August 2000
DeveloperTechNote-2, Revision 0-8

Creating Software Package Files for the Cobalt RaQ 3 and RaQ 4 Families of 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 Technical Note describes how to create software packages for the Cobalt RaQ 3 and RaQ 4 families of server appliances, which are referred to as the *RaQ 3* and *RaQ 4* in this technical note. In addition, it describes some optional mechanisms that can be added while creating package files. This technical note can be found at `ftp://ftp/cobaltnet.com/pub/developer/TechNotes/DTN2-0-7.pdf`

Note

RaQ 3 and RaQ 4 server appliances use x86 architecture and are not binary-compatible with earlier Cobalt server appliances that use MIPS architecture. For information on creating software packages for Cobalt's RaQ, RaQ 2, or Qube family server appliances, see Cobalt's Developer Technical Note-1 located at `ftp://ftp.cobaltnet.com/pub/developer/TechNotes/`

Table of Contents

1.1 Audience

This Technical Note is intended for application developers creating software to be used with Cobalt Network's RaQ 3 and RaQ 3 server appliances.

1.2 What's New in this Technical Note

This Technical Note covers information only for the RaQ 3 and RaQ 4 server appliance. New features include a URL that you can use to link users to pertinent web sites, such as an administration or a help screen for your software.

Note
If the software you are using was created solely for use on the MIPS architecture, you must recompile binaries so that they work with the x86 architecture and a Linux 2.2 kernel.

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, URL field, 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 format that you install on the Cobalt server appliance; see "Overview of RedHat Package Manager (RPM) Files" below 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.
- `index.html` file—a web page that you create that gets linked from the installed software page of the server administration.

2.1 Sample and Template Package Files

To start creating package files, download the `sample-1.1.i386.pkg` and `template-1.1.i386.pkg` files:

```
ftp://ftp.cobaltnet.com/pub/developer/sample-1.1.i386.pkg
ftp://ftp.cobaltnet.com/pub/developer/template-1.1.i386.pkg
```

`i386` is the standard naming convention for the Linux platform. Cobalt created `sample-1.1.i386.pkg` to demonstrate the package mechanism; it contains live code that simply causes the server LED to blink for 120 seconds one time. You can use `template-1.1.i386.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-1.1.i386.pkg
or
tar -xzvf template-1.1.i386.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`, `index.html`, `uninstall_me`, `register_me`, 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-1.1.i386.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 8.

The `packing_list` file for `sample-1.1.i386.pkg` looks like this:

```
Package: Cobalt Developer Sample Package for x86 platforms
Version: 1.0
List File: sample
REBOOT: no
URL: /.cobalt/sample/index.html
SCRIPT: upgrade_me
RPM: blink-0-1.i386.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-0-1.i386.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`:

```
# Things this 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 | egrep -e "(3000R|3001R|3100R)" > /dev/null
  if [ $? != 0 ]; then
    echo "4015 This package is for the RaQ 3 and RaQ 4 only"
    exit 1
  fi
else
  echo "4015 This package is for the RaQ 3 and RaQ 4only"
  exit 1
fi

RPMS=`ls $UPGRADE_DIR/RPMS`

# Check the RPMs installed on this system
for rpm in $RPMS; do
  rpm -U --test --rcfile /usr/lib/rpm/rpmrc $UPGRADE_DIR/RPMS/$rpm --nodeps &> /dev/
null
  # Since this could be patching over a previous patch, we need to ignore
  # rpms that the other package installed.
  if [ $? != 0 ]; then
    echo "4015 Problem verifying package component: $UPGRADE_DIR/RPMS/$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 --rcfile /usr/lib/rpm/rpmrc $UPGRADE_DIR/RPMS/$rpm --nodeps --force &> /dev/
null
  if [ $? != 0 ]; then
    echo "4015 Problem installing package component: $rpm"
    exit 1
  else
    echo $rpm >> $UPGRADE_DIR/.installed_rpms
  fi
done

# Copy the info page
mkdir -p /usr/admserv/html/.cobalt/test/
cp $UPGRADE_DIR/index.html /usr/admserv/html/.cobalt/test/index.html

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 `sample-1.1.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
```

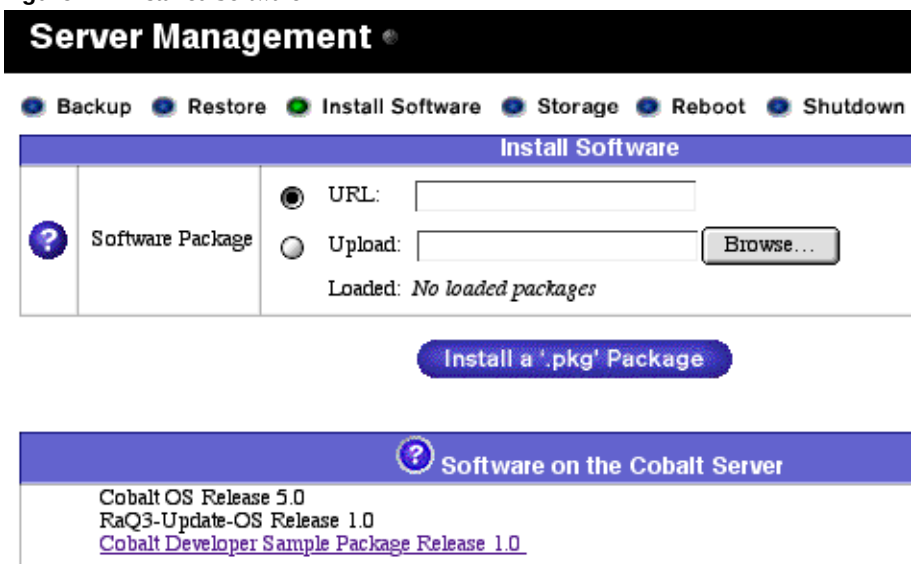
Note

Each server appliance model has a unique software configuration. The RaQ 3 and RaQ 4 software model numbers are 3000R, 3001R, or 3100R. The `upgrade_me` installation script determines the model for the server by reading the label file `/etc/build` on the server. Do **not** modify `/etc/build` on the server.

2.4 Example of the index.html file

The `index.html` file is a file that gets linked from the installed software page to a web page that you create. You can use `index.html` to link users to pertinent web sites, such as an administration or a help screen for your software.

Figure 1. Installed Software



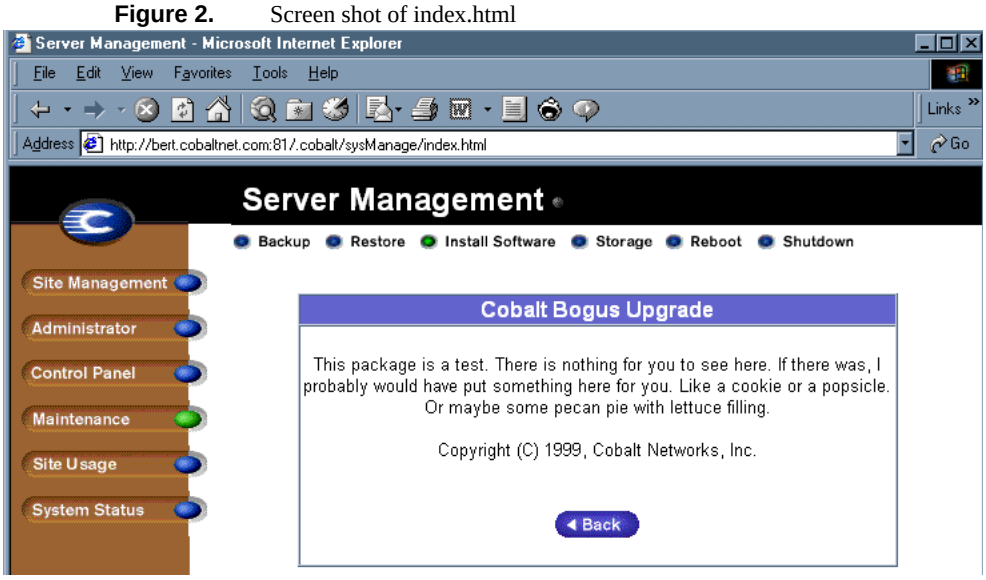
A sample `index.html` file is shown below. You must modify it to make it fit your needs.

> **Note**
> If you do not have a URL tag in the packing list, it will not generate a link.

```
<HTML>

<HEAD>
<META NAME="Copyright" VALUE="Copyright 1998-2000, Cobalt Networks, Inc.  All rights
reserved.">
<META HTTP-EQUIV="expires" CONTENT="0">

<SCRIPT LANGUAGE="javascript">
<!-- hide
top.showInfo();
// end hide -->
</SCRIPT>
</HEAD>

<BODY BGCOLOR="#FFFFFF">

<CENTER>
<P>
<BR>
<TABLE BORDER="1" CELLSPACING="0" WIDTH="450">
   <TR>
   <TD BGCOLOR="#6666CC">
   <P ALIGN="CENTER"><FONT COLOR="#FFFFFF" FACE="Helvetica, Arial"><B>Cobalt Bogus
Upgrade</B></FONT>
   </TD>
   </TR>
   <TR>
```

```
   <TD>
   <BR>
   <FONT SIZE="2" FACE="Helvetica, Arial">
   <CENTER>This package is a test. There is nothing for you to see here. If there was,
I probably would have put something here for you. Like a cookie or a popsicle. Or maybe
some pecan pie with lettuce filling.<BR>
<BR>Copyright (C) 2000, Cobalt Networks, Inc.</CENTER>
</FONT>
   <BR><BR>
   <CENTER><A HREF="/cgi-bin/.cobalt/install/install.cgi" onMouseOver="status='Back';
return true;" ><IMG SRC="/.cobalt/images/back_but.gif" BORDER=0></A></CENTER>
   <BR>
   </TD>
   </TR>
</TABLE>

</CENTER>

<SCRIPT LANGUAGE="javascript">
<!-- hide
top.pageNum=5;
top.done=true;
top.success=true;

top.registerred=true;
// end hide -->
</SCRIPT>

</BODY>

</HTML>
```

Figure 2. Screen shot of index.html



3 Creating a Package File

This section describes how to modify the `template-1.1.i386.pkg` file so that you can customize it.

1) Use FTP to transfer `template-1.1.i386.pkg`:

   `ftp://ftp.cobaltnet.com/pub/developer/template-1.1.i386.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 i386 Sample Package #[enter actual package name instead]
Version: 1.0 #[enter actual package 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
List File: sample #[enter actual list filename instead]
RPM: null-0-0.i386.rpm #[enter actual rpm names as applicable]
   ...(list all rpm files)
URL: #[enter any URL that you want displayed}]
```

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.

```
# Things this 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 | egrep -e "(3000R|3001R|3100R)" > /dev/null
  if [ $? != 0 ]; then
    echo "4015 This package is for the RaQ 3 and RaQ 4 only"
    exit 1
```

```
  fi
else
  echo "4015 This package is for the RaQ 3 and RaQ 4 only"
  exit 1
fi

RPMS=`ls $UPGRADE_DIR/RPMS`

# Check the RPMs installed on this system
for rpm in $RPMS; do
  rpm -U --test --rcfile /usr/lib/rpm/rpmrc $UPGRADE_DIR/RPMS/$rpm --nodeps &> /dev/
null
  # Since this could be patching over a previous patch, we need to ignore
  # rpms that the other package installed.
  if [ $? != 0 ]; then
    echo "4015 Problem verifying package component: $UPGRADE_DIR/RPMS/$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 --rcfile /usr/lib/rpm/rpmrc $UPGRADE_DIR/RPMS/$rpm --nodeps --force &> /dev/
null
  if [ $? != 0 ]; then
    echo "4015 Problem installing package component: $rpm"
    exit 1
  else
    echo $rpm >> $UPGRADE_DIR/.installed_rpms
  fi
done

# Copy the info page
## You will need to modify this for your package
mkdir -p /usr/admserv/html/.cobalt/sample/
cp $UPGRADE_DIR/index.html /usr/admserv/html/.cobalt/sample/index.html

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 1 for a list of codes and their meanings.

Table 1. Results codes and their meanings

Results Code	Meaning
2999	Total success
3999	Mid-install abort

Table 1. Results codes and their meanings

Results Code	Meaning
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 11.

6) 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 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 for the RaQ 3 and RaQ 4.
- 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 <>'s e.g.
#   <Cobalt Networks> or
#   <Your Company Name>;
$vendorName = "<Cobalt Networks>";

# The name of your product.  Again, maintain the angles around the name
$vendorProductName = "<Sample PKG>";

# Version of product.  You know the < drill
$vendorProductRevision = "<1.0>";

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

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

```
# Type of event.  Most likely install:
#    [install]
$vendorEventType = "[install]";

################################################################################
################################################################################
## 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 1.1.1
#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 = "1.1.1";
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!!
## /proc/cpuinfo
```

```
open(FILE, "/proc/cpuinfo");
$tmp = 0;

while (<FILE>) {
        $tmp = $1 if /^processor\s+: (\d)/;
   $cpuModel = $1 if /^model name\s+: (.*)/;
        $cpu = $1 if /^cpu MHz\s+: (.*)/;
}

$_ = `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 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: 2000
Group: Base

%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.i386.rpm` in `/usr/src/redhat/RPMS/i386/`.