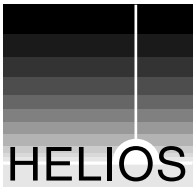

“dt”(=desktop)

(Utilities for HELIOS EtherShare 2.5)

User manual

15 December 1997

Contents

Appendix

“dt”(=desktop)

1 Introduction

License and copyright information

The “dt” utilities are not a separate product. They may be used by owners of a full HELIOS EtherShare 2.5 license only.

HELIOS Software GmbH
Steinriede 3
D—30827 Garbsen, Germany

About this documentation

The “dt” utilities are integrated in the HELIOS EtherShare 2.5 software. This manual is part of the EtherShare 2.5 documentation, it should be used together with the EtherShare 2.2 user manual and the 2.5 release notes.

About the product

This documentation describes the “dt” utilities, which mimic the functionality of some major UNIX commands for handling files, while maintaining the integrity of the desktop database. It also provides access to Macintosh specific file information from a UNIX prompt.

About YOU – the potential users of the “dt” utilities

These utilities are meant for people who have some knowledge about UNIX and Mac – from novice to expert UNIX/Mac users.

Please read appendix A 8 “Problem reports” for details about how you should report problems – should there arise any – and about what kind of information we need in order to help you find a solution.

2 Do I need the “dt” utilities?

If you access all your files from your Macintosh only, you do not need to read any further. But if you need to access files which are stored in an EtherShare volume from the UNIX prompt, there are a lot of reasons for using the “dt” utilities.

Note: “UNIX prompt” here means *any* access under UNIX, e.g. any script or program (like a backup program), which accesses files stored in an EtherShare volume.

Any manipulation to a file or folder inside a Macintosh volume using ordinary UNIX commands like “cp”, “mv”, “rm” or other programs, will cause an inconsistency between volume information and the related desktop database. Especially restoring files from a backup will trigger such an inconsistency.

In the following sections, you will find more information about this problem and about how the “dt” utilities help avoid it.

Please use the utilities *whenever* you would (usually) use any of the following UNIX commands (in case either source and/or destination files/folders are located in an EtherShare volume):

- `rm`
- `rmdir`
- `mv`
- `cp`
- `ls`
- `mkdir`

- `touch`
- `chmod`
- `chown`
- `chgrp`

You may use the “dt” utilities even if the source or destination is not located in an EtherShare volume, or if you are not sure about this. The “dt” utilities automatically select the proper mode of operation.

It is required to use the “dt” utilities together with EtherShare 2.5, PCShare 2.5 and/or EtherShare OPI 2.0. These HELIOS product versions have improved verification mechanisms to recognize potentially harmful configurations, and will disable access to volumes for which consistency between volume and desktop information cannot be assured. In addition, warning or error messages are logged to the system error log which may help you locate the cause of the problem.

What are the differences to the standard UNIX commands?

Just to help you understand the possible error messages and the specific behaviour of the “dt” utilities, here are the main differences to standard UNIX commands:

Each Macintosh file consists of two parts, the so-called “data fork” which is stored as a standard UNIX file, and a corresponding “resource fork” which is stored by EtherShare in a “.rsrc” subdirectory. Additional information for each file are stored in a corresponding entry in a volume based “desktop” database. Therefore, when using UNIX commands, each operation like move, copy, or delete, must consider both, the data and resource fork, as well as the database entry instead of just a single file.

When looking into a volume as a UNIX directory you may not notice the differences, because the resource fork subdirectory and the database files are stored as hidden UNIX files.

The “dt” utilities simulate the behaviour of the standard UNIX commands as close as possible, but there are slight differences in the number and functionality of the available options, and due to the extended functionality you may notice a different behaviour and different error messages. For example, you may see error messages about the resource fork of a file, which you would not see when using standard UNIX commands (the UNIX commands – in fact – ignore this part of the file).

For all of the “dt” utilities commands, a behaviour that differs from that of their UNIX counterparts may be visible. This is due to the fact that different semantics are used by Macintosh Finder operations on the one side and UNIX file system operations on the other. The “dt” utilities will behave like an “afpsrv” on recognized EtherShare volumes, and will behave like ordinary UNIX commands like “cp”, “mv”, “rm”, in areas where no EtherShare volume is defined. The “dt” utilities will process data and resource part of files and will also assure that ownership and permissions are compatible to the Macintosh Finder operations. Due to the limitation on file/folder name length, “dt” will only process objects with a name of 31 characters at the most on EtherShare volumes.

For additional information, please see the “EtherShare 2.2.0 manual” which is still up-to-date on this topic. Especially chapters 7.2, and 7.4 to 7.6 may be of interest.

Before you try and use any of the “dt” commands, you should be well familiar with the corresponding UNIX commands.

If you need more information about how the “dt” utilities work and how problems arise without the utilities, please read the “Technical notes” following each command description, and the appendix.

3 How to install the "dt" utilities

General remarks

The "dt" utilities are delivered as a single UNIX executable file named "dt" with an accompanying README file. Please read this file carefully to check whether there are any changes or updates which did not make it into this manual.

Please note that your platform specific copy of the "dt" utilities may require an update to other parts of your EtherShare installation to work correctly. Please check the README file and any accompanying documentation carefully.

All emulated UNIX commands are included in the single UNIX executable "dt"; the different modes like "cp" or "rm" are passed as the first argument.

Installing "dt"

The "dt" utilities do not require a special installation procedure. Please unpack the tar archive into the EtherShare installation directory ($ESDIR).

We do not recommend to sym-link regular "cp", "mv", "rm" commands to the "dt" utilities. Although this is possible, you should verify your current and future system environment very carefully, to assure proper operation of your programs, scripts and established workflows.

Especially, you should check whether your site uses special HSM, tape robot, or RAID software which may also use their own versions of these UNIX commands.

Take into account that different users and script based programs may make use of different shells and environment settings.

Before getting started...

We suggest to check for each EtherShare volume the consistency between the volume information and the volume’s desktop database. EtherShare 2.5 offers a new rebuild option, namely “-s” (scan only), which can be used for this purpose.

Simply issue `rebuild -s <volume mount point>` for every defined public or private EtherShare volume. Any output of this “scan” indicates a potentially harmful inconsistency between volume and desktop information. You should only continue after having resynchronized the desktop database by means of an ordinary rebuild.

Take some time to make yourself familiar with the utilities and the way they behave inside a Macintosh volume, between different Macintosh volumes, and between non-Macintosh and Macintosh volumes.

4 Notes about error messages

The "dt" utilities use standard error messages starting with the program name (including the command argument), followed by the file or directory currently processed, and the error message, e.g.:

```
dt rm notHere
```

may result in the error message:

```
dt rm: notHere: no such file or directory
```

The error messages used are similar to the errors issued by the standard UNIX commands. Please note that you may encounter additional error messages regarding the resource fork or the desktop database.

In some cases the "dt" utilities may issue error messages displaying the full path of a file name instead of the passed relative one. This is due to the fact that the desktop database always needs absolute file names instead of relative ones. In this case, you may see the completely resolved (including symbolic links) absolute path name in the error message.

5 Command descriptions

5.1 General remarks

In the following command descriptions the knowledge of the functionality of the corresponding UNIX commands is assumed. Please refer to your UNIX manual if you are not familiar with the commands.

If you do not need or want to know, how the “dt” utilities operate, you only need to read the main command description for each UNIX command which is emulated by the “dt” utilities. The technical description following each command is meant for advanced users or system administrators and explains some details or the special behaviour of the respective command.

With very few exceptions, “dt” behaves identically on all HELIOS-supported platforms. Please note that, therefore, some platform specific options could not be implemented.

For all commands, the standard UNIX permissions apply when accessing, removing, or overwriting a file.

“dt” checks each passed argument for the following names:

```
.Desktop
.DeskServer
.rsrc
```

These files are handled implicitly by “dt” and there is no need to specify them directly. You should *never* move, copy, or remove these files with other commands, as e.g. the standard UNIX tools. Do not manipulate them at all.

Note:	In the following, the term “volume” is used for a directory that is defined as an EtherShare volume which contains a desktop database (in contrast to a simple UNIX “directory”).

Note:	In the following, the term “volume root” means the top level folder of an EtherShare volume. The “EtherShare Applications” volume, e.g., is by default defined to reside in the “/usr/local/es/macapps” directory, which is its “volume root” directory under UNIX.

“dt” will recognize an EtherShare volume by locating its desktop database “.Desktop” file.

“dt” will recognize access to EtherShare volumes from another server only by means of available “.DeskServer” files.

“dt” will process all supplied files and folders, but only file/folder names of up to 31 characters will be kept in desktop databases.

“dt” may process files/folders differently, depending on whether the source/destination of the operation is inside/outside an EtherShare volume. The Macintosh Finder and the UNIX file system use different semantics and “dt” assures that on EtherShare volumes the Macintosh semantics will be applied. The differences are mainly visible in permissions/ownership.

Symbolic links to folders will not be processed on EtherShare volumes.

“dt” will process data and resource parts of files/folders on EtheShare volumes. In case no resource information exists, “dt” will be able to create at least minimum information. Especially copy or move operations may require more free space on destination file systems.

The following command modes are supported:

```
rm        remove a file or directory (UNIX “rm”)

rmdir     remove directory (if empty)
          (UNIX “rmdir”)

mv        move/rename a file or directory
          (UNIX “mv”)

cp        copy a file or directory (UNIX “cp”)

set       set or change the Macintosh specific file
          attributes

ls        list contents of a directory including
          Macintosh specific file attributes
          (UNIX “ls”)

mkdir     create a directory (UNIX “mkdir”)

touch     create a file or set its access time
          (UNIX “touch”)

upd       force update of Macintosh volume view

chmod     change or assign the mode of a file
          (UNIX “chmod”)

chown/    set the user and group ID of a file
chgrp     (UNIX “chown”, “chgrp”)

idinfo    display database information for the
          passed ID
```

5.2 dt (no argument)

If called without any argument, "dt" prints the usage line:

```
Usage: dt { rm | rmdir | mv | cp | set | ls |
mkdir | touch | upd | chmod | chown | chgrp |
idinfo }
```

Calling "dt" with the command argument but no arguments, prints the usage for the specific command,
e.g. calling `dt rm` prints:

```
Usage: dt rm [-fir] file ...
```

Calling "dt" with a question mark, prints the version in addition to the usage line, e.g. calling `dt -?` prints:

```
dt (c) Helios Software GmbH, Version 2.5.0
Usage: ...
```

5.3 dt rm

The "rm" command removes the directory entry specified by each file argument.

```
Usage: dt rm [-fir] file ...
```

The following options apply:

```
-f  force removing without prompting the user.

-i  prompt for confirmation for each file. If
    the answer begins with y (yes), the file
    is deleted, otherwise the file remains.
```

```
-r  recursively remove passed directory and
    its subdirectories.
```

In case of symbolic links, the link itself and not the file the link refers to is removed.

If the standard input is not a terminal, the command will operate as if the -f option were set.

Examples

```
dt rm a.out core
```

removes the directory entries: a.out and core.

```
dt rm -rf junk
```

removes the directory junk and all its contents, without prompting.

Technical notes

Missing resource forks are not reported unless the -v verbose option is set.

Volume root is automatically touched to announce the changes to any Macintosh that has mounted the volume.

When removing a directory using the “-r” option, “dt” automatically tries to remove any orphan resource forks. The volume root directory may only be removed, if this volume is not in use by any other client, either EtherShare or the “dt” utilities, otherwise a “directory not empty” error is issued.

5.4 dt rmdir

The “dt rmdir” command will remove the directory entry specified by each dirname operand, provided that this operand refers to an empty directory.

```
Usage: dt rmdir [-ps] dirname ...
```

The following options apply:

```
-p  allow users to remove the directory
    dirname and its parent directories which
    become empty.  A message is printed on the
    standard error about whether the whole path
    is removed or part of the path remains for
    some reason.

-s  suppress the message printed on the
    standard error when -p is in effect.
```

5.5 dt mv

"dt mv" moves the selected files or directories to a destination file or directory.

```
Usage:  dt mv [-fiknvz] f1 f2
        dt mv [-fiknvz] f1 ... fn d1
        dt mv [-fiknvz] d1 d2
```

The three different usages are provided for the following cases:

- move a file f1 to the destination file f2
- move the passed files "f1 ... fn" into the destination directory "d1"
- move the passed directory into the destination directory "d2", and create "d2" if it does not exist.

The following options apply:

```
-f  move the file(s) without prompting even
    if it is overwriting an existing target.
    Note that this is the default if the
    standard input is not a terminal.

-i  prompt for confirmation for each file. If
    the answer begins with y (yes), the file is
    moved, otherwise it remains where it is.

-k  keep ID, try to allocate the source ID and
    use it for the destination as well (see
    also A 7 “Using “dt” for backup/restore”).
    In case this is impossible, you will only
    get a warning if -v is set.
    You can use this parameter to preserve
    proper working of Macintosh aliases.

-n  no resource forks if no desktop (see
    A 3 “Default resource fork handling”).

-v  verbose, display extended warnings.

-z  force zero ID for destination, please
    refer to A 6 “Zero IDs” and
    A 7 “Using “dt” for backup/restore”.
```

Technical notes

The behaviour of the different UNIX implementations differs if both, the -f option and the -i option, are set. “dt mv” uses the following rule: if both, the -f option and the -i option, are set, this is not considered an error; here, the -f option will override the -i option.

If you move a directory between two EtherShare volumes, “dt” cannot simply move just the directory entry. The Macintosh Finder would have to perform two steps for this operation, namely copying a folder from one volume to the other and then deleting the folder on the source volume.

"dt" has a similar task to accomplish. For every file and folder within the directory to move, first the object must be moved to the destination volume and registered in the destination desktop. Then, the moved objects must be deleted from the source volume and unregistered from its desktop database.

Please note that any Macintosh files/folders to which aliases are pointing, will have lost their connection after successful move between volumes. This would be the same if you used the Macintosh Finder for this operation.

If source and target directory are on different file systems, "dt mv" copies the file and deletes the original; any hard links to other files are lost.

Volume root for both, source and destination, is automatically touched to announce the changes to any Macintosh that has mounted the volume.

Please note that whenever the destination is a volume, the volume specific permissions will apply, meaning that the file you have moved will automatically inherit the permissions of the directory it has been moved to.

Usually, if a file is copied to a directory without a desktop (pure UNIX directory), the file ID is set to zero. When using the -k option, the file ID is maintained (if possible).

5.6 dt cp

"dt cp" copies the passed files or directories to a destination file or directory.

```
Usage:  dt cp [-finpvz] f1 f2
        dt cp [-finpvz] f1 ... fn d1
        dt cp -r [-finpvz] d1 ... dn-1 dn
```

The three different usages are provided for the following cases:

- copy a file f1 to the destination file f2
- copy the passed files “f1 ... fn” into the destination directory “d1”
- copy the passed directory into the destination directory “dn”

The following options apply:

```
-f  copy the file(s) without prompting even
    if it is overwriting an existing target.
    Note that this is the default if the
    standard input is not a terminal.

-i  prompt for confirmation for each file. If
    the answer begins with y (yes), the file is
    copied, otherwise it is not.

-r  recursively copy passed directory.

-n  no resource forks if no desktop (see
    A 3 “Default resource fork handling”).

-p  preserve the owner and group IDs,
    permission modes, modification and access
    time.

-v  verbose, display extended warnings.

-z  force zero ID for destination, please refer
    to A 6 “Zero IDs” and
    A 7 “Using “dt” for backup/restore”.
```

Technical notes

Volume root for both, source and destination, is automatically touched to announce the changes to any Macintosh that has mounted the volume.

If your source does not have a resource fork, and your destination is a volume, a default resource fork is created. These default resources are only recognized properly by EtherShare 2.5; earlier EtherShare versions will display files with plain document icons. Please do also take into account that every default resource will require 64 bytes, thus using the minimum disk block size on the destination file system. If there is only little free disk space on the destination volume, first check whether the destination file system can store all of the files. You can roughly calculate 1K (fragment size) per file/folder. Consult your UNIX system administrator or your UNIX man pages for information on how to retrieve more exact information about your file system settings.

5.7 dt set

“dt set” modifies the additional file information stored in the file’s resource fork. For more information about the flags and fields used here, please refer to your Macintosh documentation, e.g. Apple’s “Inside Macintosh”.

You can use the EtherShare Admin to extract the creator/type information an application sets for certain icons. See also chapter “Other lists and accounting files” in your EtherShare documentation (and the contents of $ESDIR/conf/suffixes).

```
Usage:  dt set [-t type | -c creator | -f sfile]
        file ...
        dt set -a[-][isrlbp] file ...
```

The following options apply:

```
-t      set the file type for “file” to the
        passed value
```

```
-c      set the file creator for “file” to the
        passed value

-f sfile    copy the file type and creator from
            “sfile” to “file”
```

-a flag, where flag specifies the following modes:

```
i   invisible
s   system
r   read only
l   locked (norename nodelete)
b   backup
p   protected
```

Please note that you may list these information using “dt ls”. If you need to set type or creator info with non-printable values, you may as well enter these values as octal values with a preceding backslash, and use the following escape sequences:

```
\b  backspace
\n  newline
\r  carriage-return
\t  tab
\f  form-feed
\E  escape
```

Please note that for both, type and creator, all characters including 0 are valid. Each passed creator or type must be exactly 4 bytes long.

Examples

```
dt set -t “TEXT” -c “R*ch” file
```

sets type and creator for file using ASCII values.

```
dt set -c "\001\002\003\004" file
```

sets creator for file using octal values.

```
dt set -f mypic file
```

copies type and creator from mypic to file.

```
dt set -ai file
```

sets the invisible flag for file.

```
dt set -a-i file
```

clears the invisible flag for file.

Technical notes

When you call "dt set" for a file without a resource fork, a default resource fork updated with the passed parameters is created, even if the destination file does not lie in a Macintosh volume.

Please note that "dt set" does not create a missing ".rsrc" directory; use "dt touch" first on the file name in order to force creation of a related ".rsrc" subdirectory where the file's resource information can be stored.

5.8 dt ls

For each file that is a directory, the contents of the directory is listed; for each file that is an ordinary file, the name and resource related information are listed. When no argument is given, the current directory is listed.

```
Usage: dt ls [-axotTsmldi] file ...
```

The following options apply:

-a list all entries, including those beginning with a dot (.), which are normally not listed.

-x force printing of non-printable characters for type and creator – which are normally shown as underscore (_) – in the hexadecimal \xdd notation.

-o force printing of non-printable characters for type and creator – which are normally shown as underscore (_) – in the octal \ddd notation.

-t for each file, print the Macintosh creation time.

-T for each file, print the UNIX last modification time.

-s for each file, print the size of the data and the resource fork.

-m print file name stored in the desktop database instead of UNIX name. If any difference between these names is detected, both names are shown. If the target directory does not have a corresponding desktop, this option is ignored.

-l list in long format. This option is equivalent to specifying the option -tsi.

-d if the argument is a directory, list only its name and not its contents.

-i for each file, print the file or directory id (as stored in the resource fork) and the parent id (as stored in the desktop database).

```
This information may not be correct in case
files/folders were manipulated with UNIX
programs other than the “dt” utilities.
Use “rebuild -s” to verify consistency
between volume and desktop information.
```

The attributes printed in the first column consist of 8 characters, and are indicated as follows:

```
d   directory
A   alias
i   invisible
s   system
r   read only
l   locked (norename nodelete)
b   backup
p   protected
```

Please note that Macintosh aliases cannot be created with the “dt” utilities. The internal structure of aliases is not documented by Apple.

Examples

```
dt ls /usr/ethershare

d-i-rl--                    Network Trash Folder
d-------                    OpiPics
-------- 'XDOC' 'XPR3'       mydoc
```

This command prints the names of all files in the passed directory “/usr/ethershare”, showing the file attributes, the type and creator for that file and its name. Please note that directories do not have any type or creator and therefore the fields are empty for directories.

Often you may want to include more Macintosh specific information in the output. You can use the -l option for that:

```
dt ls -l /usr/ethershare
```

Here you see the attributes, the ID, parent ID, type, creator, Macintosh date, size of the data, size of the resource fork, and the file name.

```
d-i-rl- 3  2               01/15/97 09:47 512 64 Network
                                                  Trash Folder

d------ 4  2               01/16/97 13:50 512 64 OpiPics

------- 22 2 'XDOC' 'XPR3' 06/09/97 13:39 976 64 mydoc
```

Technical notes

Please note that the output format (e.g. time format) is dependent on the current settings for the local environment.

The output will not be sorted (as it would be if you used the UNIX “ls” command).

5.9 dt mkdir

The “mkdir” command creates the passed directories including the corresponding resource fork and the desktop database entry.

```
Usage: dt mkdir [-m mode] [-p] dir ...
```

The following options apply:

```
-m mode   specify the mode to be used when
          creating the directory.

-p        With this option, mkdir creates “dir”
          by creating all the non-existing
          parent directories first. The mode
          given to intermediate directories will
          be the difference between 777 and the
          bits set in umask.
```

Example

```
dt mkdir -m0444 mydir
```

creates the directrory "mydir" with the passed modes.

Technical note

If no mode parameter is set, the final directory is created in mode 777 considering the file mode creation mask "umask".

5.10 dt touch

The command "touch" sets the access and modification times for the passed files. A file is created if it does not already exist.

```
Usage: dt touch file ...
```

Example

```
dt touch  myfile
```

5.11 dt upd

The "upd" command touches the volume directory for the passed file or directory, so that all Macintoshes that have mounted this volume can display the changes. This may be necessary e.g. if a file is created under UNIX and the Macintosh would display the new icon only after closing and reopening the corresponding window.

Please note that all "dt" commands automatically update any needed volumes; so this command is only required, if files are created or changed by other, non-"dt" commands. Check whether you can adjust your workflow to "dt" commands and thus avoid potentially harmful non-"dt" commands.

```
Usage: dt upd file... | dir ...
```

Example

```
dt upd myfile
```

Technical note

The Macintosh clients only scan for changes every 10 seconds. Usually, folder information are updated within this 10 seconds period. However, depending on folder sizes, number of folders open, foreground applications, etc., this may take longer. Thus, it may take some time after issuing an “upd” command, before the changes become visible on the client. You can either click into the folder window, or close and re-open it in case the displayed information do not seem to be up-to-date. In rare situations, it may be necessary to unmount and remount the volume in question in order to let the Macintosh flush its cache entries. Make sure that you are using the latest Macintosh OS and up-to-date AppleTalk/Network drivers.

5.12 dt chmod

The “chmod” command changes the permission mode of a file. The mode may be absolute or symbolic.

```
Usage:  dt chmod [-fRn] absMode file ...
        dt chmod [-fRn] [ugoa]{+|-|=}[rwxstugo]
        file ...
```

The following options apply:

```
-f  force, do not complain if the mode change
    fails.

-R  recursively descend through directory
    arguments.

-n  change directory only – not the files
    within the directory. The default
    behaviour of the “dt chmod” command is
    as follows:
```

```
If a pure UNIX directory is changed,
the changes are applied to the directory
only. Else, if a directory on a volume
is changed, the changes are applied to the
files within this directory as well (they
are not applied to subdirectories).
The -n option makes sure that the
command is always applied to the direc-
tory only.
```

An absolute mode is specified using octal numbers (0-7), please refer to the UNIX documentation for a full description.

A symbolic mode specification is a comma-separated list (with no intervening spaces) of symbolic mode expressions of the form:

```
[who] operator [permissions]
```

In order to adjust permissions to files/directories and their related resource parts, you must own the directory in question. In case the resource directory does not match the enclosing directory’s permissions, “dt chmod” cannot adjust the “.rsrc” directory’s permissions. You have to log in as “root” in order to synchronize permissions of the two directories. Please refer to the UNIX documentation for a full description.

Examples

```
dt chmod 444 file
```

sets only read permission (0444) to “file”.

```
dt chmod a=rwx,g+s file
```

sets read (“r”), write (“w”), and execution (“x”) permission for all (“a”) users, and adds group set-ID mode (“s”) for groups’ permissions.

5.13 dt chown

The “chown” command sets the user ID of the passed file to the new user ID specified by owner, and, optionally, the group ID to the new group.

```
Usage: dt chown [-fhRn] owner[:group] file ...
```

The following options are supported:

```
-f  do not report errors.

-h  if the file is a symbolic link, change the
    owner of the symbolic link (not the owner
    of the file that is referenced by the
    symbolic link). This option is not
    supported on all platforms.

-R  recursively descend through the directory
    when setting the ownership for each file.

-n  change directory only – not the files
    within the directory. The default
    behaviour of the “dt chown” command is
    as follows:
    If a pure UNIX directory is changed,
    the changes are applied to the directory
    only. Else, if a directory on a volume
    is changed, the changes are applied to the
    files within this directory as well (they
    are not applied to subdirectories).
```

```
        The -n option makes sure that the
        command is always applied to the direc-
        tory only.

    owner[:group] (use "." or ":")
        A user ID and an optional group ID to be
        assigned. Both IDs may either be specified
        by the numeric ID or by the corresponding
        user or group name.
```

Please note that you need the necessary access rights to change the user or group ID of a file – according to UNIX semantics, only the userid "root" is allowed to change ownership settings.

Examples

```
dt chown frank:support myfile
```

changes the owner for myfile to "frank", and the group to "support".

```
dt chown 100 myfile
```

changes the owner for myfile to the user defined with the ID 100.

5.14 dt chgrp

The "chgrp" command sets the group ID of a given file to that of the new group.

```
Usage: dt chgrp [-fhRn] group file ...
```

The following options are supported:

```
-f  do not report errors.
```

```
-h  if the file is a symbolic link, change the
    group of the symbolic link (not the group
    of the file that is referenced by the
    symbolic link). This option is not
    supported on all platforms.

-R  recursively descend through the directory
    when setting the group for each file.

-n  change directory only – not the files
    within the directory. The default
    behaviour of the “dt chgrp” command is
    as follows:
    If a pure UNIX directory is changed,
    the changes are applied to the directory
    only. Else, if a directory on a volume
    is changed, the changes are applied to the
    files within this directory as well (they
    are not applied to subdirectories).
    The -n option makes sure that the
    command is always applied to the direc-
    tory only.
```

A group ID may either be specified by the numeric ID or by the corresponding group name.

Please note that you need the necessary access rights to change the group ID of a file – according to UNIX semantics, only the userid “root” is allowed to change ownership settings.

Examples

```
dt chgrp support myfile
```

changes the group to “support”.

```
dt chgrp 100 myfile
```

changes the group for myfile to the group defined with the ID 100.

5.15 dt idinfo

The main purpose of this command is problem detecting.

The Macintosh file or respectively directory ID is stored in both, the file’s resource fork and the desktop database. The ID stored in the resource fork may be listed by using the “dt ls” command.

The “idinfo” command may be used to display database information for the passed ID. If there is no difference between the resource data and the corresponding database entry, you should get the same information, otherwise the system administrator should issue a “rebuild” for the specific volume. You can use `rebuild -s <volume root>` to verify that the volume information is properly reflected in the related desktop database; see “Before getting started...” in chapter 3 above.

```
Usage: dt idinfo [-v volume] id ...
```

Please note that each volume has its own ID scope. Therefore, you must specify the correct volume for each ID you request. If you do not specify a volume with the -v option, the current working directory is assumed.

```
-v  allows to specify the volume to be used, or
    any file or directory inside the target
    volume.
```

Example

```
dt idinfo -v /usr/ethershare/etc 20
```

prints the following information:

```
volume: “/usr/ethershare” id=20 pid=18 name:
“qdoc”
```

specifying the volume detected, the ID, the parent ID for this file, and the corresponding name stored in the database.

Technical note Please note that an ID detected in the database that does not have any corresponding file in that volume does not cause any problem or data loss.

Appendix

A 1 General remarks

The information in the appendix are mainly meant for system administrators who want to integrate the "dt" utilities into custom solutions, e.g. backup scripts. You do not need these information for the simple use of the "dt" utilities (it does not hurt, however, to read on).

A 2 Macintosh files under UNIX

As described in the EtherShare manual, each volume used in EtherShare is accompanied by a corresponding desktop database, which holds additional information about each file or directory stored in this volume.

Each file or directory in a Macintosh volume consists of two parts: "the data fork" and the "resource fork".

This is invisible to the Macintosh user, both parts of a file and the corresponding information in the desktop database are handled transparently and automatically by EtherShare.

If you also use EtherShare OPI 2.0 and/or PCShare 2.5, these HELIOS products will also coordinate access to shared files/folders with EtherShare's volume desktop databases (see chapter "The EtherShare File Server" in the EtherShare manual).

When you look at a volume from the UNIX point of view, you will find two files: the data fork as the "normal" UNIX file, and the resource fork in a subdirectory ".rsrc" of the same name.

You will find more information about this in the EtherShare manual (chapter "Directory and file formats").

A 3 Default resource fork handling

Since UNIX files consist of only one part, and Macintosh files of two (see A 2 “Macintosh files under UNIX”) normally, each pure UNIX directory (“UNIX-dir”) should only have entries without a corresponding resource fork, and in a Macintosh volume (“volume-dir”) each entry should have a corresponding resource fork.

If you now copy or move an entry from a “UNIX-dir” into a “volume-dir”, “dt” always creates a default resource for this file. This is also the default behaviour, if you move or copy a file from one “volume-dir” to the other.

If you copy an entry from a “volume-dir” into a “UNIX-dir”, “dt” copies or moves the resource fork as well, even if the resource fork is not needed here. This is done in order to avoid any unintentional data loss. If you do not want to consider the resource forks here, you may use the -n option for the “mv” or “cp” command.

A 4 EtherShare and desktops

The “dt” utilities do not require EtherShare to run on the UNIX server where you issue the commands, but by default the “dt” utilities try to connect to the local desktop server on the server where you issue the commands first.

If this volume is already in use (e.g. mounted as a volume on a Macintosh) the “dt” utilities try to connect to the current server which serves this volume. This could be a different server, e.g. if this volume is mounted via NFS. Source and destination are handled completely independently, e.g. a source file for a copy command may be served by a local desktop server, while the destination directory which is lo-

cated on a NFS-mounted directory is served by another EtherShare desktop server.

The "dt" utilities are unable to connect to a desktop server, when the target directory is not mounted by any Macintosh and *no local* desktop server is running. In this case either start a local EtherShare copy or mount this volume on a Macintosh.

The "dt" utilities will do an extensive verification in order to connect to the proper desktop server. Nevertheless, please verify your environment, so that no single EtherShare volume can be accessed at the same time from different EtherShare servers.

Note: In case different desktop servers are involved in a move directory operation, it may be necessary to traverse through directories instead of just moving the directory entry – see "Technical notes" for "dt mv" for more details.

A 5 "mount points" and "auto-mounter"

Double mounts (access to the same directory under different mount points, e.g.:

```
mount   myserver:/usr/es  /es
mount   myserver:/usr/es  /home/user1/es_copy)
```

will lead to differences between volume and desktop information, resulting in inconsistencies. Please note that you may unintentionally cause this problem by accessing an already mounted volume via auto-mounter:

```
dt cp /net/myserver/usr/etherhare/file1 file2
```

You may avoid this problem by running auto-mounter on any involved server (source and destination).

A 6 Zero IDs

On any mounted volume, EtherShare automatically detects any file which has an accompanying resource fork with a zero ID (the ID set to numerical zero) when the parent directory is opened from a Macintosh.

A file without a valid ID does not have any database entry, and therefore, a new desktop database entry is created, and the resource fork is updated using the new ID.

The “dt” utilities never create zero ID resource forks in volumes unless you force this behaviour using the -z flag for the “cp” and “mv” command.

Important: This option should only be used by system administrators – it is not intended for normal use.

A 7 Using “dt” for backup/restore

One main purpose of the “dt” utilities is to avoid problems when restoring data from a backup device. Here we describe how you should use the “dt” utilities for backup/restore purposes:

backup

You may backup your data as usual directly from your volume. Please make sure that the hidden “.rsrc” directories are included. To prevent backup of incomplete data in files, make sure that the files/folders to be backed up will not be manipulated from other Macintosh or UNIX applications. Only if you intend to do a complete backup and restore of

an EtherShare volume you might include the .Desktop file. For the period of time you need to backup and restore this volume, EtherShare *must* be stopped with “stop-atalk”.

restore

Important: Do never restore your data directly into the destination volume!

Please create a restore directory on the same device (partition) where the destination volume resides, e.g. if you intend to restore files into “/data/mypics”, create a directory “/data/restore”. This “restore” directory must not be defined as a volume, or mounted.

Now restore your data using your usual restore procedure into this newly created directoy “restore”. Then, remove any “.Desktop” and “.DeskServer” files that may have been included in your backup and now exist in the “restore” directory.

Important: Do never remove these files (“.Desktop” and “.DeskServer”) from a mounted volume or manipulate them directly. This may cause not only data loss for this volume, but may cause problems for all defined volumes on the server. In case you want to backup/restore these files, make sure that EtherShare and PC-Share (if you are sharing volumes) are stopped completely prior to your backup/restore operation. Only if you stop EtherShare by means of ”stop-atalk” and stop access from other EtherShare servers, intermediate access to these files from EtherShare processes can be prevented.

Finally move the restored files with e.g.:

```
dt mv /data/restore /data/mypics
```

into their final destination. This procedure avoids any data loss and may be used on a mounted destination volume.

Usually, new (unused) file IDs are assigned when you copy files to a backup destination. It may, however, be sensible to keep the file ID (using the -k option). Thus, the file can be re-used with its original ID.

This is especially useful if you are using Macintosh aliases pointing to these files/folders. You should use this option sensibly. In case a complete desktop rebuild has been conducted after your backup operation, aliases may point to the wrong destination.

To avoid any problems, it may be a good idea to run an integrity check for the destination volume using the “rebuild -s” command before restoring any files.

A 8 Problem reports

We always appreciate feedback about how UNIX administrators and Macintosh users do use these tools and about how they understand their working. What is even more important, however, is feedback about problems that have occurred.

If you are about to contact our support department (support@helios.de), please be prepared to submit the following information:

- Customer contact: name, e-mail address, fax, phone
- HELIOS partner contact: name, e-mail address, fax, phone

- For every EtherShare/PCShare server at your site:
  EtherShare: contents of “atalk.conf”, “afpvolumes”, “Versions”
  PCShare: contents of “pcshare.cnf”, “exports.pcs”, “Versions”

- UNIX: information about the hardware and operating system

- Contents of “/etc/passwd”, “yp/passwd” (we are only interested in the location of users’ home directories)

- Output of “/etc/(v)fstab”, “exports”

- Output of “find / -name .afpvolumes -print”, “find / -name .Desktop -print”

- “rebuild -s” for every EtherShare volume

- In case any changes have been made to your initial EtherShare/PCShare configuration, we need the contents of the current configuration files.

- Send the exact and complete command sequences you issued together with the exact and complete output produced by the “dt” utilities and/or other UNIX programs you have used. (If you can use a scrollable command window, e.g. within XWindows, OpenWindows, CDE, or similar tools, this will facilitate the task of providing this information, especially since the history of earlier command sequences is also available in this window!)

- Let us have the output of “ls -la” for the directory where you applied the “dt” command, as well as for the corresponding “.rsrc” directory. In case you applied a change recursively, you can simply use “ls -laR”.
- If you used a “dt” command which processes a source and destination file/folder, please submit all the information for both, source and destination.
- Let us have the output of “ps -efl”, or “ps aux”, respectively.
- Let us have the output of “$ESDIR/swho -c” and “$PCDIR/swho -c”.
- Let us have the output of “df”.
- Specify the contents of the system messages files since the last start of EtherShare.
- Let us have a description of what you wanted to achieve, and what – you think – has happened.
- Is the result reproducible or does it emerge only occasionally? Any clue on the suspected pattern will be helpful!

In case you have additional information available, e.g. core files, trace output, please store it savely and make a note in your report – do *only send it on request*.

Depending on the severity of the problem it may be necessary to stop and start EtherShare services completely. If you use shared volumes together with PCShare, stop PCShare first, and restart it only after restarting EtherShare.