Developer Technical Note

February 2000
DeveloperTechNote1, Revision 3-3

Creating Software Package Files for Cobalt MIPS-based Server Appliances

1 Overview

New software, updates, and third-party software for Cobalt server appliances is delivered in the form of downloadable software *.pkg files, referred to as *package files*. Package files are available from Cobalt Network's web and FTP sites, and from third-party vendors. Cobalt uses package files to distribute and install software because it fits the Cobalt philosophy of ease of use: users can install software using only a web browser, which is easier and more intuitive than telnet or other Unix command line interfaces.

This note describes the typical contents of a package file, and how to create one. In addition, it describes some optional mechanisms that can be added while creating package files. This technical note can be found in `ftp://ftp/cobaltnet.com/pub/developers/DeveloperTechNote1-3-3.pdf`

Table of Contents

1.1 Audience

This Technical Note is intended for application developers creating software to be used with any of Cobalt Network's MIPS-based server appliances. For information on creating package files for Cobalt Network's x86-based RaQ 3i server appliances, see DeveloperTechNote-2, located at `ftp://ftp/cobalt.net.com/developer/`

1.2 Applicable Products

All Cobalt server appliances make use of package files. For a list of products, see "Cobalt products and their software model numbers" on page 5.

2 What is a Package File?

A package file is a single downloadable compressed collection of files used for software installation or updates for Cobalt Network's server appliances.

A typical package file is a compressed tar archive that contains the following elements:

- `packing_list` file—a text file listing the package title, version, reboot flag, uninstall script name, a list of scripts, a list file reference, and a list of `*.rpm` files included in the package file.

- `upgrade_me` file—an install shell script.
  Note: the install script name is always named `upgrade_me`, regardless of whether it is for initial software installation or an upgrade or update to existing software.
- `uninstall_me` file—an uninstall shell script
- `*.rpm` files—these are software modules in RedHat Package Manager (RPM) format that you install on the Cobalt server appliance; see "Overview of RedHat Package Manager (RPM) Files" on page 2 for details on RPM.
- `register_me` file—a file that causes you to be notified when your package file is installed, uninstalled, and in the future, activated or deactivated.
- `update_list` file—lists the name and description of the application on the Qube 2.

2.1 Sample and Template Package Files

To get started creating package files, download the `sample.pkg` and `template.pkg` files:

```
ftp://ftp.cobaltnet.com/pub/developer/sample-0.2.mips.pkg
ftp://ftp.cobaltnet.com/pub/developer/template-0.2.mips.pkg
```

Cobalt created `sample.pkg` to demonstrate the package mechanism; it contains live code that simply causes the server LED to blink for 60 seconds one time. You can use `template-0.2.mips.pkg` as a starting point for creating new package files by filling in the blanks and inserting code where appropriate.

To decompress either file, type the Linux command:

```
tar -xzvf sample-0.2.mips.pkg
```

or

```
tar -xzvf template-0.2.mips.pkg
```

Note
You must create a directory for each package file that you decompress. If you decompress two package files in the same directory, the contents of the second file will overwrite the contents of the first file.

Use a text editor to view the contents of the `packing_list`, `upgrade_me`, `uninstall_me`, `register_me`, `update_list` and `*.rpm` files.

2.2 Overview of RedHat Package Manager (RPM) Files

Cobalt Networks uses Redhat Package Manager (RPM) files because applications are easy to manage if they are installed using RPM utilities. For details on creating *.rpm files (also known as "redhat package module" files), see *Maximum RPM*, by Marc Ewing and Erik Troan. *Maximum RPM* is the definitive technical reference for the RPM packaging system; it provides information on RPM's history, usage, and internals from both the user and packager perspectives. Also, see `http://www.redhat.com/` for the most up-to-date information about RPM technology.

To view a list of all RPM files on the server, type:

```
rpm -qa
```

To find the name of the RPM that installed any *filename*, type:

```
rpm -qf /directory/filename
```

2.3 Example of the Install Process using sample.pkg

Following is a brief explanation of what happens when you install a package file. For information on creating your package file, see "Creating a Package File" on page 5.

The `packing_list` file for `sample-0.2.mips.pkg` looks like this:

```
Package: Cobalt Blink Package for MIPS platforms
Version: 1.0
REBOOT: no
UNINSTALL: uninstall_me
SCRIPT: upgrade_me
SCRIPT: register_me
SCRIPT: update_list
List File: sample
RPM: blink-0.1-1.mips.rpm
```

The installer program parses the `SCRIPT:` tag in the `packing_list` file and executes the associated shell script, `upgrade_me`. If multiple scripts are specified, they will be run in order. These scripts are executed by the *bash* shell (Bourne shell). If a script does not exit cleanly and fails to return 0, the installation is aborted and an error message is displayed to the end user. If the installation script aborts, it aborts cleanly and leaves the server in a predictable state.

In this example, the installer program performs the following tasks:

1) Parses the `packing_list` file.
2) Verifies the specified RPM file, `blink-01-1.mips.rpm` is present.
3) Executes the install script `upgrade_me`, which installs all the RPM files in the package and generates a `.installed_rpms` file. The `.installed_rpms` file lists the names of the RPM files that were successfully installed. This is the code for `upgrade_me`:

```
#!/bin/sh
# upgrade_me
   # Things this upgrade_me script must do:
   # 1) Check appropriate Build
   # 2) Check to make sure none of its RPMS are installed
   # 3) Install its RPMS
   # 4) Generate the .installed_rpms file
   # 5) return status 0 on success

   #if [ -e /etc/build ]; then
       ## Replace the model in the grep statement below
       ## with the appropriate model(s) for your install
       ## Please reference Table 1 from the Tech note
       #cat /etc/build | grep "2800R" > /dev/null
           #if [ $? != 0 ]; then
           #echo "4015 This package for [RaQ] only..."
           #exit 1
           #fi
   #fi

   RPMS=`ls $UPGRADE_DIR/RPMS`
   # Check the RPMs installed on this system
```

```
for rpm in $RPMS; do
rpm -U --test $UPGRADE_DIR/RPMS/$rpm --nodeps --replacefiles
      if [ $? != 0 ]; then
      echo "4015 Problem verifying package component: $rpm"
      exit 1
      fi
done

# Create the file for the list of RPMS we are going to install.
echo -n "" > $UPGRADE_DIR/.installed_rpms

# Perform the installation.
for rpm in $RPMS; do
rpm -U $UPGRADE_DIR/RPMS/$rpm --nodeps --replacefiles
if [ $? != 0 ]; then
     echo "4015 Problem installing package component: $rpm"
     exit 1
else
     echo $rpm >> $UPGRADE_DIR/.installed_rpms
     fi
done

exit 0
```

4) Reads the `.installed_rpms` file and creates a text record in `/var/lib/cobalt/` that indicates the installation was performed by parsing the following tags:

```
Package:
List File:
Version:
```

from the `packing_list`. This file is named by combining the `List file:` tag, version number, and `.md5lst` extension. In this example, the file name is `sample1.0.md5lst`. It lists the name and version of the package (from the `Package:` and `Version:` tags). It includes the RPM files associated with that package along with their MD5 checksum. It is from this file that the web user interface gets a list of the software installed on the server.

5) Copies `uninstall_me` script to `/var/lib/cobalt/uninstallers/`. This script is used when a newer version of the package is installed so the old one can perform any necessary cleanup before the new software is installed. This is the code for `uninstall_me`:

```
#!/bin/sh
## uninstall_me

   ## do the following test for each RPM you've installed
   if [ `rpm -e --nodeps blink` ]
   then
     echo "4015 Error uninstalling RPM"
     exit 1;
   fi
   # Repeat the line below for each RPM file
   if [ `rpm -e --nodeps {OTHER RPMS}` ]
   then
     echo "4015 Error uninstalling RPM"
     exit 1;
   fi
```

```
# clean up the md5list for each rpm that was installed as well
# this is very important!
rm -f /var/lib/cobalt/blink-0.1-1.md5lst
/usr/admserv/cgi-bin/.cobalt/install/install.cgi < /dev/null >
/dev/null
```

Note
Each server appliance model may have a unique software configuration. The `upgrade_me` installation script determines the model for the server by reading the label file `/etc/build` on the server. If this file does not exist on the server, then the server is a first generation Qube 2700WG; see Table 1. Do **not** modify `/etc/build` on the server.

Table 1. Cobalt products and their software model numbers

Cobalt Server Product	Model Number
Qube	2700WG
Qube - Japanese	2700WGJ
Qube 2	2800WG
Qube 2 - Japanese	2800WGJ
Qube 2 - French	2800WGF
RaQ - Japanese	2700RJ
RaQ upgraded to RaQ 2 software	2799R
RaQ upgraded to RaQ 2 - Japanese	2799RJ
RaQ 2	2800R
RaQ 2 - Japanese	2800RJ
Cache Qube	2700C
Cache RaQ	2700CR
Cache RaQ 2	2800C
RaQ 3i	3000R

3 Creating a Package File

This section describes how to modify the `template-0.2.mips.pkg` file so that you can customize it.

1) Use FTP to transfer `template.pkg`:

   `ftp://ftp.cobaltnet.com/pub/developer/template-0.2.mips.pkg`

2) Change `packing_list` to reflect your options including *.rpm files, reboot and register options.

Note
If you have alredy created RPM files for your software, add each RPM file to `packing_list` and follow the procedures for modifying the default `upgrade_me` and `uninstall_me` scripts.

```
Package: Cobalt Sample Package for MIPS #[enter actual package name instead]
Version: 1.0 #[enter actual package version number instead]
REBOOT: no #[yes/no] #[use Yes if the server must reboot after the package is loaded]
UNINSTALL: uninstall_me #[don't change this line]
SCRIPT: upgrade_me #[don't change this line]
SCRIPT: register_me
SCRIPT: update_list
List File: sample #[enter actual list filename instead]
RPM: null-0-0.mips.rpm #[enter actual rpm names as applicable]
   ...(list all rpm files)
```

Note
You must have at least one RPM for your package file to work. If necessary, include a null RPM; see "Creating a Null RPM File" on page 14. Create a subdirectory named `RPMS`. You **must** put all RPM files including null RPM files into the`RPMS` subdirectory.

3) Check the `upgrade_me` file to make sure it meets your needs.

Note
If you are using RPM files, the default `upgrade_me` script should work. It installs all the RPM files in alphabetical order. If you are not using RPM files or if you need RPM files installed in a different order, you can write your own installation script provided that it returns the same values and is a Bourne shell script. If you write your own installation script, you must also write a corresponding `uninstall_me` file.

```
#!/bin/sh
# upgrade_me
        # Things this upgrade_me script must do:
        # 1) Check to make sure none of its RPMS are installed
        # 2) Install its RPMS
        # 3) Generate the .installed_rpms file
        # 4) return status 0 on success
        if [ -e /etc/build ]; then
               cat /etc/build | grep "model#" > /dev/null
               if [ $? != 0 ]; then
                       echo "4015 This package for [product] only..."
                       exit 1
               fi
       fi

        RPMS=`ls $UPGRADE_DIR/RPMS`
        # Check the RPMs installed on this system
        for rpm in $RPMS; do
                rpm -U --test $UPGRADE_DIR/RPMS/$rpm --nodeps --replacefiles
                if [ $? != 0 ]; then
                        echo "4015 Problem verifying package component: $rpm"
                        exit 1
                fi
        done

        # Create the file for the list of RPMS we're going to install.
        echo -n "" > $UPGRADE_DIR/.installed_rpms

        # Perform the installation.
        for rpm in $RPMS; do
                rpm -U $UPGRADE_DIR/RPMS/$rpm --nodeps --replacefiles
                if [ $? != 0 ]; then
                        echo "4015 Problem installing package component: $rpm"
                        exit 1
                else
                        echo $rpm >> $UPGRADE_DIR/.installed_rpms
                fi
        done
exit 0
```

The return statement consists of a four-digit number followed by an error message. The four-digit number is recognized by the install script, `/usr/local/sbin/cobalt_upgrade`, as the state of the installation. The string following the four-digit results code is passed directly to the web browser in the bottom frame. If the results code is not 2999, an error GIF image is displayed with the returned text. See Table 2 for a list of codes and their meanings.

Table 2. Results codes and their meanings

Results Code	Meaning
2999	Total success
3999	Mid-install abort
4999	Complete failure
4015	Problem installing package component: *filename.rpm*

4) Edit `uninstall_me` for your files; you can modify this code example.

```
#!/bin/sh
## uninstall_me

   ## do the following test for each RPM you've installed
   if [ `rpm -e --nodeps null-0-0` ]
   then
        echo "4015 Error uninstalling RPM"
        exit 1;
   fi

# clean up the md5list for each rpm that was installed as well
# this is very important!
   rm -f /var/lib/cobalt/null-0-0.md5lst
```

a) Duplicate this code segment and edit it for each *.rpm file that you include in your package file.

```
## do the following test for each RPM you've installed
   if [ `rpm -e --nodeps null-0-0` ]
   then
```

```
        echo "4015 Error uninstalling RPM"
        exit 1;
    fi
```

b) Save the file with the file name `uninstall_me`.

5) Edit the `register_me` file and change the top section to reflect your company and vendor; see "Using the register_me File" on page 10.

6) Edit `update_list` to include the name and description of your application; see "Using the update_list File for Qube 2 Models" on page 8.

7) Create a package file. Place all of the files created in steps 1 through 4 in a directory by themselves. From this directory, run the following command to create a compressed package file:

```
tar -czvf ~/<package_name>.pkg *
```

The package file is now complete. Refer to the user's guide that came with your Cobalt server appliance for details on installing package files.

3.1 Setting Up Notification When Package Files Are Installed

The default behavior for third-party package files is to notify Cobalt and you via email each time your package file is installed. You must edit the `register_me` file and put the appropriate contact information in it.

Note

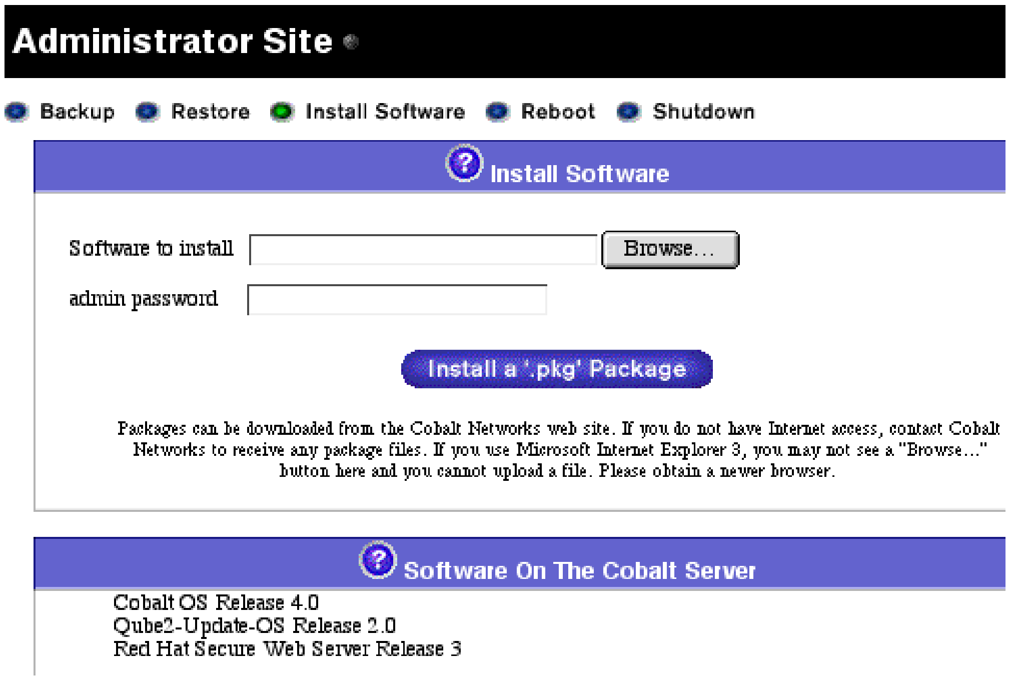
If you want to offer your customers the option of installing the package file with or without notification, you must create two package files: one with notification and one without. The customer can be offered the choice of downloading the standard one with notification or the non-standard one without notification.

To create a package file that does not include notification, delete this line from the `packing_list` file:

```
      SCRIPT: register_me
```

3.2 Using the update_list File for Qube 2 Models

The Qube 2 links to an index page from the Cobalt user interface as shown below.



Packages can be downloaded from the Cobalt Networks web site. If you do not have Internet access, contact Cobalt Networks to receive any package files. If you use Microsoft Internet Explorer 3, you may not see a "Browse..." button here and you cannot upload a file. Please obtain a newer browser.

Cobalt OS Release 4.0
Qube2-Update-OS Release 2.0
Red Hat Secure Web Server Release 3

To set this up so that the newly installed software appears in the list of programs on the Programs Page, follow these steps:

1) Edit the `APPNAME` line to include your application name **without** spaces, for example: *CobaltServerAppliances*.

2) Edit the `NAME` line to include your application name **with** spaces, for example: *Cobalt Server Appliances*.

3) Edit the description for your application.

Note
This script automatically checks the model of Cobalt server appliances and only installs this listing on Qube 2 models. Your program listing appears in the Programs Page on a Qube 2 and is ignored on other models.

This is the code for `update_list`.

```
#!/bin/sh

## update_list

## Please modify the three variables below.
## Make sure your install program then places a index.html file
## in /home/programs/<appname>/index.html

## Replace YOURNAMEHERE with the name of your application.
## Spaces are NOT allowed
APPNAME="YourProgramNameHere"

## Enter the full name of the program here. Feel free to use spaces
NAME="Your Program Name Here"

## Provide a brief description of the product
DESC="Your product description goes here"

################################################
################################################
## DO NOT MODIFY BELOW THIS LINE
################################################
################################################

## Check to make sure it's a Qube2
cat /etc/build | grep "2800WG" > /dev/null
if [ $? != 0 ]; then
   exit 0
fi

mkdir /home/programs/$APPNAME

echo "program-name: $NAME" >> /home/programs/$APPNAME/.programInfo.dat;
echo "program-description: $DESC" >> /home/programs/$APPNAME/.programInfo.dat;
exit 0
```

3.3 Using the register_me File

The `register_me` file automatically extracts useful information about the software that is being installed on the Cobalt server appliance.

You must modify the following sections with your product information:

- Email address to receive the registration messages, for example, *productxreg@yourcompany.com*
- Company email address
- Product name
- Product version number
- Software type, language (for example: English or Japanese), trial or full version, or other variables
- All sites; specify True only if the software is designed to be visible and used by all virtual sites on a RaQ server appliance. Specify False if it is intended only for a RaQ server administrator or if the software is intended for a Qube, Cache, or NasRaQ server appliance.
- Type of event, for example: install, upgrade, or other

Following is the code of the `register_me` file.

```
#!/bin/sh -- # perl, to stop looping
eval 'exec /usr/bin/perl -S $0 ${1+"$@"}'
   if 0;

##########################################################################
## PLEASE MODIFY BELOW FOR YOUR SW
```

```
# The email address which you wish to have the register mail sent to
#   please make sure that the slash remains before the @ symbol. e.g.,
#   register\@domain.com   or
#   admin\@myplace.co.uk
$vendorEmail = "register\@yourdomain.com";

# The name of your company. Please place this inside angle brackets, (<>), e.g.,
#   <Cobalt Networks> or
#   <Your Company Name>;
$vendorName = "<vendor name>";

# The name of your product. Again, maintain the angle brackets around the name.
$vendorProductName = "<product name>";

# Version of product. You know the < drill
$vendorProductRevision = "<product version>";

# What type of software is installed - i.e., trial, language, full version
$vendorProductVariant = "<variant>";

# Does this software work on all sites of a RaQ [RaQ family ONLY - set false for Qube]
#    [false] or
#    [true]
$vendorMultiSiteFlag = "[true|false]";

# Type of event. Choose one. Most likely install:
$vendorEventType = "[install|upgrade|uninstall|purchase|activate|deactivate]";

##########################################################################
##########################################################################
## DO NOT MODIFY BELOW THIS LINE!!!!!!!!!!
##########################################################################
##########################################################################

use Cobalt::Network; #perl in /usr/lib/perl5/site_perl/auto/Cobalt/Network
use Cobalt::Time;

#AUTOMATIC TAGS###############
#Register version (version of Register) 1.1.0
#date, time
#Time zone/geography
#name of server
#type of machine
#kernel version
#admin email address
#OS version (build version?)
#primary mac address
#primary ip address
#total memory
#hard disk config (size, number)
###############################

$registerVersion = "1.1.0";
my @time = localtime();
my $month = $time[4] + 1;
my $day = $time[3];
my $year = $time[5] + 1900;
my $seconds = $time[0];
my $minutes = $time[1];
my $hours = $time[2];
$installMonth = "$month";
$installDay = "$day";
$installYear = "$year";
$installHour = "$hours";
$installMinutes = "$minutes";
$installDate = "$month/$day/$year $hours:$minutes:$seconds";
$installZone = time_get_zone();
$hardDiskConfig ="";
foreach $_ (`df`) {
   /^\/dev\/(\w*)\s*(\w*)\s*.*(\/.*$)/;
   if ($1) {
   $hardDiskConfig = $hardDiskConfig . $1 . "(" . $2 . $3 . ") ";
   }
}
$hostName = `/bin/hostname`;
$adminEmail = "admin\@$hostName";  #need qualified hostname here instead of domain in
   case admins are different!!
$_ = `cat /proc/cpuinfo | grep "cpu"`;
/cpu\s*:\s*(.*)\s*/;
$cpu = $1;
/cpu model\s*:\s*(.*)\s*/;
$cpuModel = $1;
$_ = `cat /proc/cpuinfo | grep "system type"`;
/system type\s*:\s*(.*)\s*/;
$systemType = $1;
$_ = `cat /proc/meminfo | grep "MemTotal:"`;
/MemTotal:\s*(.*)\s*/;
$memory = $1;
$cobaltRelease = `cat /etc/cobalt-release`;
$cobaltBuild = `cat /etc/build`;
$cobaltKernelRelease = `cat /proc/sys/kernel/osrelease`;
$cobaltKernelVersion = `cat /proc/sys/kernel/version`;
my $IPAddr=( net_get_ipaddr() )[ 0 ];
my $MACAddr=( net_get_ipaddr() )[ 2 ];

chomp($installDate);
chomp($installZone);
chomp($adminEmail);
chomp($hardDiskConfig);
chomp($hostName);
chomp($cpu);
chomp($cpuModel);
chomp($systemType);
chomp($memory);
chomp($cobaltRelease);
chomp($cobaltBuild);
chomp($cobaltKernelRelease);
chomp($cobaltKernelVersion);
chomp($IPAddr);
chomp($MACAddr);
```

```
$toAddr = "register\@cobaltnet.com";
if ( $vendorEmail ne "register\@yourdomain.com" ) {
   $toAddr = $toAddr . ", " . $vendorEmail;
}
open SM, "| /usr/sbin/sendmail -t" or die "can't open sendmail!";

print SM "To: $toAddr\n";
print SM "From: Register.$registerVersion\n";
print SM "Subject: [RECEPTOR] $vendorProductName $vendorProductRevision\n";
print SM "\n\n";
print SM "<<COBALT RECEPTOR>>\n";

print SM "<<VENDOR VENDOR_NAME = $vendorName >>\n";
print SM "<<VENDOR PRODUCT_NAME = $vendorProductName >>\n";
print SM "<<VENDOR PRODUCT_VERSION = $vendorProductRevision >>\n";
print SM "<<VENDOR PRODUCT_VARIANT = $vendorProductVariant >>\n";
print SM "<<VENDOR MULTI_SITE = $vendorMultiSiteFlag >>\n";
print SM "<<VENDOR EVENT_TYPE = $vendorEventType >>\n";

#########################################
##CUSTOM VENDOR TAGS HERE################
print SM "<<VENDOR MY_CUSTOM_TAG = my custom value >>\n";


##END CUSTOM VENDOR TAGS#################
#########################################

print SM "\n";
print SM "<<COBALT REGISTER_VERSION = $registerVersion >>\n";
print SM "<<COBALT INSTALL_DATE = $installDate >>\n";
print SM "<<COBALT INSTALL_ZONE = $installZone >>\n";
print SM "<<COBALT INSTALL_HOUR = $installHour >>\n";
print SM "<<COBALT INSTALL_MINUTES = $installMinutes >>\n";
print SM "<<COBALT INSTALL_MONTH = $installMonth >>\n";
print SM "<<COBALT INSTALL_DAY = $installDay >>\n";
print SM "<<COBALT INSTALL_YEAR = $installYear >>\n";
print SM "<<COBALT HOSTNAME = $hostName >>\n";
print SM "<<COBALT ADMIN_EMAIL = $adminEmail >>\n";
print SM "<<COBALT CPU = $cpu >>\n";
print SM "<<COBALT CPU_MODEL = $cpuModel >>\n";
print SM "<<COBALT SYSTEM = $systemType >>\n";
print SM "<<COBALT MEMORY = $memory >>\n";
print SM "<<COBALT HARD_DISK_CONFIG = $hardDiskConfig >>\n";
print SM "<<COBALT RELEASE = $cobaltRelease >>\n";
print SM "<<COBALT BUILD = $cobaltBuild >>\n";
print SM "<<COBALT KERNEL_RELEASE = $cobaltKernelRelease >>\n";
print SM "<<COBALT KERNEL_VERSION = $cobaltKernelVersion >>\n";
print SM "<<COBALT IP_ADDR = $IPAddr >>\n";
print SM "<<COBALT MAC_ADDR = $MACAddr >>\n";
print SM "<<COBALT /RECEPTOR>>\n";
close SM;
```

4 Creating a Null RPM File

All package files must have at least one RPM file included with the package; even if you choose to use another mechanism such as tar to install your software, the package file must have at least one RPM file. If necessary, create a "null" RPM file.

1) Type this text and save it as `/usr/src/redhat/SPECS/null.spec`.

```
Summary: A null RPM to make all package files work
name: null
Version: 0
Vendor: Nobody
Release: 0
Copyright: 1999
Group: Base
Build Architecture: noarch

%description
-nothing
%changelog
%prep
%install
%post
%clean
%files
```

   This creates the null specifications file.

2) Quit the editor.

3) Type:

```
rpm -ba /usr/src/redhat/SPECS/null.spec
```

   This command creates the null RPM file `null-0.0.noarch.rpm` in `/usr/src/redhat/RPMS/noarch/`.