Relax and Recover

Kurzdokumentation

Inhaltsverzeichnis

1 Relax and Recover....2
1.1 Überblick REAR....2
1.2 REAR Testumgebung....3
1.3 REAR Framework und Workflow....4
1.4 General Workflow....5
1.5 dump Workflow....6
2 Rescue System und Backup....7
2.1 mkrescue Workflow....7
2.2 mkbackuponly Workflow....9
2.3 mkbackup Workflow....11
3 Disaster Recovery und Restore....13
3.1 recover Workflow....13
3.2 Walkthrough....14
4 Referenzhandbuch....18
4.1 ReaR Installations- und Konfigurationsdateien....18
4.2 Konfigurationsvariablen....20

Lizenz

Dieses Dokument unterliegt der Creative Commons Attribution-Share Alike 3.0 Germany Lizenz (siehe http://creativecommons.org/licenses/by-sa/3.0/de/ für den vollständigen Text).

Sie dürfen:

- das Werk bzw. den Inhalt vervielfältigen, verbreiten und öffentlich zugänglich machen
- Abwandlungen und Bearbeitungen des Werkes bzw. Inhaltes anfertigen

Zu den folgenden Bedingungen:

- **Namensnennung** — Sie müssen den Namen des Autors/Rechteinhabers in der von ihm festgelegten Weise nennen.
- **Weitergabe unter gleichen Bedingungen** — Wenn Sie das lizenzierte Werk bzw. den lizenzierten Inhalt bearbeiten, abwandeln oder in anderer Weise erkennbar als Grundlage für eigenes Schaffen verwenden, dürfen Sie die daraufhin neu entstandenen Werke bzw. Inhalte nur unter Verwendung von Lizenzbedingungen weitergeben, die mit denen dieses Lizenzvertrages identisch, vergleichbar oder kompatibel sind.

Autoren

Kai-Uwe Schurig	Initiale Version
Schlomo Schapiro	Überarbeitung und Umsetzung in Open Office

Version

Version 0.9 vom 2009-10-29, 18 Seiten insgesamt. Das Originaldokument wird im ReaR Subversion gepflegt und kann von http://rear.svn.sf.net/viewvc/rear/trunk/usr/share/rear/doc/ bezogen werden. Im PDF ist immer auch die Quellversion des Dokuments enthalten (benötigt den PDF Importer). Dazu bitte das Dokument als Hybrid PDF exportieren.

1 Relax and Recover

1.1 Überblick REAR

Dieses Kapitel gibt einen Überblick zu den Auswahlkriterien, der Historie und den wesentlichen Features von REAR.

Warum REAR?

- sauberes, modulares Konzept
- aktive Weiterentwicklung
- REAR ist im Enterprise-Umfeld entstanden und auch für dieses Umfeld gedacht
- Schnelle Reaktion des Entwicklers auf Anfragen
- Optional professioneller Support möglich (kostenpflichtig)
- Feature Request für Aufnahme von REAR in den Standardumfang vom SLES läuft

Historie REAR:

REAR ist aus den folgenden Projekten hervorgegangen:

- OpenVPN Gateway Builder (OGB) von Schlomo Schapiro
- Make CD-ROM Recovery (mkCDrec) von Gratien D'haese

Dabei wurden aus OGB das modulare Konzept und aus mkCDrec die Grundlagen für den Teil Desaster Recovery übernommen. Die Version 1.0 von REAR wurde im Jahr 2006 veröffentlicht.

Aktuelle Versionen (Stand 04/2009)

- Stabile Version 1.6 (Dezember 2007)
- Entwicklerversion 1.7.20 (April 2009)

REAR Features:

- Focus auf Desaster Recovery
- Modulares Konzept
- Keine externen Abhängigkeiten (Nutzung von Standard-Tools der Linux-Distribution)
- Unterstützung für Linux Kernel > 2.6 (keine Unterstützung für Kernel 2.2/2.4)
- Benutzerfreundlich – minimale Ausgaben (Fehlermeldungen und Details siehe Logfile /tmp/rear.log)
- Unterstützung der gängigen Dateisysteme (z.B. ext[2,3,4], reiserfs, xfs, lvm2, software raid)
- Backup und Restore mit integriertem tar, Erweiterung möglich
- Ausgabe auf lokales Filesystem, remote Filesystem (NFS, CIFS), USB, PXE-Sserver, oder ISO File
- Integration in vorhandene Backup-Lösungen (z.B. IBM TSM, Veritas NetBackup, HP DataProtector)
- Open Source (GNU GPL v2)

1.2 REAR Testumgebung

Die Test für die Screenshots und Beispiele in diesem Kapitel wurden in einer VMWare-Testumgebung (VMWare Workstation 6.5.1) mit folgenden Parametern durchgeführt:

VM1
Name: sles10sp2-vm1-kus
IP: 192.168.61.128
OS: SLES 10 SP2 x86_64
Funktion: Testserver für REAR Backup/Recovery

VM2:
Name: sles10sp2-vm2-kus
IP: 192.168.61.129
OS: SLES 10 SP2 x86_64
Funktion: Backupserver (NFSV4-Share), PXE/TFTP-Server, DHCP-Server

Außerdem wurden zusätzliche Tests auch in einer weiteren Testumgebung mit physischen Linux-Servern HP DL380G5, DL360G5 und virtuellen Linux Servern in einer VMware VI3 Umgebung durchgeführt. Die beschriebenen Funktionen von REAR konnte auch dort nachgewiesen werden.

1.3 REAR Framework und Workflow

REAR wurde als modulares System entwickelt. Alle REAR-Kommandos initiieren einen entsprechenden Workflow, welcher in den folgenden Kapiteln beschrieben wird. Grundlage ist die entsprechende englische Dokumentation der REAR-Entwickler (siehe [1]), welche für dieses Konzept übersetzt, ergänzt und aktualisiert wurde.

Der Aufruf von „rear –h“ zeigt die möglichen Optionen und Workflows:

```
sles10sp2-vm1-kus:~ # rear -h
rear [Options] <command> [command options ...]
Relax & Recover Version 1.7.18 / 2009-03-15
Build: 8b8515fd3a0fb9c7f473a089abd07f9b
Copyright (C) 2006-2009
        Schlomo Schapiro, Immobilien Scout GmbH
        Gratien D'haese, IT3 Consultants
Relax & Recover comes with ABSOLUTELY NO WARRANTY; for details
see the GNU General Public License at http://www.gnu.org/licenses/gpl.html

Available Options:
-V                      version information
-d                      debug mode
-D                      debugscript mode
-S                      Step-by-step mode
-s                      Simulation mode (shows the scripts included)
-q                      Quiet mode
-r a.b.c-xx-yy          kernel version to use (current: 2.6.16.60-0.31-default)

List of commands:
dump                    Dump configuration and system information
help                    print out usage
mkbackup                Create rescue media and backup system.
mkbackuponly            Backup system without creating a (new) rescue media.
mkdeb                   Create DEB packages with this rear version
mkdist                  Create distribution tar archive with this rear version
mkrescue                Create rescue media only
mkrpm                   Create RPM packages with this rear version
mktar                   Create tar archive with this rear installation
mkvendorrpm             Create vendor RPM with this rear version
recover                 Recover the system
validate                Submit validation information
```

1.4 General Workflow

Durch den Aufruf von „rear <command> wird jeweils der entsprechende Workflow für das jeweilige gestartet.

Der generelle Workflow sieht folgendermaßen aus:

1. **Lesen der Konfiguration**

Sammeln der relevanten Systeminformationen, um eine funktionierende Konfiguration zu erstellen. Dabei werden die Konfigurationsdateien unter /etc/rear ausgewertet (in der Reihenfolge site.conf, local.conf). Die Standardwerte stehen unter /usr/share/rear/conf/default.conf.

2. **Erstellen der Arbeitsumgebung, Start Logging**

Unter /tmp/rear.$$ wird eine Arbeitsumgebung erstellt, außerdem wird das Logging nach /tmp/rear.log gestartet.

3. **Ausführen des Workflow-Scriptes**

Das Workflow-Script für das angegebene rear-Kommando wird gestartet:

/usr/share/rear/lib/<Kommando>-workflow.sh

4. **Aufräumen der Arbeitsumgebung**

In diesem Schritt wird die Arbeitsumgebung bereinigt und das Logging wird beendet.

1.5 dump Workflow

Durch den Aufruf von „rear dump“ wird das Workflow-Script /usr/share/rear/lib/dump-workflow.sh gestartet. Die Funktion besteht darin, die aktuelle REAR-Konfiguration und relevante Systeminformationen auszugeben, siehe folgendes Beispiel:

Bsp.: (rear dump mit OUTPUT=PXE,BACKUP=NETFS, NETFS_URL=nfs://192.168.61.129/backup):

```
sles10sp2-vm1-kus:~ # rear dump
Relax & Recover Version 1.7.18 / 2009-03-15
Dumping out configuration and system information
System definition:
                                    ARCH = Linux-x86_64
                                      OS = GNU/Linux
                                OS_VENDOR = SUSE_LINUX
                               OS_VERSION = 10
                           OS_VENDOR_ARCH = SUSE_LINUX/x86_64
                         OS_VENDOR_VERSION = SUSE_LINUX/10
                    OS_VENDOR_VERSION_ARCH = SUSE_LINUX/10/x86_64
Configuration tree:
                          Linux-x86_64.conf : OK
                             GNU/Linux.conf : OK
                            SUSE_LINUX.conf : missing/empty
                    SUSE_LINUX/x86_64.conf : missing/empty
                        SUSE_LINUX/10.conf : missing/empty
                SUSE_LINUX/10/x86_64.conf : missing/empty
                                 site.conf : OK
                                local.conf : OK
Backup with NETFS
                            NETFS_MOUNTCMD =
                             NETFS_OPTIONS =
                              NETFS_PREFIX = sles10sp2-vm1-kus
                          NETFS_SKIP_WARNING =
                             NETFS_UMOUNTCMD =
                                   NETFS_URL = nfs://192.168.61.129/backup
Backup program is 'tar':
                               BACKUP_PROG = tar
                          BACKUP_PROG_ARCHIVE = backup
                BACKUP_PROG_COMPRESS_OPTIONS = --gzip
                 BACKUP_PROG_COMPRESS_SUFFIX = .gz
                          BACKUP_PROG_EXCLUDE = /tmp/* /dev/shm/*
                          BACKUP_PROG_INCLUDE =
                          BACKUP_PROG_OPTIONS =
BACKUP_PROG_OPTIONS_CREATE_ARCHIVE =
```

```
     BACKUP_PROG_OPTIONS_RESTORE_ARCHIVE =
                       BACKUP_PROG_SUFFIX = .tar
Output to PXE
                          PXE_CONFIG_PATH = /tmp
                        PXE_CONFIG_PREFIX = rear-
                         PXE_CREATE_LINKS = MAC
                      PXE_REMOVE_OLD_LINKS =
                             PXE_TFTP_PATH = /tmp
                           PXE_TFTP_PREFIX = sles10sp2-vm1-kus.
                             RESULT_MAILTO =

/usr/share/rear/lib/validated/SUSE_LINUX/10/x86_64.txt
Your system is validated with the following details:
Submitted:      Gratien D'haese
Date:           2009-02-03
Features:       LVM, NETFS, ISO
Comment:        out of the box
Finished in 1 seconds.
```

2 Rescue System und Backup

2.1 mkrescue Workflow

Durch den Aufruf von „rear mkrescue“ wird die Rescue-Umgebung für die Wiederherstellung eines Systems erstellt.

Hinweis:

Mit „rear mkrescue“ wird kein Backup des Systems erstellt (dies geschieht durch mkbackuponly bzw. mkbackup).

Der generelle Workflow zu „rear mkrescue“ sieht folgendermaßen aus:

1. **Prep**
   - Einlesen der Konfigurationsdateien
   - Vorbereitung der Build-Umgebung inkl. Dateisystemlayout
   - Vorbereitung der Einbindung ext. Module (z.B. TSM Backup)
2. **Analyze DR**
   - Analyse des Systems
   - Erstellen der DR-Umgebung unter /$VAR_DIR/recovery
3. **Analyze Rescue**
   - Analyse des Systems
   - Vorbereitung der Erstellung des Rescue-Systems (Netzwerk, Abhängigkeiten zu notwendigen Prorammen)
4. **Build**
   - Erstellung des Rescue-Images
5. **Pack**
   - Packen von Kernel und initial Ramdisk (initrd)
6. **Backup (optional, nur bei mkbackup automatisch enthalten)**
   - Optionales Backup des Systems auf festgelegtes Ziel (z.B. NFS etc.)
7. **Output**
   - Kopieren / Installieren des Rescue-Systems (Kernel, initrd) und optional des Backups in die Zielumgebung (z.B. PXE-Boot-Umgebung)

8. **Cleanup**
    - Aufräumen der Arbeitsumgebung

Die Methoden für Backup und Output werden in den Konfigurationsdateien festgelegt.

Bsp.: (rear mkrescue im **Simulationsmodus** mit OUTPUT=PXE,BACKUP=NETFS, NETFS_URL = nfs://192.168.61.129/backup):

```
sles10sp2-vm1-kus:~ # rear -s mkrescue
Relax & Recover Version 1.7.18 / 2009-03-15
Simulation mode activated, ReaR base directory: /usr/share/rear
Source conf/Linux-x86_64.conf
Source conf/GNU/Linux.conf
Source prep/NETFS/default/00_warn_about_not_professional_backup_solution.sh
Source prep/default/01_progress_start.sh
Source prep/NETFS/default/05_check_NETFS_requirements.sh
Source prep/default/99_progress_stop.sh
Source dr/default/01_mk_config_dir_recovery.sh
Source dr/default/09_only_include_vg.sh
Source dr/GNU/Linux/10_describe_physical_devices.sh
Source dr/GNU/Linux/11_describe_mountpoint_device.sh
Source dr/GNU/Linux/12_describe_filesystems.sh
Source dr/GNU/Linux/13_describe_swap.sh
Source dr/GNU/Linux/15_copy_proc_partitions.sh
Source dr/GNU/Linux/21_describe_md.sh
Source dr/GNU/Linux/23_describe_lvm2.sh
Source dr/GNU/Linux/29_find_required_devices.sh
Source dr/Linux-x86_64/31_describe_device_properties.sh
Source dr/SUSE_LINUX/60_create_mkbootloader.sh
Source dr/GNU/Linux/80_copy_fstab_file.sh
Source dr/GNU/Linux/95_cfg2html.sh
Source dr/GNU/Linux/95_collect_hpacucli.sh
Source dr/GNU/Linux/96_collect_MC_serviceguard_infos.sh
Source rescue/default/00_remove_workflow_conf.sh
Source rescue/default/01_merge_skeletons.sh
Source rescue/default/10_hostname.sh
Source rescue/default/20_etc_issue.sh
Source rescue/GNU/Linux/30_dns.sh
Source rescue/GNU/Linux/31_network_devices.sh
Source rescue/GNU/Linux/35_routing.sh
Source rescue/GNU/Linux/39_check_usb_modules.sh
Source rescue/GNU/Linux/40_kernel_modules.sh
Source rescue/default/43_prepare_timesync.sh
Source rescue/GNU/Linux/50_clone_keyboard_mappings.sh
Source rescue/default/50_ssh.sh
Source build/GNU/Linux/20_copy_as_is.sh
Source build/GNU/Linux/39_copy_binaries_libraries.sh
Source build/GNU/Linux/40_copy_modules.sh
Source build/default/50_patch_sshd_config.sh
Source build/default/99_update_os_conf.sh
Source pack/GNU/Linux/00_create_symlinks.sh
Source pack/GNU/Linux/10_touch_empty_files.sh
Source pack/GNU/Linux/20_create_dotfiles.sh
Source pack/Linux-x86_64/30_copy_kernel.sh
Source pack/GNU/Linux/90_create_initramfs.sh
Source output/NETFS/default/10_mount_NETFS_path.sh
Source output/NETFS/default/20_make_prefix_dir.sh
Source output/PXE/default/80_copy_to_tftp.sh
Source output/PXE/default/81_create_pxelinux_cfg.sh
Source output/NETFS/default/95_copy_result_files.sh
Source output/default/95_email_result_files.sh
Source output/NETFS/default/98_umount_NETFS_dir.sh
Source cleanup/default/01_progress_start.sh
Source cleanup/default/99_progress_stop.sh
Finished in 1 seconds.
```

Bsp.: (rear mkrescue mit OUTPUT=PXE,BACKUP=NETFS, NETFS_URL = nfs://192.168.61.129/backup):

```
sles10sp2-vm1-kus:~ # rear mkrescue
Relax & Recover Version 1.7.18 / 2009-03-15
The preparation phase OK
Physical devices that will be recovered: /dev/sda
Creating root FS layout OK
Copy files and directories OK
Copy program files & libraries OK
Copy kernel modules OK
Create initramfs OK
Copied kernel+initrd (14M) to /tmp
Created pxelinux config 'rear-sles10sp2-vm1-kus' and symlinks for MAC adresses in /tmp
Copying resulting files to network location OK
The cleanup phase OK
Finished in 84 seconds.
```

Auf dem Backup-Ziel (im Bsp. NFS-Server) werden folgende Daten abgelegt:

```
sles10sp2-vm2-kus:/backup/sles10sp2-vm1-kus # ll
total 13664
-rw-r--r-- 1 root root      503 Apr  8 09:10 README
-rw-r--r-- 1 root root      387 Apr  8 09:10 VERSION
-rw-r--r-- 1 root root      445 Apr  8 09:10 rear-sles10sp2-vm1-kus
-rw-r--r-- 1 root root 12562175 Apr  8 09:10 sles10sp2-vm1-kus.initrd
-rw-r--r-- 1 root root  1391348 Apr  8 09:10 sles10sp2-vm1-kus.kernel
-rw-r--r-- 1 root root      387 Apr  8 09:10 sles10sp2-vm1-kus.message
```

2.2 mkbackuponly Workflow

Das Kommando „rear mkbackuponly“ erstellt ein Backup des Systems, ohne eine (neue) Rescue-Umgebung zu erstellen.

Der Workflow sieht folgendermaßen aus:

1. **Prep**
   - Einlesen der Konfigurationsdateien
   - Vorbereitung der Build-Umgebung inkl. Dateisystemlayout
   - Vorbereitung der Einbindung ext. Module (z.B. TSM Backup)
2. **Backup**
   - Backup des Systems auf festgelegtes Ziel (z.B. NFS, TSM etc.)
3. **Cleanup**
   - Aufräumen der Arbeitsumgebung

Bsp.: (rear mkbackuponly im Simulationsmodus mit OUTPUT=PXE,BACKUP=NETFS, NETFS_URL = nfs://192.168.61.129/backup)

```
sles10sp2-vm1-kus:~ # rear -s mkbackuponly
Relax & Recover Version 1.7.18 / 2009-03-15
Simulation mode activated, ReaR base directory: /usr/share/rear
Source conf/Linux-x86_64.conf
Source conf/GNU/Linux.conf
Source prep/NETFS/default/00_warn_about_not_professional_backup_solution.sh
Source prep/default/01_progress_start.sh
Source prep/NETFS/default/05_check_NETFS_requirements.sh
Source prep/default/99_progress_stop.sh
Source backup/NETFS/default/10_mount_NETFS_path.sh
Source backup/NETFS/default/20_make_prefix_dir.sh
Source backup/NETFS/GNU/Linux/30_stop_selinux.sh
Source backup/NETFS/default/40_create_include_exclude_files.sh
Source backup/NETFS/default/50_make_backup.sh
Source backup/NETFS/GNU/Linux/60_start_selinux.sh
Source backup/NETFS/default/98_umount_NETFS_dir.sh
Source cleanup/default/01_progress_start.sh
Source cleanup/default/99_progress_stop.sh
Finished in 1 seconds.
```

Bsp.: (rear mkbackuponly mit OUTPUT=PXE,BACKUP=NETFS, NETFS_URL = nfs://192.168.61.129/daten/backup):

```
sles10sp2-vm1-kus:~ # rear mkbackuponly
Relax & Recover Version 1.7.18 / 2009-03-15
The preparation phase OK
Creating archive 'nfs://192.168.61.129/backup/sles10sp2-vm1-kus/backup.tar.gz'
Transferred 1299 MB in 1369 seconds [972 KB/sec]
The cleanup phase OK
Finished in 1370 seconds.
```

Auf dem Backup-Ziel (im Bsp. NFS-Server) werden folgende Daten abgelegt:

```
sles10sp2-vm2-kus:/backup/sles10sp2-vm1-kus # ll
total 1353680
-rw-r--r-- 1 root root 1362616078 Apr  8 09:36 backup.tar.gz
-rw-r--r-- 1 root root    8208989 Apr  8 09:35 backup.txt
```

2.3 mkbackup Workflow

Das Kommando „rear mkbackup“ ist eine Kombination von „rear mkrescue“ und „rear mkbackuponly“, d.h. es werden sowohl die Rescue-Umgebung als auch das Backup der Daten erstellt und in der festgelegten Zielumgebung abgelegt.

Der entsprechende Workflow entspricht dem von „rear mkrescue“ inkl. den relevanten Punkten für die Ablage des Backups in den Punkten 6 und 7.

Bsp.: (rear mkbackup im **Simulationsmodus** mit OUTPUT=PXE,BACKUP=NETFS, NETFS_URL = nfs://192.168.61.129/daten/backup):

```
sles10sp2-vm1-kus:~ # rear -s mkbackup
Relax & Recover Version 1.7.18 / 2009-03-15
Simulation mode activated, ReaR base directory: /usr/share/rear
Source conf/Linux-x86_64.conf
Source conf/GNU/Linux.conf
Source prep/NETFS/default/00_warn_about_not_professional_backup_solution.sh
Source prep/default/01_progress_start.sh
Source prep/NETFS/default/05_check_NETFS_requirements.sh
Source prep/default/99_progress_stop.sh
Source dr/default/01_mk_config_dir_recovery.sh
Source dr/default/09_only_include_vg.sh
Source dr/GNU/Linux/10_describe_physical_devices.sh
Source dr/GNU/Linux/11_describe_mountpoint_device.sh
Source dr/GNU/Linux/12_describe_filesystems.sh
Source dr/GNU/Linux/13_describe_swap.sh
Source dr/GNU/Linux/15_copy_proc_partitions.sh
Source dr/GNU/Linux/21_describe_md.sh
Source dr/GNU/Linux/23_describe_lvm2.sh
Source dr/GNU/Linux/29_find_required_devices.sh
Source dr/Linux-x86_64/31_describe_device_properties.sh
Source dr/SUSE_LINUX/60_create_mkbootloader.sh
Source dr/GNU/Linux/80_copy_fstab_file.sh
Source dr/GNU/Linux/95_cfg2html.sh
Source dr/GNU/Linux/95_collect_hpacucli.sh
Source dr/GNU/Linux/96_collect_MC_serviceguard_infos.sh
Source rescue/default/00_remove_workflow_conf.sh
Source rescue/default/01_merge_skeletons.sh
Source rescue/default/10_hostname.sh
Source rescue/default/20_etc_issue.sh
Source rescue/GNU/Linux/30_dns.sh
Source rescue/GNU/Linux/31_network_devices.sh
Source rescue/GNU/Linux/35_routing.sh
Source rescue/GNU/Linux/39_check_usb_modules.sh
Source rescue/GNU/Linux/40_kernel_modules.sh
Source rescue/default/43_prepare_timesync.sh
Source rescue/GNU/Linux/50_clone_keyboard_mappings.sh
Source rescue/default/50_ssh.sh
Source build/GNU/Linux/20_copy_as_is.sh
Source build/GNU/Linux/39_copy_binaries_libraries.sh
Source build/GNU/Linux/40_copy_modules.sh
```

```
Source build/default/50_patch_sshd_config.sh
Source build/default/99_update_os_conf.sh
Source pack/GNU/Linux/00_create_symlinks.sh
Source pack/GNU/Linux/10_touch_empty_files.sh
Source pack/GNU/Linux/20_create_dotfiles.sh
Source pack/Linux-x86_64/30_copy_kernel.sh
Source pack/GNU/Linux/90_create_initramfs.sh
Source backup/NETFS/default/10_mount_NETFS_path.sh
Source backup/NETFS/default/20_make_prefix_dir.sh
Source backup/NETFS/GNU/Linux/30_stop_selinux.sh
Source backup/NETFS/default/40_create_include_exclude_files.sh
Source backup/NETFS/default/50_make_backup.sh
Source backup/NETFS/GNU/Linux/60_start_selinux.sh
Source backup/NETFS/default/98_umount_NETFS_dir.sh
Source output/NETFS/default/10_mount_NETFS_path.sh
Source output/NETFS/default/20_make_prefix_dir.sh
Source output/PXE/default/80_copy_to_tftp.sh
Source output/PXE/default/81_create_pxelinux_cfg.sh
Source output/NETFS/default/95_copy_result_files.sh
Source output/default/95_email_result_files.sh
Source output/NETFS/default/98_umount_NETFS_dir.sh
Source cleanup/default/01_progress_start.sh
Source cleanup/default/99_progress_stop.sh
Finished in 1 seconds.
```

Bsp.: (rear mkbackup mit OUTPUT=PXE,BACKUP=NETFS, NETFS_URL = nfs://192.168.61.129/daten/backup):

```
sles10sp2-vm1-kus:~ # rear mkbackup
Relax & Recover Version 1.7.18 / 2009-03-15

The preparation phase OK
Physical devices that will be recovered: /dev/sda
Creating root FS layout OK
Copy files and directories OK
Copy program files & libraries OK
Copy kernel modules OK
Create initramfs OK
Creating archive 'nfs://192.168.61.129/backup/sles10sp2-vm1-kus/backup.tar.gz'
Transferred 1299 MB in 477 seconds [2789 KB/sec]
Copied kernel+initrd (14M) to /tmp
Created pxelinux config 'rear-sles10sp2-vm1-kus' and symlinks for MAC adresses in /tmp
Copying resulting files to network location OK
The cleanup phase OK
Finished in 481 seconds.
```

Auf dem Backup-Ziel (im Bsp. NFS-Server) werden folgende Daten abgelegt:

```
sles10sp2-vm2-kus:/backup/sles10sp2-vm1-kus # ll
total 1353688
-rw-r--r-- 1 root root        503 Apr  8 09:55 README
-rw-r--r-- 1 root root        387 Apr  8 09:55 VERSION
-rw-r--r-- 1 root root 1362621781 Apr  8 09:55 backup.tar.gz
-rw-r--r-- 1 root root    8208989 Apr  8 09:55 backup.txt
-rw-r--r-- 1 root root        445 Apr  8 09:55 rear-sles10sp2-vm1-kus
-rw-r--r-- 1 root root   12562202 Apr  8 09:55 sles10sp2-vm1-kus.initrd
-rw-r--r-- 1 root root    1391348 Apr  8 09:55 sles10sp2-vm1-kus.kernel
-rw-r--r-- 1 root root        387 Apr  8 09:55 sles10sp2-vm1-kus.message
```

3 Disaster Recovery und Restore

3.1 recover Workflow

Das Kommando „rear recover“ ruft den Workflow für die Wiederherstellung des Systems auf. Teilweise werden die gleichen Module (Scripte) wie im Backup-Workflow benutzt, d.h. bei Modulen mit gleichem

Namen werden auch die gleichen Scripte benutzt (technisch sind das symbolische Links auf das gleiche Script).

Es werden die gleichen Konfigurationsdateien wie beim Backup benutzt (diese werden im Rescue System abgelegt). Damit wird die Integrität der Backup/Restore Module gewährleistet.

Der Workflow sieht folgendermaßen aus:

1. **Verify**
   - Überprüfung der Integrität der Recovery-Daten
   - Prüfung, ob die Hardware des wiederherzustellenden Systems den Anforderungen für ein erfolgreiches Restore entspricht (z.B. identische Festplatten und Controller etc.)
     - Wenn ja, wird der Workflow fortgesetzt
     - Wenn nicht, dann wird der Workflow an dieser Stelle abgebrochen (da sonst das wiederherzustellende System potentiell nicht starten würde)
2. **Recreate**
   - Wiederherstellung des Festplattenlayouts auf dem System (Raidkonfiguration, Partitionierung, LVM, Filesysteme)
   - Die wiederhergestellten Filesysteme werden unter /mnt/local gemounted
3. **Restore**
   - Die Daten (Dateien und Verzeichnisse) werden aus dem Backupsystem (z.B. NFS-Share, TSM-Server) wiederhergestellt
4. **Finalize**
   - Installation des Bootloaders
   - Fertigstellung des Systems
   - Recovery-Logfile wird unter /tmp im wiederhergestellten System abgelegt

Bsp.: (rear recover im **Simulationsmodus** mit OUTPUT=PXE,BACKUP=NETFS, NETFS_URL = nfs://192.168.61.129/backup):

```
sles10sp2-vm1-kus:~ # rear -s recover
Relax & Recover Version 1.7.18 / 2009-03-15
Simulation mode activated, ReaR base directory: /usr/share/rear
Source conf/Linux-x86_64.conf
Source conf/GNU/Linux.conf
Source setup/GNU/Linux/80_setup_hp_raid.sh
Source verify/NETFS/default/05_check_NETFS_requirements.sh
Source verify/GNU/Linux/05_sane_recovery_check.sh
Source verify/NETFS/default/08_start_required_daemons.sh
Source verify/GNU/Linux/10_describe_physical_devices.sh
Source verify/NETFS/default/10_mount_NETFS_path.sh

Source verify/Linux-x86_64/11_describe_device_properties.sh
Source verify/GNU/Linux/12_compare_physical_devices.sh
Source verify/NETFS/default/55_check_backup_archive.sh
Source recreate/GNU/Linux/09_disable_lvm2_md.sh
Source recreate/Linux-x86_64/10_initialize_physical_devices.sh
Source recreate/GNU/Linux/21_create_md_devices.sh
Source recreate/GNU/Linux/22_create_lvm2_devices.sh
Source recreate/GNU/Linux/31_create_filesystems.sh
Source recreate/GNU/Linux/70_mount_filesystems.sh
Source recreate/default/98_show_disk_free.sh
Source restore/NETFS/default/40_restore_backup.sh
Source restore/NETFS/default/50_selinux_autorelabel.sh
Source restore/default/90_create_missing_directories.sh
Source restore/SUSE_LINUX/91_create_missing_directories.sh
Source restore/NETFS/default/98_umount_NETFS_dir.sh
```

```
Source finalize/default/01_prepare_checks.sh
Source finalize/default/10_populate_dev.sh
Source finalize/SUSE_LINUX/x86_64/20_run_mkbootloader.sh
Source finalize/GNU/Linux/70_create_swapfiles.sh
Source finalize/default/88_check_for_mount_by_id.sh
Source finalize/default/89_finish_checks.sh
Source finalize/default/90_remount_sync.sh
Source finalize/default/98_good_bye.sh
Source finalize/default/99_copy_logfile.sh
Finished in 1 seconds.
```

Bsp.: (rear recover mit OUTPUT=PXE,BACKUP=NETFS, NETFS_URL = nfs://192.168.61.129/backup):

Voraussetzungen:

- Funktionierender und von der wiederherzustellenden Maschine per Netzwerk erreichbarer PXE/TFTP und DHCP-Server
- Die Rescue-Umgebung muss vom Backupserver (z.B. NFS, TSM) auf PXE/TFTP-Server übertragen werden.

Im Beispiel müssen die Ramdisk (sles10sp2-vm1-kus.initrd) und der Kernel (sles10sp2-vm1-kus.kernel) auf dem PXE-Server unter /tftpboot abgelegt werden. Außerdem müssen die messages und default Dateien des PXE-Servers angepasst werden (Booteintrag für sles10sp2-vm1-kus erstellen).

3.2 Walkthrough

Ein Restore eines Systems mit REAR ist in den folgenden Abbildungen dargestellt:

1. Booten der wiederherzustellenden Maschine per PXE

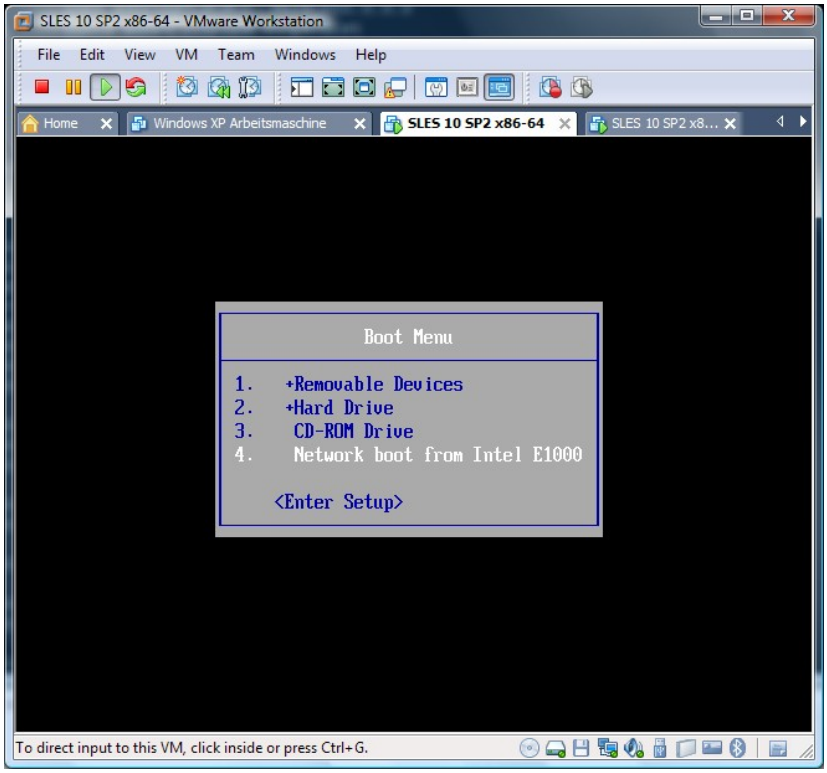

Abbildung 1 : REAR Recovery Step 1

2. Auswahl des wiederherzustellenden Systems

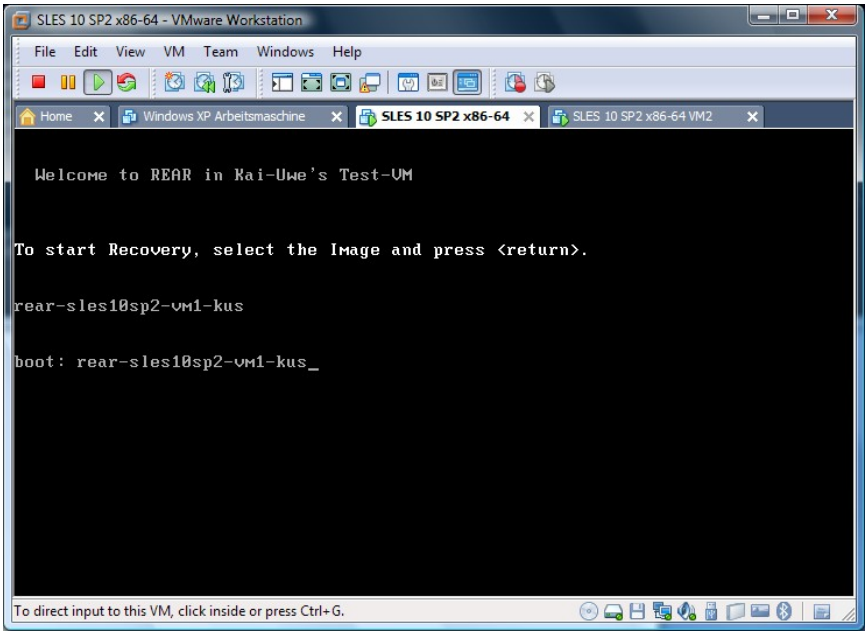

Abbildung 2 : REAR Recovery Step 2

3. Anmeldung als root ohne Passwort

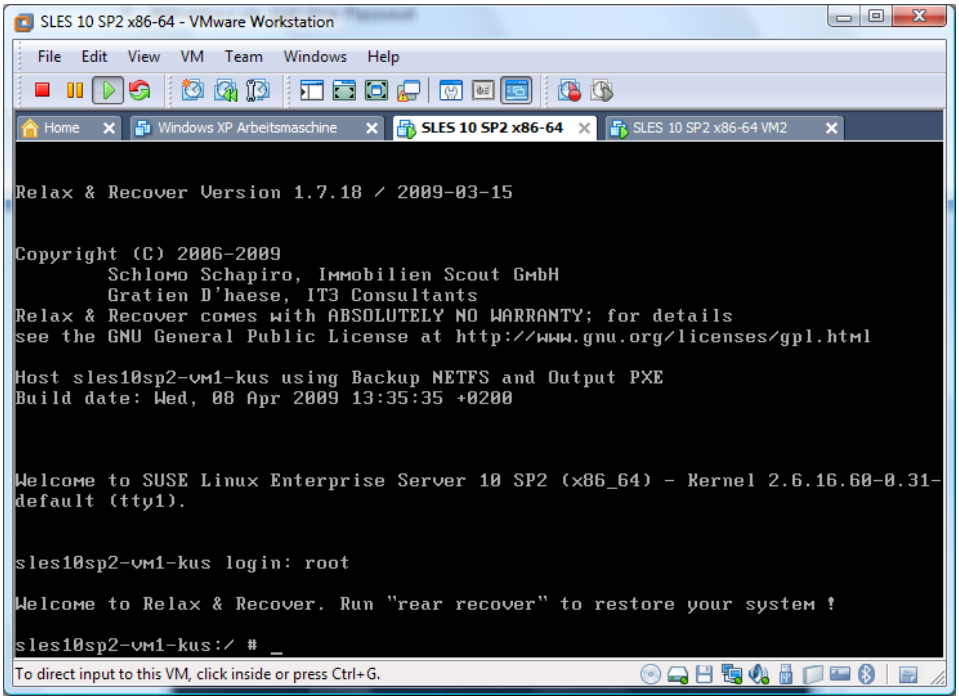

Abbildung 3 : REAR Recovery Step 3

4. Eingabe von „rear recover“ und <Return>, Recovery startet

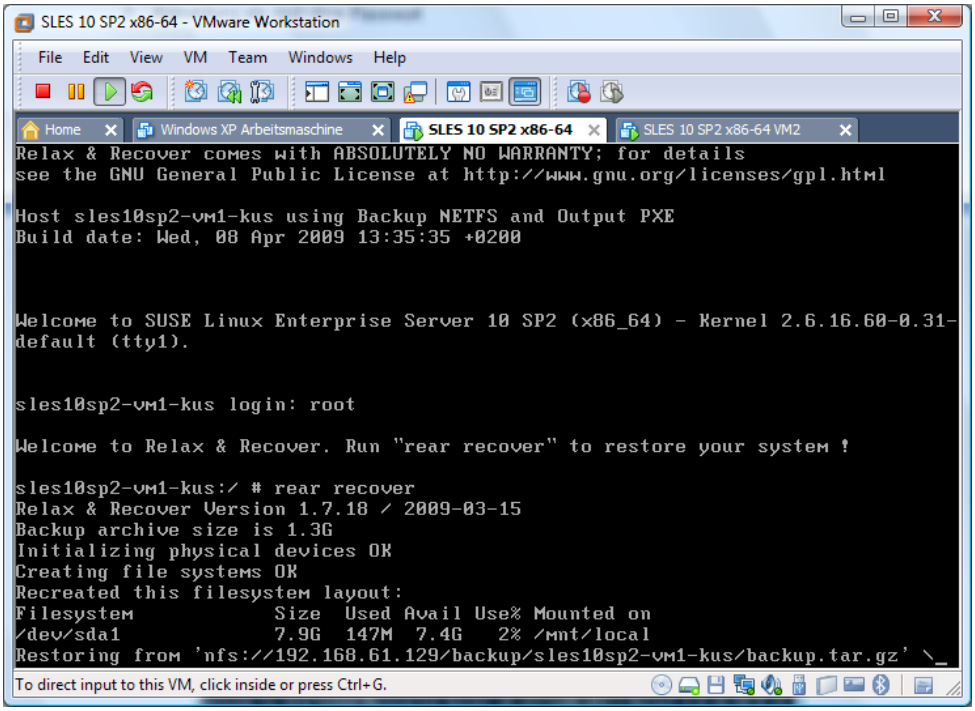

Abbildung 4 : REAR Recovery Step 4

5. Recovery beendet, Reboot des Servers durch Eingabe von „reboot“

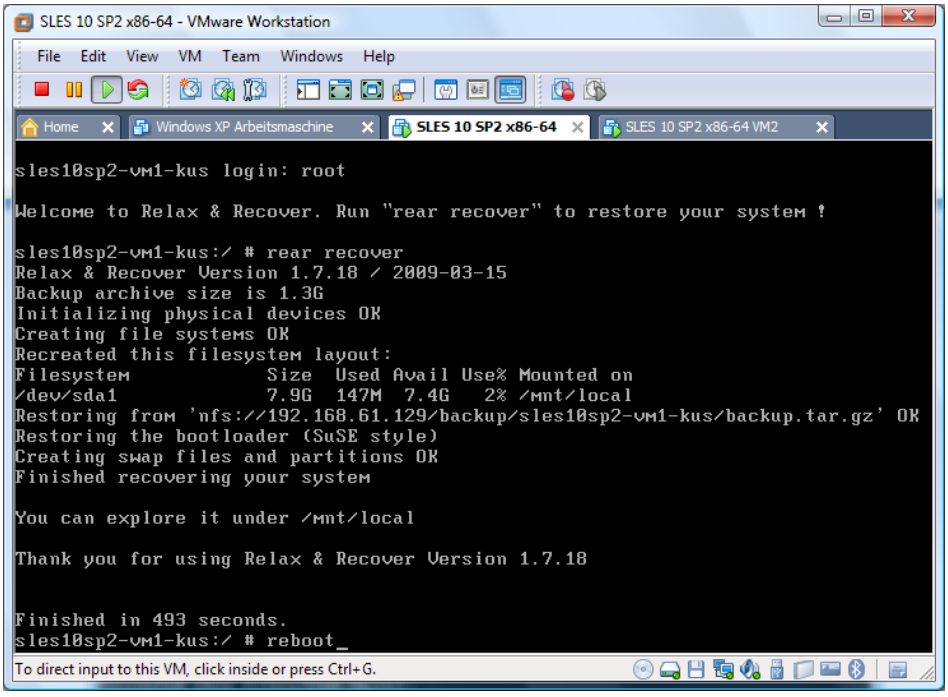

Abbildung 5 : REAR Recovery Step 5

6. Server startet nach Recovery wieder ohne Probleme

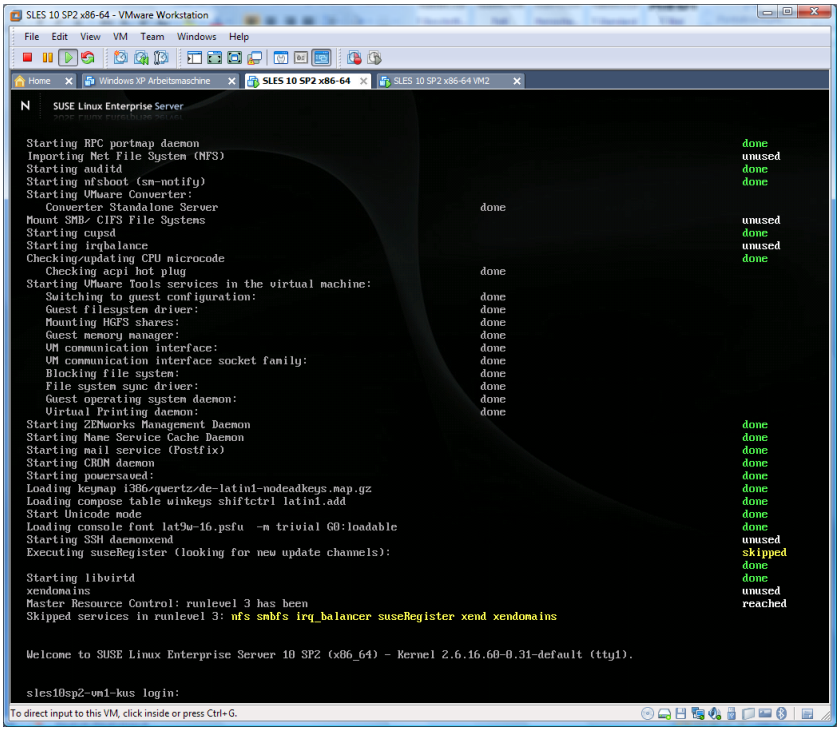

Abbildung 6 : REAR Recovery Step 6

4 Referenzhandbuch

4.1 ReaR Installations- und Konfigurationsdateien

REAR legt seine Programm- und Konfigurationsdateien weitgehend LSB-konform ab, dabei werden folgende Verzeichnisse und Dateien verwendet:

Verzeichnis	Inhalt
/etc/rear/*	Konfigurationsdateien
/usr/sbin/rear	Hauptprogram (Script)
/usr/share/rear/*	Interne Scripte (siehe Workflows)
/var/lib/rear/*	Recovery Daten
/tmp/rear.$$	Arbeitsumgebung (Build Area)

Tabelle 1 : REAR Verzeichnisse und Dateien

Der Inhalt bzw. die Funktion der wesentlichen Dateien bzw. Verzeichnisse ist in den folgenden Tabellen beschrieben:

Inhalt von /etc/rear

Datei/Verzeichnis	Inhalt
site.conf	Konfigurationsdatei (Site)
local.conf	Konfigurationsdatei (lokale Maschine)
templates/	Verzeichnis für Vorlagen (z.B. für PXE-Boot)

Tabelle 2 : Inhalt von /etc/rear

Hinweis: Die Standardkonfiguration (enthält alle Variablen mit den heweiligen default-Werten) wird nicht mehr unter /etc/rear/default.conf sondern unter /usr/share/rear/conf/default.conf abgelegt.

Einstellungen in /etc/rear/site.conf oder /etc/rear/local.conf überschreiben die jeweiligen Variablen der default.conf.

Inhalt von /usr/share/rear

Datei/Verzeichnis	Inhalt
skel/default/	Standard Filesystemgerüst für Rescuesystem
skel/$(uname -i)/	Architektur-Spezifisches Filesystemgerüst für Rescuesystem
skel/$OS_$OS_VER/	Betriebssystem-Spezifisches Filesystemgerüst für Rescuesystem
skel/$BACKUP/	Backupsoftware-Spezifisches Filesystemgerüst für Rescuesystem
skel/$OUTPUT/	Outputmethoden-Spezifisches Filesystemgerüst für Rescuesystem
lib/*.sh	Funktionsdefinitionen (pro Funktion ein File)
prep/default/*.sh prep//$(uname -i)/*.sh prep/$OS_$OS_VER/*.sh prep/$BACKUP/*.sh prep/$OUTPUT/*.sh	Scripte für Prep-Phase (siehe Workflow) Werden entsprechend Workflow in ihrer alphabetischen Reihenfolge ausgeführt Namenskonvention: ##_name.sh (## entspricht Zahl zwischen 00 und 99)
dr/…	Scripte für Analyze-DR-Phase (siehe Workflow), Ausführungsreihenfolge und Namenskonvention wie bei prep
rescue/…	Scripte für Analyze-Rescue-Phase (siehe Workflow), Ausführungsreihenfolge und Namenskonvention wie bei prep
build/…	Scripte für Build-Phase (siehe Workflow), Ausführungsreihenfolge und Namenskonvention wie bei prep
pack/…	Scripte für Build-Phase (siehe Workflow), Ausführungsreihenfolge und Namenskonvention wie bei prep
backup/$BACKUP/*.sh	Scripte für Backup-Phase (siehe Workflow), Ausführungsreihenfolge und Namenskonvention wie bei prep
output/$OUTPUT/*.sh	Scripte für Output-Phase (siehe Workflow), Ausführungsreihenfolge und Namenskonvention wie bei prep
cleanup/…	Scripte für Cleanup-Phase (siehe Workflow), Ausführungsreihenfolge und Namenskonvention wie bei prep
verify/…	Scripte für Verify-Phase (siehe Workflow), Ausführungsreihenfolge und Namenskonvention wie bei prep
recreate/...	Scripte für Recreate-Phase (siehe Workflow), Ausführungsreihenfolge und Namenskonvention wie bei prep
restore/$BACKUP/...	Scripte für Restore-Phase (siehe Workflow), Ausführungsreihenfolge und Namenskonvention wie bei prep
finalize/...	Scripte für Finalize-Phase (siehe Workflow), Ausführungsreihenfolge und Namenskonvention wie bei prep

Tabelle 3 : Inhalt von /usr/share/rear

Inhalt von /var/lib/rear

Datei/Verzeichnis	Inhalt
/var/lib/rear	Datenverzeichnis für Recoveryinformationen
/var/lib/rear/recovery	Recoverydaten des Systems

Tabelle 4 : Inhalt von /var/lib/rear

4.2 Konfigurationsvariablen

Backup und Output Methoden:

Die zu verwendenden Methoden für Backup und Output müssen in den Konfigurationsdateien /etc/rear/site.conf oder /etc/rear/local.conf definiert werden.

Die gültigen Methoden sind in der nachfolgenden Tabelle aufgeführt.

(Hinweis: Es werden nur die mit REAR 1.7 Stand April 2009 bereits implementieren Methoden beschrieben, weitere Methoden sind in Entwicklung)

Name	Typ	Beschreibung	Hinweis
REQUESTRESTORE	BACKUP	Depends on an external backup program (rear does nothing)	
NETFS	BACKUP	Copy files to NFS/CIFS share,	NETFS_URL muss angegeben werden
EXTERNAL	BACKUP	Copy files to an external system	
TSM	BACKUP	Use Tivoli Storage Manager	
NBU	BACKUP	Use Symantec Netbackup	
DP	BACKUP	Use HP Data Protector	
PXE	OUTPUT	Create PXE bootable files on TFTP server	
USB	OUTPUT	Create bootable USB device	
ISO	OUTPUT	Create an ISO image (default)	

Tabelle 5 : REAR Backup- und Outputmethoden

Inter-Modul-Kommunikation

Die folgenden Variablen werden innerhalb des Workflows von den einzelnen Modulen zur übergreifenden Kommunikation verwendet.

Name	Typ	Beschreibung	Beispiel
CONFIG_DIR	STRING (RO)	Configuration directory	/etc/rear
SHARE_DIR	STRING (RO)	Shared data directory	/usr/share/rear
BUILD_DIR	STRING (RO)	Build directory	/tmp/rear.$$
ROOTFS_DIR	STRING (RO)	Root FS directory for rescue system	/tmp/rear.$$/initrd
REQUIRED_PROGS	LIST	program files to copy	bash ip route grep ls
MODULES MODULES_LOAD	LIST	modules to copy modules to load in the recovery system	e1000 cciss mptspi
COPY_AS_IS	LIST	files (with path) to copy as-is (to Rescue-Media)	/etc/localtime
COPY_AS_IS_EXCLUDED	LIST	Specific files to be excluded from image (Rescue Media)	/tmp
VAR_DIR	STRING (RO)	introduced VAR_DIR (/var/lib/rear) for moving /etc/rear/recovery to /var/lib/rear/recovery (system recovery data) going over all *.sh scripts to modify $CONFIG_DIR/recovery into $VAR_DIR/recovery	/var/lib/rear
TIMESYNC	Scalar	Enable time synchronisation in the rescue system. =NTP NTP client (ntpd) =RDATE RDATE client You must also set the RDATE server in the TIMESYNC_SOURCE variable.	
PROGS	Array	Include program binaries on rescue media. Use this array to add custom programs to the	PROGS=("${PROGS[@]}" dos2unix /opt/myapp/bin/myapp

Name	Typ	Beschreibung	Beispiel
		rescue system. The library dependancies (as shown by ldd) are automatically included. You must include the previous content or risk creating a non-functioning rescue system: PROGS=("${PROGS[@]}" dos2unix /opt/myapp/bin/myapp) The programs are searched for in the $PATH and copied to /bin on the rescue media.	)
LIBS	Array	Include libraries on rescue media. Use this array to add custom libraries to the rescue system. The library dependancies (as shown by ldd) are automatically included. You mustinclude the previous content or risk creating a non-functioning rescue system: The libraries are copied to /lib on the rescue media. Note: 64bit and 32bit libraries are both speficied here and later on automatically sorted into their corresponding /lib and /lib64 directories.	LIBS=("${LIBS[@]}" /usr/lib/libmyownapp.so)
PING	Boolean	Enable/Disable ping tests for network servers as part of the backup and restore process. You will have to disable PING if your network disallows ping or you backup servers in a DMZ etc.	

Tabelle 6 : REAR Variablen für Intermodulkommunikation

Variablen mit der Kennzeichnung RO (ReadOnly) werden vom REAR-Framework verwaltet und durch die einzelnen Module im Workflow nicht geändert werden.

Weitere (interne) Variablen sind in der Datei /usr/share/rear/conf/default.conf dokumentiert.