

# Distributed Media-Manager

## Storage - Library - Management

### Charakteristik:

Der Distributed Media-Manager (kurz: DIMM) ist ein UNIX oder NT- Prozess, der programm- und kommandozeilengesteuert die Kontrolle über mehrere verteilte Datenträgersysteme anbietet und die Basis für ein integriertes Archivsystem bildet. Hauptmerkmal ist die Connectivity zu Standardsoftware wie Workflow-Management, Bilddatenbanken in heterogener Rechnerumgebung. Erreicht wird dies durch eine durch eine komfortable Skriptierungs-Schnittstelle, die via TCP/IP alle Funktionen des DIMM von außen verfügbar macht. Auch der Einsatz als Standalone-Archivlösung ist durch komfortable grafische Client-Programme für MacOS / Windows 95,NT realisiert.

Als Stagesysteme werden Tape-Libraries, Jukeboxes und andere SCSI-2 - kompatible Geräte unterstützt. Der DIMM legt auf Anforderung gewünschte Medien in freie Laufwerke und bietet durch volle Unterstützung moderner Bandlaufwerke wie DLT™ 4000 / 7000 und DAT optimale Performance beim Aufsuchen und Zurückladen von archivierten Daten. Es können Datenbestände über mehrere Libraries verwaltet werden, parallele Verwendung von mehreren Laufwerken ist ebenfalls implementiert.

### Funktionsmerkmale Library-Interface:

- Serverprozess für Solaris ab 2.4 / Windows NT ab 4.0
- Kommandozeilen-Schnittstelle für UNIX / MacOS / NT / W95 / Win 3.11
- Gut dokumentiertes Script-Interface mit kompletter Fehlerbehandlung zur Integration von Library-Verwaltung in existierende Programmumgebungen.
- Grafisches Interface zur Librarysteuerung für MacOS / NT / W95 / W3.11, UNIX-Version auf Anfrage
- Unterstützung von SCSI-2 kompatiblen Libraries von einfachen 7-fach DAT-Wechslern bis hin zu großen Robotern mit Barcode-Leser, Import/Export-Slots und multiplen kapazitätsstarken Laufwerken wie DLT™ 7000.
- Unterstützung der Highspeed-Positionierung von SCSI-2 kompatiblen Bandlaufwerken wie DLT™.
- Parallele Unterstützung von mehreren Laufwerken, gleichzeitiges Archivieren und Restaurieren auf verschiedenen Laufwerken.
- Unterstützung von mehreren Libraries zum Verwalten von Library-übergreifenden Datenbeständen.
- Unterstützung von Einzellaufwerken (z.B. zum manuellen Erzeugen von Clone-Bändern)
- Dokumentierte Programmierschnittstelle zur Integration von DIMM-Aufrufen in Applikationen für UNIX / MacOS / Windows
- Duplizierung von Medien (Cloning)
- Einfache Installation (über pkgadd unter Solaris, Installationsprogramm unter NT)

# Distributed Media-Manager

## Archiv - Handling

Auf der eigentlich Gerätesteuerung aufgesetzt ist das Archivsystem, das eine abstrakte und medienunabhängige Verwaltung von Daten erlaubt. Daten können datei- oder verzeichnisweise als sogenannte Media-Jobs archiviert und restauriert werden. Das eigentliche Anfordern und Verwalten von Medien wird vom Archivsystem gekapselt. Die Daten werden blockweise abgelegt, so daß auch einzelne Dateien aus größeren (z.B. > 1GB) Archiv-Aufträgen mit optimaler Geschwindigkeit restauriert werden können.

Das eigentliche Datenaufzeichnungsformat ist das bewährte Standard-Tar Format. Zur Unterstützung von hierarchischer Speicherarchitektur kann beim Archivieren festgelegt werden, welche Speichertechnologie (z.B. Bandlaufwerke, MO-Disks, DVD) verwendet werden soll.

Archivaufträge können über mehrere Mechanismen erzeugt werden:

1. Spooling:

Kommandodateien mit Informationen über zu archivierende Dateien, Verzeichnisse und Archivierungszeitpunkt werden in ein Spoolverzeichnis des Servers kopiert, das der DIMM überwacht und auswertet. Nach erfolgter Archivierung wird eine Antwortdatei vom Dimm erzeugt, in der Fehlerstatus, verwendete Medien und Bandpositionen hinterlegt sind.

2. Client Tool:

Aus dem grafischen Client-Tool heraus können Archivjobs erzeugt und gestartet werden, existierende Archiv-Jobs können erneut archiviert werden, um z.B. Projekte in verschiedenen Entwicklungsstadien im Archiv zu hinterlegen.

3. Command Line Interface:

Gleiche Funktionalität wie 2. jedoch als Script aus Standardapplikationen heraus aufrufbar.

Mittels grafischer Clientprogramme für die wichtigsten Anwendungsplattformen können die archivierten Daten jobweise und verzeichnisweise wie ein normales Dateisystem durchsucht und zurückgespielt werden. Dabei kann auch auf ältere Versionen von Jobs und Dateien, die mehrfach archiviert wurden, zugegriffen werden.

Besondere Sorgfalt wurde auf die Unterstützung von MacOS im Umfeld mit UNIX-Fileservern gelegt (z.B. Helios Ethershare™, Netatalk). Die Behandlung der Resourcefork wird ebenso unterstützt wie der korrekte Wiederaufbau von Desktop-Dateien beim Restore und die Ikonen-Darstellung von archivierten Dokumenten.

## Funktionsmerkmale Archivsystem:

- Abstrakte medienunabhängige Job-orientierte Datenverwaltung
- Kommandozeilen-Schnittstelle für UNIX / MacOS / NT / W95 / Win 3.11
- Gut dokumentiertes Script-Interface mit kompletter Fehlerbehandlung zur Integration von Archivfunktionalität in existierende Programmumgebungen.
- Grafisches File-Browser ähnliches Interface zur Datensichtung, Archivierung und Restaurierung für MacOS / NT / W95 / W3.11. UNIX-Version auf Anfrage
- Dokumentierte Programmierschnittstelle zur Integration von DIMM-Aufrufen in Applikationen für UNIX / MacOS / Windows
- Unterstützung von Apple Filesystem unter Helios EtherShare™ und Netatalk
- Zugriff auf Daten auch mit Standard-Tools wie tar und mt möglich.
- Unterstützung von verschiedenen Speichertechnologien (Bänder, MO, DVD)
- Disaster-Recovery nach komplettem Server-Crash von Band
- Blockweise Datenverwaltung für zeitoptimales Restore (je nach Laufwerks- und Librarytyp zwischen 30 s und 3 min).
- Parallele Archiv- und Restore-Jobs auf gleichen / verschiedenen Laufwerken
- Zugriff auf alle Versionen von mehrfach archivierten Dateien.
- Kompaktierung von Bändern nach Löschen von Dateien möglich.
- Nachträgliches Aufstocken der Kapazität durch Erweiterung von Libraries oder zusätzliche Libraries möglich.

# Distributed Media-Manager

## Scripting Interface

Das Script-Interface ist eine komfortable und leicht lesbare Schnittstelle zur Integration des DIMM in bestehende oder geplante Rechnerumgebungen. Ein Aufruf besteht aus einem Text-Kommando mit eventuellen Parametern und einer Rückgabe mit Fehlernummer und optionalem Fehlertext. Alle vom DIMM verwalteten Geräte wie Libraries und Laufwerke erhalten eine frei wählbare Klartextbezeichnung wie „DLT1“ oder „ATL-Library“, alternativ können Bandlaufwerke auch über ihren Systemnamen (z.B. /dev/rmt/2n) angesprochen werden. Zusätzlich zu den eigentlichen Befehlen wie „load“, „locate“ usw. existieren umfangreiche Kommandos zum Auswerten von Status, Dateninhalten und aktueller Aktivität.

Das Script-Interface kann sowohl von UNIX oder NT-Shells aufgerufen werden, als auch von Clientprogrammen unter MacOS und Windows. Zusätzlich können Entwickler durch Einbinden einer Bibliothek die Library- und Archiv-Funktionalität innerhalb ihrer Anwendung nutzen.

Beispiele für grundlegende Positionier und Ladebefehle:

```
list_drives           // Erzeuge Liste von bekannten Laufwerken und Wechslern
0:
DLT1 <Scsi-II Tape> /dev/rmt/1n
DLT2 <Scsi-II Tape> /dev/rmt/2n
ATL <Scsi-II Changer>

mag_to_drv 0 0 0 ATL // Lege das Band aus dem 1. Magazin in das 1. Laufwerk im
0: Ok               // Wechsler „ATL“ ein.

load DLT1           // Lade das aktuelle Medium im Laufwerk DLT1
0: Ok               // Kein Fehler (Fehlernummer = 0, Fehlertext = „Ok“)

read_media DLT1     // Lese das Medien-Label des Bandes im Laufwerk DLT1
0: MEDIA01         // Band hat das Label „MEDIA01“

filelocate 25 DLT1 // fahre die 25. Datei auf dem Band an (= mt -f ... fsf 25)
0: Ok

filepos DLT1        // Gib die aktuelle Dateinummer
0: 25              // Dateinummer ist 25

filepos /dev/rmt/1n // Wie zuvor, jedoch Verwendung des System-Namens
0: 25              // Dateinummer ist 25

pos DLT1            // Gib die aktuelle Blockposition des Bandes
0: 25443           // Blockposition ist 25443

mag_to_drv 0 0 0 ATL // Lege das Band aus dem 1. Magazin in das 1. Laufwerk im
0: Ok               // Wechsler „ATL“ ein.

drv_to_exp 1 3 0 ATL // Lege das Band aus dem 2. Laufwerk in den 4. Import/Export -
0: Ok               // Slot im Wechsler „ATL“ ein.
```

Beispiele für medienbezogene Befehle:

```
lock_tape MEDIA01           // Lege das Band mit der Kennung MEDIA01 in das nächste
0: /dev/rmt/1n             // freie verfügbare Laufwerk und sperre es für weitere Aufrufe.
                           // Band ist im Laufwerk /dev/rmt/1n

lock_tape MEDIA01           // 2. Anforderung
11: is locked              // Schlägt fehl, ein bereits „gelocktes“ Medium kann nicht ein
                           // weiteres mal angefordert werden

release_tape MEDIA01       // Freigabe des Lockzustandes für Medium „MEDIA01“
0: released

label_tape MEDIA02         // Erzeuge ein neues Band mit dem Label MEDIA02
0: /dev/rmt/2n             // Frisch gelabeltes Band liegt im Laufwerk /dev/rmt/2n

clone DLT1 DLT2            // Kopiere die Daten vom Band im Laufwerk DLT1 auf das Band
12: has label MEDIA02     // im Laufwerk DLT2.
                           // Fehler: auf das Band im Laufwerk DLT2 wird nichts kopiert
                           // da es ein gültiges Label besitzt (dies ist eine Schutzfunktion
                           // des DIMM zum Verhindern von ungewolltem Überschreiben)

list_media                 // Liefere eine Liste aller gelabelten Medien
0:                          // Liste enthält alle bekannten Medien mit aktuellem Zustand
MEDIA00 is full
MEDIA01
MEDIA02 is empty

list_bc_labels ATL        // Liefere eine Liste aller verfügbaren Medien im Wechsler ATL
0:                          // Liste aller bekannten Medien mit Barcode und Kennung
M0: ASE400                 // des aktuellen Aufenthaltsorts (M0 ist das 1. Magazin, M1 das
M1: ASE401                 // 2., D1 das 2. Laufwerk, E3 der 4. Import/Export-Slot
...
D1: ASE446
E3: ASE447
```

Beispiele für Archiv-Befehle:

Besonders die Archiv- und Restore-Befehle werden aufgrund der Informationsmenge bei Parametern und Rückgabe schnell unübersichtlich. In der Regel werden sie vom Archiv-Client-Tool oder aus Programmanwendungen heraus aufgerufen, so daß der normale Anwender nicht mit ihnen in Berührung kommt. Dennoch zeigen die Beispiele, daß auch relativ komplexe und abstrakte Aufgaben über das Script-Interface automatisiert werden können.

```
archiv_job T2.1 /usr4/projects/p01234      // Erzeuge einen Archivjob mit der
// Medienkennung T (für Tape, Zielmedium ist also ein Band)
// und der Jobkennung 2.1 und archiviere alle Daten unterhalb
// /usr4/projects/p01234 incl. Unterverzeichnisse mit dieser
// Jobkennung
```

```
0: LB:<MEDIA01> BC:<ASE401> FSF:744 POS:343506
LB:<MEDIA01> BC:<ASE401> FSF:745 POS:345053
LB:<MEDIA02> BC:<ASE402> FSF:1 POS:2
```

```
// Die Daten wurden erfolgreich archiviert und zwar am Ende
// des Bandes mit dem Label MEDIA01 (Barcode ASE401,
// Dateinummer 744/745, Blockposition 343506/345053) und
// am Anfang des Bandes mit dem Label MEDIA02 (Barcode
// ASE401, Dateinummer 1 Blockposition 2)
```

```
list_jobs                                // Zeige alle archivierten Jobs
```

```
0:
```

```
NA:<T1.0> TI:13.05.98 SI:12050 DI:</usr4/projects/p0010
```

```
NA:<T1.1> TI:26.05.98 SI:9149 DI:</usr4/projects/p01221
```

```
NA:<T1.2> TI:05.06.98 SI:2153 DI:</usr4/projects/p01234
```

```
...
```

```
// Die Liste enthält alle archivierten Jobs mit Zeitpunkt, Größe in
// kB sowie Startverzeichnis
```

```
list_old_jobs T1.2                       // Zeige alle bereits archivierten Jobs mit der Kennung T2.1
```

```
0:
```

```
NA:<T1.2> TI:26.05.98 SI:2149 DI:</usr4/projects/p01234
```

```
NA:<T1.2> TI:05.06.98 SI:2153 DI:</usr4/projects/p01234
```

```
...
```

```
list_files_by_job T1.2 /usr4/projects/p01234/Bilder
```

```
// Zeige Inhaltsverzeichnis des Jobs 2.1 im Verzeichnisse
```

```
// /usr4/projects/p01234/Bilder an
```

```
0:
```

```
NA:<Urlaub.tiff> TI:18.04.96 SI:5504562
```

```
NA:<Sonne.eps> TI:... SI:...
```

```
...
```

```
// Die Liste enthält alle archivierten Dateien und Verzeichnisse
mit Alter, Größe in Byte, Typ/Creator bei MacOS usw.
```

```
restore_job /usr2/export/daten T1.1 T2.1
```

```
0: Ok
```

```
// Lese alle Daten der Jobs 1.1 und 2.1 zurück und zwar nicht
// ins Originalverzeichnis sondern nach /usr2/export/daten
```