The Rembo Wizard 2.0

Automatic Operations

How To

by

Petri Mäkijärvi

December 9, 2002

Abstract:
The Rembo Wizard 2.0 is a free plug-in module for the Rembo Toolkit 2.0, a PXE-enabled, Pre-OS platform for the system hard disk management for Windows and Linux PC-computers.
This document explains how system administrator can use The Rembo Wizard's various automatic operations. For example, the document explains how it is possible to configure a system so that when a failed hard disk is changed, the operating system gets installed automatically when the system is switched back on.

Introduction

The Rembo Wizard is a free plug-in module for the Rembo Toolkit.

From the Main Menu, follow *Start Admin operations -> The Rembo Wizard Configuration.* The configuration dialog has a section *Automated operations*, which will allow you to select from the following automated operations that The Rembo Wizard can do for you:

- **Automatic System Start-Up**
  Typically always *on*, this automated operation allows the operating system to be started automatically, with an possibility to activate The Rembo Wizard if needed.
- **Unattended Installation**
  Installing a server cluster? Want to re-install client's system without moving from you office? Unattended installation is a tool for you.
- **Automatic Installation After A Hard Disk Failure**
  Your industrial computer's hard disk breaks down in the middle of the night. Luckily there is a spare disk and a technician is available to change it. But how to restore the system on it? The Rembo Wizard can help.
- **Fixing A Public Access System To A Known State**
  Organizing a conference? Everybody wants to read their e-mail with your public access workstations but each evening the machines are in a sorry state. Get them fixed automatically for you by The Rembo Wizard.

AutoBoot – automatic system start-up

You can set a *Delay* for automatic boot of the operating system. The delay is by default 10 seconds. It is worthwhile to note that this delay is used for all automatic operations. This gives you a "panic switch" in the case of an unattended installation, for example.

Use The Rembo Wizard as a simple boot loader which works both for Windows and for Linux by setting the *AutoBoot.*

For Windows the boot operation is always from the hard disk, using the Master Boot Record (MBR).

For Linux, you can select between the hard disk booting and the network kernel boot. The hard disk boot is both LILO and GRUB aware. The Master Boot Record is not used for the Linux automated boot. Instead, the */etc/lilo.conf* is analyzed first and the default boot parameters extracted. The Rembo Wizard searches the defined kernel (and the initial ram disk, if defined) and launches the Linux kernel with the default root file system parameter. If there is no */etc/lilo.conf*, then */boot/grub/menu.lst* is analyzed in a similar manner.

The Linux network kernel boot supported by The Rembo Wizard is a great way to make sure that a group of machines, such as server clusters and control system computers all work with the same, unique kernel and ram disk. Each time The Rembo Wizard takes a base image, it stores the LILO or GRUB defined default kernel and initial ram disk on the server.

Unattended - once-only installation of the base image

If you and your organization have several machines of the same type to install, The Rembo Wizard *Unattended* installation tool is your dream come true. Although nothing comes free in this world, with careful preparation of your DHCP-server and the Rembo Server, you can arrange an impressive installation procedure:

1. Note down all the MAC hardware addresses of the batch of computers
2. Program the MAC addresses in your DHCP server
3. Program the MAC address in you Rembo Toolkit server, arranging the computers in one or several groups.
4. Install the reference machine with an operating system and with your selected programs.
5. Take a base image on the group level from the reference machine.
6. For each new machine other than the reference machine, create a directory on the Rembo Toolkit server using the MAC address of the machine as the name of the directory
7. Copy the *autoload* file of the reference machine into this directory
8. Edit the *autoload* file for each new machine to install and set the *Unattended* flag ***true***.
9. Connect each new machine to the network and turn them on.
10. The Rembo Wizard will install the group level reference system on the hard disk.
11. Once the installation is finished, The Rembo Wizard will automatically turn the *Unattended* flag ***false*** in the *autoload* file.

Please note that the *Unattended* installation works always with the base image.

Additional data partition and automated operations

If you have an additional data partition in your installation, The Rembo Wizard creates the partition but it cannot format it. There are two reasons for this:

- ☑ By its design definition, The Rembo Wizard **never** works with other partitions but the system partition.
- ☑ If there is a data partition of NTFS 3 or greater type to create, you would need a Rembo image to format the data partition with Rembo's *DeviceCleanEx*().

Supposing that you would still like to get the data partition formatted automatically by The Rembo Wizard. It is possible to arrange that by writing your own routine in the *autoload* file, either at the group level or at the target system's own *autoload* file. Following is the example code for the case where you have created a FAT32 partition for the data on the second primary partition of the first system disk. The Rembo Wizard takes care of the system disk partitioning at the primary partition level and it takes of the installation of the system on the system partition (in this example, partition 1). Then your code is called.

```
// This is user's code
bool userCodeLoaded;
if (userCodeLoaded)
    goto userCodeEnd;
// Hook at the end of Unattended installation
void UnattendedHook ( void ) {
    Printf ("UnattendedHook() executing<br>");
    SysLog( NetInfo.IPAddress+": UnattendedHook() HDClean(0,2)");
    HDClean(0,2);
} // UnattendedHook()
userCodeLoaded = true;
userCodeEnd:    ;
// end of user's code
```

Immediately after the above hook's execution, the system is booted, providing that you have the *AutoBoot* option set. If the data partition is of NTFS-type, then the hook's code would be quite complicated, but not impossible to write. You would need to take a dummy image of the reference system's data partition. Then you would use the dummy image to obtain the structure required by the *DeviceCleanEx()* Rembo function which must be used instead of *HDClean()* for the NTFS file systems.

AutoRepair - base image reinstallation if a new hard disk detected

All systems that have the following type persistent variable stored in their host-level *autoload*-file can take the advantage of the *AutoRepair* feature of The Rembo Wizard:

```
str PartitionTable = "NTFS:2048256 EMPTY:0 EMPTY:0 EMPTY:0";
```

The Rembo Wizard calls the above parameter as disk's *Partition Table Signature*. It is created automatically each time you take a base image of the system. Also The Rembo Wizard checks the partition table signature each time when you enter into the configuration dialog utility on the client machine.

Once the system is installed and running in normal conditions, it is not likely that the partition table of the system changes. This is especially true in computers that would be ideal candidates for the *AutoRepair*-operation:

- Clustered computer's nodes
- Control system computers

If we take the example of a node in a large cluster of computers, there is likely no screen, keyboard or mouse attached to the node that has suddenly broken down. But typically there is a Hot-Plug SCSI-disk which is now broken, just to make an example. *AutoRepair*-operation will help us now to quickly get the system up and running:

1. We take a spare disk with no partitioning and replace the broken disk
2. Power on the system
3. The Rembo Wizard gets in execution
4. It reads the disk's partition table and compares it to the *Partition Table Signature* stored on the server.
5. A partition table mismatch is detected.
6. The Rembo Wizard warns on the display (which is not necessarily connected) during the *AutoBoot* delay that the system is going to be re-installed.
7. An installation similar to the *Unattended*-installation starts.
8. Once the system is reinstalled, it is booted (*AutoBoot* must be on).

An other example would be with the control system computers that could be repaired from a hard disk failure by the maintenance personnel without any particular tools or knowledge of the actual system that is running on the dedicated system that is backed up with The Rembo Wizard.

AutoRepair-operation provides similar user level code hook **option** as with the *Unattended*-installation. Following is an example code in the host's *autoload*-file:

```
// This is user's code
bool userCodeLoaded;
if (userCodeLoaded)
    goto userCodeEnd;
// Hook at the end of AutoRepair installation
void AutoRepairHook ( void ) {
    Printf ("AutoRepairHook() executing<br>");
    SysLog( NetInfo.IPAddress+": AutoRepairHook()HDClean(0,2)");
    HDClean(0,2);
} // AutoRepairHook ()
userCodeLoaded = true;
userCodeEnd:    ;
// end of user's code
```

AutoFix - compare and fix the system partition

Typical usage of this feature is with the public access computers and such. For example, you are organizing a conference and you must provide public access computers for people so that they can read their e-mails from their home institute. With the *AutoFix*-feature of The Rembo Wizard you can make sure that each morning the participants would find a fresh, working installation.

Unlike the *Unattended*-installation, which is once-only automated installation operation the *AutoFix*-feature is executed at every reboot. The system partition is not formatted but its content is compared against the base image:

- Files that exists on the disk but not on the base image are deleted
- Files that exists in the base image but not on the disk are returned back to the disk
- Files on the hard disk that are declared different than those in the base image after a MD5 checksum verification are replaced on the disk with the original version of each file.

Please note that with this procedure the system partition never gets formatted. It would be a good idea to reinstall the system from the base image before the next big conference (in our example) in order to avoid system disk's fragmentation.

AutoFix -operation provides similar user level code hook **option** as with the *Unattended*-installation. Following is an example code in the host's *autoload*-file:

```
// This is user's code
bool userCodeLoaded;
if (userCodeLoaded)
    goto userCodeEnd;
// Hook at the end of AutoFix operation
void AutoFixHook ( void ) {
    Printf ("AutoFixHook() executing<br>");
    SysLog( NetInfo.IPAddress+": AutoFixHook()executing");
} // AutoFixHook ()
userCodeLoaded = true;
userCodeEnd:    ;
// end of user's code
```

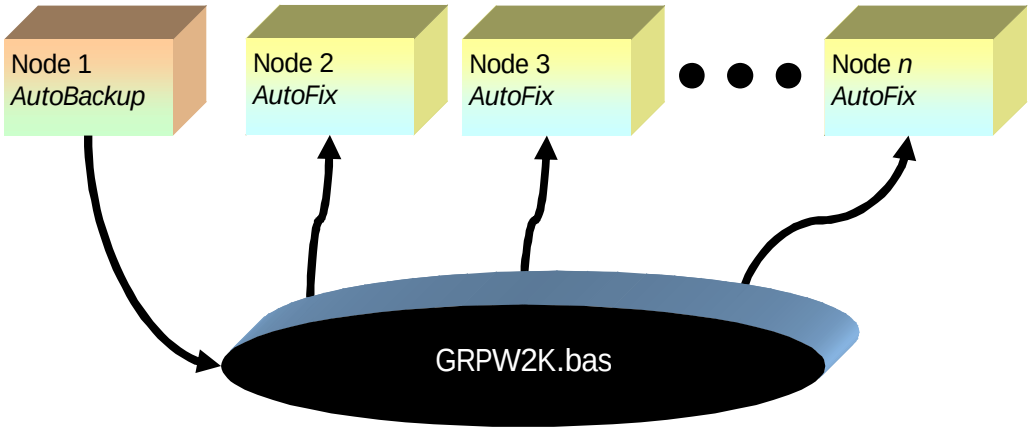
AutoBackup - make a new base image at each reboot

Apart the very obvious usage of forcing a backup on user's computer, this feature has been created for a more complex usage. Nothing prevents you *AutoBackup* in single-shot cases, but following is an explanation of cluster-computing level synchronizing, using both *AutoBackup* and *AutoFix* features of The Rembo Wizard.

Let's suppose that you have a cluster of identical computers. They can be general purpose login servers, a Beowulf cluster, an OpenMosix cluster, a group of control system computers, or such. You may have originally set up all the computers to have an equal configuration using the disk cloning techniques provided by The Rembo Wizard and you may even use network kernel booting to assure the identical configuration of all the computers in the cluster.
Typically the configuration of a cluster evolves; new user logins are defined, common shell resources are modified, new NFS-mount points created, new drivers installed, and much more. You may have already a script based mechanism to keep the systems on identical state. But you should ask yourself following questions:

1. Is your scripting based update mechanism absolutely reliable and vaccinated against human errors or locked resources?
2. What if the system disk of a node gets broken? "*You'd repair it with Rembo*". Sure, but when you did take the last Rembo backup? Probably when you installed the cluster… How do you plan to reach the state of the other machines on the system that you are repairing? *(In practice you would have to stop a working machine, take a Rembo backup of it and then repair the broken machine with the new image. Quite time consuming.)*

The solution to the above problems is to use The Rembo Wizard's *AutoBackup* feature together with the *AutoFix* (case 1) or with the *AutoRepair* (case 2) features. You would select one machine on the cluster as the reference machine, on which you would do all the necessary maintenance modifications. On this machine you would use *AutoBackup*-feature. All other machines would be set to use *AutoFix* or *AutoRepair*-features.



In the above picture, *Node 1* is the reference machine of the cluster of computers. Periodically we take a new base image of its system partition using the *AutoBackup*-feature. All other nodes, from *Node 2* to *Node n* are periodically fixed to the same state as the reference node using the *AutoFix*-feature.

Of course, you cannot fix the other systems from the base image while The Rembo Wizard is creating the base image. The most obvious solution to this would be to reboot the reference machine *Node 1* 24 hours before the other nodes of the system. This would give you the advantage of being alerted by The Rembo Wizard if something goes wrong in the base image creation before other nodes are applying the base image.

This can be a problem on many computer clusters where maintenance rebooting of systems is scheduled for once per **entire** cluster. The problem can be resolved with a configuration which is also the most secure one:

1. *Node 1* Machine Type is set to something else than with the rest of the nodes in the cluster. For example, *BUP* instead of *GRP* in the above picture's example.
2. Between two successive maintenance reboots of the cluster, the system administrator will have time to receive and study reports sent by The Rembo Wizard. If everything goes well, it is enough to rename *BUPW2K*.bas to *GRPW2K.bas*, again in the example of the above picture.

Can I set both *AutoFix* **and** *AutoRepair* on? Yes, it is possible but only by editing *autoload*-file by hand. If you pass through the configuration dialog, it allows you to select but one automatic operation at a time. Selecting both *AutoFix* and *AutoRepair* options will give the priority to *AutoRepair*-option over the *AutoFix*-option. If the disk is changed and it will be *AutoRepair*'ed, it will be also *AutoFix*'ed (uselessly, of course).

AutoBackup -operation provides similar user level code hook **option** as with the *Unattended*-installation. Following is an example code in the host's *autoload*-file:

```
// This is user's code
bool userCodeLoaded;
if (userCodeLoaded)
    goto userCodeEnd;
// Hook at the end of AutoBackup operation
void AutoBackupHook ( void ) {
    Printf ("AutoBackupHook() executing<br>");
    SysLog( NetInfo.IPAddress+": AutoBackupHook()executing");
} // AutoBackupHook ()
userCodeLoaded = true;
userCodeEnd:    ;
// end of user's code
```

Server side shared files expansion pitfall with AutoBackup

As with any Rembo Toolkit 2.0 or higher server, shared files repositories continue to expand on the server. With systems where The Rembo Wizard's System Snapshot is taken manually, this can be a problem in long term and the server's file system must be surveyed. With *AutoBackup*-feature the shared files repository expansion becomes something almost programmed and therefore you have to analyze the repository's expansion on regular basis and take action accordingly. Following is the how the Rembo Toolkit server is used to check and fix its own file system.

Stop the Rembo Toolkit server.

```
/etc/init.d/rembo stop
```

Following commands will keep the server disabled and you client computers should not be rebooted during the shared file system reparations. Press *Ctrl*+C to stop the application when the Rembo Toolkit server says "listening on socket…".
Run the *fsck* equivalent of the Rembo Toolkit, first in reporting only mode:

```
cd /usr/local/rembo
./rembo –d –v 3 –chkshared
```

If there are no **big** warnings, you can run the repair mode on the shared file system

```
./rembo.-d –v 3 -fixshared
```

You can get an idea what can be gained if the shared file system is backed by issuing the command

```
./rembo.-d –v 3 -statshared
```

Finally, if you consider that the gains are worth of the small risk involved (something **can** go wrong and therefore you **must** have a recent backup of the shared file system), you can pack the shared file system

```
./rembo.-d –v 3 -packshared
```

Start the Rembo Toolkit server.

```
/etc/init.d/rembo start
```

✍