This option shows for Update and Synchronize functions only. Since these functions transfer the files only if they detect that the source and target files are different, NFM must be configured on how to determine if the files are different. The difference can be determined by the files Size, Timestamp, and/or Checksum. One more can be selected. If none are selected then the file will always be transferred. Note: Files marked as text files in the Fileset will not have their sizes compared if the files are on OSes that do not use the same text file format.

Synchronize

This option shows for the Synchronize function only. The option specified the direction that the synchronization will take place. It can be set to "Source to Target", "Target to Source", or "Bidirectional". Since these functions transfer the files only if they detect that the source and target files are different, NFM must be configured on how to determine if the files are different. The difference can be determined by the files

Size, Timestamp, and/or Checksum. One more can be selected. If none are selected then the file will always be transferred.

☐ **Synchronize missing files**

This option shows for the Synchronize function only. Check this box if you want missing files or directories to be deleted from the destination if the original file or directory is missing. This option is not available for "Bidirectional" synchronization.

Timeout This option shows for an execute type function only. This field allows a time-out value (in seconds) to be specified. If the command is not finished executing within this amount of time, the command is killed and an error is reported.

Error Level This option shows for an execute type function only. Normally, when an execute function finishes with any value other than zero, the function is flagged as an error, with an error type audit message. This field allows setting a threshold to determine when this occurs. If the actual exit code is less than the value specified in this field, the function will not be considered in error. There will not be an error audit message and the function will not be retried during a plan retry operation. The function will still finish with the exit code, which will still filter up to affect the plans return code.

Template For a custom FTP, set this field to the custom control file provided by the user to perform the indicated FTP operation.

Plan Name For a `Run Plan` type function, this field specifies the plan name.

Buttons

The following buttons are available from the `Plans` screen:

New Plan Clears all fields on the screen to allow for the creation of a `new plan`.

New Phase Inserts a `new phase` immediately following the highlighted item.

New Function Inserts a `new function` immediately following the highlighted item.

Move Up Swaps the currently selected item with the one immediately above it.

Move Down	Swaps the currently selected item with the one immediately below it.
Delete Line	Deletes the currently selected item.
Summary	Takes the session directly to the Plan Summary screen.
Apply	Applies all changes made on this screen. This will cause an add or an update to occur, depending on whether or not the plan preexisted.
Cancel	Cancels all changes on this screen since the last time this plan record was displayed or applied.
Delete	Deletes the currently displayed plan record.
Notes	Displays a pop-up text window for storing notes in.

System settings

System settings may be accessed from the Management drop down menu. These folders display and maintain overall system information as well as settings that are used to customize NFM for a particular customer's environment.

Info

The Info folder contains display only information regarding the version and license. The following fields are displayed in this folder:

NFM Server Version This field shows the current version number of the NFM server.

Nodes Defined This field shows the number of nodes currently defined to the system.

Maximum Available This corresponds to the Nodes defined field to the left. This is the maximum number of nodes which can be defined to the server based on the licensing agreement.

Clients Defined This field shows the number of nodes that are enabled as NFM clients currently defined to the system.

Maximum Available This corresponds to the Clients defined field to the left. This is the maximum number of NFM client enabled nodes that can be defined to the server based on the licensing agreement.

The remaining fields display license specific information.

Server

The `Server` folder contains overall configuration items for the server. The following fields are maintained on this screen:

`NFM Server Node Name` This field should be set to the `name` of the node record that represents the NFM server computer.

☐ **`Enable client initiated commands`**

Check this box if you want to enable the remote server interface feature that is further explained in Chapter 8, "Tools."

The remaining section of this folder allows for user customized connection entries to be created. These entries serve the following two purposes:

1. They designate the port and other criteria for the NFM server to listen for incoming socket connections from NFM clients. This is primarily to service incoming requests from the remote command line interface which is enabled by the checkbox described above.

2. The first entry that is marked for SSL provides default global parameters for outgoing SSL connections. This includes the private key and certificate fields along with the remaining fields that modify their behavior.

Note: It is not necessary to create connection entries for either of these purposes, if the default values are acceptable and do not need to be customized. The absence of any defined connection entries here causes the following default behavior:

- For incoming socket connections from the NFM clients. Two entries are automatically created. A listen entry is assumed for port 8008 to service non-SSL connection requests. Another listen entry is assumed for port 8009 to service SSL connections. For the SSL entry, the default certificates are in effect and the remaining options default. If any customized entries are created, neither of these defaults is created, so all necessary connections must be specified.

- For outgoing SSL connections to NFM clients, the default SSL certificates and options are in effect (the receiving NFM clients may still use different certificates configured at the NFM client sites).

The following attributes are available for each connection entry:

`PORT=[#]` This field should be set to the desired port for the NFM server to use when listening for incoming connections. This field is not used for outgoing connections. Outgoing connections use the port specified in the individual node or model definition. This field may be left blank to default to `8008` for non-SSL connections and `8009` for SSL connections.

`SSL=[Y/N]` For incoming connections this indicates that this port is used exclusively for SSL or non-SSL traffic. The NFM client

configuration at the nodes must use connections whose port and SSL flag match for connections directed to this NFM server.

Also, the first connection entry in this list that is marked for SSL has special significance. The remaining key and certificate fields will be used for all outgoing connections destined to SSL nodes as determined by their node or model definitions.

Note: The remaining fields are all used for SSL type connections only.

Private Key file name This PEM file contains the private encryption key that is used to secure communication connections. If the file name provided does not specify a path, the file must exist in the "`ssl`" subdirectory of the NFM server home directory. If this field is left blank the file "`NFMPrivateKey.pem`" is used by default.

Certificate file name This PEM file contains the certificate that will be sent to the remote system for authentication. If this field is blank the file "`NFMCertificate.pem`" is used by default.

Root Certificates file name This PEM file can contain any additional root certificates and/or any certificates to use with the "Whitelist" option.

Verify Peer This checkbox determines if the SSL will verify the remote system's certificate.

Self-Signed Certificates This checkbox indicates whether or not to allow self signed certificates. The default is to not allow "N".

Whitelist This checkbox determines if the Root Certificates file will also be used as a "White List" for the incoming peer certificates.

Password This specifies a password used to access the "Private Key" file if it requires one.

Audit

The `Audit` folder contains options that are specific to the audit keeping feature of NFM. The following fields are maintained on this screen:

Housecleaning time of day `Time of day(00:00-23:59)` to perform activities such as removing or archiving old audit records and finished plan instances.

Audit & Jobs expire after # days This number specifies how old audit records and completed plan instances should be (in days) before they are automatically deleted or archived.

Expiration action ○ **Delete** ○ **Archive** Select the desired action for audits when the expiration date is reached. `Delete` simply removes the audit entries. `Archive` causes the audits to be moved to the default archive location specified in the next field.

Move audits to This designates the default archive location. An archive location is a named entry that represents a physical directory in the NFM server file system. A predefined entry "ARCHIVE" is available. Additional locations may be created by the user using the NFM command line interface (reference the sub-command `putasource` in Chapter 8, "Tools").

Maximum audit size # bytes This specifies a maximum size (in bytes) for the text of a single audit message. Most audit messages are relatively small, containing enough text to convey a simple error, or some other informational message. The exception to this (and the reason for this limit) is the audits produced from program execution within a plan. The entire text output from any given program execution is captured and stored as a single audit that may contain any number of lines of output. This setting causes an audit to be truncated if necessary (a message indicating truncation was done is appended to the audit).

Very large audits have the potential to fill up the file system that holds the NFM audit records. For this reason, it is recommended that when a program is to be run that is known to generate a large amount of output, the output be redirected to a file during the execution of the command.

☐ **Audit Configuration Changes**

This option causes audit messages to be generated when an operator modifies any part of NFM's configuration.

Transfers

The `Transfers` folder contains options that are specific to file transfers. The following fields are maintained on this screen:

Maximum Active Jobs This field should be set to the maximum number of `plans` that can be running simultaneously. This field must be set to a non-zero value.

Maximum Active Nodes This field should be set to the maximum number of nodes that can be actively involved in file transfers at any given moment. This does not apply to nodes involved in a `multicasting` or `streaming` file transfer. A value of `zero` implies no limit. There are two primary reasons for setting this field:

1. To limit the total amount of bandwidth, NFM utilizes with regards to other network activity. With limited simultaneous NFM connections other network traffic should respond better.

2. Even if other network activity is not a consideration, attempting too many connections may overwhelm the network or one of the nodes to the point it actually slows down all of the file transfers. In addition, if the network is not fast enough to keep up, then individual connection timeouts may occur during file transfers.

The limit can be set below the actual number of nodes that may be attempted during a transfer. Once the limit is reached any additional nodes will simply wait for others to complete before transferring. This will be visible from the individual status bars on the `Plan Monitor` screen discussed in Chapter 5. This feature is independent of the `Maximum Active Nodes` field that may be specified on a per node group basis (described earlier in this chapter under "Node Groups").

☐ **Enable Prioritization**

This enables the proritization option of individual transfers to be implemented. If this option is `off`, all transfers have equal priority regardless of the setting specified in the plans. Although the prioritization scheme effectively gives some file transfers priority over other ones, its usage may limit NFM's overall bandwidth allocation with regards to other network traffic beyond the scope of NFM. For this reason, the ability to turn `off` prioritization globally has been provided.

☐ **Compress Transfers by Default**

This specifies the system wide default use of compression during file transfers. This option can be overridden on any given `fileset` from the `Filesets` screen.

☐ **Encryption Transfers by Default**

This specifies the system wide default use of encryption during file transfers. This option can be overridden for any given node on the `Node Detail` screen.

☐ **Store External Attributes**

This specifies the system wide default use of the store external attributes during file transfers. This option can be overridden for any given `fileset` from the `Filesets` screen.

☐ **Continue Plans on Full Database**

This option, if enabled, allows plans to continue processing even if the NFM database runs out of space and can no longer store additional data (including audits, tracking markers, etc.). By default, if the system runs out of space for database records, all plan activity will go on hold (users will clearly see this happen from the `Plan Activity` or `Plan monitor` screen). Once space is available, plan activity will resume as if nothing had happened. With this option in effect, plans will continue their activity, however, it will not be properly audited and tracked.

☐ **Automatically Resume Interrupted Plans after # seconds**

A plan instance becomes "Interrupted" if the NFM server computer is rebooted or if a backup NFM server takes over for a crashed system. The interrupted state of a plan instance is very similar to the cancelled state. It may be resumed manually from the `Plan Activity` screen, or by checking this option, all interrupted plan may be automatically resumed, when the NFM server starts/restarts.

The delay in seconds may be specified to allow other action (such as human intervention) to occur after a reboot, but prior to resuming failed plans.

☐ **Change Detection Checksum Type**
An Update or Synchronize function can detect if a file has been modified by computing a checksum on the file on both the source and target locations. If the checksum on both copies of the file do not match then the file has been modified. The checksum can be based on a combination of one to three different computations based on CRC64 (64-bit), MD5 (128-bit), of SHA (160-bit).

Change-only Transfers
When Change-only Transfers are enabled via one of the file transfer functions in a plan, the block size, checksum size, and minimum file size can be changed here. The defaults for block and checksum size should normally be alone. The minimum file size can be used to specify a file size that a file must be larger than before the Change-only Transfer is used. That is, depending on certain situations it may be quicker just to transfer small files than using the Change-only Transfer option.

Statistics

The `Statistics` folder contains options that are specific to statistics options. The following fields are maintained on this screen:

☐ **Enable General Statistics**
This enables the gathering of statistics used for the dashboard feature.

Detailed General Statistics expire after # days This specifies the amount of time before detailed statistics records are automatically deleted.

Summary General Statistics expire after # days This specifies the amount of time before summary statistics records are automatically deleted.

☐ **Enable Performance Statistics**
This enables the gathering of statistics used for the performance graphs. The feature is being deprecated in favor of the dashboard, but is kept for backward compatibility. It is recommended not to use this.

Quick

The `Quick` folder contains options that are specific to the `Quick` menu functions. The following fields are maintained on this screen:

Quick Transfer Template This field is used to configure the NFM plan that represents a Quick Transfer.

Quick Execute Template This field is used to configure the NFM plan that represents a Quick Execute.

The use of these two fields is further explained in Chapter 9, "Quick Functions," in section "Quick configuration."

AUTH

The authorization tab selects and configures which user authentication method is to be used for accessing the NFM system. The first field: "**`Configure user authentication type`**" reflects which method is currently in use: NFM (the default) or LDAP. The remaining fields will change based on which method is chosen allowing for the full configuration of the chosen method.

Note: This field cannot actually be changed from this screen since it would almost certainly disable the current user's session. This field is used to select and configure the details for the method. Once the method is fully configured, and optionally tested, the authentication method may be changed with the NFM command line interface. Refer to Chapter 8, "Tools", section "NFM command line interface", for the command: `nfm set,auth`.

NFM

NFM by default stores and maintains its own set of user passwords. This authentication type allows for a series of minimum password requirements to be implemented, along with controls to automatically suspend users under certain conditions related to bad logon attempts, inactivity, etc. The following settings are available:

`Enable User Password Restrictions` This checkbox allows the remaining settings to become active. By default, NFM does not enforce password restrictions.

`Maximum Failed Logon Attempts` If a user enters the wrong password this many times in a row, their logon will automatically become suspended. This will require an administrator to re-enable the user before they can logon again (this is done from the "User Detail" screen, "NFM Permissions" tab). A setting of "0" indicates no limit. Default=5.

`Password Expiration (normal users)` Once a password has been added or changed, it must be changed again on a regular basis within this interval (# of days) or it will become expired. If it does become expired, a user will be forced to change it the next time they attempt to logon. "0" indicates no expiration. Default=90.

`Password Expiration (administrators)` The password expiration (# of days) for administrators. Default=30

`User Inactivity Suspension` The number of days of inactivity (no user logons) that will result in a user becoming automatically suspended (requiring re-enabling by an administrator). Default=90.

`User Inactivity Deletion` The number of days of inactivity that will result in a user becoming automatically deleted. Default=0 (never).

`Minimum Length` Passwords must be at least this many characters. A value of "0" indicates that no password is required. Default=7.

Minimum Alpha Characters Passwords must contain at least this many alphabetic characters. Default=1.

Minimum Non-alpha Characters Passwords must contain at least this many non-alphabetic characters. Default=1.

Old Password Retention Indicates the number of previous passwords, on a per-user basis, that will be stored in the database. This password history is kept available for the benefit of the "Minimum Repetition Length" setting below. Default=13.

Minimum Repetition Length While setting a new password, no segment of characters of this length or longer may be repeated from any of the previously stored passwords for this user. For the purposes of comparison, case insensitivity, and reverse order of the characters will be considered. Default=4.

Enforce Prohibited Words With this option enabled, a user will be unable to set a password that matches a word on the "Prohibited Words List". It can optionally be set to deny a password that contains a word on the list, based on the next setting. The "Prohibited Words List" must be created and maintained by the customer. It may be imported from a simple text list of words. Default=off.

Do not allow passwords that Match/Contain prohibited words This field is set to "Match" or "Contain" to control the use of the "Prohibited Words List" as described above. Default="Match".

Ignore words smaller than # characters. If the previous setting denies passwords that "Contain" words on the list, then this number represents the smallest length of words in the list to consider while looking for a match. The purpose of this is so that the customer can import a dictionary list of words, for example, without having to remove small words like "a" or "I", which could easily exist in a password. Default=4.

View Prohibited Words List

This button allows viewing (but not changing) the current prohibited words list in a pop-up windows.

LDAP

NFM can be configured to perform user authentication by an LDAP lookup when logging on to, and using the NFM user interface (both the GUI and the command line interface). The fields configured on this tab supply the information needed by NFM to verify the user ID and password with the LDAP server.

If LDAP authentication is selected for use with NFM, it is not actually necessary to create an NFM user record for each potential user. Instead, LDAP users may

inherit the settings of an existing NFM user record (see "NFM User Attribute" below).

The following fields are maintained on this screen:

LDAP Server This is the DNS name of the LDAP server.

Port # This is the port number served by the LDAP server if other than the default. (`0` = use default).

Search Base This is the base entry where NFM will begin its LDAP search.

Scope This is the scope of the LDAP search below the LDAP Base. Select from the provided menu: (`Base | One Level | Subtree`).

Search DN This is the distinguished name that NFM will use to bind to the LDAP server. This DN must have "search" permission.

Search Timeout This is the maximum amount of time in seconds that a user has to respond with an LDAP user or password.

UID Attribute This is the name of the LDAP attribute that will contain the user ID supplied for the NFM login. If left empty the default value `"uid"` will be used.

NFM User Attribute This is the name of an LDAP attribute that contains an existing NFM user ID that will be used to inherit user settings (permissions, preferences, etc.) from. If this field is specified, each user that logs in will be expected to have the field of this name set to the NFM user that they will inherit settings from. In this case it will not be necessary to have an actual NFM user defined with the login user's name. (Note that multiple LDAP users may inherit settings from the same NFM user).

If this field is left blank then each user that logs in must have an actual NFM user defined for them, whose name matches their LDAP login ID.

Access Attribute This is the name of an optional LDAP attribute that will be checked for permission to access the NFM application. If left empty this option is not used.

Access Value This is the value that must be present in the LDAP access Attribute to allow access to the NFM application. If left empty this option is not used.

Cache Size This value defines the number of cached LDAP authentication requests that may exist at any given moment. The NFM LDAP cache exists to improve the performance of the NFM user sessions by not forcing each and every command to be individually authenticated through the LDAP server. Ideally this

value should be greater than or equal to the number of users that may access the NFM server.

Cache Timeout (See description of the cache above). This defines how long an authentication request may remain cached prior to forcing it to be re-authenticated through the LDAP server. A long timeout will improve performance by minimizing lookups, however, if a user is denied access to NFM via an LDAP configuration change, it will take this long (the timeout value) before the change takes effect.

Node Passwords

This tab allows assignment of node password restrictions similar to those used for user passwords. Also, like user passwords, a node password may be set to expire. However, unlike node passwords, this will not actually prohibit the use of the node. Rather it will generate warnings when any such nodes are being used. The following fields are available:

NFM by default stores and maintains its own set of user passwords. This authentication type allows for a series of minimum password requirements to be implemented, along with controls to automatically suspend users under certain conditions related to bad logon attempts, inactivity, etc. The following settings are available:

Enable Node Password Restrictions This checkbox allows the remaining settings to become active. By default, NFM does not enforce node restrictions.

Password Expiration Once a password has been added or changed, it must be changed again on a regular basis within this interval (# of days) or it will become expired. If it does become expired, audit warnings will be generated on a daily basis indicating so. Warnings will also be generated by any plan that uses the node. This will NOT however cause the attempted action with the node to fail. "0" indicates no expiration, which is the default.

Minimum Length Passwords must be at least this many characters. A value of "0" indicates that no password is required. Default=7.

Minimum Alpha Characters Passwords must contain at least this many alphabetic characters. Default=1.

Minimum Non-alpha Characters Passwords must contain at least this many non-alphabetic characters. Default=1.

Old Password Retention Indicates the number of previous passwords, on a per-node basis, that will be stored in the database. This password history is kept available for the benefit of the "Minimum Repetition Length" setting below. Default=13.

Minimum Repetition Length While setting a new password, no segment of characters of this length or longer may be repeated from any of the previously stored passwords for this node. For the purposes of

comparison, case insensitivity, and reverse order of the characters will be considered. Default=4.

Enforce Prohibited Words With this option enabled, a user will be unable to set a password that matches a word on the "Prohibited Words List". It can optionally be set to deny a password that contains a word on the list, based on the next setting. The "Prohibited Words List" must be created and maintained by the customer. It may be imported from a simple text list of words. Default=off.

Do not allow passwords that Match/Contain prohibited words This field is set to "Match" or "Contain" to control the use of the "Prohibited Words List" as described above. Default="Match".

Ignore words smaller than # characters. If the previous setting denies passwords that "Contain" words on the list, then this number represents the smallest length of words in the list to consider while looking for a match. The purpose of this is so that the customer can import a dictionary list of words, for example, without having to remove small words like "a" or "I", which could easily exist in a password. Default=4.

View Prohibited Words List

This button allows viewing (but not changing) the current prohibited words list in a pop-up windows.

Note: The "Prohibited Words List" if active, is the same one used for node passwords.

Buttons

The following buttons are available from the `System settings` screen, they apply to changes made in any of the folders:

Apply `Applies` all changes made on this screen.

Cancel `Cancels` all changes on this screen since the last time this screen was displayed or applied.

Multicast Profiles

Overview

`Multicast` profiles may be accessed from the `Management` drop down menu.

When a fileset is transferred from a single source client to a large group of target clients, the IP multicast protocol can provide dramatically increased performance. To realize this performance, several communications parameters must be carefully tuned for the network topology, traffic and end-node capabilities in use during the transfer.

IP multicast is similar to broadcast in that the source node transmits a single packet that is then received by multiple target nodes. IP multicast differs from broadcast in two important ways:

1. broadcast packets cannot be "forwarded" across a router. IP multicast packets can be forwarded across multiple routers.
2. broadcast packets can only be received by nodes on the same subnet, even if they are on the same physical LAN. Nodes on any subnet can receive IP multicast packets even if they are not on the same physical LAN.

IP multicast may be used effectively when:

1. The same fileset is to be sent to multiple targets.
2. The source node is an NFM client.
3. The target nodes are NFM clients.
4. Both source and target clients have enough temporary space to store a compressed version of the fileset in the clients' /tmp directories.
5. The network is configured to properly forward IP multicast packets.
6. The network has sufficient available bandwidth to effectively carry IP multicast traffic.

A multicast profile is a collection of communication parameters that allow precise tuning to match the network. A single multicast profile may be specified for multiple functions in multiple plans. This is similar to the reuse of models in the definition of nodes.

Unlike the TCP/IP protocol used for one-to-one transfers, the IP multicast protocol is not inherently reliable. IP multicast packets may be dropped by the network or by the target client's IP stack. The base IP multicast protocol makes no provision for ensuring that the data arrives at its destination or is in the proper sequence.

NFM clients implement an algorithm for reliable file transfer between NFM clients using IP multicast. The basic transmission algorithm consists of prepare, announce, transmit, confirm and apply steps. The transmit and confirm steps are automatically repeated as necessary.

Only the transmit step uses IP multicast. During the announce and confirmation steps, the source client opens a TCP/IP connection to each target node. The source client can open multiple simultaneous TCP/IP connections. The Announcement Channels and Confirmation Channels fields of the Multicast Profile control the number of simultaneous connections.

To reduce the computational load on the source client and facilitate partial retransmission of data, a `fileset` is prepared on the source client before it is sent via IP `multicast`. This preparation of a `fileset` for `multicast` transfer includes compression, encryption and storage of the `fileset` on the source client. This occurs only once on the source client at the start of the IP `multicast` transfer. The `apply` of a `fileset` includes decryption, decompression, storage of the files and removal of the temporary file on the target clients. This occurs once on each target client after all of the `fileset` data has been received.

In some cases, an IP `y` transfer to a particular client may not be possible. If an IP `y` transfer function fails, the NFM server will retry the transfer using the default (`non-multicast`) transfer method.

The suggested values below are intended only to provide a starting point. Some experimentation will be necessary to achieve optimal results on a particular network under a particular load.

All multicast transfer types

The fields in this section apply to all `multicast` transfer types.

`Name`
This is the user-selected name that identifies this `multicast` profile. Type in a new entry, or select an existing entry from the drop down list.

`Data Payload Size`
This is the number of data bytes that will be transmitted in a single IP `multicast` packet. An additional four to eight bytes of overhead will also be sent in each IP `multicast` packet. Values below `128` may yield poor performance due to the larger number of packets that must be transmitted and confirmed. Values larger than `1024` may cause many packets to be dropped by the network or the target client's IP stack if `non-multicast` traffic is present. Values in the range of `256` to `1024` bytes will often give good results.
Suggested value: `1024`

`Inter-Packet delay`
This is the number of milliseconds that the source client will delay between transmitting data packets. This delay is important to allow the target clients time to read and store the data. This delay also allows for TCP/IP traffic to be processed during the delay. Longer delays will cause slower overall transmission speeds. Delays that are too short may cause many packets to be dropped resulting in more retransmission of those packets.
Suggested value: `20`

`Initial Transmit Count`
This is the number of times that the `fileset` will be transmitted before beginning the first confirmation step. On lightly loaded, reliable networks, a value of `1` may be sufficient. For networks with higher

traffic or less reliability, or when there are very many target clients, a value of 2 or more may provide better overall performance.
Suggested value: 1

Time To Live (TTL)
This value corresponds to the number of routers that the IP multicast packets must cross to reach their farthest destination. The allowable range is from 1 to 255.
Suggested value: 10

Inactivity Timeout
This is the number of seconds that the target client will wait to receive multicast data. If no multicast data is received within this time, the target client will stop attempting to receive multicast data.
Suggested value: 600

Use CRC-32 Data Integrity Check
This box should be checked to enable CRC-32 checking of each data packet. This protects against data corruption during transmission. On highly reliable networks, this may not be needed.
Suggested value: yes

Progressive Delay
Checking this box will cause the source client to delay progressively longer each time a block is retransmitted. The progression is 25 percent additional delay for each retransmission up to a maximum of five times the initial delay. This allows setting of a lower initial Inter-Packet delay while dynamically accommodating slower clients or unpredictable network traffic.
Suggested value: yes

Randomize Retransmit
Checking this box will cause retransmitted packets to be sent in a pseudo random order. This often relieves cyclical problems such as dropping every n'th packet. Most cyclical problems result from network or target client processing limitations.
Suggested value: yes

Reliable / Unreliable
Choosing reliable guarantees that either the fileset will be received correctly or any failure will be reported. Choosing unreliable will not guarantee that the fileset will be received and failures at the target clients may not be reported.
Suggested value: Reliable

Reliable multicast transfers

Fallback Threshold
It may be ineffective to continue multicasting packets that are consistently dropped before being saved at the target client. In this case, the source clients can fallback to non-multicast (TCP/IP) for the

remaining data blocks. During confirmation, the source client evaluates the number of outstanding data blocks (packets) that the target client has received since the last confirmation. If the percentage is less than this value, the source client will begin fallback data transmission for the outstanding data.
Suggested value: `20`

`Confirmation Channels`
The source client can perform multiple simultaneous confirmations with multiple target clients. Each of these confirmations uses a "channel" consisting of one system thread and one socket. Source client resources are limited. In particular, the source client may be limited in the number of simultaneous open sockets or the number of available threads. If either of these numbers is less than the number of target clients, the source client should be restricted to less than the number available.
Suggested value: `2`

`Overlapped`
The confirmations may be performed while simultaneously `multicasting` data. When the number of target clients is much larger than the number of confirmation channels, this option may increase overall performance on some networks. Note that some `multicast` data will be dropped because of the higher priority TCP/IP traffic used by the confirmation channels.
Suggested value: `no`

`Announced`
This is required for reliable transfers.
Suggested value: `announced`

`Announcement Channels`
Similar to confirmation channels, announcement channels are used by the source client to set up the `multicast` transfer with the target clients.
Suggested value: `2`

`Default Base Address / Use Base Address`
The NFM server will select a `multicast` address for the transmission. The address will be selected from the `multicast` pool (`224.0.1.1` through `239.255.255.254`). The next available address (not already in use by the NFM server) from the specified base address will be used. The default base address is `224.0.1.1`. A specific base address may be specified to avoid conflicts with other IP `multicast` applications or to facilitate network diagnostics.
Suggested value: `default base address`

Unreliable multicast transfers

Unreliable transfers may be announced. See the Announcement Channels and Base Address sections above.
Suggested value: `announced`

Unreliable, unannounced transfers may be used to send data to non-NFM applications.

Default Multicast Address / Use Multicast Address
The NFM server will assign a multicast address for the transmission. The address will be either the default (224.0.1.1) or the value specified. If this address is in use by another NFM multicast transmission, the transfer will fail.

Raw Data Only
When selected, this option will cause the source client to send only the content of the data files without any information about file names or permissions.

Notes This window can be used to store text information specific to this Multicast profile.

Buttons

The following buttons are available from the Multicast Profile Detail screen:

Summary	Takes the session directly to the Multicast Profile Summary screen.
Apply	Applies all changes made on this screen. This will cause an add or update to occur, depending on whether or not the profile record preexisted.
Cancel	Cancels all changes on this screen since the last time this profile record was displayed or applied.
Delete	Deletes the currently displayed profile record.

Initial setup

The following steps should be performed during the initial configuration of the system:

1. The initial user root on the system should define an administrator type user record with a password for them and re-logon to the system using it. After making sure other valid administrator users can log on, the user root can be deleted.

2. A model should be added that reflects the computer running the NFM server software. After choosing a name, set the transfer method to SOCKETS and make sure the NFM Client box is checked on.

3. Create a node that represents the NFM server computer. This node should reference the model created in Step 2. The Primary communications name should be set to the IP host name or address. You should now be able to communicate with the NFM client software running on the NFM server computer. Using the browse function on an unspecified `fileset` to browse the client node can test this.

4. On the `System settings` screen, update the `NFM Server Node Name` field to reflect the system node record (it should be the only entry in the drop down box).

At this point you should now be able to add the various other models and node records that will define your environment. Node group, filesets and plans may then be created.

The next chapter "Operations" will step you through the process of scheduling plans to actually run as well as monitoring and controlling them. Two additional configuration type items `Day schedule` and `Hour schedule` are also included in this chapter as they are more easily explained after understanding the scheduling process.

Servers

NFM server configuration records are required when running in a multi-server environment, where the servers will be running collectively to form a virtual server running across multiple computers. This is described in detail in section "NFM multiple server support" in Chapter 6, "Advanced Functions." Please read that section thoroughly before making any configuration changes here.

When running a single NFM server, or when using multiple NFM servers that are operating independently, it is not necessary to change the configuration using this section.

`Servers` may be accessed from the `Management` drop down menu. The following information can be configured:

`Server Name` This is a user selected name for an NFM server. Type in a new entry, or select an existing entry from the drop down list. A default entry with the name "SERVER01" should already be present. This entry which represents the current primary server is created automatically at install time.

`Communications Name` This field should be set to the IP address or DNS name of the corresponding server. This field is required for other servers to be able to communicate with this server.

`Machine ID` This display only field should reflect the machine ID of the corresponding server. This field should be set automatically once the NFM server on the computer has started and detected a proper configuration.

Backup Server This field designates a backup server to this server definition. In the event that the assigned server fails to start any scheduled plans within a specified time limit, the backup server will automatically do so instead (running the plans on itself). For more information on Backup Servers, please see section "Backup Servers" in Chapter 6, "Advanced Functions."

Takeover delay (minutes) This field specifies how long the backup server waits, on a per plan instance basis, before assuming the responsibility of running a plan that the designated server failed to start. The backup server will attempt to factor in the difference, if any, between the clocks on the two computers. Example:

- Server A is set to 8:00 p.m.
- Server B is set to 8:03 p.m.
- Server B is a backup to A with a take over delay of 5 minutes.
- Plans update stores instance 104 is scheduled to run at 8:00 p.m.

If Server A never starts the plan, Server B would start the plan at 8:08 p.m. its time (or 8:05 p.m. Server A's time). If Server B cannot fetch the current time from Server A, it will start the plan at its time of 8:05 p.m. (or 8:02 p.m. Server A's time). It is therefore a good idea to keep the clocks in sync, and also, to set the take over delay long enough to be greater than any skew between the clocks.

Usergroups This display panel shows all of the usergroups that currently have the field Home Server set to this server record name. (See section "Usergroups" earlier in this chapter.)

Non-members Lists all the nodes and node groups that are not currently a member of the selected group.

Buttons

The following buttons are available from the Server screen:

Summary	Takes the session directly to the Server Summary screen.
Apply	Applies all changes made on this screen. This will cause an add or update to occur, depending on whether or not the Server pre-existed.
Cancel	Cancels all changes on this screen since the last time this Server record was displayed or applied.
New	Clears all of the fields to allow for the creation of a new Server record.
Delete	Deletes the currently displayed Server record.

Watch Files

Overview

The NFM watch file feature allows files to be transferred automatically as needed, rather than at pre-scheduled times. This is done by using a file discovery mechanism on the source node, which automatically invokes file transfers when the appropriate files appear in the source location. The transfers themselves rely on the conventional running of a plan. This allows the process to be customized as desired by the user.

This feature eliminates the need for the customer to repeatedly run plans to `poll` for data which can create large amounts of plan activity (audits, etc.) when there may be nothing going on. It also provides the ability to transfer files very quickly once they are available.

NOTE: The Watch File feature on the mainframe is currently restricted to the HFS file system.

`Servers` may be accessed from the `Management` drop down menu. The following information can be configured:

Configuration

The following steps are required to implement this new feature:

1. Create the `fileset` containing the file names or wildcard expressions to `watch` for on the source node or nodes. Any files matching any entries within the `fileset` will trigger the watch file action. Since the transfer will attempt to copy all files specified in the `fileset`, a wildcard expression is generally used. Refer to Chapter 4, "Configuration;" section `FILESETS` for specific information related to the creation of a `fileset`.

2. Create a node group that contains all of the desired source nodes on which file watching will operate. This must be done even if only one node will be used as the source.

3. Create a plan to copy the `fileset` from the source node to the desired target node. It is recommended that the plan be configured to run directly when the file trigger occurs. An alternative to running the plan directly is to configure an NFM remote server interface command to be invoked instead. This alternative approach is discussed at the end of this section. The plan should have the following characteristics

- At a minimum, the plan should include a function that transfers the files from the source node (where the file trigger occurred) to a target node. Although a watch file configuration may be used to watch any number of source nodes, via the node group, it will run a separate instance of the specified plan as needed for each node.

- It is not necessary to put a specific node name in the source node field for the transfer function. You may instead use the special designation `$CTLNODE`. This value if found, is automatically substituted with the actual node name of the source node that triggered this plan instance. This allows the creation of a generic plan that can be used for any number of nodes.

- It is recommended that the plan transfer function make use of the same `fileset` that is specified in the watch file configuration, however this is not required. Another `fileset` could be used if, for example, additional files need to be sent during the transfer, that are not included in the trigger file list.

- It is very important to delete the source files after transferring them. This is best accomplished by checking the field "`Delete source file after completion`" in the plan function. If the trigger files are not deleted after transferring them, the trigger will continually be reset, which will cause the plan to run repeatedly.

The plan that is triggered by the appearance of a file is not required to transfer the file; this is just a likely scenario. The trigger mechanism can be used to invoke any plan. It is however important to make sure that the plan removes the trigger file at some point before its completion.

Refer to Chapter 4, "Configuration," in section "Plans" for more information on creating a plan.

4. Create the watch file configuration (described in detail in the following section). Once created, the `watchfile` mechanism will automatically activate (unless the disabled checkbox is set). All of the source nodes will be contacted and configured to begin watching for the files. It is therefore recommended that the initial node group contain a single test node to make certain it is working as expected. Additional nodes may later be added to the node group.

Watch file configurations may be accessed from the `Management` drop down menu. The `Watch File Detail` screen allows viewing or changing of individual watch file records.

`Watch File` This is a user selected name for a watch file configuration. Type in a new entry, or select an existing entry from the drop down list.

`Fileset Name` This field should be set to `fileset` whose entries will be `watched` for on the specified nodes. The `fileset` may contain fixed names or wildcard expressions.

Use Plan Check this field and select a plan to invoke when a file trigger occurs on one of the designated nodes.

Use NFMI Command Alternatively to using a plan, check this field and type in an NFMI command to use instead when a file trigger occurs. See an additional description of this option later in this section.

Poll Interval & Static Interval These fields specify the discovery checking (or polling) interval and static interval. Both fields are in seconds and default to 10. See the section below for a detailed description of these fields.

Disabled Checking this field keeps this watch file from becoming active.

Buttons

The following buttons are available from the Watch File Detail screen:

Summary	Takes the session directly to the Watch File Summary screen.
Apply	Applies all changes made on this screen. This will cause an add or update to occur, depending on whether or not the Watch File pre-existed.
Cancel	Cancels all changes on this screen since the last time this Watch File record was displayed or applied.
New	Clears all of the fields to allow for the creation of a new Watch File record.
Delete	Deletes the currently displayed Watch File record.

More on Watch File processing

Once a watch file configuration is properly defined it will send the appropriate configuration, including the list of files, to each of the designated nodes. The NFM client on each node will automatically scan for files matching the list every # seconds (where # is the polling interval). Once the NFM client determines that there are any matching files, it waits for an additional # seconds (the static interval) and rechecks to see if the list of matching files has changed. If it determines that anything has changed, it will continue waiting for the static interval. The file trigger will not occur until the list of matching files remains unchanged for the full amount of the static interval. This helps prevent a transfer from occurring while the files are still actively being changed.

The list of files will be considered changed if any of the following conditions is detected:

- There are new matching files that were not present during the last check.
- Files that were detected previously are no longer present.
- The size of a previously detected file has changed.
- The time stamp of a previously detected file has changed.

If it is possible to control the manner in which the files are created, certain procedures will also help prevent the premature transfer of a file:

- Have the file created and written to under a different name that does not match entries on the trigger list. Once the file is complete, rename it to its final name.
- Similarly, have the file created in a different directory on the same device and then copied to its final location for pick up.
- Set the trigger to a secondary, dummy file name. Create the dummy file after the actual file or files are finished. Remove the dummy file as part of the transfer processing.

Monitoring Watch File activity

Since plans are used to perform the transfers, the plan instances will automatically appear in the "`Plan Activity Screen`" as they become active.

If an error in the configuration or other condition prevents this from happening, audit messages will be created. These are viewable from the "`Audit Trail`" (or `History`) screen. Watch for these to appear if expected plan instances do not get created. Keep in mind that with the default interval values, it may take more than 20 seconds for a transfer to be attempted after a matching file is placed in the source directory.

Using an NFMI command instead of a plan

As specified in the watch file configuration, an `NFMI` command may be invoked instead of a plan when a file trigger occurs. This command will most likely perform an `nfm callplan` to initiate a file transfer of the `fileset`, but it may also take full advantage of the NFM command line interface, which may allow some options to be used, that might not otherwise be available from just running a particular plan.

To us this option, you must first configure the remote command. See Chapter 8, "Tools," section "NFM remote server interface" for a full description of setting up a remote command. Build a command as if it were being issued directly from one of the watch file nodes. The command will be issued automatically, on behalf of this node, when the file trigger occurs.

Configuring watch files at the end node

Everything in this section so far describes configuring the watch file feature on the NFM server using the user interface (GUI). The necessary information is automatically sent to the NFM clients on the appropriate nodes and stored locally in text configuration files.

It is also possible to set the configuration on an individual node directly on the NFM client computer, which basically involves editing the appropriate text files and then restarting the NFM client to begin watch file processing. This might be necessary under the following conditions:

- The node is configured to us a call-in type connection (`CONNECT=CALL`).
- The `DEMAND` field is set to `Y`.
- The `CHECKIN` field from the client configuration file is set to `N`.

This combination would mean that client node cannot be contacted directly from the NFM server and the node is not configured to check in to the NFM server at startup. (The `CHECKIN` option being enabled would allow the automatic download of a server configured watch file configuration). This sort of setup might be deployed, if it is desirable that the NFM client never tries to contact the NFM server, until such time that a watch file condition is triggered. For more information on the client configuration file options, see Chapter 6, "Advanced Functions," section "Encryption and connection options."

The text file `watchfiles.cfg` in the NFM client home directory contains each of the watch file configurations, active for this node, in a separate stanza that contains basically the same data combined from a server watch file record along with the file names specified in the `fileset` definition. Create a locally defined entry by copying a stanza from an automated entry and making the following changes:

- Assign a new name for the first field `WATCHFILES=`*`Name`*`.` The name must not match an existing `watch file` record that might later be assigned to this node or the stanza may be over-written by the server one of the same name.

- Set the next field to `AUTOENTRY=N.` This marks the stanza as being locally defined and it will not be affected by any updates from the server. Any stanzas that do not have this setting will be subject to automatic deletion if updated configurations are sent down from the NFM server.

NFM Client Configuration

Overview

NFM keeps the majority of its configuration stored on the NFM server, with most of the data being stored in its database, including everything previously described in this chapter. This provides a centralized approach for deploying and maintaining the system.

Some of the configuration items, however, must reside at the end nodes themselves. This includes the following types of information:

- Parameters necessary to establish communications from (or to) the NFM server. This could include what port to listen to for IP connections, or whether or not to use SSL, few examples. These types of items are usually required to match a definition stored in the nodes corresponding NODE or MODEL record at the NFM server.
- Security items that need to be locally administered, like a node password, or an SSL private key file.
- Certain other configuration that may be desirable to offer local configuration, including watch file lists and FTP server definitions.

Most of this information is stored in a text editable configuration file stored on the NFM client computer.

NFM client configuration file

The NFM client configuration file, usually `nfm.cfg`, resides in the NFM client's home (or install) directory. An initial configuration file is created automatically when the NFM client is installed and looks approximately like this:

```
SERVADDR=
SERVNAME=nfmserver
GATEWAY=
NFMNODE=unknown
MAXTHREADS=10
PARMS=
CHECKIN=N
HTTP_TUNNEL=N

CONNECT=LISTEN, PORT=8008, INTERVAL=0, DURATION=60, TIMEOUT=5, SSL=N, KEY=,
CERTIFICATE=, PASSWORDFILE=, VERIFYPEER=N,
```

There are two general types of line entries:

1. Lines that contain a single NAME=VALUE pair usually define a setting that has global meaning.
2. Lines that contain a series of NAME=VALUE pairs, and usually begin with CONNECT=*connection_type* define a connection entries. A connection entry usually defines a means by which a connection is made to or from this node. Each of the individual settings in the line only affects the current connection entry. The currently available connection types are `LISTEN`, and `CALL`, and must be set to match the node or model connection entries defined on the NFM server.

Note: Some of the global settings, may also be specified (using the same name) on a per connection entry. In such a case, the global setting would provide a default value for all connection entries, unless a more specific, connection entry value was supplied that would override the default setting. These are documented where available.

The configuration file may be edited as needed. The NFM client must be stopped and restarted before a configuration change will take effect.

The default configuration file establishes a single connection entry that listens for incoming connections on the default port (8008). If the default port is suitable and the node may be connected to from the NFM server and other nodes across the network using this port, then there would be no need to have to change the configuration file.

There are numerous references in the following section to "NFMI or check-in processing". This refers to the use of the remote command line interface (the NFMI program) and the CHECKIN option (described below), both of which require the NFM client to make a connection to the NFM server.

Global settings

The following global settings are available for the NFM client configuration file:

SERVADDR=[IP address or hostname] This field should be set to the IP address or hostname that the NFM server is running on. It is only necessary if NFMI or check-in type processing is required. If this field is set it indicates that the NFM server is directly accessible for such purposes (without having to use the proxy).

SERVNAME=[server name] This field should be set to the name of the NFM server. It is only necessary if NFMI or check-in type processing is required, and if the NFM server is not directly accessible. In this situation the field GATEWAY should also be set.

GATEWAY=[IP address or hostname] This field should be set to the IP address or hostname that the NFM proxy program is running on. It is only necessary if NFMI or check-in type processing is required, or as a way to indicate the location for "call-in" type nodes to reach the NFM proxy. If the SERVADDR field is blank, then this field along with SERVNAME is used by NFMI or check-in processing to reach the NFM server. If SERVADDR is set and this field is set, than SERVADDR is used for NFMI and check-in, and this field is used for "call-in node" type nodes.

Note: The per-connection settings for GATEWAY and NODENAME (below) may alternately be used for each "call-in" type connection. This would accommodate having a single NFM client being reachable from two different NFM servers if desired.

NFMNODE=[name] This field should be set to the NFM node name that represents this node. This field is only for a "call-in" type node in conjunction with the GATEWAY field to indicate communications using the NFM proxy.

MAXTHREADS=[#] This field indicates how many threads should be created for thread pooling. Thread pooling is enabled by default and can dramatically improve the efficiency of the NFM client when dealing

with a large number of simultaneous connections. Setting this value to 0 turns off thread pooling. It is unlikely that this field will need to be changed.

PARMS=[command line arguments] Any fields configured for this setting will cause the NFM client to start as if these fields were typed in following the NFM command (no quoting of multiple values is necessary). The NFM client program can be started in a stand-alone manner with the help option to get a list of start up arguments:

```
nfmclient -h
```

It is unlikely that it will ever be necessary to use this option unless being directed to do so by a support person.

CHECKIN=[Y/N] This field indicates whether or not to perform check-in processing when the NFM client is first started. This option is used primarily for watch file processing with a call-in type node. This is discussed in more detail in the previous section "Watch Files" earlier in this chapter. This option must also be enabled to invoke an automatic plan or command on the server when the node first starts. See the fields "Automatic Plan" and "Automatic Command" in the node or model configuration sections.

HTTP_TUNNEL=[Y/N] This field enables the HTTP tunnel service system wide for this computer. This service must be enabled for any NFM programs (server or client) on this computer that wish to use the HTTP tunnel service. Other configuration items determine whether or not individual connections will use the tunnel, this setting simply makes it available.

HTTP_TUNNEL_PORT=[#] Use this field to designate a particular port to be used for the HTTP tunnel service. The default port number 8113 will be used if this field is not set, or is set to 0.

Connection entries

The following settings define connection entries used by the NFM client to facilitate communications to and from the NFM server and other NFM clients. These entries should largely match the connection entries along with certain other options defined in the NFM node or model definition for this node. The following fields are available:

CONNECT=[type] This field should be set to the required connection type. One of the following:

LISTEN This defines a basic listen type connection entry that accepts incoming socket connections. This should be used in conjunction with a client type node (a node that has its "Primary Transfer Method" set to "SOCKETS" or "NFM/HTTP"), and has its connection entry set to Type=Listen, in the node or model record.

CALL This defines a "call-in" type connection that establishes a socket connection in to a NFM proxy program running on another computer. This should be used in conjunction with a client type node that has its connection entry set to Type=Call. Please reference the next section for a complete description of the NFM proxy program and call-in type nodes.

PROXY This entry enables the NFM client to also server as an NFM proxy. Please reference the next section for a complete description of the NFM proxy program and call-in type nodes.

PORT=[#] This field should be set to desired port for this connection. It may be left blank to default to `8008` for non-SSL connections and `8009` for SSL connections. The connection entry

HTTP_TUNNEL=[Y/N] This field indicates whether or not this particular connection should use the HTTP tunnel protocol to communicate with the NFM server or with the NFM proxy. It is only necessary for a check-in type connection. The handling of incoming connections is not aware of the use of the HTTP tunnel since it connects to a normal listen connection.

SSL=[Y/N] Indicates if this is an SSL connection.

Note: The remaining fields are all used for SSL type connections only.

KEY=[file name] This PEM file contains the private encryption key that is used to secure communication connections. If the file name provided does not specify a full path, the file must exist in the "`ssl`" subdirectory of the NFM client home directory. If this field is left blank the file "`NFMPrivateKey.pem`" is used by default.

CERTIFICATE=[file name] This PEM file contains the certificate that will be sent to the remote system for authentication. If this field is blank the file "`NFMCertificate.pem`" is used by default.

ROOTCERTS=[file name] This PEM file can contain any additional root certificates and/or any certificates to use with the "Whitelist" option.

VERIFYPEER=[Y/N] This field indicates whether or not the SSL will verify the remote system's certificate.

SELFSIGNEDCERTS=[Y/N] This field indicates whether or not to allow self signed certificates. The default is to not allow "N".

WHITELIST=[Y/N] This field determines if the Root Certificates file will also be used as a "White List" for the incoming peer certificates.

PASSWORDFILE=[file name] This specifies a file containing the password used to access the "Private Key" file if it requires one. This password file is created using the "nfmi –sslpassword" command.

NFM proxy program

The NFM proxy program provides connectivity to NFM clients and other entities that may not be otherwise accessible via a direct TCP/IP connection. This is typically because of one or more of the following factors:

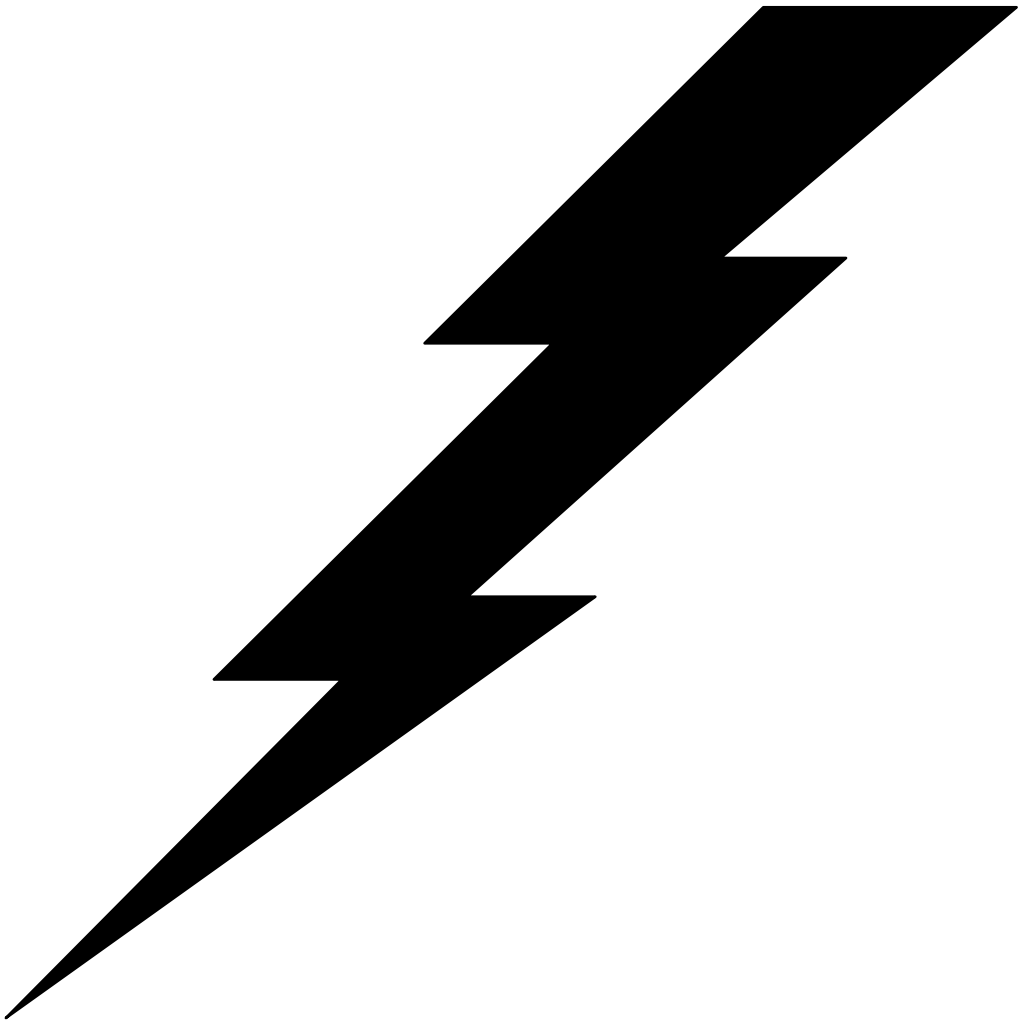
- An NFM client is running behind a firewall.
- An NFM client is running on a computer that does not generally have a fixed IP address (this would be typical for the Android or iPad client for example).

These NFM clients should be configured to make a "call-in" type connection whereby they connect in to the NFM proxy program (typically running on the NFM server computer). The NFM server and possibly other NFM clients can then connect to these clients by connecting to the NFM proxy program creating a virtual end to end connection with the call-in type nodes.

Although the NFM proxy program is referred to as a separate program, it is now actually built in to the NFM client program. This means that any NFM client program can also serve as an NFM proxy by adding a connection entry (`CONNECT=PROXY`) as described above. A typical customer setup would be to have the NFM client running on the NFM server to be enabled in this way. A call-in type node would connect to this proxy and the NFM server would connect to the proxy on the local loop back to reach the nodes.

It is however possible that the NFM server itself is also behind a firewall. Such a setup would require running the NFM proxy on a different computer that all computers can reach. This would require the following steps:

- Install a copy of the NFM client on the "gateway" computer. Configure it with the proxy connection entry. Remove the other connection entries if no additional NFM activity is required on this computer.
- On the call-in NFM clients, configure the appropriate fields to enable connectivity to the NFM proxy. This includes the call-in connection entry as well as certain other fields referenced in the previous section.
- On the NFM server, configure the gateway field of the "call-in" node or model connection entries with the IP address of the gateway computer.



5: Operations

Overview

The following chapter discusses operations using the NFM user interface. This consists of scheduling a plan to run and monitoring and controlling plans. A section describing the viewing of audit records (History) follows this. There is also a section describing the configuration and use of custom schedule tables (See "day schedule" and "hour schedule").

Scheduling

This Plan Scheduler screen may be accessed from the Scheduling menu. This screen will allow you to schedule one or more 'instances' of a plan to actually run. After you have entered the appropriate information, click on the Schedule or Run Now button to submit a plan.

Plan Name Enter the desired plan name.

Source and Target Node(s) These fields will override the default source and target node(s) fields specified in the Plan header. If there is no source or target node(s) fields in the Plan header, and the Plan function do not explicitly define the nodes to work with, one or both of these fields will be required at this point.

Initial State This sets the initial state of the plan. The default value of Ready is usually used, but the plan may be submitted in the Hold state that means a user must manually resume it. This is described further in controlling plans.

History This field is used to optionally override the corresponding history field in the Plan header. By specifying Default the Plan header history value is used. (See "Plans" for a description of the history field.)

Day Schedule This specifies a customized calendar that is used in conjunction with automatically rescheduling the plan at predefined intervals. This is described further in the section Day Schedules later in this chapter.

Hour Schedule This specifies a customized hour table that is used in conjunction with automatically rescheduling the plan at predefined intervals. This is described further in the section Hour Schedules later in this chapter.

Repeat Every # [interval] These two fields instruct the scheduler to automatically reschedule this plan on a given `interval`. The `interval` can range from seconds to years. Once the plan is submitted, it will only show once in the submitted plan list. Each time the plan is through running it will resubmit itself on the appropriate time interval. When a more elaborate schedule is needed, day and hour tables may be specified.

.

Use the calendar and `Start time` boxes on the right portion of the screen to indicate the initial scheduled time for the plan to run. If a plan is setup to restart automatically on a given interval, this interval will be based on the initial start time. Once the plan is submitted a unique numerical `instance ID` is assigned to the plan. Any automatic resubmission of the plan will also generate a unique ID.

Buttons

The following buttons are available on the `Plan Scheduler` screen:

Schedule	Schedule the plan for execution.
Run Now	Run the plan immediately. The calendar and start time fields are ignored.
Monitor	Takes the session directly to the `Plan monitor` screen and displays the plan instance just submitted. In order to view a different plan instance, go to the `Plan Activity` screen and double click on an entry from the list.
Clear	Resets the fields on the screen to their default values including copying in the current date and time to the `Start` boxes.

Plan Monitor

This `Plan Monitor` screen is not directly accessible from the pull down menu but is reached in one of two ways. Once a plan is submitted from the plan scheduler, the user may click on the monitor button to go directly to this screen and display that plan instance. Otherwise, any user may go to the `Plan Activity` screen, where all submitted plans are listed, and double click on an entry to bring it up on the `Plan Monitor` screen. See the following section for more on the `Plan Activity` screen.

This screen shows the current status of an individual plan instance. This screen always displays the instance ID, plan name and current status, along with the plans scheduled start time. The remaining contents of the screen are divided into

folders that display different types of information about the plan instance. The initially selected folder when changing to this screen will depend upon the current state of the plan (`scheduled`, `active` or `finished`). The folders will automatically switch when the plan changes state. The user may manually change to any of the following folders:

Header The `Header` folder contains a summary of the information that was entered from the plan scheduler. If the plan has not started yet, a countdown is displayed at the top of the folder.

Live The `Live` folder is empty unless the plan is currently running. If it is running this folder displays a list of all the individual activities ongoing, including the nodes, file transfers and executions in progress, file names, byte counts, percentage completed graphs, etc.

Audit The `Audit` folder looks similar to the `Audit Trail` screen (discussed later). This displays the entire audit records currently generated from this plan instance.

Legend The `Legend` folder is for help only. It displays a visual depiction and description for many of the symbols and color codes that are displayed on the `Plan Monitor` screen.

Buttons The following buttons may be available on the `Plan Monitor` screen, based on the state of the plan instance:

Apply This button in conjunction with the refresh rate box to the left may be used to change the refresh rate of this screen. This button is always available.

Hold This button puts a scheduled or active plan on hold. If the plan has not started yet, nothing will occur when its start time is reached. If the plan is active, each of the activities displayed will stop what they are doing and wait. The nodes will change color from green to light blue to dark blue. Light blue indicates a hold is pending while dark blue means it has reached a held state. It may take time before the individual node processes reach a held state if they are in the middle of program execution or certain types of non-client file transfers (an FTP transfer cannot be held in the middle of a file transfer).

Release This button `releases` a plan that was previously put on hold, or was submitted initially in the held state. This button does not free up an individually held node (see below).

Nodes This button brings up the `Node control` window that is used to provide control over individual nodes.

Node Control window

The Node control window provides the ability to hold, resume and exclude individual nodes from the current plan instance. This window is available before and during the plans execution. One or more nodes should be highlighted from the list and then the following buttons are available:

Enable — This button will resume a held node or it will enable a previously excluded node (see below). Also if the entire plan instance was put on hold (see above) the individual node is still enabled. Note, a file transfer must have the source and target node both enabled before copying will continue.

Hold — This button puts the selected nodes on hold.

Exclude — This button excludes the selected nodes from any further activity in this plan instance. This is similar to the node being defined as off-line in the node record, but it only applies to this particular plan instance.

Close — Close the Node control window.

Plan Activity

The Plan Activity screen may be accessed from the Scheduling drop down menu. This screen is essentially the summary screen for all plan instances. Various information about each instance is listed along with its current state and return code (if finished). Double clicking on an entry at any time, will transfer control to the Plan Monitor screen for the selected plan instance. Like the Plan Monitor screen, this screen automatically refreshes itself to display changing plan information.

Buttons

Most of the buttons on this screen affect the highlighted node entries. Changes may be performed on multiple entries by selecting more than one (shift-click, control-click) and pressing the appropriate buttons. The availability of different actions will be dependent upon the states of the selected plan instances:

Details — Takes the session directly to the Plan Monitor screen to display the first selected entry. This is the same as double clicking on an entry.

Node Control — This brings up the Node Control window discussed in the previous section for controlling individual nodes in the first selected entry.

Cancel — Cancels the currently selected plan entries.

Delete	Remove the selected plan entries from the list, also deletes any logs or temporary files associated with the plan instance. This does not delete audit records generated by this plan.
Hold	Puts the selected plan entries on hold. This has the same effect as clicking the hold button from the `Plan Monitor` screen.
Resume	Resumes previously held, cancelled, or interrupted plan entries.
Retry	Restarts the same plan instance from the beginning and attempts to perform only those actions that failed previously.
Display Options	Displays a window that allows you to change the fields displayed on this screen.
Usergroups	This provides a set of checkboxes to filter out plan instances based upon which usergroup they are a part of.
Legend	This displays a pop-up window that describes the various symbols and fields displayed on this screen.

Day Schedule

The `Day Schedule` screen may be accessed from the `Scheduling` drop down menu. This screen maintains custom calendars that tailor the exact days to run plans that are to be automatically repeated. Once created, the schedule is specified in the `Day Schedule` field on the plan scheduler to associate it with a given plan.

If a day schedule is specified during plan submission, and the `Repeat every` fields are left blank. This has the same effect as if the `Repeat every` field had been set to `Repeat every 1 day(s)'`. This would cause the plan to run at the same time of day as it was originally scheduled for, only on the days that are checked on in the `day schedule` record. If the `Repeat every` fields are set to some other value, then the time interval that is implied would only occur on those days checked on in the day schedule. This is much easier explained by example:

Suppose a day schedule was created that had every Monday through Friday checked on for a given year, but had holidays unchecked (in other words it reflected workdays only). A plan is submitted with this schedule and the `Repeat every` field indicated every `1`-hour. The result would be that the

plan would be run once every hour from its originally scheduled time, but only on workdays.

A day schedule such as the one in the example, could be shared by any number of different plans, since the day schedule is a separate entity from the original starting time and `Repeat every` fields used for separate plans.

The fields on this screen have the following characteristics:

Name This is a user selected `name` which defines this day schedule. Type in a new entry, or select an existing entry

Year Select the `year` being configured.

The following actions may now be performed:

- Clicking on individual days will reverse their current status, turning them on initially or turning them off if they were already on.
- Clicking on a day of the week within a month will reverse all of those days for that month.
- Clicking on a month title, will reverse all the entries within the month.
- Clicking on a day of the week box at the top of the screen will reverse the status of all of those days for the entire year.

Buttons

The following buttons are available on the `Day schedule` screen:

Apply	`Applies` all changes made on this screen. This will cause an add or an update to occur, depending on whether or not the day schedule preexisted
Cancel	`Cancels` all changes on this screen since the last time this day schedule record was displayed or applied.
New	Clears all the fields on the screen for creating a `new` entry.
Delete	`Deletes` the currently displayed day schedule record.
Notes	Displays a pop-up text window for storing `notes` in.

Hour Schedule

The `Hour Schedule` screen may be accessed from the `Scheduling` drop down menu. This screen maintains custom schedulers like the day schedule but for use on an hourly basis instead. Once created, the schedule is specified in the `Hour Schedule` field on the plan scheduler to associate it with a given plan.

If an Hour Schedule is specified during plan submission, and the Repeat every fields are left blank. This has the same effect as if the Repeat every field had been set to Repeat every 1 hour(s). This would cause the plan to run at the same offset within the hour it was originally scheduled for, but only during those hours of the day checked on in the Hour Schedule record. If the Repeat every fields are set to some other value, then the time interval that is implied would only occur on those hours checked on in the day schedule. In addition, this hour table can be combined with the day table to customize scheduling even further. If we modify the example from the day schedule a little further:

> Suppose we submit a plan with the previously created day schedule that represented workdays, and we also include an hour schedule that has 9:00 a.m. to 5:00 p.m. checked on. We also set the Repeat every field to run the plan every 10 minutes. The result would be that the plan would be run once every 10 minutes, but only between 9:00 a.m. and 5:00 p.m., and only on workdays.

A day schedule such as the one in the example could be shared by any number of different plans, since the day schedule is a separate entity from the original starting time and Repeat every field used for separate plans.

Only the name field needs be typed in on this screen, then the appropriate hour boxes can be checked on or off

Buttons

The following buttons are available on the Hour Schedule screen:

Apply	Applies all changes made on this screen. This will cause an add or an update to occur, depending on whether or not the hour schedule preexisted.
Cancel	Cancels all changes on this screen since the last time this hour schedule record was displayed or applied.
Delete	Deletes the currently displayed hour schedule record.
Notes	Displays a pop-up text window for storing notes in.

History

Audit trail

Audit messages may be created in a variety of ways. The majority of audit messages will be generated from plan activity. The amount of detail that will be included will depend upon the configuration of the history field on a per-plan

basis. Each audit message will be one of three types, which includes errors, warnings, and informational audits. Audits are also generated during configuration changes if this option is enabled in the `System Settings` screen. In addition, the NFM command line interface (described in Chapter 8, "Tools") describes how custom audit messages may be created by the users as part of a plan or in a stand-alone manner.

The `Audit trail` screen is used to filter and view audit messages. Unlike the `Plan Monitor Audit` folder which views only a given plan instance's audits, this screen allows viewing all of the audit records, or to display a subset of them based on a particular filtering criteria. These filters include the following:

`Level` Enter the desired combination for the type of audits to display, including, `All`, `Warnings`, and `Errors.`

`State` Use this option to select only unread audits if desired. See the description below for double clicking (reading) an audit message.

`Search` Enter one of the additional search criteria to further limit the display of audits. Choose to select by `Node`, `Fileset`, `Plan`, `Instance ID` or use the `Find` field to search for audit messages containing a particular string of text.

`Date and Time range` Specify a starting or ending time (or both) to narrow the display of audits to a particular time interval.

The following fields also control the data being displayed:

`Display up to # entries` Enter a number representing the maximum amount of entries to display. If the number of entries that match the search criteria is greater than this number, a message will be displayed that indicates that there are additional older or newer messages that are not being displayed. If a starting date range is in use, than newer entries above the maximum display count will not be shown. If a starting date range is not used, older entries will not be shown.

`Location` This field allows the user to display audits from an alternate data source other than the current NFM database. This may be an archive location or any directory that contains audit tables copied from the database directory or perhaps from another NFM server. Alternate data sources may be added with the NFM command line interface (reference the sub-command `putasource,` in Chapter 8, "Tools"). Also, see the "Audit" tab in the `System Settings` screen for more information on automatically archiving audit records (described in Chapter 4, "System Settings").

Buttons

The following buttons are available from the `Node Detail` screen:

`Display` After entering the desired filters, click on this button to retrieve audits matching the search criteria.

Note, unlike the `Plan Monitor Audits` folder, which automatically refreshes as new audits are created, this screen requires you to click on `Display` because the output may be extremely large.

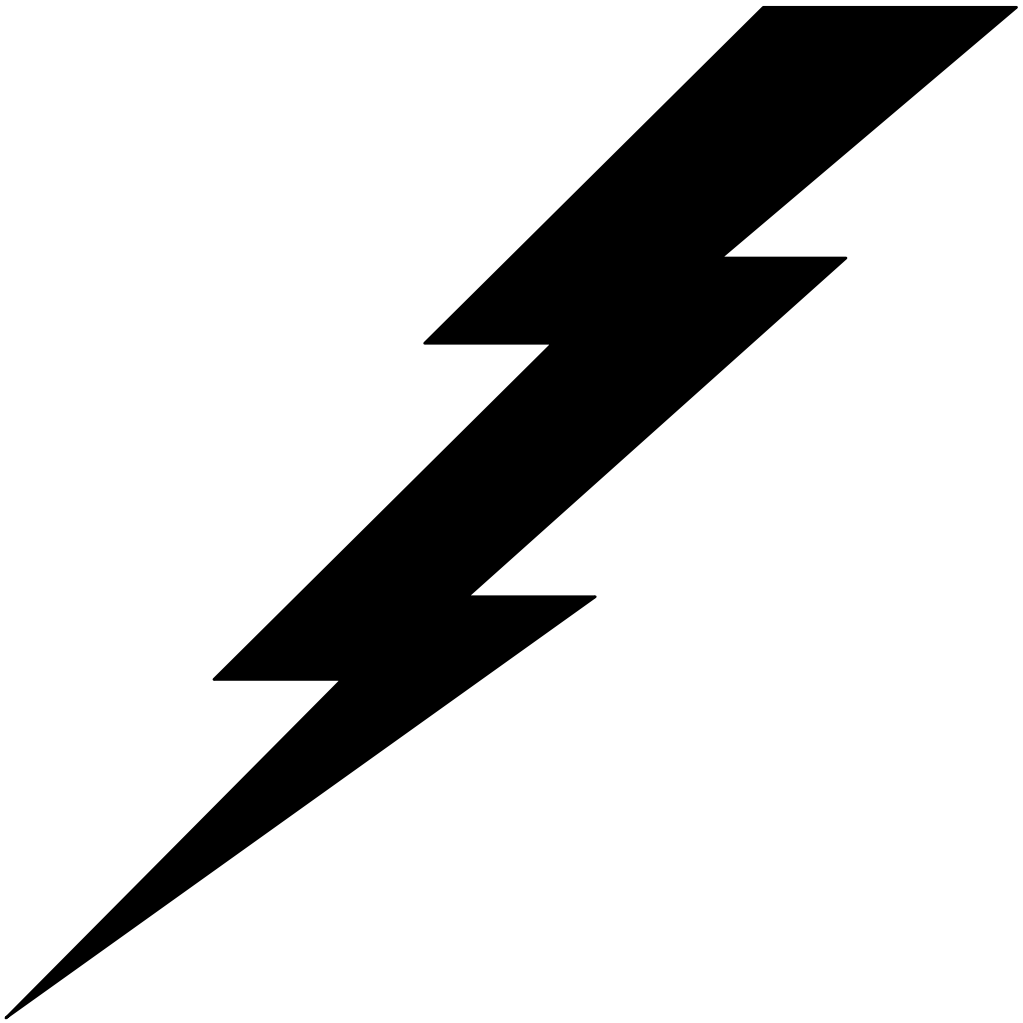
`Usergroups` This provides a set of checkboxes to filter out audits from the display based upon which usergroup they are part of.

`Double click on an audit message` This will bring up the audit message in a separate window, this may be done for any of the following reasons:

1. Additional information may be displayed for the audit including some return codes.

2. If this is a multiple line audit, the remaining lines will be displayed in the window. A multiple line audit can be recognized from the list because it has three trailing periods at the end of it. Multiple line audits are generated from output of executed commands in plans and on certain audit configuration changes.

3. Double clicking on an entry marks it as being read. Note only Warnings and Errors are marked initially as unread. This allows for an acknowledgment procedure (optional) to be implemented by support personnel.



6: Advanced Functions

Overview

This chapter describes several advanced topics related to NFM plans, and includes several examples that should aid the user in designing plans to accomplish certain goals. The following topics will be discussed:

- Using wildcard entries within filesets.
- Using environment variables within plans.
- Working with node groups.
- Plan execution flow.
- Error recovery.
- File streaming.
- Data encryption and compression.

Using wildcards

NFM supports using wildcard entries to represent file names within fileset definitions. This is done by specifying a wildcard pattern within any of the Source File fields while defining a fileset definition (as discussed in Chapter 4, "Configuration," section on "Filesets"). Wildcard patterns within NFM make use of a name containing some combination of regular characters, question marks '`?`', and asterisks '`*`'. These are interpreted similar to Windows and UNIX conventions, whereby question marks may substitute for any text character, and asterisks may substitute for any group of text characters (including no characters).

Two consecutive asterisks "`**`" have special significance. They may be used to invoke a recursive search into subdirectories to look for an expression match. Put another way, the directory separators can be considered to be part of the wildcard segment denoted by the double asterisks. So the expression "`/tmp/**.dat`" would pick up the files "`/tmp/inv.dat`" and "`/tmp/sub1/sub2/plans.dat`". Also, if a file transfer is used that specifies a wildcard of this type, the target node will automatically create missing subdirectories as needed, as long as the subdirectories are part of the file name and not part of the file prefix.

There are some specific rules that should be noted:

- Do not use a dot '`.`' as part of a filename, unless the file name actually contains a dot. NFM does not recognize the dot as being an implied part of a filename with or without an extension, the way Windows and 4690 does. Consider the following example:

The directory "`/tmp`" contains the following files:

```
file1
```

```
file1.dat
file2
file2.dat
```

Using "`*.*`" with Windows would imply all four files, the expression "`file1.*`" would imply the first two. NFM does not treat the dot any different than any other characters, therefore the proper alternative expressions would be "`*`" and "`file1*`" respectively.

- Wildcard entries may be used in the 'Target Files' fields, only when they are also used in the 'Source Files' fields. This allows copying the wildcard segment of the source name to be used in the target name.

- If wildcards are specified for an operation and no matching files are found, this is not considered to be an error by default. To cause NFM to generate an error under these circumstances, the source file should be prefaced with a plus character '+'. This character is only used to indicate this option; it is not treated as part of the file name. Example:

```
Source Prefix:      /tmp/
Source File:        +inventory.*
```

The above two fields would indicate all files matching "`/tmp/inventory.*`" and any operation that did not find any matching files will generate an error.

Using environment variables

The NFM environment variables enhance the functionality of the NFM system by providing a flexible way to pass information to functions of an NFM plan instance. There are two types of environment variables available, user configured and NFM provided. The user-configured variables consist of those variables defined on the model and node definitions. This provides a means of customizing filenames and executed commands so they can perform different operations based on the current node they are working with. The NFM provided environment variables are automatically generated during a plan execution. These variables are listed below.

Environment variables can be used in two different ways. Variables can be referenced in an executed command of plan function as well as prefix and filename fields that are referenced from within a `fileset`. This is done in the form `$(VARIABLE)` inserted anywhere in the text of these fields. In this case the substitution is done on the NFM server prior to the command or filenames being used during plan execution. All NFM environment variables are also passed down to the NFM client during a plan execute type function, and are available in local context as real operating system environment variables.

User configured environment variables

The user configured environment variables are defined on the `Node` and `Model detail` screens. An environment variable assigned to a model will be available for substitution to any nodes defined to use that model, unless the

same environment variable name is specified in the Node record where it will take precedence. Consider the following example:

I have 100 store nodes, half of which are Windows/95 computers, the other half Windows/NT. I have a software package that resides in `C:/myapp` on the 95 computers, but resides in `D:/myapp` on the NT computers. The exception to this is two of the NT computers; store099 and store100 have it installed in `F:/myapp`. By setting up the following environment variables, I can send a single file to every stores correct location with a single command.

```
Nodes STORE001 through STORE050 share the same model called WIN95.
Nodes STORE051 through STORE100 share the same model called WINNT.
The WIN95 model record has the environment variable MYAPP=C:/myapp.
The WINNT model record has the environment variable MYAPP=D:/myapp.
The node records STORE099 and STORE100 each have the environment variable
   MYAPP=F:/myapp.  (This overrides the default model value of D:/myapp for
   these two stores.)
Now define a fileset definition called MYAPPFILES with the following:
   Source File=/tmp/data.001
   Rename File=$(MYAPP)/data.001
Finally define a plan with a function that copies the following:
   Transfer MYAPPFILES from source_node CENTRAL to target_node ALLSTORES
(ALLSTORES is a node group that contains all 100 stores).
```

Variable substitution during plan execution will provide the desired resulting copies:

```
CENTRAL:/tmp/data.001 → STORE001:C:/myapp/data/data.001
. . .
CENTRAL:/tmp/data.001 → STORE050:C:/myapp/data/data.001
CENTRAL:/tmp/data.001 → STORE051:D:/myapp/data/data.001
. . .
CENTRAL:/tmp/data.001 → STORE098:D:/myapp/data/data.001
CENTRAL:/tmp/data.001 → STORE099:F:/myapp/data/data.001
CENTRAL:/tmp/data.001 → STORE100:F:/myapp/data/data.001
```

NFM provided environment variables

The following list represents the NFM provided environment variables. These, along with the user defined environment variables from the current source or target node (or their models), are available for substitutions during a plan execute function. When used as part of the command itself they should be referenced as above with the `$(VARIABLE)` convention. The variables are also sent down to the target client node and established as operating system environment variables, so they will be available from within a batch or script using the appropriate operating system syntax.

`NFMBIN`	The NFM server executables directory.
`NFMDATA`	The NFM server data directory.

NFMEXECS Current executable command string.
NFMFILEO Current file ordinal #.
NFMFNAME Current Source File name not including any directories (after any wildcard expansion).
NFMPASS Repeating phase current iteration #.
NFMPHASE Current phase name.
NFMPHSRC Highest function return code so far in current phase.
NFMPLAN Current plan name.
NFMPLNID Current plan instance ID number.
NFMPLID0 Plan instance ID number (5 digits, leading 0's).
NFMPROD The NFM server product directory.
NFMPRPRC Previous phase return code.
NFMRETRY Plan is in retry (0=no, 1=yes)
NFMSCOMM Current source node communications name.
NFMSERV The name of the node representing the server.
NFMSMODL Source node model name.
NFMSNAME Current Source File's base name (no 'Source Prefix') after expanding wildcard entries (see example below).
NFMSNAP Source NAP name.
NFMSNODE Source node name.
NFMSOPER Source node operating system name.
NFMSRVR The NFM server name.
NFMTCOMM Target node communications name.
NFMTMODL Target node model name.
NFMTNAP Target NAP name.
NFMTNODE Target node name.
NFMTOPER Target node operating system name.
NFMUGID The usergroup ID of the user who submitted this plan.
NFMUSER The name of the user who submitted this plan.
NFMUSERG The name of the usergroup of this plan.
NFMWILD Current source file's first wildcard segment of base name after expanding wildcard entries.
NFMWILD# Current source file's nth wildcard segment of base name after expanding wildcard entries.

(The remaining variables represent the date and time on the NFM server when the current function was started):

NFMTM_YR Year (4 digits).
NFMTM_Y2 Year (2 digits).
NFMTM_MO Month (2 digit, 01-12).
NFMTM_DY Day (2 digit, 01-31).
NFMTM_JD Julian date or day of year (3 digit, 001-366).
NFMTM_HR Hour (2 digit 01-12).
NFMTM_AP AM/PM indicator ("AM"or "PM").
NFMTM_HM Hour in military time (00 - 23).
NFMTM_MI Minutes (00 - 59).
NFMTM_SE Seconds (00 - 59).
NFMTM_BN Numeric time stamp.
NFMTM_WN Day of the week #.
NFMTM_WD Day of the week abbreviation.

The following example shows the use of the date and time environment variables:

I have a directory on a node called "/tmp" that I wish to backup once a day. I do not know ahead of time the names of the files so I want to select every file in the directory by using wildcards. I would like to rename each file by adding the current day of the year (Julian date) as a string appended to the end of each file name. By doing this I will create a collective backup whereby each new day that this plan is run I will be creating a different extension and therefore a new copy of each file in the target directory.

```
Nodes will be called STORE001 and STOREBACK.
Define a fileset definition called backfiles with the following:
   Source Prefix=/tmp/          Target Prefix=/tmp.back/
   Source File=*                Target File=$(NFMSNAME).$(NFMTM_JD)
Plan backplan will copy the fileset backfiles from STORE001 to STOREBACK.
```

For this example, let's say we have three files: file1, file2, and file3 in the "/tmp" directory currently. Let's also assume it is February 1st. The Source File specification of asterisks indicates to pick up each file in the directory. As each file is copied the target filename substitutes the $(NFMSNAME) with the source files base name, and substitutes the $(NFMTM_JD) with the current Julian day of year to cause the desired copying.

```
STORE001:/tmp/file1 → STOREBACK:/tmp.back/file1.032
STORE001:/tmp/file2 → STOREBACK:/tmp.back/file2.032
STORE001:/tmp/file3 → STOREBACK:/tmp.back/file3.032
```

Working with node groups

Overview

Working with node groups within a plan involves some special consideration. The following section describes the characteristics of each type of function with regards to node groups.

File transfers

The `Use node extension` flag that can be checked in a `fileset`, definition controls the behavior of transferring to and from node groups. There are five different scenarios:

`Copying from a node to a node (extensions on/off)`

Extensions flag has no effect files are simply copied. File names are unchanged unless the target field is specified.

`Copying from a node to a node group (extensions off)`

Each node in the group will receive an identical copy of each file. File names are unchanged unless the `target` field is specified.

`Copying from a node to a node group (extensions on)`

The `source node` contains a separate (and unique) copy of each file destined for each node in the group. Each file exist on the `source node` with an extension in the form `filename.NODENAME`. The `.NODENAME` extensions do not show up when viewing `filesets` but are assumed to exist. Each file that is copied is renamed to its name as seen in the `fileset` name or target field (in other words the `.NODENAME` extensions is stripped off).

`Copying from a node group to a node (extensions on)`

This is the exact reverse operation of the previous scenario. The same named file is retrieved from each node in the group and the `.NODENAME` extension is tacked on to the filename as it is created on the `target node`.

`Copying from a node group to a node (extensions off)`

This would seemingly be an unwise combination as it implies copying several files to the same target file. The only practical use of this scenario would be when using wildcards to select files for copy. This would mean that the file naming conventions were already in place to prevent name collisions and each of the source files already had a unique name. This option should be implemented carefully to avoid files overlaying one another.

File deletion

During a delete on `target` operation, if the `target` is a node group, the files will be deleted on each node in the group. The same applies for a delete on `source` operation, if the `source node` is a group. There is also special consideration here to working with a `fileset` that has node extensions turned on. In order to properly delete `target` files that have node extensions, specify

the appropriate `node group` in the `source node` field. Likewise, to delete `source` files that have node extensions, specify the appropriate `node group` in the `target node` field. See the example after the following section on "Program execution."

Program execution

The `target node/group` specifies on which node(s) the program will execute. The source `node/group` may be used to specify multiple instances of a programs execution in the following way. If the source specifies a node group and the target specifies a node, separate instances of a program will be started (on the `target node`) for each node in the group. In this case the reserved environment variable `NFMSNODE` may be specified in the program string. For each instance of execution the actual node name will be substituted allowing the program to uniquely perform activity related to its specific instance.

Node group example

The following example demonstrates a combination of some of these activities and how they work with node groups. We have the following requirements:

```
Copy out separate price files (price.dat) from node CENTRAL to 10 different
stores (nodes: 'STORE001' to 'STORE010', all contained in group 'GROUP1'.

Copy a batch file (procstore.bat) to each of these stores and run it to install
the price.dat file and also create an inventory file inventory.dat to be copied
back up to CENTRAL.

Copy the inventory.dat file back up from each store, and run a local program
that independently produces a separate report for each store from its inventory
file.
```

We will need the following `filesets` defined:

```
PRICEFILES:    Source prefix=/tmp/                Target prefix=C:/data/
               Source file=price.dat              Target file=(blank)
          Use Node Extensions is ON

PROCSTORE:     Source prefix=/tmp/                Target prefix=C:/tmp/
               Source file=procstore.bat          Target file=(blank)
          Use Node Extensions is OFF

INVFILES:      Source prefix=C:/data/             Target prefix=/tmp/
               Source file=inventory.dat          Target file=(blank)
          Use Node Extensions is ON
```

The plan we will use contains a single phase with the following functions defined:

```
1. Transfer       source=CENTRAL    target=GROUP1         fileset=PRICEFILES
2. Transfer       source=CENTRAL    target=GROUP1         fileset=PROCSTORE
3. Execute        target=GROUP1     command=C:/tmp/procstore.bat
4. Transfer       source=GROUP1     target=CENTRAL        fileset=INVFILES
5. Delete on source                 source=GROUP1         fileset=INVFILES
6. Execute        source=GROUP1     target=CENTRAL
   command="invreport /tmp/inventory.dat.$(NFMSNODE)"
7. Delete on target  source=GROUP1  target=CENTRAL        fileset=INVFILES
```

Now to examine the resulting actions from each of these functions:

1. This function expected to find separate copies of the price file and copy them out to each store in the group as follows (note the extension being stripped off during the copy):

```
CENTRAL:/tmp/price.dat.STORE001 → STORE001:C:/data/price.dat
CENTRAL:/tmp/price.dat.STORE002 → STORE002:C:/data/price.dat
CENTRAL:/tmp/price.dat.STORE003 → STORE003:C:/data/price.dat
```

2. This function copied out the same batch file to all the stores:

```
CENTRAL:/tmp/procstore.bat → STORE001:C:/tmp/procstore.bat
CENTRAL:/tmp/procstore.bat → STORE002:C:/tmp/procstore.bat
CENTRAL:/tmp/procstore.bat → STORE003:C:/tmp/procstore.bat
. . .
```

3. This function executes the batch program `procstore.bat` just copied down at each of the stores. This created the local file `C:/data/inventory.dat` at each store.

4. This function copied the unique inventory files from each store to central (note the extension being added during the copy):

```
STORE001:C:/data/inventory.dat       →       CENTRAL:/tmp/inventory.dat.STORE001
STORE002:C:/data/inventory.dat → CENTRAL:/tmp/inventory.dat.STORE002
STORE003:C:/date/inventory.dat → CENTRAL:/tmp/inventory.dat.STORE003
. . .
```

5. This function deleted the local file `C:/data/inventory.dat` at each store.

6. This function executed multiple instances of the program `invreport` on `CENTRAL` and passed each copy of the program a separate node name representing the different entries of the source node group. This was made possible by use of the reserved environment variable `NFMSNODE` discussed in the previous section:

```
(Executing on CENTRAL) invreport STORE001
(Executing on CENTRAL) invreport STORE002
(Executing on CENTRAL) invreport STORE003
. . .
```

7. Finally, this function deletes the multiple inventory files that were copied up to central with their extensions. This only does multiple deletes because of the source node group being set to `GROUP1`:

```
(Deleting on CENTRAL) /tmp/inventory.dat.STORE001
(Deleting on CENTRAL) /tmp/inventory.dat.STORE002
(Deleting on CENTRAL) /tmp/inventory.dat.STORE003
. . .
```

Plan execution flow

Overview

Execution flow within a plan can be controlled by several options. This can most easily be described by first understanding the default behavior of phases, functions, and node group considerations.

Default execution flow

Phases within a plan run sequentially by default. Each one must finish before the next one begins. Functions within a phase always run sequentially. For two functions to run at the same time they must be in separate phases. In most instances, when a function deals with a node group, multiple tasks (one for each node) perform the function at the same time. The function cannot complete until each task is finished. The exception to this is when the next function in the phase is also working with the same node group and no termination logic for the current function is in effect. In this case individual task may continue into the next function. This allows for speeding up a plan by not having to wait on a slow node.

Terminating

phases

Functions may specify a termination threshold. If the return code from the function is greater than or equal to a specified value, no other functions within the phase will execute. This may be specified explicitly for each function or may be specified in the phase as the default for each function. If specified in the function, it will override the default value specified in the phase.

Phase logic

Several alternatives for running phases simultaneously exist. Phases may specify a dependent phase that they run after or run with. In addition, whether or not the phase runs at all may be specified based on the return code from the previous phase. This behavior is examined in more detail with a description of each option and some examples.

Run phase after phase

A phase specifies that it will run after another if the return code allows for it. Consider the example:

`Phase2` runs after `Phase1` finishes with return code greater than `4`.
`Phase3` runs after `Phase1` finishes with return code less than or equal to `4`.

This allows for a simple `if`, `then`, `else` logic. `Phase1` will run, and then either `Phase2` or `Phase3` will run depending on the return code.

Run phase with phase

Phases may run simultaneously by specifying more than one to run after another or to specify running with another phase. Phases may also specify running after multiple phases complete. This is best described by examples:

1. `Phase2` runs after Phase1 finishes with return code less than `8`.
   `Phase3` runs after Phase1 finishes with return code less than `8`.

   Effect: `Phase2` and `Phase3` run simultaneously after `Phase 1` finishes. They are completely independent from one another.

2. `Phase2` runs after `Phase1` finishes with return code less than `8`.
   `Phase3` runs with `Phase2` with the asynchronously option on.

   Effect: Same results as example 1.

3. `Phase2` runs after `Phase1` finishes with return code greater than `0`.
   `Phase3` runs with `Phase2` with the synchronously option on.
   `Phase4` runs after `Phase3` finishes with return code less than `8`.

   Effect: By specifying `Phase3` runs with `Phase2` synchronously, `Phase2` and `Phase3` are grouped such that no further dependent phases will run until they are both finished. `Phase4` will not run until `Phase2` and `Phase3` both complete and only if `Phase3`'s return code is less than `8`. In this example it does not matter what `Phase2`'s return code is.

4. `Phase2` runs after `Phase1` finishes with return code greater than `0`.
   `Phase3` runs with `Phase2` with the synchronously option on, propagate return code on.
   `Phase4` runs after `Phase3` finishes with return code less than `8`.

Effect: Same as #3 except that the return codes from both `Phase2` and `Phase3` are combined. Each phases return code is considered the same as the highest one between the two. Thus, `Phase4` will not run until both `Phase2` and `Phase3` are complete, and only if they both finish with return codes less than `8`.

Error recovery

Overview

This section describes the tools and techniques used in detecting, avoiding, and correcting errors that may occur while running a plan. This section does not address specific errors; see Chapter 8, "Troubleshooting," for a look at specific error scenarios.

Early detection

Invalid plans

When a plan is being written, it is possible to combine things in such a way as to create an invalid plan, or one that would certainly fail upon execution. Some examples of this might be:

- Referencing a configuration item that does not exist from within a plan or from within another configuration item. Example, specifying a node whose corresponding model has been accidentally deleted.
- Specifying an invalid combination, like transferring files from one node group to another, or copying directly between two FTP type nodes.
- Specifying a transfer without specifying a target node, and not providing a default in the plan header or during plan submission.

These types of errors are all caught during what is called a plan installation process. This installation is automatically performed during the following events:

Scheduling a plan When a plan is scheduled to run in the future, the installation process verifies the validity of all the elements. If any problems are found, a pop-up window will appear indicating that the plan is `REJECTED` and gives the cause for the rejection. When this occurs, no plan instance is actually created. After fixing the problem the plan may be resubmitted and thus, starts the procedure over again.

When a plan actually runs When a plan is scheduled to run in the future, and it has passed the initial installation performed during scheduling, it is possible that something may occur before it actually runs that makes it invalid. Example, a plan is scheduled to run in one hour, the plan instance is created and the plan is in the `Scheduled` state. But 15 minutes later, someone deletes one of the required nodes for this plan. For this reason, the plan again goes through the installation procedure when it actually starts running. When a failure occurs in this case the plan instance goes into a `REJECTED` state.

In order to help catch this kind of problem ahead of time, NFM automatically recalculates the requirements for all scheduled plans, whenever a change is made to NFM's configuration that may adversely affect a plan. In our example, a warning message would be generated, and a warning indicator would automatically appear next to the plan instance in the `Plan Activity` screen. Too see the warning message itself, you would need to go to the `Plan Monitor` screen and clicked on the `Audits` folder. If, however, the node were re-added prior to the plan running, the warning would automatically be removed.

Excluding and Holding

There are several options available to prevent errors from occurring if a problem is anticipated. These include the following:

Excluding a node

If a node is known to be unavailable or soon to be unavailable, changing the node record to mark it `offline` will keep it from being included in any plan activity (even if a plan has already been scheduled to use that node). This keeps you from having to remove the node from a node group when the downtime may only be temporary.

A node can also be excluded from a particular plan instance; this can be accomplished from the `Node Control` window of `Plan Monitor` or `Plan Activity` screen.

If either of these methods is used to exclude a node, the node may be enabled on a per plan instance basis with the `Node Control` window, but only if the plan instance has not began running yet.

When a running plan encounters an error communicating with an NFM node, NFM will automatically mark the node as `excluded` for the duration of the plan instance (except when `Hold on Error` is specified, see below).

Holding a node or nodes

Holding a plan instance or a particular node will suspend operations, providing a chance to potentially fix problems even while the plan is still active. If an entire plan instance is put on hold, individual nodes can later be enabled, or excluded if needed, on a node-by-node basis. `The Hold on Error` option that may be selected on an individual function can be used to make a node automatically become held if an error occurs. This allows an operator to potentially fix the problem and then enable the node. This will cause the node to retry the errant operation, and if successful, proceed as if there never was an error (a warning will be generated however).

Holding nodes either manually or automatically does have benefits; it implies operator intervention on a timely basis, because proceeding steps in a plan can be held up while nodes are on hold. It may be more desirable to let a plan finish running and then retry failed operations later. The next section deals with the plan retry option.

Retrying a Plan

NFM offers the ability to rerun certain parts of a plan instance that had one or more failures occur during its previous run. This is done from the `Plan Activity` screen, by highlighting a plan instance that is already in the finished state, and clicking the `retry` button. The plan must have had the `retry` enabled field checked (in the `plan header`) prior to its initial run.

During a `retry`, NFM scans for points of failures and retries the functions that caused them. If the failure was due to a NODE communications problem, the function will pick up where it previously left off for that node only. If it is an individual file problem, the retry will only attempt the operations specifically for the failed files. If the failure persists or a new problem occurs, NFM will mark the problems accordingly for additional retries. A plan may be retried any number of times.

Once a problem is fixed, the plan will continue running from that point on as if it was running for the first time. The following behavior is in effect during a retry operation:

- Once a function has been retried, any subsequent functions in the current phase are run again, regardless of whether they ran previously. The exception to this is when a subsequent function works with the same node group as the fixed function. In this case the subsequent function only works with the nodes that had errors in the current function.
- Once a function has been retried, any subsequent phases to the current phase will be run again, regardless of whether they ran previously.
- If a failure on an individual file caused the original function to skip the remaining files, then once the file failure is fixed, all remaining files will be acted upon; otherwise, only the failing files are retried.
- If a failure occurs in the middle of copying a file, and the function has `checkpoint restart` enabled. The retry will continue the copy from the point of failure within the file; otherwise, it will copy the entire file.
- If a non-zero return code occurs from an execute function, it will be considered a failure and a retry will cause the entire function to run again.

Since NFM will continue to do subsequent activity after a retry, it may be desirable to set up a plan in such a way that certain functions only run when everything has worked. The following shows this example - consider the following plan:

```
Plan: GETINV1
Phase1:
Function 1: Copy inventory file from node group ALLSTORES to node CENTRAL (exit
            phase if rc >= 4).
Function 2: Delete inventory file from store group ALLSTORES (exit phase if rc
            >= 4).
Function 3: Execute function on CENTRAL that concatenates all files together
            and generates a report from them.
```

The node group ALLSTORES contains nodes STORE001 through STORE010. During its initial run plan GETINV1 fails to connect to STORE009 and STORE010. Function 1 will have copied up inventory files for STORE001 through STORE008. On these 8 nodes the next function will delete these 8 files (Function 2 continues on the nodes that worked, because Function 1 and 2 both work with the same node group). Function 3 however, will not run at all because a return code of 6 (node failure) will be in effect after Function 2 finishes. Then run retry on the plan instance. This time STORE009 works but STORE010 is still bad. STORE009's inventory file is deleted but Function 3 still does not run.

Now we have finally called up STORE010 and found out his Windows NT computer caught a virus but the problem is now fixed. We retry the plan again and now the copy works followed by the delete. The current rc will be 0 and Function 3 will now run.

Look at a deviation from that example. STORE001 through STORE009 are finished but STORE010 is hopeless (the virus has wiped the entire hard disk). We would like to retry the plan again but ignore STORE010 and continue to perform Function 3. Using the node control button we go and exclude STORE010 from the current plan instance. Now do the retry. NFM will realize that an attempt to copy from STORE010 is necessary, but the node has been taken off-line. This will cause NFM to continue on as if a retry was attempted and worked (since no error occurred, we are at rc=0). Function 3 will now run (and of course, should be designed to not fail because of a missing STORE010 inventory file).

File streaming

Overview

A streaming file transfer allows a sequentially written file to be transferred as it receives data from another application or command. This allows a virtual mirroring of the file to another node and enables data to be transferred in near real time as it is being created. The next sections detail how to start and stop a streaming transfer as well as a list of restrictions to the process.

Starting a streaming transfer

By simply checking the box next to Streaming Interval in a plan transfer function, and assigning a numeric-streaming interval, a streaming transfer is

designated. (See "Plans: Functions," in Chapter 4, "Configuration.") Once a streaming file transfer starts, it will copy whatever portion of the file that exists to the target node (it will not fail if the file has yet to be created). It will then check the file every # seconds (# being the streaming interval), and will continue copying any new data appended to the file until such time as the transfer is terminated.

Stopping a streaming transfer

A streaming transfer is terminated in one of two ways:

1. When a streaming transfer is started, a tag file is created in the same directory as the file being transferred on the source node in the form `filename.ntag`. If this file is deleted either manually or by some other means, the transfer will terminate.

2. Another function either in another plan or another phase of this plan can perform a `release` function specifying the source node and `fileset` name. It should not be a function following the transfer function because that function will never run while the transfer is in progress. The following example shows this:

   `Phase 1` - runs in sequence.
   Function: Begin streaming transfer of datafile.
   `Phase 2` - runs with `Phase 1`.
   Function: Delay until 5:00 p.m.
   Function: Release datafile on source node.

   This plan will start and stop a streaming file transfer.

Restrictions

The following restrictions apply to a streaming file transfer:

- The application writing to the file must have the file open in a shareable mode.
- The application writing to the file must be flushing the file to disk on a regular basis. This is especially an issue on Windows and 4690 computers that will often not write a file until it has been closed.
- A `fileset` used for this type of transfer should only specify a single file entry.

Data compression

Compression for file transfers is enabled/disabled on a per fileset basis. Compression is turned to one of three states with a check box '`Compress Transfer`' in the fileset records (see "Filesets" under Chapter 4, "Configuration"). The three states are:

`on` Any transfer of this fileset will be compressed.

`off` Any transfer of this fileset will not be compressed.

default Compression of this filesets will occur if the 'Compress Transfers' checkbox is set to on in the 'System Settings' screen (see "System settings" in Chapter 4, "Configuration").

The decision to turn compress on or off will probably be a result of an end-users knowledge of the type of data being transferred. Compression may greatly reduce the amount of time it takes to transfer a file, but if the file is already compressed, then compression may not yield an improvement and simply take up extra CPU utilization. Data compression is only available for transfers between NFM client (sockets) type nodes.

Encryption and connection options

Overview

NFM provides two forms of encryption for socket connections among NFM components, including server to client connections, and client-to-client (peer-to-peer) connections involving file transfers. The NFM, or native type of encryption protects sensitive data sent over the network (file transfer data, etc.). The Secure Sockets Layer or SSL encryption implements the OpenSSL Version 2.3. and is more suitable for use over insecure networks like the Internet. Encryption is not available for non-NFM clients like FTP or HTTP nodes.

NFM simplifies the configuration for encryption by simply having the user designate the encryption type, on a per node (or per model) basis. When a connection is established between two nodes that have their encryption type specified differently, the more secure of the two is used for the connection. For example, if one node has encryption set to NFM and another has encryption off, a connection between the two will use NFM encryption. Likewise, if one node has encryption set to SSL and another has encryption off or set to NFM, the connection will use SSL.

In addition to the type of encryption required, there are other considerations involved with making connections between entities over an insecure network where firewalls may limit the connection options available. NFM offers several options for configuring its clients to address these needs. Depending upon which options are required, it may be necessary to alter the NFM client configuration file on each node as well as the node's server configuration record. This section will describe different connection options and the necessary setup to implement them.

Configuration

The following areas are all involved with configuring a nodes encryption and connection characteristics:

- The Node setting for encryption (in the Model or Node definition) specifying the node with encryption as Off, NFM, or SSL
- The Connections entries (in the Model or Node definition) that dictate the default and available means of connecting.
- The NFM client configuration file that exist at each NFM client node.

NFM client configuration file

The NFM client configuration file, resides in the NFM client's main install directory and is usually called `nfm.cfg`. A default configuration file looks something like this:

```
NFMSERVER=nfmserver
NFMNODE=unknown
CONNECT=LISTEN, PORT=8008, INTERVAL=, DURATION=, SSL=0, KEY=, CERTIFICATE=,
```

The first part contains the following fields:

`NFMSERVER=[address]` This field should be set to the IP address or hostname that the NFM server is running on.

`NFMNODE=[name]` This field should be set to the NFM node name that represents this node.

(NOTE: These two fields are only required if a `CONNECT` record of type `PERMANENT` is defined below).

The remaining portion of the file contains one or more connection entries. These entries should equivalent to the connection entries defined in the NFM node or model definition for this node. The following fields are available:

`CONNECT=[type]` This field should be set to the required connection type. Currently supported values are `LISTEN` and `PERMANENT`.

`PORT=[number]` This field should be set to desired port for this connection.

`SSL=[Y/N]` Indicates if this is an SSL connection.

The remaining fields are not currently implemented.

Examples

The remainder of this section displays some sample configuration setups to demonstrate the various options available.

Sample 1 – No SSL required

No SSL is required. Store 1 has encryption turned off. Store 2 has encryption set to NFM.

`STORE001` – Node record has encryption set off. No connection definitions are required and no changes to the configuration file at the client are required because the node by default will listen for a connection on port `8008`.

`STORE002` – Node record has encryption set to NFM. No connection definitions are required and no changes to the configuration file at the client are required because NFM encryption takes place over the default port of `8008` that the client is listening to.

Sample 2 – SSL required, node set to type listen.

Store 1 needs to use SSL. Store 1 is reachable across the network

STORE001 – Node record has encryption set to SSL. No connection definitions are required because NFM assumes the client is listening on port 8009 (the default SSL port). The following change is made to the configuration file at the node site:

```
NFMSERVER=nfmserver
NFMNODE=unknown
CONNECT=LISTEN, PORT=8009, SSL=1,
```

The first two lines are ignored because the connection type is LISTEN. The connection entry tells the NFM client to start listening on port 8009 (the default SSL port) for an incoming connection. The SSL=1 indicates SSL connection type only. The use of this configuration will disable this NFM client from doing any non-SSL connections and thus make it more secure.

Sample 3 – SSL required, node set to type connect (PERMANENT).

Store 1 needs to use SSL. Store 1 is not reachable across the network because it is behind a firewall. The NFM server has port 8019 opened in the firewall for incoming connections (the default NFM Server listening port for connect type nodes, using SSL).

STORE001 – Node record has encryption set to SSL. One connection definition is defined as follows:

```
 Type=[Permanent]  Port [8019] SSL [Checked]
```

The configuration file at the node site looks like this:

```
NFMSERVER=213.48.55.69
NFMNODE=STORE001
CONNECT=PERMANENT, PORT=8019, SSL=1,
```

The NFMSERVER and NFMNODE entries are now required since the NFM client at the store will be making a connection in to the server on port 8019. Once connected (using SSL), it will inform NFM that the connection represents store STORE001. Any activity that NFM needs to perform with this node will be initiated over this permanent connection.

Sample 4 – Some non-SSL nodes, some SSL nodes.

This example shows the unique requirement for configuring the NFM client running on the NFM server so that it can communicate with both SSL and non-SSL nodes at the same time. Store 1 and Store 2 need SSL connections. Two nodes, test1 and test2 are within the firewall and we do not want to use SSL with them. The NFM client running on the server computer needs to be able to receive connections from any of the four nodes.

STORE001 and STORE002 are set up as described in Sample 2.

TEST1 and TEST2 are set up as described in Sample 3.
Node SERV1 (the NFM node representing the NFM server computer) is setup as follows:

Node record has encryption set to Off. Two separate connection definitions are defined as follows:

```
 Type=[Listen]  Port [8008] SSL [Off]
 Type=[Listen]  Port [8009] SSL [Checked]
```

The configuration file in the NFM client directory at the NFM server node looks like this:

```
NFMSERVER=nfmserver
NFMNODE=unknown
CONNECT=LISTEN, PORT=8008, SSL=0,
CONNECT=LISTEN, PORT=8009, SSL=1,
```

Once again the first two lines are ignored since all connect entries are of type LISTEN. The next two lines configure two listeners so that the node will be able to handle connection request from SSL and non-SSL nodes. The reason that the NFM node SERV1 has encryption set to Off, is because that allows for SSL to not be attempted when SERV1 is communicating with TEST1 and TEST2. When SERV1 communicates with STORE001 and STORE002 it will know to use SSL anyway.

Sample 5 – FTP-S Server port enabled.

This example shows the process to enable the client to also be an FTP-S server. It can be configured to have multiple FTP ports. Authentication is done via the NFM server that means the FTP users must have NFM accounts. The required FTP password will be the users NFM password.

```
NFMSERVER=nfmserver
NFMNODE=STORE001
CONNECT=FTPSERVER, PORT=21, IPADDR=, SERVADDR=204.62.234.99, SERVPORT=8018, SSL=Y
```

The NFMSERVER line is ignored because the connection type is FTPSERVER. The NFMNODE line is used to identify this node to the server for user authentications and audits. The NFMNODE name specified must be configured on the server being used for authentication. The CONNECT line tells the NFM client to start listening on port 21 (the standard FTP port) for an incoming connection. The IPPADDR= indicates to accept FTP connections on any available TCP/IP interface. If an IP address is specified (IPADDR=204.62.234.99) then FTP connections will only be accepted on that interface. The SERVADDR specifies the IP address of the NFM server that will handle user authentication and audits for connections handled by this entry. The SERVPORT is the IP port where the NFMi server is listening. User authentication and audits are handled by the NFMi server.. The SSL=Y indicates that communications with the NFMi server will be using SSL.

More than one FTP-S server port can be configured. Each entry must specify a different PORT value (unless the IPADDR option is used to specify different IP interfaces).

User home directories default to <NFM client working directory>/home/<user name>. This can be modified by configuring the "Home Directory" setting in the nodes configuration on the server. Configuring a specific path such as "/home/ftp" will give every user the same home directory. You can use environment variables such as "/home/$(NFMUSER)" to specify different home directories for users.

Encrypting the NFM login session

In addition to encryption of file transfers as described above, NFM also allows a user to have their NFM session to be encrypted as well. This can be accomplished if a user is logged on through the GUI interface and entering a user name and password and then checking the encryption connection button.

NFM multiple server support

Overview

Multiple NFM servers may be configured to work collectively. This type of implementation presents itself to the end user as a single "virtual" NFM server. This allows administrators or other authorized users complete access to all NFM activity across multiple servers with a single login through the existing user interface. This single system look and feel is maintained even while the actual workload (plan processing, file transfers, etc.) may be spread out among multiple physical servers.

This is accomplished by having the multiple servers share a common NFM database. All data in the database is available to all servers in the collective, which makes the single virtual system look identical to all servers.

There are two types of NFM server designations, "Primary" and "Secondary." A primary server is one that controls and accesses its own database in a local file system. A secondary server accesses an NFM database remotely over a network connection to a primary NFM server computer. Among a group of NFM servers working collectively, there can only be one primary server, and any number of secondary servers.

The division of work among the servers is controlled using the existing usergroups (the physical partitions that allow NFM items to be logically subdivided among different groups of users). Each usergroup may be assigned a "home" server computer. Any time a plan from the usergroup is scheduled to run, the home server computer will actually perform the work. The plan may still be monitored and controlled from a different server, but the actual processes and connections involved will occur on the home server.

In addition to plan activity, an NFM server provides a user interface session to users that are logged in through the browser interface (or stand-alone interface). A user that is logging in to NFM may actually connect in to any of the NFM servers (`primary` or `secondary`) that are participating in the collective virtual server, and they will see the same content. It is recommended however, that users connect in to the "home" server of the usergroup that they belong to (or will spend most time working with). There are two reasons for this. First, a user will get better session performance logging in to the server that is running their plans (especially while monitoring plan activity). Second, and perhaps more important, if a secondary server loses network connectivity to the primary (and thus the database), it is still possible for it to perform processing independently. This means that the server may still be able to run plans, connecting to local nodes, and so on, even without being in contact with the primary server. In the event that this occurs, a user will need to be logged in directly to this NFM server to monitor and control local activity.

Configuration

When an NFM server is installed on a computer, it automatically creates a local database and assumes the role of a primary NFM server. Configuring a multi-server setup basically consists of separately installing multiple NFM servers (each one being a separate primary by default), and then re-configuring all but one of them to serve as secondary servers.

Since a stand-alone NFM server is already serving as a primary, no special consideration needs to be given about its role when it is installed. A customer may choose to add secondary servers years later to an existing NFM server. However, when installing a new server that may or may not become a secondary, consideration should be given to what its initial role will be. A primary server that is converted to a secondary server will lose any database items that were created on it while it existed as a primary. (Actually, existing database items may be saved off and re-integrated back into a collective setup, using the export and import functions, but this will involve additional setup work).

The following steps should be taken to create a secondary server and integrate it into a multiple server environment.

1. The primary server should already be installed, configured, and working properly.

2. A server that will become the secondary server should have the NFM server software installed on it. This will temporarily create an additional primary server running independently. Do not create any configuration items (nodes, models, etc.) on this server, as they will be lost after reconfiguring the server.

   Note: Existing data will not be erased; it will simply not be referenced any longer once the server is accessing the remote database. If necessary, the data can be recovered.

3. Log into NFM on the primary server and create a secondary server record representing the secondary server being added (choose a unique name for it (the name "SERVER02" is used here as an example). Refer to the section "Servers" in Chapter 4, "Configuration" for more details on working with server configuration records.

4. Stop the NFM server process on the secondary server.

5. Edit the NFM server configuration file on the secondary server. The file should be in one of the following locations and should initially look similar to the one shown below.

Unix: `/var/tps/dba/nfmfcp.cfg`
Windows: `C:\Program Files\TPS Systems\NFMServer\dba\nfmfcp.cfg`

```
SERVERNAME=SERVER01
FCPCONFIG=/var/tps/dba/nfmfcp.cfg
```

This represents a default configuration for a primary server. Change the value of the field SERVERNAME to the exact name chosen for the new server record created in Step 3 above.

```
SERVERNAME=SERVER02
FCPCONFIG=/var/tps/dba/nfmfcp.cfg
```

Note: The second field references the NFM database configuration file, which will be changed in the next step.

6. Edit the NFM database configuration file on the secondary server. The file should be in one of the following locations and should initially look similar to the one shown below.

Unix: `/var/tps/dba/nfmfcp.cfg`
Windows: `C:\Program Files\TPS Systems\NFMServer\dba\nfmfcp.cfg`

```
TYPE=LOCAL
DATAPATH=/var/tps/dba/data
ADDR=
PORT=
```

This represents a configuration for a database found on the local file system. Change the TYPE from LOCAL to REMOTE and set the ADDR field to the IP address or hostname of the primary NFM server computer:

```
TYPE=REMOTE
DATAPATH=/var/tps/dba/data
ADDR=204.23.13.66
PORT=
```

- The DATAPATH field is now ignored since the TYPE field is set to REMOTE.
- The ADDR & PORT fields are now active. The PORT field may be left blank which will default to 16016.

7. Restart the NFM server process on the secondary server.

At this point you should now be able to login to the secondary server with the user interface and see the same content you would see if you were logged into the primary server. You will of course need to log in as a user that is defined to the primary server, as you should now be using the shared NFM database. Take note of any initial messages that are displayed while logging in as they may indicate a problem with the setup. No plans will run on this server until the following Step 8 is completed.

8. Create one or more new usergroups whose workload will be performed on this secondary server (you may change an existing usergroup to now utilize the new server). To perform this step, use the Usergroup menus under configuration (see "Usergroups" in Chapter 4, "Configuration"). Assign a server to a usergroup by setting the "Home Server" field in the "`Home Server`" tab in the `Usergroup Detail` screen to the appropriate server record. After this is done, the corresponding server should automatically perform any plans that are scheduled for this usergroup.

Backup Servers

Any existing server may be defined as a backup to another server. This can be done in the `Server Detail` screen (see section "Server," in Chapter 4, "Configuration" for the specific field settings). The following restrictions should be considered when setting up a backup server.

- In order to operate properly, a `backup server` must have the same access to needed resources as the server that it is a backup of. This includes any node connections, as well as any local files or programs that are involved with a plan's functions.

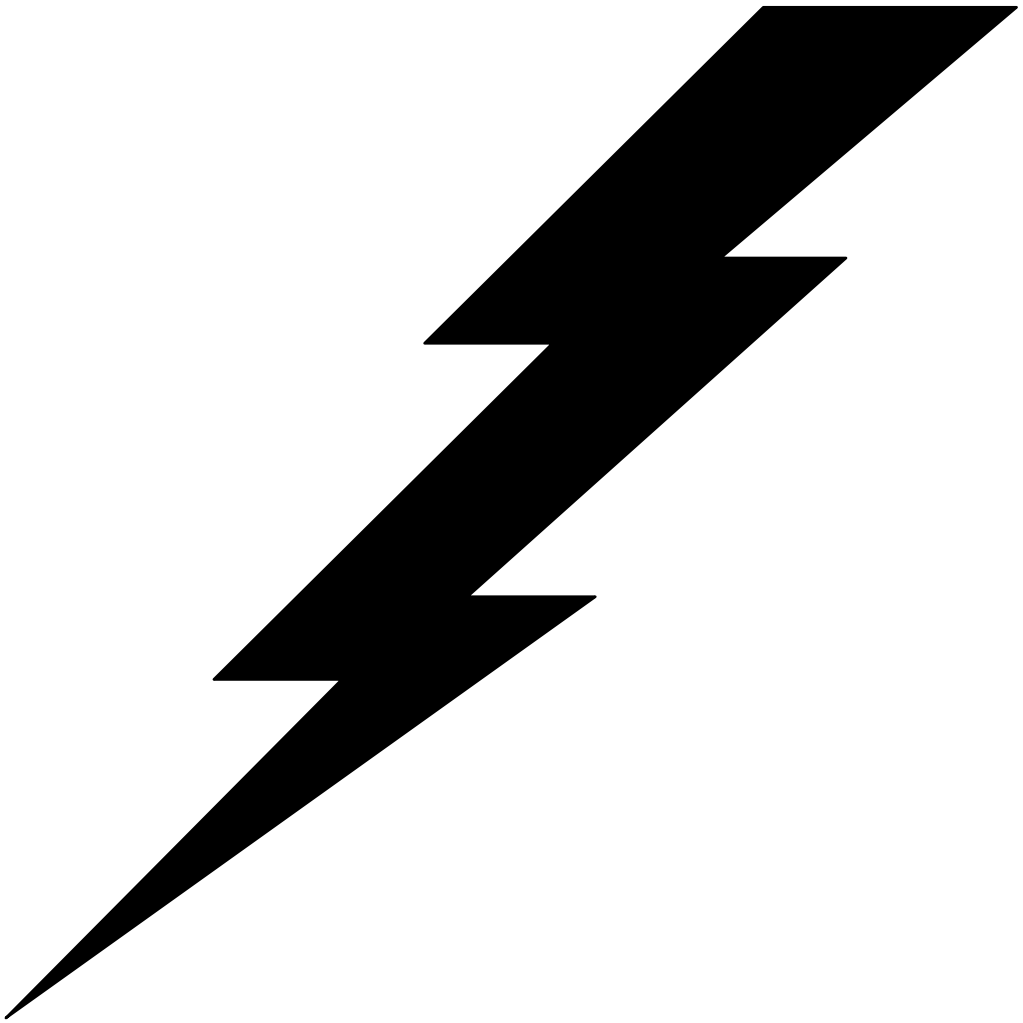
- A `backup server` will automatically begin running plans after the takeover interval (defined in the server) has expired on a per plan instance basis. If the assigned server recovers, it may continue to perform plans even while some of the plans it missed may have been started by the backup.

- Because a `backup server` keeps the NFM database replicated, it will determine when to start a plan instance based on its own copy of the data. If the master database is unavailable, the backup will continue to operate on the locally cached data. This is ideal of course if the primary server has crashed and the master database is unavailable. However, if communications is lost between a server and its backup, it is possible that both servers will attempt to run the same plan instance. Care should be taken in the configuration of the take over interval, as well as the general design of the plans to minimize the impact of this.

- A server that is designated as a backup may still be a fully functioning server in its own right (it does not have to exist just to be a backup). Assuming that a server and its backup are generally doing unrelated

work, the effects of having to do backup work on a server should simply be an increased amount of workload.

- A server that is a backup to another may also have its own backup; however, the workload from any given server will only be taken over by its immediate backup and not cascade to a backup of a backup. Two servers may be designated as backups to each other.



7: Bandwidth Management

Overview

This section describes the bandwidth management features of NFM. This includes two parts. This first section, bandwidth control, describes the mechanisms used by NFM to tune the amount of bandwidth NFM uses. The second section, bandwidth monitoring/reporting, describes the tools available with NFM to examine its bandwidth usage.

Bandwidth Control

Overview

Performing file transfers across a network can obviously be very bandwidth consuming. This problem is of course even more intensified when transferring files to or from a node group. Left unchecked, NFM will attempt to do all transfers in a running plan, simultaneously, and as fast as is possible, being almost always limited in speed, to whatever the network can handle. The source node or nodes, will attempt to write all data until the TCP/IP stack halts the flow (flow control) and waits for the network to catch up. Because of these factors, doing file transfers can have a severe impact on other network activity.

NFM provides two primary mechanisms that enable the user to control the amount of data NFM attempts to transfer at any given moment. This consists of limiting the number of active nodes and data pacing or throttling.

Maximum active nodes

NFM allows the user to configure the maximum active nodes in two places. There is an overall "`Maximum Active Nodes`" field in the system settings that can be used to limit the total number of nodes in use at a given time within all of NFM. Once a particular node starts a file transfer, it will reserve one of these slots until it is finished and then free up the slot for another node to begin. This limit is system wide; no matter how many plans may be actively trying to do transfers. Any node that is halted, waiting for a slot, will be clearly marked as such while monitoring a plan. (See "System settings" in Chapter 4, "Configuration," for setting this field.)

In addition to this, there is a "`Maximum Active Nodes`" field that is available on a per node group basis. This works in the same way but allows the user to give different node groups, a different number of slots from which to work with. This allows assigning some groups with more bandwidth than others. (See "Node groups" in Chapter 4, "Configuration," for more information.)

This provides an efficient way to increase performance for other network activity without necessarily impacting the performance of NFM. Other applications are able to better compete for network resources, but if no other activity is occurring, NFM will still try to use as much bandwidth as is available.

In addition, even if NFM is the only application using the network, setting the maximum active nodes may actually make NFM more efficiently use the existing bandwidth. This would be true if too many simultaneous network connections adversely affected the network as a whole. So, put another way, setting this value might make NFM finish a set of file transfers faster, than if the value was not set at all.

Data Pacing

NFM implements a configurable pacing method that can be tuned on an individual node basis. When this is active, a source node that is sending a file, will only send a certain number of data blocks, before waiting for a reply from the receiving node (the data block size is also configurable). Although this creates a slight overhead, it can dramatically increase performance of non-NFM network activity, because the NFM sending nodes will not be able to flood outgoing TCP/IP queues with data, without letting other applications get their data written out in a timely fashion.

Like the "`maximum active nodes`," this mechanism is extremely efficient because it will only significantly slow down NFM transfers if other network activity is present. (For more information on implementing this option, read the descriptions for the "Transfer Block Size" and "Transmit Window Count" fields for the applicable node models, under "Models" in Chapter 4, "Configuration.")

Bandwidth Monitoring

Overview

NFM provides a reporting mechanism that offers detailed statistics about its own bandwidth usage. This can be used to analyze network usage and aid in tuning the bandwidth control mechanisms. In addition, the data can offer the user-detailed accounting of network usage by selecting and reporting the data using multiple criteria. The user could for example, report exactly how much bandwidth was used by a given node over a certain time period, or perhaps, how much bandwidth was used in transferring a particular set of files or even an individual file.

The reporting is available in two ways. A performance graph is available through the GUI that can view real time statistics as they are occurring, or view historical data. Also the statistics information can be exported in a batch fashion to simple text files, where it would be available for outside applications. Both of these allow extensive selection criteria to decide what to report (by time interval, node, fileset, individual file, etc.). The amount of statistics data that is maintained can be configured by the user with the ability to automatically purge or compress and archive data automatically at set times.

Extracts

Extracts are configurable records that define the parameters for a query of performance data (whether through graph or batch). The extracts define two major parts of any given query; selection and formatting. The `Selection` part provides the ability to filter the available performance data to isolate only the data that is desired. The `Formatting` portion is used to create custom reports (in batch), or to indicate which binary fields are desired to look at (in graph form).

Extracts may be accessed from the `Performance` drop down menu of the `Management` sub-menu. The following fields and folders are maintained on this screen:

`Extract Name` This is a user selected name for an extract. Type in a new entry, or select an existing entry from the drop down list.

`Selection` Click on this folder to display or edit the desired criteria for selecting performance data. The available fields and their field type are:

`/Byte Count`	numeric (bytes)
`/File Name`	string
`/Fileset Name`	string
`/Plan Name`	string
`/Plan Instance`	numeric
`/Source Node CPU Time`	numeric (milliseconds)
`/Source Node Disk Read Time`	numeric (milliseconds)
`/Source Node Name`	string
`/Source Node Network Write`	numeric (milliseconds)
`/Target Node CPU Time`	numeric (milliseconds)
`/Target Node Disk Write Time`	numeric (milliseconds)
`/Target Node Name`	string
`/Target Node Network Write`	numeric (milliseconds)

The radio buttons for each field allow one of the following choices:

`N/A`	Not applicable.
`Equals` or `Starts With`	String selections.
`<, =, >, Not Equal`	Numeric selections.

When a query is performed, the available data records must pass each of the above field selection checks to be included in the output.

`Formatting` Click on this folder to display or edit the desired formatting for a query done with this extract. The available fields and their usage is as follows:

`Output File Format`
Determines how the fields are separated for batch output. This may be set to one of the following: `Comma separated`, `Tab separated`, or, `Fixed Length`. The `Fixed Length` will use the length parameter specified per line entry (as discussed below).

`Show Zeros`
This checkbox will determine whether or not to generate entries that contain zero binary data, or to only display non-zero information (the default) with regards to batch output (has no effect on graph). The following example describes this better:

Let's assume a file of size 100 bytes was transferred to a specific target node at 1:30, 2:30 and then 3:30 p.m. Now let's say a query was performed to look at the byte count of transfers from 12:00 (noon) to 6:00 p.m., on a one (1) hour interval for just this target node. If this box is not checked then only the records containing data would be displayed something like the following (this particular display would show from an extract that included the fields `timestamp` followed by `byte count`, with a comma separated output format, see below):

```
03/01/10 14:00:00,100
03/01/10 15:00:00,100
03/01/10 16:00:00,100
```

Checking the `Show Zeros` option would cause an entry to be printed for each time interval regardless of the lack of actual performance data present as follows:

```
03/01/10 12:00:00,0
03/01/10 13:00:00,0
03/01/10 14:00:00,100
03/01/10 15:00:00,100
03/01/10 16:00:00,100
03/01/10 17:00:00,0
03/01/10 18:00:00,0
```

This option might be necessary for input to another application that expects to see uniform counts whether they are zero or not.

Fields

Multiple line entries may be entered in this portion of the folder to define the different fields (and their order) that will be output during a batch query. The binary fields chosen here will reflect which entries will show on a graph query (string entries cannot be graphed of course).

To add a new line (and thus a new field entry), fill in the appropriate fields in the bottom of the screen and click on the `Add` button. To change an existing line, click on the line to highlight it and change the appropriate fields at the bottom of the screen and then click on the `Change` button. To change the order of the entries, highlight a line and then click on `Move Up` or `Move Down` button. Use `Delete Line` to delete a highlighted entry, and use the `Clear` button, to remove all entries.

The available field entries that may be defined (as reflected in the `Field Name` drop down box) are:

```
Timestamp
Constant
Active Nodes
Byte Count
File Name
```

```
Fileset Name
Plan Name
Plan Instance
Source Node CPU Time
Source Node Disk Read Time
Source Node Name
Source Node Network Write
Target Node CPU Time
Target Node Disk Write Time
Target Node Name
Target Node Network Write Time
```

Most of these entries match the list of selection criteria described in the previous section.

Format
This field describes the display format used for this field in a batch query. The following choices are available:

```
ASCII Decimal, zero filled
ASCII Decimal, space filled
Constant
Active Nodes
```

Buttons

The following buttons are available on the Extracts screen:

Apply Applies all changes made on this screen. This will cause an add or update to occur, depending on whether or not the extract preexisted.

Cancel Cancels all changes on this screen since the last time this extract was displayed or applied.

Delete Deletes the currently displayed extract.

Graph Takes the session directory to the Graph screen, where queries may be performed (described in the next section).

Graphs

The Graph screen is where queries are performed using the GUI. This screen has folders for looking at actual graphs and for looking at batch output. Batch output may also be obtained using the nfm command line facility (using the get,stats subcommands, see Chapter 8, "Tools").

The Graph screen may be accessed from the Performance drop down menu of the Management sub-menu, or by clicking on the Graph button from the Extracts screen. The following sections define the different folders and their use.

Graph Click on this folder to display an actual graph. Enter the following fields and then click on the Run button to see the graph:

From Date and Time
Determines the beginning time for the graph data to display. If the box before the From field is checked it will use the date specified, otherwise if the box is not checked, the graph will start with the earliest existing data.

To Date and Time
Determines the ending time for the graph data to display. If the box before the To field is checked it will use the date specified, otherwise if the box is not checked, the graph will continually update itself showing near real time data as it is generated.

Total Each # [time interval]
Tells the graph on what intervals to display the graph data (eg., every 10 minutes, every two (2) hours). Larger intervals will display faster but will contain less specific information.

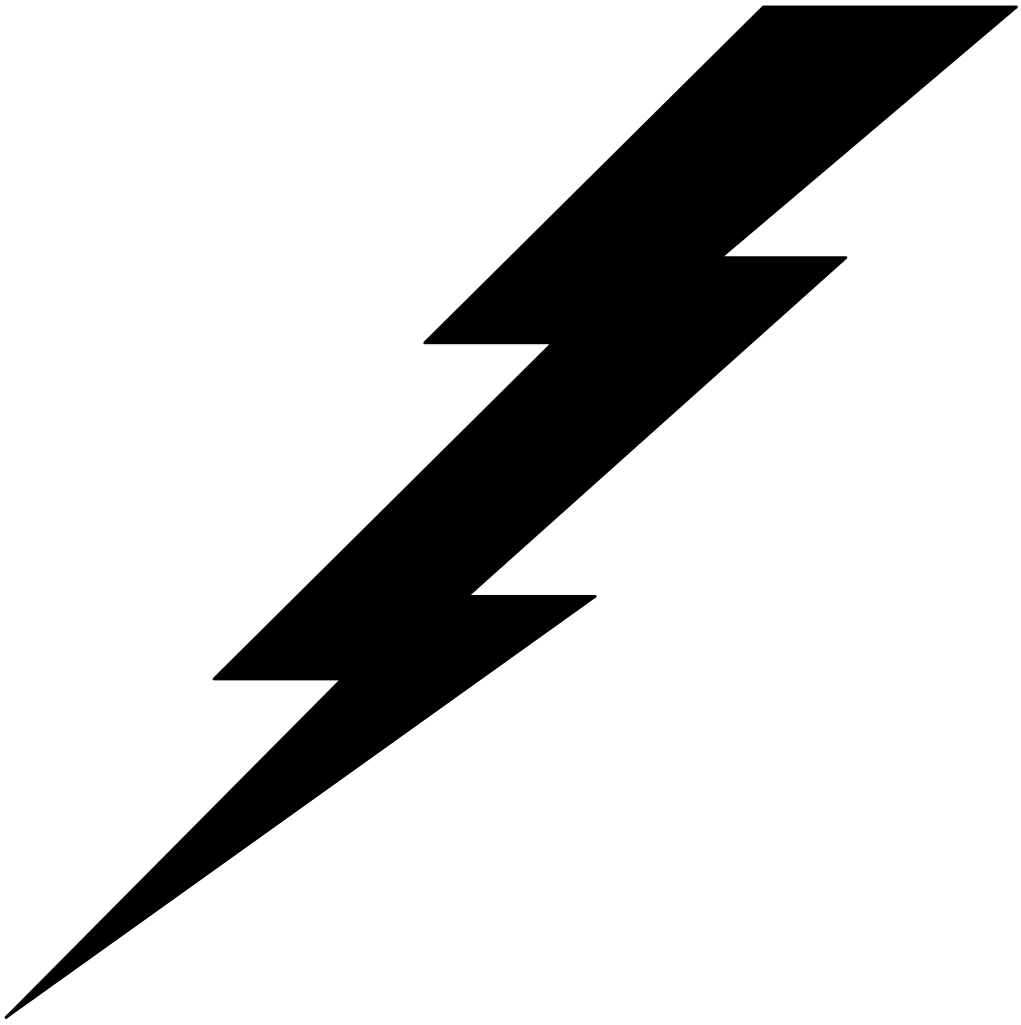
The graph will display different colored lines representing the different binary fields that are present in the extract format list. The column to the left of the graph will display a field description, and marker information that represents only one of the binary fields which defaults to the first (or only) one specified in the extract formatting list. To change this column to one of the other binary fields, and to display the color code information, click on the Legend folder described below.

The row below the graph will show corresponding time intervals for the graph data. The graph will attempt to make these intervals match the intervals requested in the "Total Each" field described above, but may display fewer entries due to constraints of what will fit on the screen.

Legend Click on this folder to view a description of each of the lines displayed in the graph, including the color and some general information about the corresponding data. Also, by checking the box next to one of the alternate fields, the column back on the graph can show the scale for that particular field.

Table This folder displays the exact numeric data that was used to display the most recent graph. This can be useful to view values that may not be easily visible on the graph or to compare numerical values in the different fields. This output is similar to the output from the Batch folder described below, but it does not contain any of the string fields.

Batch This folder is used to generate batch data based on the extract chosen. The format of this output will be based on the fields selected in the format section of the extract. Similar to the graph, this folder allows inputting the start and stop time, as well as the time interval desired for the display. Also, the field at the bottom "Output file name" allows the data to be directed to a file.



8: Tools

Data reformatting tools

nfmparse

The `nfmparse` program is a stand-alone program that runs from the command line on an NFM server computer. This command can be used to separate records from a single data file into multiple files that may be destined for multiple nodes. Parsing an input file requires that `header` records exist, between the data for each node. This header record must contain two things: 1) a string constant that distinguishes it as a header record, and 2) a node identification field that can be mapped to an output file for each node. The `nfmparse` command also performs the reverse operation of combining files into a single data file. This can be a simple concatenation, or an option can be specified to insert a header record between the data that is built from each input file.

Each of these operations allows changing the record size automatically and converting between fixed and variable length records if required. The `nfmparse` command may be executed in a stand-alone environment, from a script or bat file, or it may be specified from within an NFM plan as an execution type function.

Command syntax

The syntax of the `nfmparse` command is as follows:

```
nfmparse option option option . . .
```

The options are specified as either `<name>` or `<name>=<value>` pairs depending upon the option (do not include the < and >). These options may be specified on the command line or, more commonly, in a text file by use of the `options` option. The parsing operation is performed by default. Specifying the option `combine` will cause the combine operation to be performed. The `<name>` portion of the pair must be one of the option names below.

The `<value>` portion of the pair may be either a literal string or an environment variable. An environment variable is indicated by a `$` in the first position (UNIX style). If no environment variable with the given name exists, the variable name will be used in its place. Numeric or single character values may be entered in decimal, hex or octal.

NOTE: These are "scripting" environment variables, not the NFM environment variables available when running plans.

There are two "classes" of options; those that are only specified once, and those that are specified for each header "type" within the input file. If multiple header "types" occur in the input file for a node, there will be multiple output files for that store.

One time options

parse (no value) Specifies parse operation (the default).

combine (no value) Specifies combine operation.

Options=<text file> Names a text file containing more options. Within option files, the # character is a comment delimiter which makes all characters to the right of the # comments. Option files can include other option files with this option. There is no automatic protection against recursion.

NOTE: The remaining options may have different meaning for parse versus combine operations as noted with each option. These options are set up in such a way that the same options file used for parse can be used for combine, and would simply reverse the operation, combining the parsed output files back into the collective file.

file=<filename> Input file name. The default, when this option is not specified, is to use standard input. For combine, this is the output file name. This should have the following format:

1. Records in the file are either.
   A) Delimited by CR or NL or CRNL or some other simple string.
   B) Of a known fixed length (same for headers and data).
2. Header records separate data for individual stores.
3. Header records can be distinguished from data records by simple pattern matching at specific offsets (often 0) within the header record.
4. There may be multiple header types in each input file.
5. Each header record contains some identifying node identification sequence, which can be used in either a table-lookup or text-substitution scheme to map it to a TPS®/NFM node name.

delimited=<string> This string delimits input records. The following common substitutions are available: \r = carriage return, \n = newline, \t = tab. Common values would be delimited=\r\n for Windows style files or delimited=\n for UNIX style files. See the fixed option below. For combine, this is the output delimiter.

fixed=<#> Input records are fixed length this many bytes. Exactly one of "delimited" or "fixed" must be specified. Example: fixed=256 would indicate fixed length records of 256 bytes. For combine, this is the output record length.

pad=<#> For combine operation only, if "fixed" is specified (see above), records that are shorter than the fixed value will be padded with this numerical ASCII value. For parse, this field has no meaning.

varlenencode For combine operation, this flag turns on the variable length encoding feature which when used with delimited input files, will cause a two byte binary length to be imbedded into the output file. This encoding is

identical to that expected by a file being transferred to an OS/390 that has the Variable Length Encoding flag specified in the fileset, OS/390 folder.

nodesubstitution=<conversion_string>
A string that specifies a conversion to create a node name from the node identification value in the header record. This string allows placing constant characters before and after the node ID, by placing the characters `%s` in the string where the node ID is put. Example: If a store number of three (3) characters such as `015` was the ID field and we wished to map that to a node name of `STORE015` we could specify the string `STORE%s` as this string. For `combine`, this works in reverse and would strip off `STORE` from the node name to create the ID `015`. If the node name cannot be built directly from the node ID, it may be necessary to use the `nodetable` option described below.

nodetable=<file name>
Name of a file containing node ID field to NFM node name mapping. This is a text file containing `<ID>` space `<nodename>` (without the `<>`) for each valid NFM node name. This can be used where simple pre-pending/appending operations are not insufficient. Blank lines and comments are not supported in this file. For `combine`, this translates in reverse. Exactly one of `nodesubstitution` or `nodetable` must be specified. See `storeposition` and `storelength` below.

filetable=<file name>
Name of a file containing file ID to file name mapping. During a combine operation, this optional field allows an input file name (as specified in the `membername` field described in the per-header options described below) to be converted to a file ID field to be stored in the inserted header record of the output file. This field is only used if the per-header options `filenameposition` and `filenamelength` are in effect. This text file should contain entries in the form `<ID>` space `<filename>` (without the `<>`) for each valid input file.

memberdirectory=<directory name>
Directory for output files. The default, when this option is not specified, is to create output files in the current directory. An output file of the form `<membername>.<node name>` will be created for each header instance encountered in the input file. This is convenient when using TPS®/NFM's `user file extensions` option, which expects files in this format. All output files constructed from a single input file are placed in this directory. For `combine`, this is the directory for the input files

maxrecord=<#> This is the maximum input record buffer size. Records longer than this value minus 1 will not be handled properly. This applies to both delimited and fixed records. The default for this value is `2048` yielding a maximum usable input record length of `2047` bytes. For `combine`, this is the output record buffer size.

maxreclen=<#> Similar to `maxrecord`, this field limits the input and output maximum record sizes. The following differences exist:

1. During a parse or a combine, if the input file is delimited and the file does not physically end with a delimiter, the remaining portion of the file is processed as a record anyway.

2. During a combine operation, if an output delimiter is being used, the total output record size including the delimiter will not exceed `maxreclen`. The "`maxrecord`" size could be exceeded including the delimiter.
3. An output record that exceeds `maxreclen` and is wrapped into an additional record, will not end up with a delimiter on it (using "`maxrecord`" it would have one).

Although "`maxreclen`" is intended to replace "`maxrecord`", "`maxrecord`" will remain available for backward compatibility.

`maxtruncate` During a combine or parse operation, any record that exceeds the length specified by "`maxrecord`" or "`maxreclen`" will by default be wrapped into additional records as needed. Use of this option "`maxtruncate`" will cause the record to be truncated at the maximum length with any additional data discarded.

`partialdelimiter` Two different situations may result in a partial output record being created which are:

1. During a combine operation when an input file does not end with a delimiter.
2. During a combine or parse operation when an output record is truncated due to the use of the "`maxtruncate`" keyword.

By default, these partial records are written out with no delimiters. Use of this keyword "`partialdelimeter`" will cause a delimiter to be added to these output records.

`verbose=<#>` Output `verbose` message to standard output. The default for this value is `verbose=0` meaning `none` or `OFF`. Setting `verbose=1` will cause high level tracing messages to be printed. Setting `verbose=2` causes additional per-record information to be printed.

Per-header options

`header=<string>` File header sentinel string. The `nfmparse` program will examine each input record for the presence of this string in the location specified in `headerposition` below. When an input record matches, a new output file is created with `membername` below for the NFM node identified by `storeposition` and `storelength` below. This also sets the values below to their defaults for this header when it is found in the options stream. For `combine` operation, the value of this field is not really used, however, it should be set to something anyway to indicate the beginning of a new group of header options.

`headerposition=<#>` Offset of head sentinel within header record. Required for each header. For combine this field is not used.

`headerinit=<string>` This field is used for combine operation only. This string can be used to specify constant data that will make up the inserted record placed between records in the combined output file. This

record will have the appropriate node identification string described below, copied over it before it is written out. If this is a fixed length output file, than this string will be truncated or padded as needed to accommodate the correct record size. If this is a variable length output file (`delimited`), than this string will represent the output record size for each of these inserted records. Make sure to use double quotes to include spaces. Also, this option must be specified to cause inserted records to be created. If this option is not included, the output file will simply be the input files concatenated together.

`membername=<name>` This specifies the output file base name for this header type. Required for each header. This name combined with the node name in the form `<membername>.<NFM node name>` will represent the actual member file names which will be created in `memberdirectory`. For `combine`, this is the input file base name.

`storeposition=<#>` Offset of node ID field within header record. Required for each header.

`storelength=<#>` Length of ID field within header. Required for each header.

`memberlength=<#>` Output record length. This implies that output records will be fixed at this length. See also `memberpad` and `memberdelimiter` below. For `combine` operation, this is the input record length.

`memberpad=<#>` Input records of length less than `memberlength` will be padded to `memberlength` with this character. This should specify the numerical ASCII value. The default is `ASCII space` (`32` or `0x20`). This field is only used if `memberlength` (above) is specified. For `combine` operation this field is not used.

`memberdelimiter=<string>`
Output record delimiter string. This implies that output records will be `delimited`. Either `memberlength` or `memberdelimiter` must be specified for each header. Output records are not required to be of the same `delimited`/`fixed` organization as input records. See the `delimited` option above. For `combine` operation, this is the input record delimiter string.

`membereof`
This option specifies that entire files (output files for parse, input files for combine) be created/preserved in roughly their original form as one contiguous block of data. The size of this block of data is maintained through the use of the `datasizeposition` option (described below), which must be used in conjunction with this option. Using `membereof` has the following effect on combine & parse operations:

Combine: Each input file is written as a series of records to the output file based on the combined file format. If the combined file format is fixed length, then as many records of the fixed size as necessary are created to hold the file including the final record which may contain a partial block of data combined with the necessary pad characters to create a full record. The original size of the file (in bytes) is stored in the header record inserted prior to the file data.

Parse: While parsing a combined file, when a header sentinel string identifies a header record that uses the `membereof` option, the size of the data that makes up the individual file is extracted from the header at the position specified by `filesizeposition`. As many records are read as are necessary to extract the originally stored entire file that is then written out to its original length.

`filesizeposition=<#>`
Offset of the 4 byte binary size field within the header record. This option should only be used in conjunction with the `membereof` option described above.

`filenameposition=<#>` Offset of filename or file ID within header record. This maintains the file information for the data that follows either by name or by ID (typically used in conjunction with the `filetable` option). The `filenameposition` and `filenamelength` typically describe the same field used as the header identifier or sentinel string (referenced by `header` and `headerposition`). This is especially the case when a single configuration file is used to describe combining or parsing multiple types of files. This field is optional for each header.

`filenamelength=<#>` Length of filename or file ID within header. This field is optional for each header.

`dateposition=<#>` Offset of date field within header record. During a combine operation, this field is set to the modification time of the individual input file. This field is not currently used during a parse operation. This field is optional for each header.

`dateformat=<string>` This field describes the desired format for storing the date field specified by `dateposition`. The string should contain any combination of the following strings and constant data used as separators:

MM	(month 01-12)
DD	(day 01-31)
YY	(2 digit year)
YYYY	(4 digit year)
hh	(hour)
mm	(minute)
ss	(second)

Example: MM/DD/YY hh:mm
Might display: 03/12/63 17:00

`copyheader=<#>` Include header records in output files. The default, `copyheader=0`, does not cause header records to be copied to output files. If `copyheader=1`, the header records will be copied to the output files. For `combine`, this field has no meaning.

`group=<group_name>` This is an optional name of an NFM group to create during the parse operation. This group will contain all of the NFM node names generated from the header records of the input file for this header

type. Note that the environment variables $NFMUSER and $NFMPASSWORD will need to be set prior to running nfmparse to allow this program to write the new group entry. For combine operation, this field has no meaning.

inputfilespec=<string>
This field is used for combine operation only. This is a wildcard representation of all files in the directory (specified by) to be included as input for the combine operation. This uses Dos type wildcards including * to represent any text and ? to represent individual characters. This should specify files that match the member name, for example: <membername>.*. This field can however, be more specific to grab only certain files that match the membername like <membername>.store01??.

Exit values

The nfmparse program terminates with an exit code of 0 when no errors were detected, all input was processed and there were no errors writing output.

If a configuration error is detected, nfmparse terminates with an exit code of -1.

Other errors are reported in the exit code as bit-mask values:

1	Warnings generated.
2	One more store IDs were not found in the node table.
4	No files matched the inputfilespec field.
8	Some input records were not written to output files.
16	Failed to add an NFM node group.
32	Some data was truncated.
64	One or more nodes invalid while adding NFM node group.
128	Severe error occurred.

NFM command line interface

Overview

The nfm command provides a command line interface to the NFM server functionality. This program can accomplish almost any tasks that can be performed through the GUI user interface. Some tasks, like importing and exporting data to and from the NFM database, can only be done through this command line interface. The following section details the use of the nfm command.

The nfm program is located in /usr/lpp/tps/nfm/bin on AIX and /opt/tpsnfm/bin on most other versions of UNIX, and should be available on the path from a symbolic link in /usr/bin/. On Windows the nfm program exists as C:/Program files/TPS Systems/NFMServer/nfm.exe by default. The following environment variables should be set by the calling process to use this command (some are optional as described):

NFMUSER Set to the appropriate NFM user. This is required.

NFMPASSWORD Set to that users password. This environment variable may be left `unset` if the password value is empty.

NFMPFILE An alternate method of supplying the password. This may specify a file that contains the encrypted password. (See the subcommand `pfile` below for a description of how to create a password file).

NFMUGID Set to the ID of the current usergroup. This is similar to selecting which usergroup you are currently "logged into" while using the GUI and becomes the default place to load items from if they are not explicitly referenced from another usergroup. This field may be left blank to imply `usergroup 0 ([Public]` usergroup).

Input

The general syntax of the command is as follows:

```
nfm subcommand,parm1,parm2,parm3 . . .
```

The `parms` are dependent upon which subcommand is specified. Commas separate the subcommand and all `parms`. If you need to specify a comma or any white space in a parameter, the entire parameter should have double quotes surrounding it.

Any parameter that allows a default may be left blank by simply specifying nothing between the commas. If at any point in the command, the entire rest of the command may be defaulted, than the remaining commas do not have to be specified. Example:

```
nfm submitjob,planname,,,,,,,,,
```

is identical too:

```
nfm submitjob,planname
```

Some subcommands require multiple lines of input, like `putfileset` and `putplan`. In order to allow for a new line to be entered from a shell or script, the entire set of parameters, after `nfm,` should be encased in single quotes. This will allow you to simply type the `newlines` and stay in the command until the end quote is reached.

Example:

```
nfm 'putplan,H,planname,,SUMMARY,
P,phase1,N,0,DEFAULT,,,GREATER,,0,N,N,
F,TRANSFER,FILESET,FROM,TO,N,0,N,0,1,1'
```

Immediately following the description of each parameter in the subcommand section are two fields in parenthesis. This first indicates whether or not the field is required or optional (`req` or `opt`). The second is a type `specifier` that indicates any special format the parameter must be in, and will be set to one of the following:

`string:`	The parameter is simply a string, like plan name. Remember to double quote it if it contains white space or commas.
`dec:`	The field is a decimal numeral.
`date:`	The field represents a date and time in the following

format: `MM/DD/YYYY:HH:MM:SS`

Output

The output from the `nfm` command will vary based on the subcommand. It may be multiple lines but the final line will always be the error message line and will look like this:

```
:error:#:error_message
```

The `:error` indicates this to be the error message line. The `:#` indicates an NFM type `error code`, or `zero` for success. The `error_message` will contain a error message if the return code is `non-zero`. The following examples show the failure and success of deleting a user as well as a command with multiple line output:

```
in> nfm deleteuser,skippie
out>     :error:200:Record not found

in> nfm deleteuser,doug
out>     :error:0:

in> nfm listusers
out>     christine
    jack
    sam
    :error:0:
```

Unless otherwise indicated in the following section, all subcommands will generate a single line of output as listed in the first two examples above.

Subcommands

The following subcommands are available. Note, the use of the term `job` is synonymous with a submitted instance of a plan:

export	Export records from NFM database to text file.
exportx	Export records using wildcard specifiers.
exportc	Export records from NFM database to text file. Allows comma separated values.
import	Import records from text file into NFM database.
browsedir	Display directory listing from a node.
browseroot	Display root directory from a node.
callplan	Run a plan and wait for its completion.
canceljob	Cancel a running plan instance.
cancelplan	Cancel all plan instances of a particular plan name.
delay	Wait for specified number of seconds.
deleteasource, deleteday, deletefileset, deletehour, deletejob, deletemodel, deletenode, deletenodegroup, deleteplan, deleteuser	Delete specified record items.
enablejobnode	Continue a held node in a plan instance**.**

excludejobnode	Exclude a particular node in a plan instance.
genkeys	Generate file encryption public/private key pair.
getaudit	Print a particular audit.
getday, getfileset, gethour, getjobstatus, getmodel, getnode, getplan	Display specified record items.
holdjob	Hold a plan instance.
holdjobnode	Hold a particular node in a plan instance.
listaudits	Print a series of audits.
listclients, listdays, listfilesets, listhours, listjobs, listjobsx, listmodels, listnodes, listnodegroups, listnodemembers, listplans, listusers	List all the items of the specified record type.
pfile	Creates a password file.
putasource, putaudit, putday, putfileset, puthour, putmodel, putnode, putplan	Add or update specified record items.
resumejob	Continue a held or canceled plan instance.
retryjob	Retry failed operations in a finished plan instance.
setjobenv	Set an environment variable for subsequent use within a plan.
statnode	Contact a node and retrieve NFM client version number.
submitplan	Submit a plan (create a plan instance).

The following section details the subcommands available with the `nfm` command.

export,TEXTFILE,DATABASEFILE,DATAPATH,[ENTRY1,ENTRY2,ENTRY3…]

Description: Export NFM database records (`plans`, `filesets`, `audits`, etc.) to a text file representation. This allows supplying data like audit records to some outside processing from NFM. When used in conjunction with import, this allows copying records between database files in different locations or on different NFM server computers. This can also be used to make changes to NFM configurations using a text editor, or generate NFM records like `filesets` from other software.

Note, the `export`, `exportx` and `import` functionality is only available through the NFM command line interface. Unlike most other NFM commands, there is no direct way to invoke them from the GUI. The NFM command is of course available to be run from a plan execute function just as any stand-alone command.

TEXTFILE	(`req`, `string`) The name of the text file to create. Note, this text file will be overwritten if it already exists.
DATABASEFILE	(`req`, `string`) The name of the NFM database file, can be one of the following: **AUDITS, DAYS, ENVIRONMENTS, FILESETS, HOURS, JOBNODES, JOBS, METHODS, MODELS, MODEMS, NODEGROUPS, NODES, PLANS, QPLANS, USERGROUPS, USERSN.**
DATAPATH	(`opt`, `string`) This parameter can be used to specify a particular directory in which to find the database file. If left blank, records will be

exported from NFM's current database directory. This option allows you to copy records to and from a backup copy of a database file for example.

ENTRY1,ENTRY2,ENTRY3…
(opt, strings) You can specify one or more names of records to export here. If no entries are supplied, all of the records in the specified database will be exported. The exportx subcommand specified below, additionally allows you to specify entries using wildcards.

Important: When exporting records to a text file any existing text file is deleted; however, when importing to a database file, records are added to existing ones.

exportx,TEXTFILE,DATABASEFILE,DATAPATH,[ENTRY1,ENTRY2,ENTRY3…]

Description: Export NFM database records with wildcards. This is functionally equivalent to the export subcommand (see above) with one notable exception. The ENTRY fields may contain wildcards (* and ?) to allow selecting multiple similar named records. When using exportx from a UNIX shell, it is important that you provide double quotes around each of the ENTRY fields to keep the shell from trying to expand the wildcards.

Important: When exporting records to a text file any existing text file is deleted; however, when importing to a database file, records are added to existing ones.

exportc,TEXTFILE,DATABASEFILE,DATAPATH,CSV

Description: Export NFM database records with optional comma separated values. This is functionally equivalent to the export subcommand (see above) with the additional option for specifying comma separated values (or CSV). Specify a "1" for this parameter to indicate that output fields should be comma separated instead of being in the form "NAME=VALUE". The following additional changes will be in the output file:

- A header line will be created that contains the tag names of each of the output fields. These will also be comma separated.
- Comma's that are in the fields will be escaped with a backslash '\'.
- Backslashes that are in the fields will be escaped with a backslash '\'.

import,TEXTFILE,DATABASEFILE,DATAPATH,NOMODTIME

Description: Import NFM database records from a text file. This is the reverse operation of export (see above).

TEXTFILE (req, string) The name of the text file to read records from. Note, all of the records in this text file will be imported.

DATABASEFILE (req, string) The name of the NFM database file to create records in. See export above for a list. Note, that unlike export overwrites the output text file entirely, the import function will only overwrite existing database records that are the same name as the ones in the text file. It will have no affect on other existing records in the database.

DATAPATH (opt, string) This parameter can be used to specify a particular directory in which to find the database file. If left blank, records will be imported too NFM's current database directory.

NOMODTIME (opt, string) Tells NFM not to update the modification time of each record (if it applies), but instead to set it to the value stored in the input text file. Specify Y or 1 to enable this option. The default is to set the modification time to the current time.

Important: When exporting records to a text file any existing text file is deleted; however, when importing to a database file, records are added to existing ones.

browsedir,NODE,DIRECTORY,START#,MAX#,EXPANDFLAG

Description: Get a directory list from an NFM client node.

NODE (req, string) The node name to browse.
DIRECTORY (req, string) The starting directory for the browse. (Use browseroot first to get the root directory specification if you want to browse from the root directory).
START# (opt, dec) The first file to list, default=1.
MAX# (opt, dec) The last file to list default= the total number of files.
EXPANDFLAG (opt, dec) Specify 1 to expand all subdirectories, default=0.

Output: Multiple lines followed by error message:

```
FILENAME,TYPE,SIZE,DATE\n
FILENAME,TYPE,SIZE,DATE\n
. . .
```

FILENAME File name.
FILETYPE File type. Set to one of the following:
0 = normal file.
1 = directory.
2 = Windows driver letter (considered a subdirectory of the root directory).
SIZE The file size.
DATE The files modification date and time.

browseroot,NODE

Description: Return the default root directory for the specified NFM client node.

NODE (req, string) The node name to browse.

Output: Single line followed by error message:

```
DIRECTORY
```

NOTE: This will come back as an empty string on a Windows client and / on a Unix client.

callplan,PLANNAME,JSRCNODE,JTRGNODE,HISTORY,EXITRC,SHOWAUDITS\n
NAME,VALUE\n
NAME,VALUE\n
...

Description: Run a plan immediately, wait for it to complete before returning.

PLANNAME	(req,string)	The plan name to submit.
JSRCNODE	(opt, string)	Source node name.
JTRGNODE	(opt, string)	Target node name.
HISTORY	(opt, string)	History, set to DEFAULT, NONE, SUMMARY or DETAIL.
EXITRC	(opt, string)	Specify Y to cause the nfm command to exit with the return code of the plan. Default is N. The disadvantage of using this option is that a non-zero return code will not be enough information to determine if the plan actually ran or not (since the return code could be an indication of a failure to start the plan). In this case examination of the output message may be necessary.
SHOWAUDITS	(opt, string)	Specify Y to cause the audit messages generated by the plan to be printed out to this commands output while the plan is running. Default is N.
NAME,VALUE	(opt, string)	Environment variable name and value pairs (one per line), allows passing in information as NFM environment variables. Note, there is no equivalent functionality from the GUI Scheduler screen.

Output: Single line followed by error message:

ID,JOBRC

ID	Plan instance ID.
JOBRC	NFM return code of plan.

canceljob,ID

Description: Cancels the specified plan instance. The plan must be scheduled or active.

ID	(req, dec)	The plan instance ID.

cancelplan,NAME

Description: Cancels all plan instances of a particular plan name that are either scheduled or active.

NAME	(req, string)	The plan name.

delay,#
or
delay,HH:MM

Description: Delay for a period of time. The amount of time specified may be a number of seconds (by simply specifying a numerical value), or an absolute time of day specified by military time `00:00` through `23:59`. If the absolute time specified has already passed when the command runs, it will delay until that time the following day.

This sub-command may be used by itself or in combination with any other sub-command to delay its function. For example, to automatically cancel all plans called `updatestores` at 5:00 p.m., use the following command:

```
nfm delay,17:00,cancelplan,updatestores
```

deleteasource,NAME
deleteday,NAME
deletefileset,NAME
deletehour,NAME
deletejob,ID
deletemodel,NAME
deletenode,NAME
deletenodegroup,GROUPNAME
deleteplan,NAME
deleteuser,NAME

Description: Delete the specified object.

NAME or **ID** (`req`, `string`) The objects name or ID.

enablejobnode,ID,NODENAME
excludejobnode,ID,NODENAME
holdjobnode,ID,NODENAME

Description: Changes the state of a node with regard to the submitted plan specified.

ID (`req`, `dec`) The plan instance ID.
NODENAME (`req`, `string`) The node name.

genkeys,[NODENAME or GROUPNAME],PRIVKEY,PUBKEY,OVERWRITE,NOBACKUPS

Description: Generate new, and unique, public/private key pair on the specified node or nodes (using node group name).

NODENAME or GROUPNAME
(`req`, `string`) The node name or node group name to update.
PRIVKEY (`opt`, `string`) The name of private key file to create.
PUBKEY (`opt`, `string`) The name of public key file to create.
OVERWRITE (`opt`, `string`) Specify the optional string "OVERWRITE" to cause an existing target key file of the same name to be overwritten. SEE THE WARNING BELOW.

NOBACKUPS (opt, string) Specify the optional string "NOBACKUPS" to cause NFM to not automatically create a backup of the generated keys on the NFM server computer (stored in the NFM server "key_backups" subdirectory). SEE THE WARNING BELOW.

If a node group is specified, the individual nodes in the group will be processed one at a time. Before processing a large group of nodes, it is recommended that the user test with a single node. If this is done, make sure to delete the generated keys (or use test names) prior to trying again with the same node. This command will fail if the target key names already exist, and overwrite is not specified.

The PRIVKEY & PUBKEY options, if used, should specify file names without path information. This will cause the files to be created in the default location (NFM client "keys" directory). Although, not recommended, you may specify a full path to the file, but if you do so, any configuration or reference to these key files will also require the path information.

WARNING: Extreme caution must be exercised when using the options OVERWRITE or NOBACKUPS. The overwrite option will cause an existing key file of the same name to be deleted and lost forever (a backup may be available, based on the initial creation of that key file). Once a private key file is deleted with no backups available, any data that was encrypted with the corresponding public key will become permanently lost. Similarly, the no backups option will prevent NFM from storing a copy of the newly created keys on the NFM server computer. The automatic backup step is provided as a safeguard in the event that the local key files at the NFM client are accidentally deleted or otherwise become unavailable.

Since NFM does not automatically create any key files, the use of this command with the PRIVKEY, PUBKEY, OVERWRITE & NOBACKUPS options all left blank will create the initial default key set (public.pem & private.pem) for each node specified, along with a backup copy created at the NFM server.

Please note, the option to create a private key that is password protected is not available through this remote command. It must be done locally with the nfme utility.

getaudit,ID

Description: Prints a particular audit record.

ID (req, dec) The ID of the audit.

Output: The format of the output is identical to that of the listaudits subcommand; see the description further on in this section.

getday,NAME,YEAR

Description: Return the contents of a day schedule table.

NAME (req, string) The day schedule name.
YEAR (req, dec) The requested year for the day schedule.

Output: Single line followed by error message:

NAME,YEAR,DAYS,MODIFICATION_TIME
Days specify a 0 (off) or a 1 (on) for all the days of the year.

getfileset,NAME

Description: Returns the contents of a fileset.

NAME (req, string) The fileset name.

Output: Multiple lines followed by error message:

```
1,FSETNAME,UNIQUEFILES,SPREFIX,EXTATTRS,\n
2,TPREFIX,\n
3,DIRECTORY,FILENAME,\n
4,TARGETNAME\n
6,DSNAME,RECZ,BLKZ,DEVTYPE,VOLSERN,PRIMARY,
  SECONDARY,SPACEUNITS,RECFORMAT,VLENCODE,
  TRUNCATE,DCBMODEL,DSORG,DIRBLOCKS,RELEASE,
..CONTIG,DCB_DD,LIKE,PASSWORD,REFDD,UNIT,VOLREF,
..BUFNO,NOASATRANS,DISPSTAT,DISPNORM,DISPCOND,
..DATACLASS,MGMTCLASS,STORCLASS,ALLOCPDSE,
  RETDAYS,EXPIREDATE,TRAILBLANK,VOLCOUNT,
  WRAPRECORD\n
```

See the individual field descriptions below.

putfileset,1,FSETNAME,EXTENSIONS,COMPRESS,SPREFIX,EXTATTRS\n 2,TPREFIX,\n 3,FILETYPE,FILENAME,\n 4,TARGETNAME\n 6,DSNAME,RECZ,BLKZ,DEVTYPE,VOLSERN,PRIMARY, SECONDARY,SPACEUNITS,RECFORMAT,VLENCODE,TRUNCATE, DCBMODEL,DSORG,DIRBLOCKS,RELEASE,CONTIG,DCB_DD, LIKE,PASSWORD,REFDD,UNIT,VOLREF,BUFNO,NOASATRANS, DISPSTAT,DISPNORM,DISPCOND,DATACLASS,MGMTCLASS, STORCLASS,ALLOCPDSE,RETDAYS,EXPIREDATE,TRAILBLANK, VOLCOUNT,WRAPRECORD\n

Description: Updates the contents of a fileset.

Field descriptions for GETFILESET and PUTFILESET:

KEY (req, string) An identifier for this line of input set to one of the following:

- 1 Indicates this is the header record. The first line must be this.
- 2 Indicates this is a target header record. If one exists it must be the seconds line.
- 3 Indicates this is a file record.
- 4 Indicates this is a file target record.
- 6 Indicates this is a file host record.

For each fileset record (type 3) entered, the following optional entries 4 through 6 are automatically grouped with the 3 record constituting a single files records.

Header fields:

FSETNAME	(req, string)	The fileset name.
EXTENSIONS	(opt, string)	Specify Y to enable node extensions option. Default is N.
COMPRESS	(opt, string)	Compression option. Specify COMPRESSON, COMPRESSOFF or COMPRESSDEF to set compression on, off or system default for transfers of this fileset.
SPREFIX	(opt, string)	The source prefix string.
EXTATTRS	(opt, dec)	Specify 1 to enable external attribute feature. Default is 0 or off.

Target Header fields:

TPREFIX	(opt, string)	The target prefix string.

Source File fields:

FILETYPE	(opt, string)	Specify Y to indicate that the file is a directory. Default is N (regular file).
FILENAME	(req, string)	The file name.

Target File fields:

TARGET	(opt, string)	The target file string.

Host fields:

DSNAME	(opt, string)	The data set name.
RECZ	(opt, dec)	The record size.
BLKZ	(opt, dec)	The block size.
DEVTYPE	(opt, string)	The device type.
VOLSERN	(opt, string)	The volume serial number.
PRIMARY	(opt, string)	Primary space allocation unit.
SECONDARY	(opt, string)	Secondary space allocation unit.
SPACEUNITS	(opt, dec)	Specify one of the following: 1 Indicates Blocks. 2 Indicates Tracks. 3 Indicates Cylinders.
RECFORMAT	(opt, string)	Record format.
VLENCODE	(opt, dec)	Specify 1 to enable variable length encoding. Default is 0 or off.
TRUNCATE	(opt, dec)	Specify 1 to allow truncation. Default is 0 or off.

gethour,NAME

Description: Return the contents of an hour schedule table.

NAME (req, string) The hour schedule name.

Output: Single line followed by error message:

NAME,YEAR,DAYS,MODIFICATION_TIME
Days specify a 0 (off) or a 1 (on) for all the days of the year.

getjobstatus,ID

Description: Retrieve the status of a submitted plan ID.

ID (req, dec) The submitted plan ID.

Output: Multiple line output followed by error message:

PLANNAME,CSTATE,STARTTIME\n
PLANNAME,ID,USERNAME,SUBTIME,STARTTIME,
CSTATE,JOBRC,SRCNODE,TRGNODE,RECURSI,
RECURSCT,DAYTAB,HOURTAB,HISTORY\n

NOTE: This description is only true for a job that is not in the ACTIVE state.

Line 1:

PLANNAME Plan name.
CSTATE Current state, set to one of the following: READY, HELD, ACTIVE, ABORTED, REJECTED, CANCELLED and FINISHED.
STARTTIME Starting time of job.

Line 2:

PLANNAME Plan name.
ID Plan instance ID.
USERNAME User that submitted plan.
SUBTIME Time plan was submitted.
STARTTIME Time plan started.
CSTATE Current state of plan.
JOBRC NFM return code of plan.
SRCNODE Default source node.
TRGNODE Default target node.
RECURSI Recursion interval.
RECURSCT Recursion count.
DAYTAB Day table.
HOURTAB Hour table.
HISTORY History option.

getmodel,NAME

Description: Retrieve the contents of a model record.

NAME (req, string) The model schedule name.

Output: Single line followed by error message:

```
NAME,OS_NAME,PRIMARY_METHOD,BACKUP_METHOD,NFM_CLIENT
    NFM_CLIENT: Y=yes N=no
```

getnode,NAME

Description Retrieve the contents of a node record.

NAME (req, string) The model schedule name.

Output: Single line followed by error message:

```
NAME,MODEL,COMNAME,COMNAME2,OFFLINE,
    ONHOLD,DIAGNOSTIC,ENCRYPT,\n
```

getplan,NAME

Description: Retrieves the contents of a plan.

NAME (req, string) The plan name.

Output: Multiple lines followed by error message:

```
H,PLANNAME,SRCNODE,TRGNODE,HISTORY,1INSTANCE
    RETRY,LONGNAME,DIAGNOSTIC\n
P,PHASENAME,TERMFLAG,TERMVAL,CONDFLAG,
    CONDPHASE,CONDREL,CONDVAL,DUALASYNC,
    DUALPROPRC,CUTOFFINT,CUTOFFVAL,REPEATINT,
    REPEATVAL,REPEATMAX,REPCONDFLAG,REPCONDREL,
    REPCONDVAL\n
F,FUNCTION,PARMS,SRCNODE,TRGNODE,
    TERMFLAG,TERMVAL,HOLDERR,PRIORITY,
    CPRFLAG,RENSFLAG,OVERWRITE,TMPNAMES,
    STREAMING,RELATIVE,ASYNC,SHELL,TEMPLATE,
    MCPROF,TIMEOUT,ERRORLEVEL,PRESTIME,APPEND,
    CLEAR,DELETESRC,PRESATTRS,\n
```

See the individual field descriptions below.

```
putplan,H,PLANNAME,SRCNODE,TRGNODE,HISTORY,1INSTANCE,
          RETRY,LONGNAME,DIAGNOSTIC\n
      P,PHASENAME,TERMFLAG,TERMVAL,CONDFLAG,CONDPHASE,
        CONDREL,CONDVAL,DUALASYNC,DUALPROPRC,CUTOFFINT,
        CUTOFFVAL,REPEATINT,REPEATVAL,REPEATMAX,REPCONDFLAG,
        REPCONDREL,REPCONDVAL,\n
      F,FUNCTION,PARMS,SRCNODE,TRGNODE,TERMFLAG,TERMVAL,
        HOLDERR,PRIORITY,CPRFLAG,RENSFLAG,OVERWRITE,TMPNAMES,
        STREAMING,RELATIVE,ASYNC,SHELL,TEMPLATE,MCPROF,
        TIMEOUT,ERRORLEVEL,PRESTIME,APPEND,CLEAR,DELETESRC,
        PRESATTRS,\n
```

Field descriptions for `GETPLAN` and `PUTPLAN`:

Key: (`req`, `string`) An identifier for this line of input set to one of the following:
`H` Indicates this is the `header` record. The first line must be this.
`P` Indicates this is a `phase` record.
`F` Indicates this is a `function` record.

Header fields:

`PLANNAME` (`req`, `string`) The plan name.
`SRCNODE` (`opt`, `string`) The default source node for this plan..
`TRGNODE` (`opt`, `string`) The default target node for this plan.
`HISTORY` (`opt`, `string`) The default history level for this plan. Set to `NONE`, `SUMMARY` or `DETAIL`.
`1INSTANCE` (`opt`, `string`) Set to `Y` to restrict only one instance of this plan being scheduled at a time. Default is `N`.
`RETRY` (`opt`, `string`) Set to `Y` to restrict only one instance of this plan being scheduled at a time. Default is `N`.
`LONGNAME` (`opt`, `string`) Not normally used, should be blank.
`DIAGNOSTIC` (`opt`, `string`) Set to `Y` to enable diagnostic mode for this plan. Default is `N`.

Phase fields:

`PHASENAME` (`req`, `string`) The name of this phase.
`TERMFLAG` (`opt`, `string`) Set to `Y` to set default break from phase threshold. Default is `N`.
`TERMVAL` (`opt`, `dec`) Termination threshold corresponding to `TERMFLAG`.
`CONDFLAG` (`opt`, `string`) Indicates when this phase runs. Speicfy `DEFAULT`, `SEQPHASE` or `DUALPHASE` to represent `runs in sequence`, `runs after phase` or `runs with phase` respectively.
`CONDPHASE` (`opt`, `string`) If `CONDFLAG` is set to `SEQPHASE` or `DUALPHASE`, this field should be set to the `runs after` or `runs with` phase name.
`CONDREL` (`opt`, `string`) If `CONDFLAG` is set to `SEQPHASE`, this field represents the conditional run logic for the previous phase. Set to `GREATER`, `GREATEQ`, `LESSER`, `LESSEQ`, `EQUAL` or `NOTEQUAL`.
`CONDVAL` (`opt`, `dec`) Used with `CONDREL` to determine if a sequential phase will run. Set to the appropriate return code.
`DUALASYNC` (`opt`, `dec`) If `CONDFLAG` is set to `DUALPHASE`, this field turns on asynchronous logic. Set to `N` for synchronous, `Y` for asynchronous.
`DUALPROPRC` (`opt`, `dec`) If `CONDFLAG` is set to `DUALPHASE`, this field turns on propagates `rc` logic. Set to `Y` for yes. Default is `N`.

Function Fields:

`F,function,parms,srcnode,trgnode,termflag,termval,holderr\`

`FUNCTION` (`req`, `string`) Specifies the function type. Select `TRANSFER`, `EXECUTE`, `DELETE` or `RUN`.

PARMS (req, string) Specify a fileset name for a TRANSFER, DELETE, RENAME, or CUSTOM FTP type function. Specify a command string for an EXECUTE type function. Specify a number of seconds for a DELAY type function.
SRCNODE (opt, string) The source node for this function unless a default is implied.
TRGNODE (opt, string) The target node for this function unless a default is implied.
TERMFLAG (opt, string) Set to Y to set a break from this phase threshold. Default is N.
TERMVAL (opt, dec) Termination threshold corresponding to TERMFLAG.
HOLDERR (opt, string) Set to Y to turn on hold on error logic. Default is N.
PRIORITY (opt, dec) Sets the functions priority. Specify 0, 1 or 2 for low, medium or high respectively.
CPRFLAG (opt, string) Set to Y to turn on checkpoint restart logic for this function. Default is N.
RENSFLAG (opt, string) Set to Y to turn on redirect naming option. Default is N.
OVERWRITE (opt, string) Set to Y to make overwriting files during transfers automatic. Set to N to make this an error condition.
TMPNAMES (opt, string) Set to Y to enable using temporary names during transfers. Default is N.
STREAMING (opt, dec) Specify a non-zero number as the streaming interval to define a streaming type transfer. Default is 0 or streaming off.
RELATIVE (opt, string) Set to Y to make a DELAY function wait for a specified number of seconds. Set to N to wait until an absolute time of day represented by number of seconds.
ASYNC (opt, string) Set to Y to cause an EXECUTE type function to not wait for the program to complete but return control immediately. Default is N, wait on program.
SHELL (opt, string) Set to Y to turn on the Use Shell option. Default is N.
TEMPLATE (opt, string) The name of an existing FTP custom template for CUSTOM FTP type function.
MCPROF (opt, string) The name of an existing Multicast profile for TRANSFER type function.
TIMOUT (opt, dec) Timout value in seconds for EXECUTE type function.
ERRORLEVEL (opt, dec) Error level value for EXECUTE type function.

listaudits,LEVEL,STATE,PLANNAME,PLANID,FSETNAME,NODENAME,STARTTIME,ENDTIME,COUNT,SSTRING

Description: List audit records.

LEVEL (opt, dec) One of the following numbers to represent which audit type (Informational, Warning or Error) or which combination of types to display as follows:

1 Informational audits only.
2 Warning audits only.
3 Informational and Warning audits.
4 Error audits only.
5 Informational and Error audits.
6 Warning and Error audits.
7 Display all types (Same as leaving this field blank).

STATE	(opt, string) Display only audits in the read or unread state, specify READ or UNREAD, otherwise leave this field blank to display both states.
PLANNAME	(opt, string) Specify a plan name to display only audits created by any instance of a specific plan.
PLANID	(opt, dec) Specify a plan instance ID to display only audits created by a specific plan instance.
FSETNAME	(opt, string) Specify a particular fileset name to only display audits related to a specific fileset.
NODENAME	(opt, string) Specify a particular node name to only display audits related to that node.
STARTTIME	(opt, date) Specify a particular date and time from which to display audits created past that time.
ENDTIME	(opt, date) Specify a particular date and time from which to display audits created prior to that time. Use in combination with the previous field to create a date range.
COUNT	(opt, dec) Use this field to specify a maximum number of audits to display that match the other criteria, default = no maximum.
SSTRING	(opt, string) Specify any desired string of characters that are searched for in the text part of the audit messages. Only audits containing this text are displayed.

Output: Multiple lines followed by error message:

TIMESTAMP,ID,LEVEL,STATE,PLANNAME,PLANID,FSETNAME,NODENAME,PLANRC, NFMERRNUM,SYSRC,TEXT\n

TIMESTAMP	Reflects the creation time of the audit.
ID	The ID of the individual audit (do not confuse with plan ID). Note, this field is used as input for the getaudit subcommand, which would be necessary for retrieving the entire contents of a multi-line audit.
LEVEL	The audit type, either INFO, WARNING, or ERROR.
STATE	The acknowledgement status of the audit, either READ or UNREAD.
PLANNAME	Name of the plan (if any) that generated this audit.
PLANID	The plan instance ID that generated this audit. A value of 0 indicates the audit was not generated from within a plan.
FSETNAME	The fileset name (if any) associated with this audit.
NODENAME	The node name (if any) associated with this audit.
PLANRC	The plan type return code associated with the audit.
NFMERRNUM	The NFM error code value.
SYSRC	An operating operating system or subsystem specific return code.
TEXT	The actual text of the audit. If the text ends with three periods "…" it is probably a multi-line audit and only the first line is being displayed. Use the getaudit subcommand to retrieve the entire audit message. Multi-line audits are most commonly generated from program execution output.

listclients (List nodes that are nfm clients.)
listdays
listfilesets
listhours
listjobs
listmodels

listnodes
listplans
listnodegroups
listusers

Description: List the various objects.

Output: Multiple lines followed by error message:

```
item_name,
item_name,
item_name,
. . .
```

listjobsx,SCHEDULED,HELD,COMPLETED,ACTIVE

Description: List all current plan instances.

SCHEDULED,HELD,COMPLETED,ACTIVE Put a `1` in each desired field to filter the list based on the plan instance state. Note, `COMPLETED` includes aborted, canceled and rejected.

listnodemembers,GROUPNAME

Description: List the members of a group.

GROUPNAME (`req`, `string`) The groupname.

Output: Multiple lines followed by error message:

```
member,
member,
member,
. . .
```

pfile,USERNAME,PASSWORD,FILE

Description: Create an encrypted password file.

USERNAME (`req`, `string`) The user name for whom the password applies.
PASSWORD (`req`, `string`) The password.
FILE (`req`, `string`) The name of the file to create. This should be a fully qualified path.

putasource,SOURCENAME,DIRECTORY

Description: Create a new audit source location.

SOURCENAME (`req`, `string`) The name of the audit source.

DIRECTORY (req, string) The directory location (this should not include a final director separator). This directory may be suffixed with the NFM year environment variable similar to the following example:

"/var/tps/dba/archive/$(NFMTM_YR)"
(Note, this is how the pre-supplied default entry "ARCHIVE" is defined).

This signifies special processing when this audit source is used as the archive target location (specified in the System Settings). This causes the audits to be placed in subdirectories representing the current year during archiving. This also allows the user to select the desired year in the location drop-down box on the history screen to aid in viewing audits.

putaudit,MESSAGE,ERRNUM,FILESET,PLAN_ID,LEVEL,NODENAME,PLANNAME, PLAN_RC,SYSTEM_RC

Description: Create a custom audit message.

MESSAGE (req, string) The message itself.
ERRNUM (opt, dec) The NFM error code.
FILESET (opt, string) A fileset name.
PLAN_ID (opt, dec) The plan instance ID. Use the environment variable NFMPLNID for this command executed from a plan.
LEVEL (opt, string) Specify INFO, WARNING or ERROR.
NODENAME (opt, string) A node name. Use the environment variables NFMSNODE or NFMTNODE to specify the source or target node when using this command executed from a plan.
PLANNAME (opt, string) The current plan name. Use the environment variable NFMPLAN for this command executed from a plan.
PLAN_RC (opt, dec) Specify the desired plan type return code.
SYSTEM_RC (opt, dec) Specify a system dependent return code.

NOTE: If this command is called from within a plan, either directly or from within a script, it will be desirable for certain fields to automatically default. For this reason the PLAN_ID, PLANNAME and NODENAME fields should be left blank. This will cause them to automatically be filled in with the current plan name, instance ID, and target node name.

putday
puthour
putmodel
putnode

Description: Each of these put commands takes the same input as their respective get commands produce as output (except for the error message). Refer to the appropriate get command for the format.

holdjob,ID

Description: Puts the specified plan instance on hold. The plan must be scheduled or active.

ID (req, dec) The plan instance ID.

resumejob,ID

Description	Resume running a held or canceled plan instance. If the plans start time is still in the future, it becomes scheduled again. If the plan had already been active, it resumes running where it left off.
ID	(req, dec) The plan instance ID.

retryjob,ID

Description	Perform a retry operation on a completed plan that had errors.
ID	(req, dec) The plan instance ID.

setjobenv,NAME=VALUE

Description	Set an environment variable for subsequent use within a plan.
NAME	(req, string) The name of the environment variable to set.
VALUE	(req, string) The value to set this environment variable to.
	This command allows you to set virtually any string into an environment variable available for subsequent plan operations. This would, for example, allow you to use Unix or Windows commands to create a target file name for a transfer that would immediately follow this function. This command is only available when called from within a plan.

statnode,NODENAME

Description	Contact a node and print out NFM client version number.
ID	(req, dec) The node name.

submitplan,PLANNAME,STARTTIME,RECURSI,RECURSC,RECURDT, RECURHT,JSRCNODE,JTRGNODE,ISTATE, HISTORY\n NAME,VALUE\n NAME,VALUE\n ...

Description:	Schedule a plan for execution.
PLANNAME	(req, string) The plan name to submit.
STARTTIME	(opt, date) The starting time for the plan. The default if not entered will be the current time.
RECURSI	(opt, string) Recursion interval, set to SECOND, MINUTE, HOUR, DAY, WEEK, MONTH or YEAR.
RECURSC	(opt, dec) The number of the above intervals to repeat every.

RECURDT	(opt, string)	Day schedule name.
RECURHT	(opt, string)	Hour schedule name.
JSRCNODE	(opt, string)	Source node name.
JTRGNODE	(opt, string)	Target node name.
ISTATE	(opt, string)	Initial state, set to READY or HOLD (default=READY).
HISTORY	(opt, string)	History, set to DEFAULT, NONE, SUMMARY or DETAIL.
NAME,VALUE	(opt, string)	Environment variable name and value pairs (one per line), allows passing in information as NFM environment variables. Note, there is no equivalent functionality from the GUI Scheduler screen.

NFM remote server interface

Overview

The NFM remote server interface or nfmi command provides a command line interface to the NFM server from any NFM client computer. This can be used when it is desirable to have the NFM client computer initiate NFM activity rather than scheduling it from the NFM server. The following section details the use of the nfmi command.

Usage

The nfmi program ships with the NFM client package, and is located in /usr/lpp/tps/nfmc/bin on AIX and /opt/tps/nfmc/bin on most other versions of UNIX, and should be available on the path from a symbolic link in /usr/bin/. On Windows the nfmi program exists as C:/Program files/TPS Systems/NFMClient/nfmi.exe by default. The nfmi program has the following syntax.

```
nfmi [-i] parm=value parm=value . . .
```

The nfmi program sends a command string and a separate list of assigned parameter pairs in the form parm=value as seen above, up to the NFM server for execution. The command string consists of the command itself as well as any command line arguments. Each command is actually an existing executable program (most likely a script or batch file) that resides in /var/tps/nfm/commands on the NFM server computer (on Windows this directory is in the main NFM install location). The remaining command line arguments are passed to the command, and the parameter pairs are available to the program in the form of OS environment variables. The -i optionally instructs nfmi to run in interactive mode.

The nfmi program performs the following steps once started:

1. The nfmi program scans for a default configuration file on the local NFM client called nfmi.cfg (located in /var/tps/nfmc or the NFM clients main install location). This text file may contain any number of parameter pairs in the form parm=value (one per line). These are read into nfmi's parameter list. There are certain parameter names that are reserved indicating additional meaning or action. These are:

   NFMSERV=*hostname* Indicates the IP address or hostname of the NFM server computer to contact. This is a required field.

   NFMNODE=*nodename* Specifies the NFM node name that the NFM server knows this node as. This is a required field.

CONFIG=*filename* This instructs `nfmi` to also read in an additional configuration file of the specified name. This is optional, also, this is the only parameter name that may appear more than once.

COMMAND=*command string* This is one of the methods of entering a command to run. If `nfmi` is not run interactively, than this or the following parameter is required.

CMDFILE=*filename* This is another way of entering a command. This filename may contain one or several commands to run in sequence. Note: If clients will be consistently running the same set of commands. It is recommended that those commands be in the command script on the server, so they appear as a single command to the client.

EXITRC=YES This option causes the `nfmi` program to exit with the remote command's exit value. This option does not apply to running `nfmi` in interactive mode (with `-i`). If `nfmi` runs multiple commands (with the `CMDFILE` option), the exit value will be set to the last command that generated a non-zero value.

EXITFAIL=YES If the `CMDFILE` option is used to run multiple commands, this option instructs `nfmi` to halt processing of subsequent commands if any of the commands generates a non-zero return code.

2. Any parameter pairs typed in, as arguments to `nfmi` will be added to the parameter list.

3. If `nfmi` is not run interactively, it will send up each command string along with the parameter pairs to the NFM server, execute the command, print the output locally, and then exit.

4. If `nfmi` is run interactively, the prompt `nfmi>` will appear and the user will be required to type commands. Certain commands are interpreted locally, these are:

   help Prints a list of these local commands. It will then run the `help` command on the server if it exists.

   quit Quit the `nfmi` program.

   set *parm=value* Allows the user to type in a parameter value.

   Anything else typed as input is assumed to be a command string and is sent to the server for execution.

Configuration

Several steps must be performed to enable the NFM remote server interface to function. These include the following:

1. The actual commands must be created and placed in the NFM servers remote command directory (see the sample in the next section). When a command is run from an NFM client, it will be done for a particular user so that NFM user restrictions can be enforced. It is possible to make certain remote commands limited to certain users. To do this, create a subdirectory in the command directory that matches the users name. Place the appropriate commands in this directory. These commands will now be

available only for that user. Commands in the base directory will be available for all users.

2. The NFM server must have this service turned on. This is accomplished through the NFM user interface in the System Settings. Under the Server folder, the check box labeled `Enable client initiated commands` must be checked on.

3. The following two steps must be done for every NFM client node that is to be enabled for this service:

    - A configuration file, as discussed above, should be created on each client with the minimum required parameters `SERVADDR` and `NFMNODE` set.
    - On the NFM server, the node (or corresponding model) record must have the fields `Nfmi User Name` and `Nfmi Password` fields set to a valid NFM login ID. Any imbedded NFM commands issued on the server as part of a remote command will run with the authority of this user. (A new user record may be added for each node not representing a real person if desired).

Sample

The following sample shows the configuration and usage of the `nfmi` command.

At the request of each 4690 store, send the file `c:/tmp/inventory` to the NFM server `/tmp` directory. Append the store name (node name) to the end of the file.

Set up the appropriate fileset and plan to accommodate the transfer, leave the source node blank, as it will be supplied by each separate command.

```
Fileset INVFILE:
    Source prefix=c:/tmp/          Target prefix=/tmp/
    Source file=INVENTORY          Target file=INVENTORY.$(NFMSNODE)

Plan UPLOADINV:
    Function transfer    Source Node=(blank)    Target Node=CENTRAL
```

Node extensions cannot be used to append the node name because they only apply to a group operation, whereas, each of these transfers will occur as a separate instance of the plan. Instead the target file uses the reserved environment variable supplied by NFM to indicate the current source node name.

The command that will actually run will be called `uploadinv`. The appropriate script by this name will be placed in the NFM servers remote command directory and contain the following lines.

```
#! /bin/ksh
nfm callplan,UPLOADINV,$NFMNODE
```

or on windows the file `uploadinv.cmd`

```
nfm callplan,UPLOADINV,%NFMNODE%
```

This will cause the plan to be run by using the command `uploadinv` with `nfmi`. The NFMNODE environment variable will pass the desired source node name to the `nfm` command. This in turn will be substituted for the file extension `$(NFMSNODE)` in the `fileset` target field when the plan is run resulting in the desired copy from the 4690 computer to the NFM server.

`c:/tmp/INVENTORY` ➔ `/tmp/INVENTORY.store001`

Each store could perform a similar copy at their request. Although the example here shows a single operation performed in the command script (in this case the `nfm` command), the scripts can contain several commands that will all be executed from a single command sent up from `nfmi`. These can be any commands available on the system, not just the `nfm` command. For more information on the `nfm` command line interface, see the description earlier in this chapter. For more information on the NFM environment variables and their use from within plans, (see "Using environment variables," Chapter 6).

NFM file encryption

Overview

The NFM file encryption feature allows files to be stored on disk in an encrypted form protecting them from unauthorized access. This functionality should not be confused with the current transit encryption that NFM provides to safeguard data as it is being transmitted, which includes the use of SSL or NFM native encryption that may be designated in the node and model definitions.

NFM file encryption is based on OpenSSL technology and currently utilizes an AES algorithm with a 32 bit key to provide a commercial grade, industry standard level of encryption. The encryption options allow for either a pass-phrase to be used, or a public/private key file to be designated.

The following methods are available to perform file encryption or decryption:

- Automatically encrypt/decrypt during file transfers in a plan.
- Using specific encrypt or decrypt functions in a plan.
- Using the stand-alone utility (nfme).

Automatic

NFM can perform the following tasks as part of a file transfer:

- Decrypt a currently encrypted source file resulting in a plain text target file.
- Encrypt a plain text source file resulting in an encrypted target file.

The following characteristics of these operations should be noted:

- A decrypt on the source file may be combined with an encrypt on the target file if the user wishes to re-encrypt a file using different options. This would be typical for example if a file is encrypted with the source node's default private key, and the user wishes to have it re-encrypted using the target node's private key.
- Neither of these operations alters the source file in any way. The plan functions "encrypt" and "decrypt" may be used to do this.
- If either of the operations is specified, the data in transit is always going to be encrypted, even if the node or model configurations do not call for this.

Automatic encrypt/decrypt is configured by specifying the appropriate options in the fileset definition. Please see "Filesets" section "Encryption Folder" in Chapter 4 "Configuration" for more information.

Plan Functions

An NFM plan can perform specific encrypt or decrypt functions as desired. These are useful when an encryption or decryption operation is not associated with a transfer or needs to be scheduled as a separate task. These functions reference a fileset, and utilize the same file encryption options used for automatic encryption. See "Plans" section "Functions" and "Filesets" section "Encryption Folder" in Chapter 4 "Configuration" for more information.

Key File Creation

When performing file encryption that uses public/private key files (as opposed to passwords), NFM will use the files "public.pem" and "private.pem" in the NFM client "keys" subdirectory by default unless an alternate set of keys is specified. These default keys must be manually created by the user as a one time configuration step. This can be done in one of two ways:

- Locally at the NFM client using the "nfme" utility. This is described in the next section.
- Remotely from the NFM server. This is done through the NFM command line interface with the "genkeys" subcommand. Refer to section "NFM command line interface" in Chapter 8 "Tools" for more information.

nfme utility

The NFM file encryption utility or `nfme` program provides a command line interface to perform local file encryption/decryption on any NFM client computer that supports the file encryption feature. This command is also used to perform certain maintenance type activities associated with file encryption. The following section details the use of the `nfme` command.

The `nfme` program ships with the NFM client package, and is located in `/usr/lpp/tps/nfmc/bin` on AIX and `/opt/tps/nfmc/bin` on most other versions of UNIX, and should be available on the path from a symbolic link in `/usr/bin/`. On Windows the `nfme` program exists as `C:/Program files/TPS Systems/NFMClient/nfme.exe` by default.

Configuration

Although the `nfme` program is a stand-alone utility, it must be able to communicate to the NFM server in order to perform any local encryption or

decryption operations, as well as any key generation commands. For this reason, the following configuration changes may be necessary before using it:

1. The NFM client configuration file must include the necessary information for the NFM client to contact the NFM server. This usually consists of setting the fields `SERVADDR` and `NFMNODE`. Please reference the section "NFM Client Configuration file" in Chapter 4 "Configuration" for more information on this.

2. On the NFM server, the node (or corresponding model) record must have the fields `Nfmi User Name` and `Nfmi Password` fields set to a valid NFM login ID. Any subsequent encrypt or decrypt operations performed at the NFM client will first require successful user authentication based on these fields and will be done subject to any permissions or path restrictions that would otherwise be enforced for this user accessing this node.

Additionally, all encryption or decryption operations performed by the `nfme` program, will be audited at the NFM server. This will produce error type audits for any errors encountered (including unauthorized access attempts), as well as informational audits for successful operations.

Usage

The following section describes the syntax and available options for using the `nfme` program:

```
nfme [options] file . . .
```

Primary Command Options:

`(none)`	encrypt the specified files
`-d`	decrypt the specified files
`-i`	provide information about encrypted files
`--genrsa`	generate public and private key files
`--genpub`	generate public key file from private key file

Miscellaneous Options:

`-k`	use public/private keys (no password)
`-p password`	specify password for encrypting/decrypting a file (otherwise you will be prompted)
`-t`	turn on diagnostic trace
`-v`	turn on verbose
`--private=[name]`	specify private key file name
`--public=[name]`	specify public key file name
`--private_pass=[pass]`	password to access/create private key with
`--sslver`	display OpenSSL version being used

More about specific operations

`Encrypting/decrypting files` (specified by default and with –d)

These operations encrypt or decrypt the specified list of files. By default, an encrypt operation will result in the removal of the plain text file and the

creation of the encrypted file as the original name with an extension of ".tef" added. To safely guard against any potential data loss, it is recommended that the user make a backup copy of the plain text file until confirming that an encrypt followed by a decrypt results in a successful restoration of the original file.

A decrypt operation automatically assumes a ".tef" extension exists, so it is not necessary to specify the extension during the command. The encrypted file is removed after the plain text file is created.

The following options regulate the behavior of an encrypt or decrypt operation:

The "-k" option directs the encryption to use a private key for encryption rather than a password. The absence of "-k" forces the use of a password. It is not necessary to specify "-k" for a decrypt operation, because the file is examined to determine which method to use.

The "--public" and "--private" options may be used to specify a key file other than the default ones. Each NFM client creates a default (and unique) set of keys (public.pem and private.pem) which are used automatically if no alternate key files are specified. A key file specification without a path is automatically found in the default subdirectory "keys/" off of the NFM client base directory, however a full path may be specified to an alternate location.

Creating or altering key files

The "—genrsa" creates a key file pair (public & private) used for encrypting and decrypting respectively. The NFM client automatically creates a unique set of keys, which are used by default, so this is only needed if an alternate key set is desired. The "—public" and "—private" options must also be used to specify alternate names, otherwise an error will probably result in trying to overwrite the default keys. The "—private_pass" option should be used with the "—genrsa" if the user wants to create a private key that is password protected. The "—private_pass" option is otherwise used in conjunction with any command that needs to access a private key that is password protected.

The "—genpub" command recreates the a public key file from a private key file. This command is not normally necessary unless the public key file is accidentally deleted.

User exits

Overview

User exits are customer written programs that can be invoked by an NFM client to perform security checks or other customized tasks before or after doing particular actions (such as sending a file, receiving a file, etc). Information regarding the current action including an account user name and password are provided to the user exit. The user exit can then process the information as necessary, and if it chooses, disallow the action from occurring.

Program interface

The user exit program, if configured, is loaded once when the NFM client is started. It stays loaded and is called at the following key points by the NFM client:

1. Immediately after the client program is loaded. This will occur only once (initialization request).
2. Before a file is opened for sending (pre-read request).
3. After a file is sent (post read request).
4. Before a file is opened for receiving (pre-write request).
5. After a file is received (post write request).
6. Before a program execution (pre-exec request).
7. After a program execution (post exec request).

A separate area of memory (or control block) is associated with each request and is used to relay information back and forth between the NFM client and the user exit. The following section lists the available fields in the control block and describes their usage (For the exact layout of this control block, including field sizes and offsets, please refer to the `nfmcexit.h` file shipped with the NFM client.

User exit control block

(Note: Fields that are described as text are null terminated strings that may be empty).

`LENGTH` (numeric, read only) The size in bytes of the control block. The size of the control block may increase in subsequent versions, however the size and offsets of the existing fields will not change.

`ACTION` (numeric, read only) A number representing which action is about to occur, or has already occurred. Set to one of the following:

- **`0`** Initialization. This is a single call made to the user exit after it is first loaded. It is at this point in the user exit that any one-time initialization should be performed.
- **`1`** Read request. A file open associated with sending a source file from this node.
- **`2`** Write request. A file open associated with receiving a target file to this node.
- **`4`** Exec request. Associated with executing a program on this node.

`FINISHED` (numeric, read only) Indicates whether or not the action has occurred. Set to one of the following:

- **`0`** Indicates that the action is about to occur. It is at this time that the exit may return a zero value to deny the action from occurring (see return code below).
- **`1`** Indicates that the action has already occurred.

USERID (text, read only) This field contains the optional `Account User Name` field from the Node Access Point (or NAP) record used to refer to this node in the NFM plan.

USERPASSWORD (text, read only) This field contains the optional `Account Password` field from the Node Access Point (or NAP) definition used to refer to this node in the NFM plan.

FILENAME (text, read only) This field contains a file name if the `ACTION` field is set to `1` (read) or `2` (write).

MESSAGE (text, modifiable) This field may be modified by the user exit to contain a readable error message that will be displayed in an audit message when the action that is about to be performed is denied. This only occurs when the exit returns a zero value.

REQUESTNUMBER (numeric, read only) This field contains a number unique to this request. This number will be the same for the before and after calls made from the same request.

NFMERROR (numeric, read only) This field may contain an NFM error number during an after call (`FINISHED=1`) from a specific action that failed.

USER AREA (undefined, 32 byte, modifiable) This area of the control block is available for the user exit to use for its own needs. Because the entire control block is unique per request, this area offers the user exit a place to pass information between the before and after calls for a given request. Additionally, this user area is copied from the user area of the initialization call to the user exit. This allows the user exit to perform one-time initialization (when `ACTION` is set to zero), and then set values into the user area that will be available for later calls to access.

User exit return value

During a call prior to a specific request (when `FINISHED=0`), the user exit may return with a zero to indicate that the request is to be denied. If this happens, the request will not be performed, and the NFM server will generate an audit error message indicating so. If a message has been copied to the `MESSAGE` field, it will be displayed as part of the audit error allowing the user exit to display a specific reason for denying the requested action..

The value returned from the user exit during the initialization call (`ACTION=0`), or during any of the post action calls (`FINISHED=1`) has no effect.

Restrictions

The operating environment of the user exit is restrictive. The actual NFM client threads that are performing the operations call the user exit. Because of this,

great care must be taken to preserve the integrity of the calling process and threads. The following rules must be adhered to:

1. The user exit must be dynamically loadable, supporting the following loading conventions:
   - `dlopen()` on Unix.
   - `LoadLibrary()` on Windows.
   - `fetch()` on OS/390 z/OS

2. The program must be fully reentrant. It may be called simultaneously from multiple threads. It must perform any necessary locking/synchronization while accessing outside resources.

3. It must operate fast enough to prevent performance problems.

4. It must not have any "leaks" (memory, file handles, or other system resources).

5. It must not change the security context (effective user) or any other attribute of the calling thread/process. This could negatively effect or disable the NFM client.

6. It cannot cause/throw any exceptions that will cause thread/process termination. This WILL disable the NFM client.

Sample program

The following files are shipped with the NFM client and can be found in the products base directory:

`nfmcexit.h` This include file contains definitions suitable for writing a user exit program for C or C++. Included is a structure definition for the user interface control block described above along with some useful defines.

`nfmpqc.c` This is a sample user exit program written in C. It has been compiled and tested on the various platforms that the NFM client runs on. This particular program simply examines the account user name provided, and if it is equal to "lame," it denies each of the available actions described in the previous section. This program may be used as a starting point for developing a user exit.

Configuring the user exit

Once the user exit has been built and placed in the appropriate location for loading (this process will vary by platform), the following steps should be done to enable its use:

1. Edit the NFM client configuration file, to include a line similar to the following one: (See "Encryption and connection options" in Chapter 6, "Advanced Functions" for a general description of the client configuration file.)

```
USEREXIT=PERMISSION, LIB=libnfmpqc.o, PARM1=NFMPQC
```

The field descriptions are:

USEREXIT=PERMISSION This indicates a permission query type exit. This is the only valid setting at this time.

LIB=[loadable object name] This contains the name of the loadable program object. This will be different based on the platform as follows:

libnfmpqc.o	For Unix this represents a shared object module in `/usr/lib` or elsewhere in the `LIBPATH`.
nfmpqc.dll	For Windows this refers to a `DLL`.
nfmpqc	For `OS/390` this refers to a load module.

NOTE: At this time the 4690 and VOS implementations of the NFM client do not support user exits.

PARM1=NFMPQC This must match the actual name used for the entry point into the user exit.

2. In order to use the `USERID` and `USERPASSWORD` fields in the user exit, a Node Access Point (or NAP) must be configured on the NFM server, and subsequently used from within a plan to refer to this node for action. When this occurs, the NAP's `Account User Name` and `Account Password` values will be available in these two fields in the control block, when the user exit runs.. For more information on configuring a NAP, please see "Node Access Point" in Chapter 4, "Configuration."

Email Client

Overview

The NFM Server includes an SMTP Client executable that can be used to send email from the command line of the NFM server computer. This makes it available for use within plans so that, if setup properly, it may be used to notify individuals if a plan encounters errors or other situations that require attention. The SMTP program can be found in the default NFM server program location (`C:\Program Files\TPS Systems\NFMServer\` on Windows or `/usr/lpp/tps/nfm/bin` on Unix).

Usage

The following section describes the syntax and available options for using the SMTP program:

```
smtpclient [options] [recipients] . . .
```

Message Header Options:

`-s SUBJECT`	subject line of message (Subject:)
`-t EMAIL`	address of the recipient (To:)
`-f EMAIL`	address of the sender (From:)

`-r EMAIL`	address for replies (Reply-to:)
`-e EMAIL`	address for errors (Errors-to:)
`-c EMAIL`	address for copies (Cc:)
`-n EMAIL`	address of the sender (Sender:)
`-o EMAIL`	address of the organization (Organization:)

Message Body Options:

`-B #`	number of blank lines for message body
`-T TEXT`	line of text for message body
`-F FILE`	text file name to insert in message body
`-A FILE`	file to send as attachment
`-M`	use MIME-style translation to quoted-printable

Message Options:

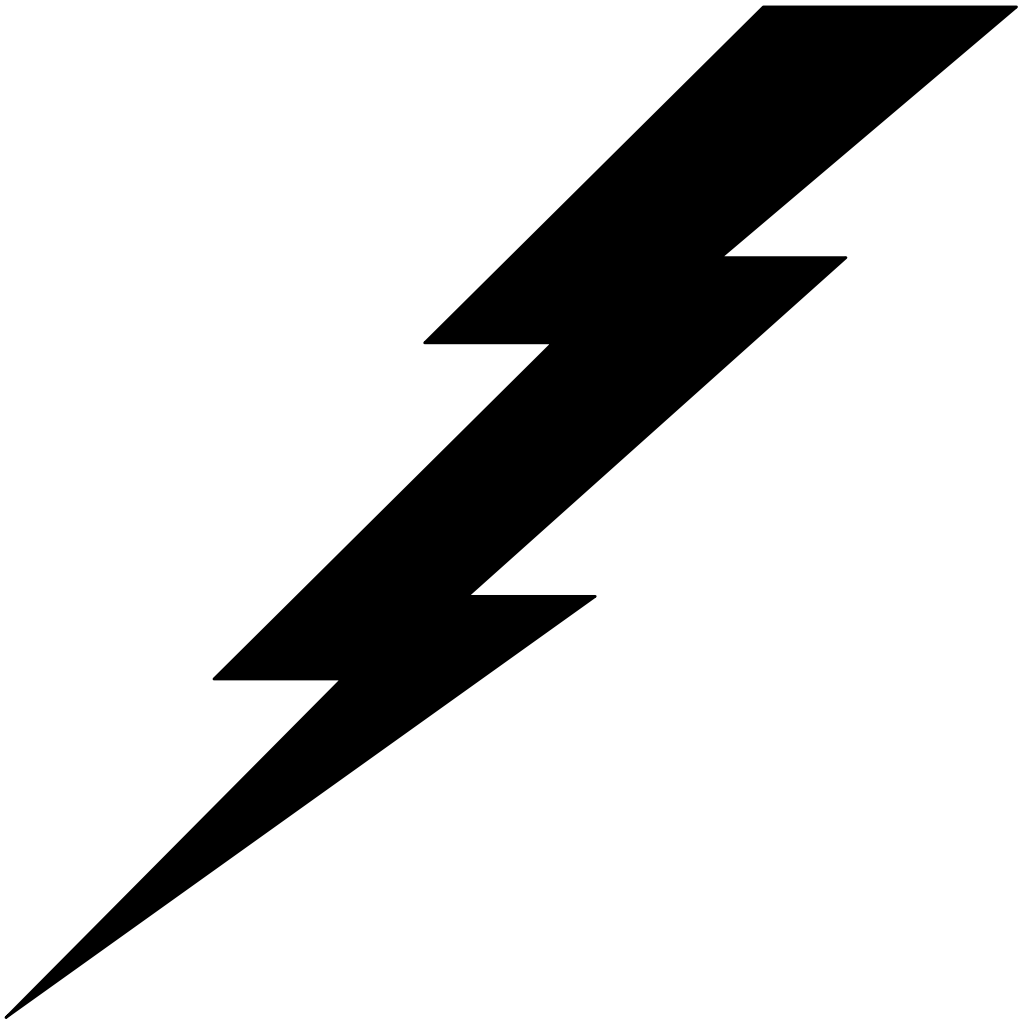
`-p #`	set priority (High:1-Normal:3-Low:5)
`-R`	request email read notification

Processing Options:

`-S SERVER`	TCP/IP host where SMTP server can be contacted
`-P PORT`	TCP/IP port where SMTP server can be contacted

Giving Feedback:

`-q`	disable normal output to terminal
`-v`	enable verbose logging messages
`-V`	display version string
`-?`	display this page



9: Quick functions

Overview

NFM quick functions enable users to perform simple operations on demand (non-scheduled). The following types of operations are available:

`Quick Transfer` User can perform a simple file transfer (one file) from a source node to a target node.

`Quick Execute` User can execute a non-interactive command on a target node (similar to the execute function within a plan).

`Quick Plan` User can run a pre-defined plan.

The quick functions, in addition to offering a simplified way for users to perform certain tasks, also allows for a class of users to be created that can benefit from the services of NFM while being completely restricted as to which resources they may access. Users can be restricted to any combination of the above types of functions as well as being restricted to certain nodes, and even certain directories of certain nodes, based on their user ID (this is fully explained under Configuration, later in this chapter).

Users performing the quick transfer and execute type functions, can enter the appropriate parameters for the operation, selecting nodes from pop-up lists and file descriptions from browse menus, or typing both in manually. Once these parameters are set, the entire operation may be saved under a user-selected name, to simplify performing the same operation in the future by being able to simply select the name from a menu.

The next sections provide detailed instructions for operating the different quick function menus. Following those is a section on configuration that is performed by administrators to make and assign different quick operations to different users.

Quick transfer

The `Quick Transfer` screen may be accessed from the `Quick` menu. This screen will allow you to immediately perform a single file transfer from a source to target node. After entering the appropriate fields, click on the appropriate button for the desired action.

`Transfer From Node` Select the desired source node name.

`Transfer From File` Type in (or browse for) the full file name that represents the source file of the transfer. This file cannot be a directory name and it cannot contain wildcards.

File Type Check the Binary or Text box to describe the mode to transfer this file. Text mode is necessary when transferring a text file between nodes that store text files differently from one another; such as Unix to Windows.

Host File Parameters This button will appear, if the source node is a Host or mainframe type node. Click on this button to bring up an additional window that allows defining any necessary parameters for referencing a Host type dataset. For the source node, usually just the Dataset Name field is necessary. For a complete description of each of these fields, please reference the Host folder description of defining Filesets in Chapter 4, "Configuration."

Transfer To Node Select the desired target node name.

Transfer To File Type in (or browse for) the full file name that represents the target file of the transfer. You may select a target directory without specifying the file base name itself. This will cause the base name of the source file to be created in the target directory. If you type in this directory, make certain you include the trailing slash (/) to indicate it is a directory; otherwise, NFM will assume it a target file name and may try to create a file by that name (if you are browsing, the trailing slash is added automatically.

Host File Parameters This button will appear, if the target node is a Host or mainframe type node. Click on this button to bring up an additional window that allows defining any necessary parameters for creating a Host type dataset. For a complete description of each of these fields, please reference the Host folder description of defining Filesets in Chapter 4, "Configuration."

Wait Option Select the appropriate button, Wait for it to finish or Just start it to indicate the desired option.

Buttons

The following buttons are available on the Quick Transfer screen:

Go	Begin the file transfer.
Save	Save these parameters for use at a later time (you will be prompted for a new or existing name). This information may be saved at any time, before or after performing the file transfer.

Quick execute

The Quick Execute screen may be accessed from the Quick menu. This screen will allow you to immediately perform a single execution of a program on a target node. After entering the appropriate fields, click on the appropriate button for the desired action.

Execute on node Select the desired target node name.

Command Type in the command to run.

Use Shell This option directs NFM to run the command using the native operating system shell or command line interface. This is necessary for doing things like redirecting output or causing environment variable substitution.

Wait for command to finish This option will cause the screen to lock until the command is finished running.

NOTE: Any output that may be generated by the command is not automatically displayed. It can however, be viewed by going to the Quick Monitor screen, clicking on the appropriate instance, and then clicking on the audit message that contains the output. The Quick Monitor screen is described later in this section.

Buttons

The following buttons are available on the Quick Execute screen:

Go	Begin executing the command.
Save	Save these parameters for use at a later time (you will be prompted for a new or existing name). This information may be saved at any time, before or after performing the command.

Quick list & Quick plan

The Quick List screen may be accessed from the Quick menu. This screen will list all the previously saved quick entries that are available to a given user. This includes Quick Transfers, Quick Executes and Quick Plans. Quick Plans cannot be directly created by end-users, but must instead be pre-configured by an administrator. Once an entry is highlighted, click on the appropriate button for the desired action.

Buttons

The following buttons are available on the Quick Execute screen:

Go	Begin executing the quick function directly. This will take you to the appropriate screen based on the Quick function type and begin processing directly.
Edit	This button will take you to the appropriate screen based on the Quick function type. The saved parameters for the function are displayed and may be changed. Clicking on go will begin processing the function.

Quick monitor

The Quick Monitor screen may be accessed from the Quick menu. This screen is identical in functionality to the Plan Activity screen available from the Scheduling menu in NFM. This will display the actual plan information and status that resulted from any of the quick functions performed. Unlike the main Plan Activity screen, this list will only include plans that were submitted by the current user. Reference the Plan Activity and Plan Monitor sections of Chapter 5, "Operations," for a full description of the information and available options for this screen.

Quick functions configuration

Some configuration steps are necessary before Quick functions will be available for general use. Quick functions are actually running plans to perform the operation, although the users may be unaware of this. This keeps the operation simple and straightforward from the user's point of view, but also logs the operation in the normal manner with a plan instance ID, audits, etc. The Plan type Quick function simply refers to an existing plan to run. For these types of items, the name of the Quick plan is simply the name of the real plan that is being run. Transfer and Execute type Quick functions use special plans referred to as templates. These plans are specifically designed for quick functions. They usually have a single function that use reserved environment variable names for the file and program name fields.

When a user initiates a Quick Transfer or Quick execute, the fields that they supply are substituted for the appropriate environment variables in the templates. Other fields like File type (for transfers) and Use shell and Wait for completion (for executes) override the similar options in the template plans and filesets.

The system is shipped with three templates: QuickTransfer, QuickExecute and QuickFileset. The first two are the plans used for

Quick transfers and Quick executes. The QuickFileset is a template fileset that is referenced from within the plan QuickTransfer. The QuickFileset specifies the reserved environment variables $(NFMSFILE) and $(NFMTFILE) that get substituted with the actual file names during a Quick Transfer. Likewise, the QuickExecute plan contains the reserved environment variable $(NFMECMD) that gets substituted for the actual command during a Quick execute.

These templates are just samples, but are ready for use. These can be altered to some degree, for example, compression could be turned on in the fileset, or history settings could be altered on the plans, or even additional functions could be included in the plans. Additional templates can be created. The template actually used for a transfer or execute operation will be determined from the following three criteria:

1. The System Settings has fields to specify a system wide default for Quick Transfer Template and Quick Execute Template. These will be used if the following two items are blank.

2. The Quick Folder (described below), allows specifying a default transfer and execute template on a per user basis. These would override the system settings defaults.

3. A template name may be assigned on a per quick function basis. This field can be selected while saving a Transfer or Execute definition from the appropriate Quick screen. This field will only display for an administrator who is editing the function.

Quick folder

The Quick Folder is available from the User Detail screen. This can be used to designate exactly which quick functions are available on a per user basis. The following information can be maintained on this folder:

Transfer Template This designates a default template plan to be used by this user for each Quick Transfer that they perform (see note below).

Execute Template This designates a default template plan to be used by this user for each Quick Execute that they perform.

NOTE: On the System Settings screen, the Quick folder allows setting of a system wide Transfer and Execute template. If these are set, it is not necessary to set the above two fields unless you wish to override the system defaults.

Available Quick Actions This box on the right of the screen, lists all of the Quick functions (from all users) that are available to be added to the current user. These are all the Quick functions on the

system and are not available to this user until added. The sub-fields of each function include:

Name The Quick Functions assigned name. This name does not have to be unique across the system, only for a given user. Multiple users may have separate Quick Functions with the same name.

Type Type of function. Either Transfer, Execute or Plan.

User The user name to which this Quick Function is currently assigned to.

This list will initially show all quick functions of type Plan that will in fact be a list of all plans defined to NFM. This is because any plan can be assigned as a Quick plan to individual users. The list will not show any Transfer or Execute type functions until they are first created.

For an administrator to create and then assign the Transfer and Execute types of Quick functions they should first go directly to the Quick menu and create appropriate entries on the Transfer and Execute screens (as discussed in the previous section). After returning to the Quick Folder, these entries should now be available in this list.

Allowed Quick Actions This box on the left of the screen lists all of the Quick functions that have already been added to the current user and are available for them to use. These entries may have previously been added using this folder, or may have been created by the individual user from the Quick menu screens directly.

Buttons

The following buttons are available on the Quick Folder:

Add Add item(s) that are highlighted in the Available Quick Actions list to the current user (adding them to the Allowed Quick Actions list).

Edit After highlighting an entry in the Available Quick Actions list, this button will take you directly to the appropriate Quick menu screen (based on the function type) where the function may now be edited or launched.

Remove Remove the highlighted item(s) from the Available Quick Actions list.

The remaining buttons beneath the Quick Folder apply to the entire user record as described in the Users section of Chapter 4, "Configuration." Note that any changes made on the Quick Folder do not get saved until the Apply button is used.

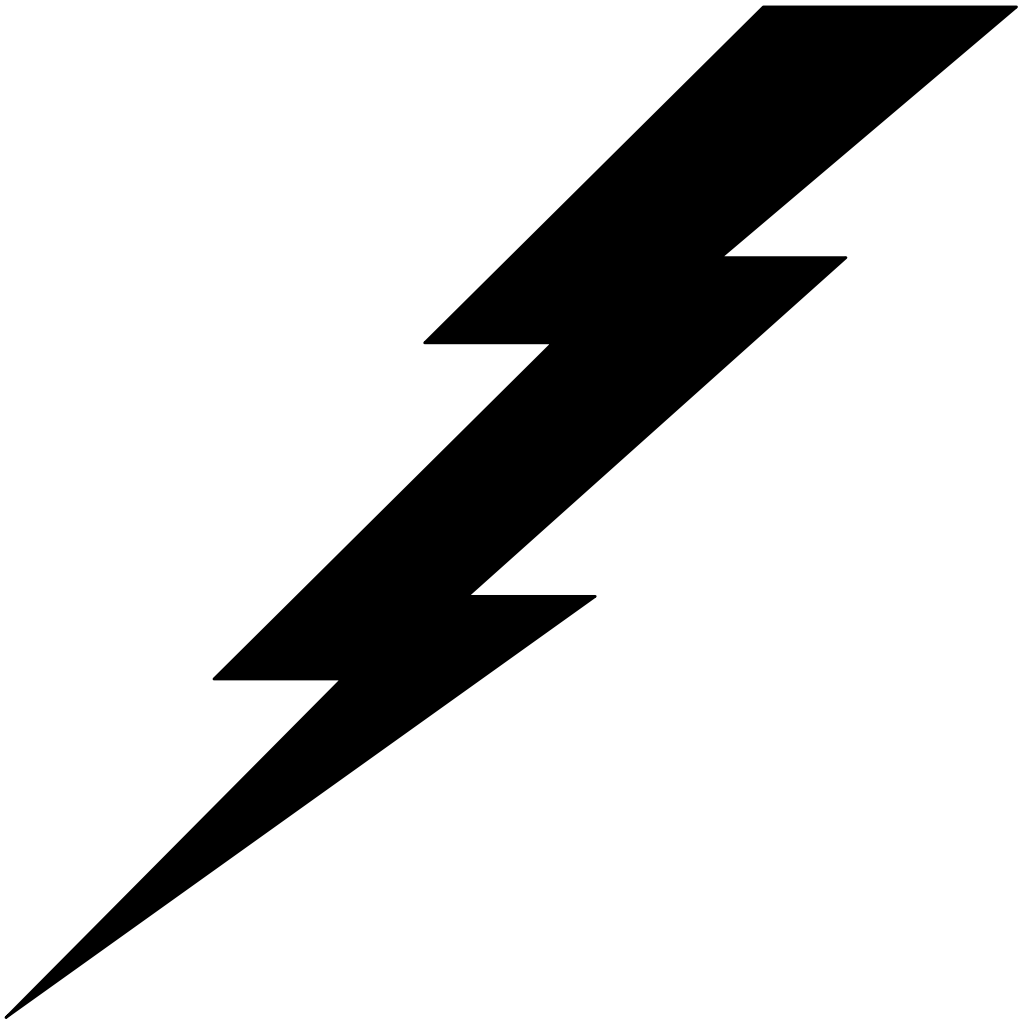
User home directories

During Quick operations, users are already restricted from accessing certain nodes based on their assigned read and write node groups (any pop-up list will only show those available to them). For Quick transfers, users may further be restricted to a particular directory (or sub-directories within) on any given node. This is done by creating a Home directory for each such node.

When a home directory is in effect, any reference to a filename during a quick transfer is automatically appended to this Home Directory. This is true for browsing or typing a filename. The Home Directory is not displayed to the end user, so when they see what would appear to be the root directory, they are actually in the home directory. It is impossible to browse, or refer to by typing, a filename outside their home directory.

The Home Directory field exists in the model record (this allows for adding a home directory to a several nodes). The Home Directory only applies to Quick transfers and does not affect regular plan transfers. The Home Directory may contain an environment variable that represents the current user to automatically create user home directories. Using the environment variable NFMUSER does this. A Home Directory field set to /home/$(NFMUSER)/ for example would restrict a user named todd to the directory /home/todd/.

NOTE: In the example just mentioned, all Quick transfers would be restricted to this matching Home Directory. If it is necessary to allow other users have Quick access outside this scheme, it may be necessary to have different Node and Model records created (pointing to the same physical node) to permit different types of access.



10: Troubleshooting

Overview

NFM was designed to be very comprehensive with regards to capturing and reporting errors in a very user-friendly fashion. Most errors that occur will result in either a pop-up window display (if the error occurred during an interactive session), or with easily readable audit messages for things occurring during plan activity or any other unattended operation.

This chapter attempts to focus on the most common problems that may occur where the cause may not be obvious, and suggested procedures for diagnosing and making corrections. The problems are divided by category into the following sections:

- Problems logging on to NFM.
- Problems communicating with NFM nodes.
- Problems related to system date and time.
- Unexpected results from a plan.
- When all else fails.

Problems logging on

Browser option

Logging on to the NFM system using the browser option involves more steps occurring under the covers than with the stand-alone option. Because of this more things can go wrong. If you have an error such as "`The page cannot be displayed`," it is recommended that you try the stand-alone option first as this may give you back a more descriptive error if it is a communications problem. If the stand-alone option works but the browser does not, then the `http` server may not be configured correctly or may not be currently running. If you are able to successfully get to the sign on screen using the browser option, then you have successfully established an `http` session and there is no need to try the stand-alone option.

Stand-alone option

If you receive one of the following errors:

```
Unknown host name.
Unable to connect to host host_name: Operation timed out.
Unable to connect to host host_name: Host unreachable.
```

Or, you receive anything else that would seem to indicate a communications error, the IP name or address is probably configured wrong or the network is down. Try pinging the address from the command line, if that does not work there is a network problem. If you get the error:

```
Unable to connect to host host_name:8100 Connection
refused.
```

It is likely the TPS® Command server is not running, check to make sure it is installed and running.

If one of these problems still persist, you may need to check with the network administrator. Several of these problems could also occur (most likely operation timed out) if you are attempting to cross a firewall that is not configured to allow this type of socket connection. This could occur even if a ping works. NFM can be configured to use a different port than the default of `8100` if this port is prohibited by a firewall.

If during the logon attempt you get the following message:

```
Command rejected, failed to execute command nfm.
```

This is a probable indication that the TPS® Command server is working fine but that the NFM server product is not installed or running.

Problems communicating with NFM nodes

NFM clients

Any of the errors listed above with regards to IP communications problems may show up as audit error messages during attempts by NFM to communicate with NFM nodes during file transfers or other network activity. Once again, try the `ping` command to see if it is a network problem. If the transfer is taking place between two remote nodes (neither one is the NFM server), make sure the nodes can ping each other.

If you are getting the `connection refused` message, the NFM client is probably not installed or not running on the node in question. A simple way to test connectivity to an NFM node is to perform the following command on the NFM server systems command line: `nfm,statnode,`*`node_name`*. This contacts the node and prints the version of the NFM client running (see the NFM command line interface in Chapter 7, "Tools").

If you are still getting network errors even though the pings work, consider the firewall problem as stated in the previous section. In the case of contacting NFM clients, the default port value is `8008`. If desired, this can be changed, see the description of the communications name in node configuration.

NFM non-client nodes

If problems are encountered communicating with FTP type nodes, you should try to perform the FTP command manually. If this does not work there may be a problem with the way FTP is configured for either the FTP site or the computer attempting to make the FTP connection. You may need to reference the appropriate operating system documentation to determine the problem.

Problems related to system date & time

It is fairly important that all computers involved in NFM keep their system date and time relatively accurate. This is especially true between the NFM server and any computer running the user interface (browser or stand-alone). This is important because the NFM interface keeps its own clock synchronized with the one on the NFM server, it does however, adjust the display to show local time zone information. This is why the clock displayed on each NFM screen may be close but not exactly matches the time shown by the local operating system.

If the date and time are off by more than half an hour, you may see one of the following conditions occur when you try to schedule a plan:

1. The NFM system will warn you that you are attempting to schedule something in the past. This message is intended to keep someone from accidentally starting a plan immediately if it was not the intent.

2. You notice that a plan has not started yet, even though the start time has already passed looking at the NFM clock display.

Unexpected results from a plan

If a plan does not accomplish its intended goals or does something unexpected, run the plan with the `History` option set to `Detail` (defined in plan header or set from plan scheduler). This causes NFM to record in full, each operation a plan performed. For a file transfer, it records the exact source and target filename used for each node involved. This will also record when events are being skipped due to termination logic that was configured. This will help identify items that may not be set up correctly. It would probably be a good idea to keep this option configured at all times, unless the resulting amount of data generated would be undesirable.

If a plan that has been running terminates and information that should have appeared in the audit trail does not, check to make sure the file system or disk drive containing NFM's databases has not filled up.

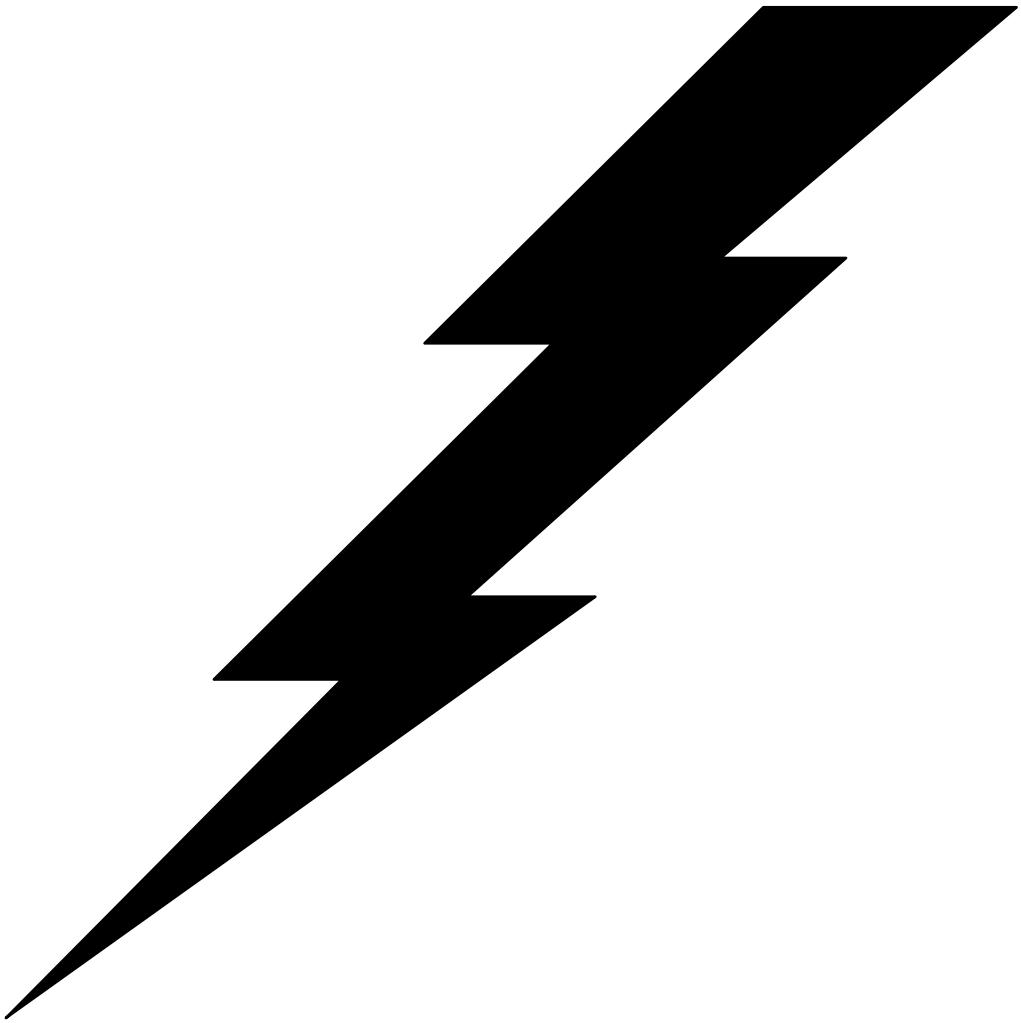
If environment variables are used but do not seem to contain the values that are expected, try executing a simple function `set` on any target node that is in question (you must check the `Use Shell` box). This will make the entire environment variable list in effect for that client to print out to an audit message (you must click on the audit entry with the trailing '…' to see the list). This is available because NFM sends all environment variables down to an execute function and they become locally defined.

When all else fails

If a problem requires a trained help desk person or needs to be reported to TPS® Systems support, it may be necessary to turn on NFM's diagnostics and recreate the problem. The NFM server and each of the NFM clients can individually generate diagnostic logs. Diagnostics may be enabled on a per-node, or per-plan basis depending on the nature of the problem. Turning on diagnostics for one or more nodes is accomplished by checking the diagnostic box in the node's configuration (or its corresponding model). This causes everything the node does to generate diagnostic logs. If the NFM server node has diagnostics checked, it also generates diagnostic output for the NFM server as well. Another option is to check the diagnostic box in a plan's configuration (in the plan header). This has the same effect as turning on diagnostics for each node except that it only generates output for the specific plan's activity.

For Windows installs the diagnostic logs will be created in the `logs` subdirectory located off of each components main install directory. For Unix-Type OS installs the diagnostic logs will be created in the `/var/tps/nfmc/logs` subdirectory. For an AS/400 install the diagnostic logs will be created in `logs` subdirectory of the components home directory. The logs on the mainframe will either go to files, or the job output (JES spool) for the mainframe task/job. If the mainframe client is started with "`-T`" in the "`PARM=`" parameter, it goes to the JES spool. Otherwise, it goes to one or more separate files with the naming format of "`TPS01.NFMC.LOGS.<something>`" (unless changed with "`PARM='-H home'`", then it uses "`<home>.LOGS.<something>`"). The "`something`" part will include the NFM plan instance ID. It looks like A0211000 for plan instance 211. The client will also create a file named "`TPS01.NFMC.LOGS.STARTLOG`" that will contain the NFM client's version number. Note that on any OS you can use the "-H<Home>" parameter to set the home directory to any value. The `logs` subdirectory with the logs will then be created in that location. While it is possible that direct examination of these logs may provide information to a user that will solve a problem, they were intended for use by the TPS® support staff.

The above two methods for enabling diagnostics assumes that the NFM server will be able to contact the appropriate nodes to tell them to enable diagnostic output. In the event that a problem occurs at an NFM client site prior to the establishment of a socket from the NFM server, it may be necessary to turn on the diagnostics mode using a command line option.



Appendices

Appendix A: Return codes

Plan return codes (or rc):

Description

The plan `return code` (or `rc`) is set to the highest severity level of any error that occurred during a plans execution. By setting options in the plans phases and functions, this return code can be used to alter or terminate flow of execution. The values listed here will result from the functions other than type `Execute`. Execution functions will exit with `2` if the program could not be executed, otherwise, the function will exit with the exit code of the program itself.

`0` = Successful.

`2` = Warning generated, or unexpected output from a non-NFM program during a file copy (such as FTP).

`4` = Individual file action has failed.

`6` = A node error occurred and will probably disable any further action with the node.

`8` = An error occurred on the NFM server, but the plan executing may be able to abort.

`10` = An error occurred on the NFM server severe enough to cause the plan executing to abort.

`20` = An error occurred severe enough to disable the NFM server.

Appendix B: FTP nodes

This section describes FTP type nodes, including how to set them up and additional considerations involving their use within NFM.

Overview

NFM incorporates the use of computers running FTP servers as a basic node type, available for use with normal file activity (transfers, deletes, etc.). This type of node does not require the NFM client to be installed on it in order to be used by NFM, but the support is limited compared to a regular NFM client node (described in detail later in this section).

An FTP type node is created by setting its "Primary Transfer Method" field in the node record or in the node's corresponding model record to one of the FTP types (see the list of available FTP node types later in this section. In addition, the "`Account User Name`" and "`Account Password`" fields must also be set to correspond with the appropriate FTP login values desired to access the node. As with most node settings, if a large number of nodes will be similarly defined, it may be preferable to configure these fields in the node's model rather than the node itself.

The "OS Name" field (operating system name) for the FTP node is generally not required but should be filled in if it is known. It may be specified as "generic" if it is not known. However, for an OS/390 FTP node, it is imperative that this field be set to "OS/390".

Once the appropriate node or nodes are added, they can now be specified for file activity just like an NFM client node. However, please note the limitations in the following section.

Limitations

Every effort has been made to seamlessly integrate an FTP type node into the NFM system as if it had the full functionality of an NFM client node. However, there are some limitations that cannot be avoided. This is why using an NFM client node is always preferred, and FTP should be used when it is not possible to use the NFM client. The limitations are:

- The "Execution" type function within a plan is not available for FTP nodes. An attempt to do so will generate an error accordingly.

- File transfer is only available (in either direction) between an FTP node, and an NFM client type node. It is not possible to send files directly between two FTP type nodes within NFM. (This can still be accomplished by transferring a file to an intermediate NFM client and then on to an FTP node.)

- The NFM client used in conjunction with an FTP node for sending or receiving files must have FTP client capability. This exist in one of two forms:

1. The NFM client must have built in FTP client capability, that is, it uses the FTP protocol to directly communicate with the FTP server node. This is referred to as "direct mode". This is generally available on all Windows and Unix platforms.

2. The NFM client must have an external FTP client program available that it can utilize to perform FTP functions with the FTP server node. This is referred to as "batch mode". This is generally available on all Windows, Unix and 4690 computers. Manually running the "`ftp`" program on the computer can be done to confirm availability in most cases.

- The plan monitor will not show the same level of detail for the progress bar as with an NFM client transfer.

- The following additional features are not available for an FTP type node:

  - Streaming file transfer
  - Check point restart
  - Encryption (except by default with secure FTP)
  - Compression.

FTP Node Types

There are many different types of FTP nodes available as indicated by the designated transfer method (primary or secondary) assigned to the node or inherited from the node's model. Some general traits of these different entries are described here followed by a detailed description of the individual FTP types.

Note: Any of the FTP type nodes may refer to an FTP site on the Internet assuming the connectivity is available over the local network and not blocked by any firewalls.

FTP Batch Support vs FTP Direct Support

- FTP Batch type nodes depend upon an external "ftp" or "sftp" program to perform the actual FTP operations. These programs must exist on the NFM client node that is communicating directly with the FTP type node, which is often the NFM server node. The FTP or SFTP program may be native to the operating system on the computer or may be part of an installed package as long as it supports the standard command set used by NFM to perform file operations. Use of the Direct support (described next) is preferable when it is available. Currently the SFTP (Secure FTP) functionality is only available via the batch support.

  Note: The FTP batch environment is entirely dependent upon the communicating (or partner) node having the FTP command capability. It is therefore important to understand which nodes may end up serving in this role. During a file transfer operation, the NFM client node designated on the other end of the transfer (source or target) performs the operation to the

FTP node. Most other operations that involve only the FTP node, such as delete, rename, or file browsing, will default to using the NFM server node to run the FTP commands (this may be overridden on a per node basis).

- With FTP Direct support, the NFM client node that is communicating directly to the FTP node supports the FTP protocol and no external FTP program is required. This node type is preferable to the batch support when it is available. The FTP transfer method types that use this direct support typically have "Server" in the name, like "FTP Server" (this is just a naming convention as all NFM FTP type nodes operating as FTP servers). Note: This direct support is currently available for the FTPS protocol but not SFTP.

FTP Nodes

The following list contains the FTP transfer types currently available. Please refer to the section that follows this list for information on additional node or model settings required for all FTP node types.

FTP Server The node operates as a standard FTP server. An NFM client operating as an FTP client will connect to this node using the FTP protocol directly (direct mode). This method is preferred over the batch mode entry below ("FTP").

FTP This node operates as a standard FTP server. When this selection is in use, the NFM client that is communicating to this node will use the external FTP client program that is native to the operating system (batch mode). Note: The use of the "`Custom FTP`" option (described in `plan function` settings), can only be used with this type of node.

FTPS Server The node operates as an `FTPS` server to provide secure FTP transfers. An NFM client, operating as an FTPS client, will connect to this node using the FTPS protocol directly (direct mode). This setting uses explicit mode, that is the FTPS client will "explicitly request" security from an FTPS Server, which must be configured to handle such request. Use the next entry "FTPS Implicit" to define an FTPS server operating in implicit mode.

FTPS Implicit The node operates as an `FTPS` server to provide secure FTP transfers. This entry is identical to the "FTPS Server" entry (described above) except that it will operate in implicit mode rather than explicit mode.

SFTP Server The node operates as a standard SFTP (Secure FTP) server. An NFM client will connect to this node using the SFTP protocol This method is preferred over the batch mode entry below ("SFTP").

SFTP (SSH) The node is enabled as an SFTP (Secure FTP) server, which utilizes the SSH protocol to provide secure file transfers. When this selection is in use, the NFM client that is communicating to this node will use the external SFTP client program "sftp" that is native to the operating system. This is typically referred to as "batch mode". Use of this entry implies the use of authentication keys to provide security (instead of passwords). Refer to the "sftp" documentation for more

information on this. The "SFTP Server" entry uses the SFTP protocol directly (instead of invoking an external program) and is preferred to using this method.

SFTP+password The node is enabled as an SFTP (Secure FTP) server. This entry operates similarly to the entry "SFTP (SSH)" described above, however it allows password authentication to be used instead of certificates. This entry also utilizes an external SFTP program for communications (batch mode). The "SFTP Server" entry is preferred to this method as well.

Kermit FTP The node is enabled when using Kermit to secure server using Kermit FTP.

FTP additional settings

All FTP node types should also have the following fields configured in the node or corresponding node's model:

Communications Name (primary or secondary) This field should be set to the IP host name or IP address of the FTP node.

Account User Name & Account Password These fields are required to provide a valid FTP account login for the FTP node. This information may be alternately supplied from a "Node Access Point" (or NAP) entry. The use of NAPS allows access for any number of FTP accounts on a single FTP node. Please see the section "Node Access Points" in Chapter 4 "Configuration" for more information.

FTP Attributes settings

FTP type nodes that operate in direct mode should have the following additional settings available in the `Attributes` folder of the model detail screen (see the "FTP Nodes" section above to determine which entries are included):

ftp-account (text) A response to be given if the FTP server sends an ACCT request after the user name and password fields have been provided.

ftp-alternative-to-user (text) A string wihich will be used to authenticate if the usual FTP user and password negotiation fails.

ftp-create_dirs (checkbox) Select the option to cause automatic creation of any directory that FTP fails to change into (using the "CWD" command).

ftp-method (selection box) Determines which method to use to reach a file on a FTP(S) server. The choices are:

`MULTICWD` A single CWD operation is performed for each path part in the given URL. This is how RFC-1738 says it should be done. This is the default behavior.

`NOCWD` No CWD operation is performed. A full path will be given to the server for the file commands (SIZE, RETR, STOR etc).

`SINGLECWD` A single CWD operation is performed using the full target directory.

`ftp-port (text)` Enables the use of active mode (default FTP operations are passive). Used to get the IP address to use for the FTP PORT instruction. The PORT instruction tells the remote server to connect to our specified IP address. The string may be a plain IP address, a host name, a network interface name (under Unix) or just a '-' symbol to let the library use your system's default IP address.

Fileset considerations

A Fileset definition used for an NFM client transfer should work similarly for an FTP node, but there may be some subtle differences in the naming conventions required to satisfy the FTP client and/or server programs. Sometimes these requirements may not be obvious until a transfer attempt is actually tried. Some examples of such requirements are as follows:

- NFM generally requires a leading forward slash (or drive designation for Windows) for each filename in a fileset. This is usually accomplished with the "Source Prefix" and "Target Prefix" fields, but may be done in the beginning of each filename in the "Source Files" or "Target Files" list. Some FTP servers will not accept this but instead will want a file specified as a 'relative' location from the home directory that is logged into. So it may be necessary to simply specify "`tmp/filename`" instead of "`/tmp/filename`".

- Wildcards are allowed with FTP, but in this case the source and target prefix fields will almost always be required. They should be set to the fully qualified directories to be used for the transfer (no subdirectory should be included in the wildcard expression in the "Source Files" list, only the base filename). This is important because an FTP wildcard transfer will invoke an '`lcd`' and '`cd`' command to these prefix values to establish the locations prior to doing an '`mput`' or '`mget`' command. Additionally, some FTP clients and servers will want a trailing '`/`' on these prefixes and some will not.

Most of the remaining fields used in the fileset definition have the same basic meaning for an FTP transfer as an NFM client transfer. The 'Text' or 'Binary' setting for the file corresponds to the "ascii" and "bin" options that native FTP supplies. All of the OS/390 specific fields (Record Format, Record Length, etc.) should be set the same for FTP as for NFM client transfers.

Secure FTP

NFM also provides secure FTP functionality when used in conjunction with either the "Kermit" FTP client program or the "secure FTP" (or `SFTP`) as part

of the SSH protocol standard. Similar to basic FTP, these protocols are used by specifying the appropriate "Transfer Method" fields in the node or model, in this case "Kermit FTP" and "SFTP (SSH)" respectively. The use of either of these two protocols is subject to the same restrictions listed above for standard FTP. Currently, these two protocols are only available for use with the NFM client running on Unix platforms.

As with standard FTP, NFM invokes the client programs for these protocols, through the use of customized scripts that are maintained as part of the NFM product. This allows the use of these protocols to be transparently integrated into the NFM system with the same ease of configuration and use as the native NFM client type nodes.

Acquiring the client software programs for these protocols, as well as satisfying any licensing requirements is the customer's responsibility. These two client programs are recommended for use with the NFM product due to their widespread availability and acceptance within the industry.

The term SFTP tends to be re-used for different things. As it is referenced here, it refers to the command-line client program "`sftp`" that provides secure FTP over SSH. An excellent source of information on this protocol is available at http://www.openssh.org.

For more information on Kermit, please visit http://www.columbia.edu/kermit/.

Appendix D: PPP configuration

Overview

The NFM server controls the tasks of dialing remote nodes and establishing TCP/IP connections with the NFM clients on these nodes to facilitate file transfers and other NFM activity. In a typical customer setup, these remote nodes usually have a single phone line and modem that offer a single point of connection for use with NFM. Because of this, NFM does not support file transfers directly between two remote nodes (otherwise known as peer-to-peer transfers) as it does with socket type connections. Instead, file transfers are only available to and from the remote nodes and the central NFM server computer. Files may of course be transferred to the NFM server and then sent to another computer as an additional step within a plan.

The required setup for NFM is to configure the NFM server for dialing out with a modem or set of modems (modem pool), and configuring the NFM client computers to handle incoming connections. Because NFM generally relies on the underlying operating system for modem support, this setup is very different between different platforms. Please refer to the appropriate section in the remainder of this appendix for the operating system desired. The final section "NFM Configuration" details the configuration steps necessary from within NFM once the modems are installed and configured to the operating system.

Windows configuration

Overview

NFM uses the Remote Access Services (or RAS) facility on Windows to control dial-up connections between the NFM server computer and the remote dial-up nodes running the NFM client. The remote nodes should be configured to receive incoming dial-up connections. The NFM server computer should be configured with multiple dial-up connections (or phonebook entries), one for each available modem.

The following section describes configuring the client(s) and server for Windows/XP or Windows/2000 systems. Older versions of Windows (NT and 98) are generally supported, although the configuration may vary somewhat. In general the default settings for dial-up connections work reasonably well. Follow the guidelines below for items that are specifically required for NFM, but also follow the Windows documentation as needed for settings that may be specific to your environment or network, as well as those that require special considerations.

Remote nodes (NFM clients) Setup the incoming connections configuration to allow for dial-up to occur. In the "Incoming Connections" panel (Start / Settings / Network and Dial-up Connections / Incoming Connections), configure the settings in the various tabs as follows:

General tab Specify which modem(s) to use for incoming connections.

Users tab Specify that users may remotely log in. The name and password actually used by NFM will be configured in the nodes environment variables (described in the section below, "NFM configuration").

Networking Specify the network components used for this connection. We recommend specifying only TCP/IP since that is the only protocol used by NFM.

Click on the properties tab for TCP/IP to make any changes to the TCP/IP properties required. The "TCP/IP address assignments must be configured in a way that corresponds with the properties of the computer dialing in. For example, if the dial-in computer specifies a specific IP address, you must check "Allow calling computer to specify its own IP address".

NFM server Create a phonebook entry for each modem to be included in the modem pool for use with NFM. Although phonebook entries will often contain dial-up information for a specific site (phone number, user name, etc.), the ones defined for NFM will be mostly blank, as most of the information will be supplied by NFM at connection time. In the "Make New Connection" panel (Start / Settings / Network and Dial-up Connections / Make New Connection), use the Network Connection Wizard to configure the following settings:

Network Connection Type Check "Dial-up to private Network"

Phone Number to Dial Leave blank

Connection Availability Check "For all Users"

Name for the connection Choose a name for this phonebook entry, preferably one that is based on the modem it represents (`Modem1`, Modem2, etc.). Click `Finish`.

From the connect panel now displayed, click "Properties" and configure settings in the various panels:

General tab In "Connect using:" field, it is imperative that you only check the modem for which this phonebook entry will represent.

Options tab Under "Redialing options," it is recommended that you set "Redial Attempts" to `0`. Although not mandatory, NFM has its own redial capability (Connection retry & interval) which provides additional logging as well as automatically rotating the modem's use to other nodes in between dialing attempts. This allows time for a connection problem that may be intermittent, such as a busy line, to clear up.

Networking Specify the network components used for this connection. Again we recommend specifying only TCP/IP for use by NFM.

Click on the properties tab for TCP/IP to make any changes to the TCP/IP properties required. The "TCP/IP address assignment setting should correspond with the configuration of the dial-up computer. Checking "Obtain an IP address automatically" should work okay. If, however, it is necessary to specify an IP address, then make sure the dial-up computer is configured to allow this in its incoming connection configuration. Also, if this is the case, make sure to specify a unique IP address for use with each of the modems.

Click on "Advanced," uncheck the setting "Use default gateway on remote network." This option, if checked, changes the default route for the NFM server computer and would not be desirable if the computer is also on a local network.

Repeat this process for each modem, making sure that each phonebook entry references its own unique modem.

NFM configuration

Once the appropriate connection entries are created on the server and client nodes, follow the remaining steps within the NFM user interface to facilitate the use of dial-up PPP nodes.

Create NFM modem records Go to the `Modem Detail` screen (under Management) to add modem records. In the modem field type in a name representing the modem being configured (this does not have to match the phonebook entry name, just something similar like `MODEM1`, `MODEM2`). In the "Device Name" field, type in the exact name used for the corresponding phonebook entry (Example: `Modem1`, `Modem2`, etc.) . Click '`Apply`' to add the record. Repeat this step for each modem.

The "`Off Line`" checkbox may be checked at any time to remove a modem from the available modem pool temporarily without having to delete the modem record. All other fields on this screen are unused at this time.

Create Models and Nodes Follow the instructions in Chapter 4, "Configuration," for adding a model and node combination to represent each of the dial-up NFM client computers. The requirements for creating these records for use with dial-up nodes are as follows:

- The node's corresponding model must have its Primary Transfer method set to "PPP".
- The node's Primary communication Name should be set to the appropriate dial string (phone number along with any special control sequences).
- Two specific environment variables, for login name and password, must be added either to the node or the node's model (adding them to the model is allowed, as long as all nodes using this model are configured to use the same values. These are:

  NTUSERNAME=(users name)
  NTPASSWORD=(password)

It isn't necessary to create a dial-up type node for the NFM server. By specifying a file transfer between the NFM server node (type SOCKETS) and remote node (type PPP), NFM will automatically establish a dial-up connection, using a modem from the available modem pool.

Dial-up should now be available. The Modem Summary screen, in addition to showing all of the defined modems, contains dynamic information representing the state and usage of the individual modems. This information can be useful for identifying potential problems. This following information will be displayed:

Device column Next to the name of the phonebook entry, you should see the characters [Ok]. Otherwise, you may see a string representing an error associated with the phonebook entry such as: "Invalid phonebook entry". If this is the case, look in the Windows configuration and make sure the entry is defined correctly. You should not attempt to use these modems until they all show [Ok]. Note: Taking the modem offline will keep it from displaying its state.

LocalIP & RemoteIP These fields will show the IP addresses currently in use if the modem is connected.

Node, Plan, Instance These fields will identify who is currently using the modem.

Appendix E: HTTP nodes

This section describes HTTP type nodes, including how to set them up and additional considerations involving their use within NFM. It also addresses the optional use of HTTP protocol with the NFM Graphical User Interface (GUI).

Overview

NFM incorporates the HTTP protocol for one or more of the following scenarios:

1. NFM activity (file transfers, etc.) is required for a computer that has an existing HTTP server capability, but it is undesirable to install an NFM client or other NFM software on it.
2. Same as #1 (above), however it is okay to implement a custom NFM CGI program as an extension to the HTTP server. This adds additional functionality while still not having the full range of functionality as an NFM client.
3. There are no restrictions to installing NFM software (a client) on the computer, but the customer wishes to ONLY use the HTTP protocol for any communications to/from the computer for NFM activity.
4. The customer desires to only use the HTTP protocol between the user interface and the NFM server (similar to #3).

Note: The above setups, especially 1 and 2, have significant restrictions compared to using an NFM client with its native protocol. The native NFM client is therefore the recommended approach unless circumstances prohibit it.

There are currently three basic method types corresponding to scenarios 1 through 3 listed above:

`HTTP` A basic HTTP server.

`HTTP_D` An HTTP dual server. Combines standard HTTP server with some extended features via a CGI program.

`NFM/HTTP`
An NFM/HTTP client. Performs standard NFM client functions using the NFM protocol encapsulated (or tunneled) over HTTP.

Each of the above 3 methods is also offered using HTTP over SSL (or HTTPS) with the following names **`HTTPS`**, **`HTTPS_D`**_& **`NFM/HTTPS`**.

The next section describes each of these methods in detail, including their limitations and prerequisite requirements.

HTTP & HTTPS

A node with this method serves as a standard HTTP server and provides limited file transfer capability (the computer can only support file capabilities offered by an HTTP server, as it is not running any NFM software). This limits it to a basic file upload, download & delete with the following restrictions:

- The node does not support directory functions and therefore cannot retrieve files using wildcard expressions, nor can the node be browsed.
- The location for the files being accessed on the node is usually restricted to the HTTP server's base directory tree.
- The node does not support any of the following functions or features:

    - Create/remove directory
    - File append
    - Execute programs
    - Compression
    - Attribute preservation
    - Checkpoint restart
    - Streaming transfers
    - Call-in type connectivity (where the node initiates the connection to the NFM server).
    - Watchfile support
    - Stand-alone file encryption

- A node of this type can only transfer files to/from a standard NFM client node. It cannot send files directly between itself and another node of this type. (This can still be accomplished by transferring a file to an intermediate NFM client and then on to an HTTP node.)

The HTTPS indicates that the HTTP session will run over SSL to provide a secure transfer capability. This assumes that the HTTP server being used offers this option.

Requirements

HTTP & HTTPS nodes require a standard HTTP server running on the computer. The HTTP server must be configured to allow the appropriate actions (GET & PUT) based on the direction of the transfer and the directory locations involved. The current implementation has been tested against versions of Apache and Microsoft IIS. For Apache, it is recommended that the version be 2.0 or later, for IIS, version 5.0 or higher is recommended.

Configuration

The "Primary Transfer Method" for the node or model record should be set to HTTP or HTTPS as desired to configure this node type. Please see "NODES" or "MODELS" sections in Chapter 4 "Configuration" for more information.

HTTP_D & HTTPS_D

A node with this method serves as a standard HTTP server along with certain NFM extensions provided by a custom CGI program that may be installed on the computer. The computer is not running the NFM client but does have additional NFM client capabilities provided by the CGI program. This node performs file upload and download using standard HTTP protocols, but performs most other functions using the CGI program. It has the following restrictions:

- The location for the files being accessed on the node is usually restricted to the HTTP server's base directory tree.
- The node does not support the following features for file transfers:

    - File append

  - Compression
  - Attribute preservation
  - Checkpoint restart
  - Streaming transfers
  - Stand-alone file encryption

- Call-in type connectivity (where the node initiates the connection to the NFM server).
- Watchfile support

Requirements

HTTP_D & HTTPS_D nodes require a standard HTTP server running on the computer. The HTTP server must support the GET, PUT & POST actions to operate properly.

The current implementation has been tested against versions of Apache and Microsoft IIS. For Apache, it is recommended that the version be 2.0 or later, for IIS, version 5.0 or higher is recommended.

Configuration

The "Primary Transfer Method" for the node or model record should be set to HTTP_D or HTTPS_D as desired to configure this node type. Please see "NODES" or "MODELS" sections in Chapter 4 "Configuration" for more information.

The HTTP server must be configured to allow the appropriate actions (GET, PUT & POST). Also, the HTTP server must have a copy of the program "nfmcgi.exe" placed in the cgi-bin subdirectory off of the document root directory. This program can be found in the corresponding home directory of any NFM client package that matches the operating system.

NFM/HTTP & NFM/HTTPS

This represents an NFM client node that performs all of its communications over an HTTP tunneled protocol. Unlike the other HTTP methods, this requires the NFM client to be physically installed on the computer. It provides full NFM client functionality.

Use of the HTTPS entries indicates that the HTTP session will run over SSL to provide a secure transfer capability. This assumes that the HTTP server being used offers this option.

Requirements

NFM/HTTP & NFM/HTTPS nodes require a standard HTTP server running on the computer. The HTTP server must be configured to allow POST to occur. Ideally it should also support chunked-encoding (it will work without this but it is much less efficient). The current implementation has been tested against versions of Apache and Microsoft IIS. . For Apache, it is recommended that the version be 2.0 or later, for IIS, version 7.0 or higher is recommended.

Configuration

The "Primary Transfer Method" for the node or model record should be set to NFM/HTTP or NFM/HTTPS as desired to configure this node type. Please see "NODES" or "MODELS" sections in Chapter 4 "Configuration" for more information.

Use of these node types requires the following additional configuration:

- The HTTP server must have a copy of the program "httptunnel.exe" placed in the cgi-bin subdirectory off of the document root directory. This program can be found in the corresponding home directory of any NFM client package that matches the operating system.

- The NFM server and any other clients that try to connect to this node must have the HTTP tunnel service running locally. This is accomplished by enabling the service in the local NFM client configuration file ("HTTP_TUNNEL=Y"). See the section "NFM Client Configuration" near the end of this chapter for more information.

Appendix F: AWS S3 nodes

This section describes Amazon Web Services (AWS) Simple Storage Service (S3) type nodes (i.e. the cloud), including additional considerations involving their use within NFM.

Overview

NFM incorporates the use of computers running S3 servers as a basic node type. This type of node does not require the NFM client to be installed on it in order to be used by NFM, but the support is limited compared to a regular NFM client node. NFM uses an NFM Client along with the AWS Command Line Interface (CLI) to handle normal file activity (transfers, deletes, etc.). Information on acquiring, installing, and configuring the AWS CLI can be found at http://aws.amazon.com/cli/.

An AWS type node is created by setting its "Primary Transfer Method" field in the node record or in the node's corresponding model record to "AWS CLI" (see Chapter 4, "Configuration" section "Models" or "Nodes"). Once the appropriate node or nodes are added, they can now be specified for file activity just like an NFM client node. However, please note the limitations in the following section.

Limitations

Every effort has been made to seamlessly integrate an AWS type node into the NFM system as if it had the full functionality of an NFM client node. However, there are some limitations that cannot be avoided. This is why using an NFM client node is always preferred, and AWS should be used when it is not possible to use the NFM client. The limitations are:

- The "Execution" type function within a plan is not available for AWS nodes. An attempt to do so will generate an error accordingly.
- The "Synchronize" type function within a plan is not available for AWS nodes. The "Update" can be used and it will use the AWS CLI "sync" option. The AWS CLI will attempt to do the synchronization.
- File transfer is available (in either direction) between an AWS node, and an NFM client type node as long as the following two conditions are in effect:

    1. The node on the other side of the transfer (either sending or receiving) must be an NFM client. It is not possible to send files directly between two AWS type nodes using NFM. (This can still be accomplished by transferring a file to an intermediate NFM client and then on to an AWS node.)

    2. This NFM client system must have the AWS CLI client program installed and available. This is generally available on Windows and some Unix computers. Manually running the "`aws`" program on the computer can be done to confirm availability.

- The plan monitor will not show the same level of detail for the progress bar as with an NFM client transfer.
- The "Streaming file transfer" feature is not available for AWS nodes.
- The "Check point restart" feature is not available for AWS nodes.

Fileset considerations

A Fileset definition used for an NFM client transfer should work similarly for an AWS node, but there may be some subtle differences in the naming conventions required to satisfy the AWS client and/or server programs. Sometimes these requirements may not be obvious until a transfer attempt is actually tried. Some examples of such requirements are as follows:

- NFM generally requires a leading forward slash (or drive designation for Windows) for each filename in a fileset. This is usually accomplished with the "Source Prefix" and "Target Prefix" fields, but may be done in the beginning of each filename in the "Source Files" or "Target Files" list. The AWS S3 protocol deals with "buckets" and "objects", not "directories" and "files". "Buckets" are created in the S3 cloud and "objects" are placed in the buckets. AWS object names will be discussed later.

- Wildcard usage is limited by the AWS CLI. NFM will allow wildcard use with AWS by converting them to the AWS CLI format.

Most of the remaining fields used in the fileset definition have the same basic meaning for an AWS transfer as an NFM client transfer. The 'Text', 'Binary', or 'Directory' setting for the file does not get used for AWS. As mentioned before there are no directories and all files are transferred as 'Binary'.

Security

AWS also provides secure functionality when used. An account must be created on the AWS server which will have corresponding "Access Keys". These keys are required and will be used by the AWS CLI to securely encrypt the transmission sessions between the client system and the AWS server. When an AWS type node is created the "`Account User Name`" and "`Account Password`" fields must also be set to correspond with the appropriate "Access Key" and "Secret Access Key" values to access the AWS node.

AWS Objects

As mention earlier, the AWS S3 protocol does not have directories. It has "buckets" and "objects". The naming convention used is:

//<bucket name>/<object name>

When a file is transferred to the AWS server it is treated as an object. That object is placed in a bucket. There can be multiple buckets and each can contain multiple objects. You can use the AWS Online Console to view the buckets and their contents. If an object has the "/" character in it, AWS and NFM will simulate directory browsing. Note that if you use the "\" character it is treated just like any other character by AWS and will not be used by AWS to simulate a directory. So the object name "/tmp/file" is not the same as the object name "\tmp\file".

When creating a fileset be aware that you must always specify a bucket name whether you are copying a file to or from the AWS server. Otherwise, a fileset used for an AWS node follows the same NFM rules as any other node. For example, if you need to create a bucket named "TmpBucket" then setup a fileset to be:

Source		Target	
Prefix:		Prefix:	
Files:	/	Files:	//TmpBucket

If you wish to copy all the files in the local "/tmp" directory to the AWS bucket "TmpBucket" then you can setup the fileset as:

Source		Target	
Prefix:	/tmp/	Prefix:	//TmpBucket/
Files:	*	Files:	

If the two files "stuff.txt" and "color.jpg" are in the "/tmp" directory then the "objects" "stuff.txt" and "color.jpg" will be placed in the "TmpBucket".

If you wish to also copy the directory information to the bucket then you can setup the fileset as:

Source		Target	
Prefix:	/	Prefix:	//TmpBucket/
Files:	tmp/*	Files:	

The result will be that the "objects" "tmp/stuff.txt" and "tmp/color.jpg" will be placed in the "TmpBucket".

You can also rename objects if you wish. Using the normal NFM method you just place the new name in the "Target Files" entry for the "Rename" such as:

Source		Target	
Prefix:	/tmp/	Prefix:	//TmpBucket/
Files:	stuff.txt	Files:	text/stuff.txt
	color.jpg		images/color.jpg

The result will be that the "objects" will be renamed to "text/stuff.txt" and "images/color.jpg" and will be placed in the "TmpBucket". Also, you can direct objects to different buckets such as:

Source		Target	
Prefix:	/tmp/	Prefix:	
Files:	stuff.txt	Files:	//TextBucket/stuff.txt
	color.jpg		//ImageBucket/color.jpg

Copying objects from a bucket works the same ways as mentioned above. If you wish to copy objects with names that start with "tmp/" from the "TmpBucket" then setup the fileset as:

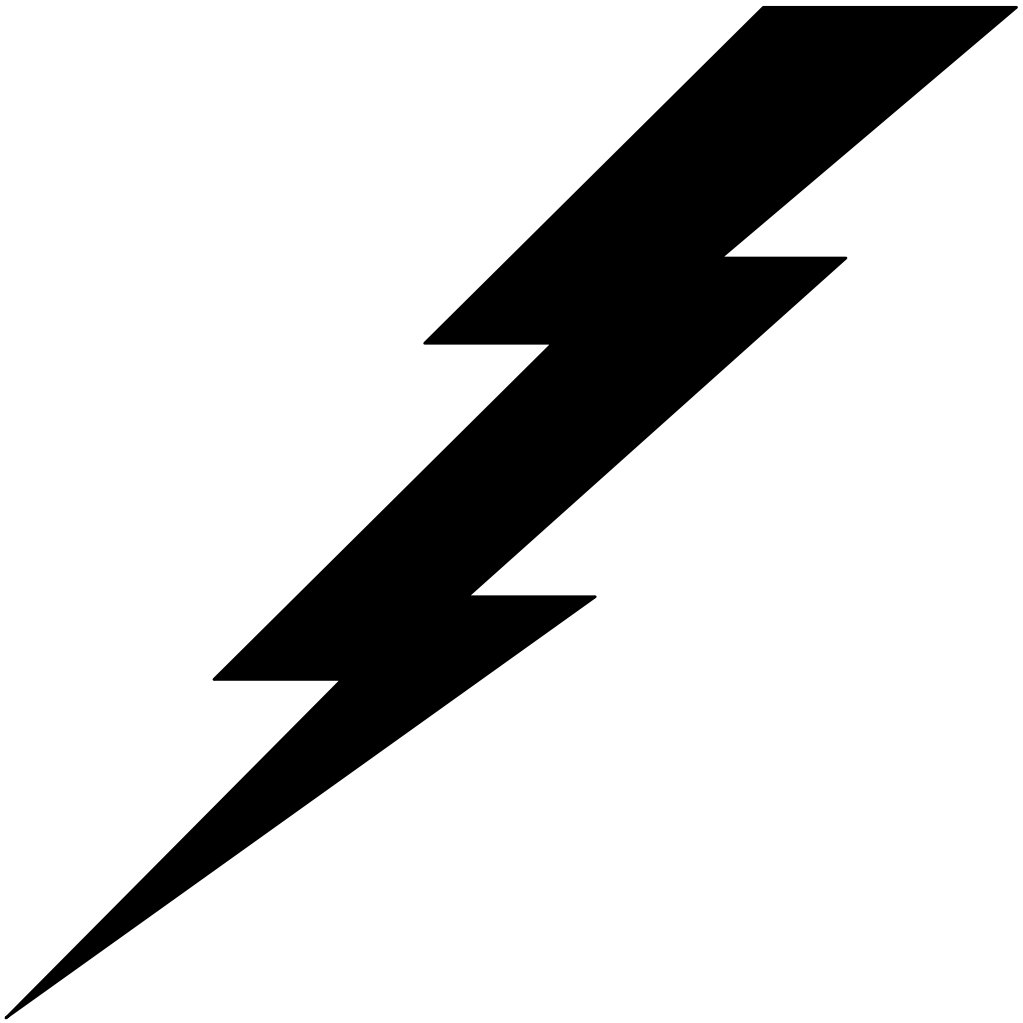
Source		Target	
Prefix:	//TmpBucket/tmp/	Prefix:	/tmp/
Files:	*	Files:	

Things to Note

- As mention earlier, the AWS S3 protocol does not have directories. It has "buckets" and "objects". NFM will not create or delete directories in a

bucket since they do not exist. NFM will however create and delete buckets and objects.

- Although NFM will not create directories when objects are copied to a bucket, it will however create directories when objects are copied from a bucket onto the system running the NFM client.
- AWS does not treat the "\" the same as the "/". The "/" is used to simulate directories but the "\" is treated just like any other character such as "a", "b", "c", etc. So, even on Windows systems you should use "/" as directory separators.
- The AWS CLI must be installed on a system along with an NFM Client and AWS account(s) must be created to acquire AWS Access Keys. You probably should set the permissions for the account(s) to FULL ACCESS.
- Although the AWS CLI has a built-in default DNS name to access the Amazon Cloud you must enter "s3.amazonaws.com" as the AWS node's "Primary Communications Name".
- If you have a "private" cloud that is AWS S3 compatible you should be able to specify a "Primary Communications Name" for that cloud.
- There are multiple ways to configure the AWS CLI account information on a client system. If you choose to configure it on the client system then do not enter anything in the nodes "`Account User Name`" and "`Account Password`" fields.
- If you enter anything in both the nodes "`Account User Name`" and "`Account Password`" fields then they will be assumed to be the accounts "Access Key" and "Secret Access Key". If something is entered in the "`Account User Name`" field but not the "`Account Password`" then it will be assumed to be an AWS "Profile".
- The AWS CLI encrypts the communication sessions using the account access keys.
- When copying to the cloud, if the "Recurse subdirectories" option is set then NFM will copy all the subdirectories and their content. This is not the case when copying from the cloud.



Index

Index