Application Note

CPE

CMS Desktop Linux

CPE-AN601-R

5300 California Avenue • Irvine, CA 92617 • Phone: 949-926-5000 • Fax: 949-926-5203

September 3, 2014

Revision History

Revision	*Date*	*Change Description*
CPE-AN601-R	09/03/14	**Updated:** • “Compiling Desktop Linux” on page 6. • “Using Desktop Linux on 64-bit Linux Systems” on page 10.
CPE-AN600-R	01/28/14	Initial release

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Table of Contents

About This Document 4
Purpose and Audience 4
Acronyms and Abbreviations 4
Document Conventions 4
References 5
Technical Support 5
Introduction 6
Compiling Desktop Linux 6
Running on Desktop Linux 7
Starting Smd 8
Stopping Smd (and all CMS Software) 9
Accessing the WebUI 9
Accessing the CMS CLI with Consoled 9
Using TR69C 9
Using Desktop Linux on 64-bit Linux Systems 10
Inspecting the Configuration File 10
Writing Desktop Linux Friendly Code 11

About This Document

Purpose and Audience

This document explains how to run the core parts of the CMS software on a standard desktop PC running Linux® distribution, or “Desktop Linux”.

This document is intended for software and system engineers.

Acronyms and Abbreviations

In most cases, acronyms and abbreviations are defined on first use.

For a comprehensive list of acronyms and other terms used in Broadcom documents, go to: http://www.broadcom.com/press/glossary.php.

Document Conventions

The following conventions may be used in this document:

Convention	*Description*
Bold	User input and actions: for example, type **exit**, click **OK,** press **Alt+C**
`Monospace`	Code: `#include <iostream>` HTML: `<td rowspan = 3>` Command line commands and parameters: `wl [-l] <command>`
< >	Placeholders for *required* elements: enter your <username> or `wl <command>`
[ ]	Indicates *optional* command-line parameters: `wl [-l]` Indicates bit and byte ranges (inclusive): [0:3] or [7:0]

References

The references in this section may be used in conjunction with this document.

Note: Broadcom provides customer access to technical documentation and software through its Customer Support Portal (CSP) and Downloads and Support site (see Technical Support).

For Broadcom documents, replace the “xx” in the document number with the largest number available in the repository to ensure that you have the most current version of the document.

Document Name	Number	Source
Broadcom Documents		
[1] Application Debugging Using GDB - V4.2 FOR BCA LINUX® ROUTERS	CPE-AN5xx-R	CSP

Technical Support

Broadcom provides customer access to a wide range of information, including technical documentation, schematic diagrams, product bill of materials, PCB layout information, and software updates through its customer support portal (https://support.broadcom.com). For a CSP account, contact your Sales or Engineering support representative.

In addition, Broadcom provides other product support through its Downloads and Support site (http://www.broadcom.com/support/).

Introduction

The core parts of the CMS software can be run on a standard desktop PC running a relatively recent (less than two years) Linux® distribution. This feature is called “Desktop Linux”.

Being able to run CMS software on the desktop significantly reduces the time needed for the edit/compile/debug cycle. Running on the desktop also allows developers to use standard software debug tools, such as gdb and valgrind, on the applications. Desktop Linux is particularly useful for developing web pages, tr69c, and the core MDM. The last two tasks are usually only done by internal Broadcom developers.

The parts of the CMS software that can run on the desktop are the data model scripts (in the data-model directory): smd, ssk, httpd, consoled, tr69c and the CMS libraries which they depend on (for example, libcms_core.so, libcms_util.so, libcms_msg.so, and so on). When a CMS application runs on the desktop, its functionality is limited. For example, if you are using the WebUI to configure the Ethernet port, httpd will not actually configure the Ethernet port. However, the data model is updated and the configuration is saved by “simulated” flash device driver which writes the config file to a plain file in the desktop file system. In general, any operation that is limited to pure software (for example, manipulation of data structures) or accesses features simulated on the desktop will work. Any operation that configures actual hardware or does things inside the kernel will be silently ignored. Typically, it is recommended to first do as much development and debug as possible using Desktop Linux. Then build images for the real target device to verify that your code actually works as desired on the hardware.

The Desktop Linux feature was initially introduced with the first release of CMS, in release 4.02L.01. Since then, this feature has been neglected and somewhat broken. It has been fixed again in 4.14L.01, but only the 963268GW profile has been tested with Desktop Linux. Other profiles are not supported, but may work after some minor fixes. Also, many features are not supported with Desktop Linux. If you would like to run a particular feature on the desktop, try to do it yourself or contact your support engineer for assistance.

Compiling Desktop Linux

Follow these steps to compile the BCA CPE reference software for Desktop Linux:

1. Do a normal compile using a standard profile, for example 963268GW. This is needed to ensure the kernel source tarball is untarred, and the proper links to the kernel header files are created.
2. Type **make userspace_clean**.
3. Create a Desktop Linux profile. This can be done in one of two ways:
   - By using the DESKTOP.arch file. Type “release/maketargets 963268GW_DESKTOP”. The maketargets script will read in the DESKTOP.arch file (located in targets/arch) and create a new profile called 963268GW_DESKTOP under the targets directory.
   - By running make menuconfig. Follow these steps:
     a. Run make menuconfig and load the 963268GW profile
     b. In the *Major Feature Selection* section, enable “Build CMS for DESKTOP LINUX”.
     c. In *WLAN Selection*, disable *Wireless Driver*, *Wireless WAPI Support*, *Wireless DHD Driver Support*, *Wireless Brand Selection*, *Wireless Control Utility*, *NVRAM Emulation*, etc. Basically, all WLAN features should be disabled for Desktop Linux builds.

d. In *Other Features*, disable *Userspace Stack Smashing Protection* support.

e. In *Management Protocols and User Interface Selection*, in *TR69 Management Protocol*, choose *TR69* (not *TR69_SSL*). Also, disable *Extensible Messaging and Presence Protocol (XMPP)*.

f. Save the profile as 963268GW_DESKTOP and exit menuconfig.

4. Overwrite the contents of the .last_profile file with the name of the desktop Linux profile using the following command: “echo 963268GW_DESKTOP > .last_profile”.

5. Cd into userspace/public/libs and type **make cms_util cms_msg cms_boardctl**.

6. Cd into userspace/private/libs/tr143_utils and type **make**.

7. Cd into userspace/private/libs/cms_core and type **make**. Unlike previous releases, saved binary objects do not have to be moved manually. As of 4.16L.03, the Makefile automatically selects the correct saved binary file.

8. Cd into userspace/private/libs and type **make mdm mdm2 cms_qdm cms_dal cms_cli nanoxml ethswctl vlanctl pwrctl fapctl atmctl xdslctl**.

9. Cd into userspace/private/apps/httpd and type **make**. At the end of the make, if any libraries are missing, the final link of the http will fail and a list of missing libraries will be shown. Make those libraries and recompile httpd.

10. Cd into userspace/private/apps and type **make smd ssk consoled**.

11. Optional: cd into userspace/private/apps/tr69c and type **make**. Unlike previous releases, saved binary objects do not have to be moved manually. As of 4.16L.03, the Makefile automatically selects the correct saved binary file.

12. If you make a change to an application or library, you can just rebuild in the directory where you made the change. However, if you make a change to the data model or any global defines (profile, make.common, and so on), you should do a “make userspace_clean”, cd into the data-model directory and type **make**, and then repeat Step 5 through Step 11.

13. When you make a change to the software, it is best to first stop Smd (described in “Stopping Smd (and all CMS Software)” on page 9), do the recompile, and then start Smd (described in “Starting Smd” on page 8).

Running on Desktop Linux

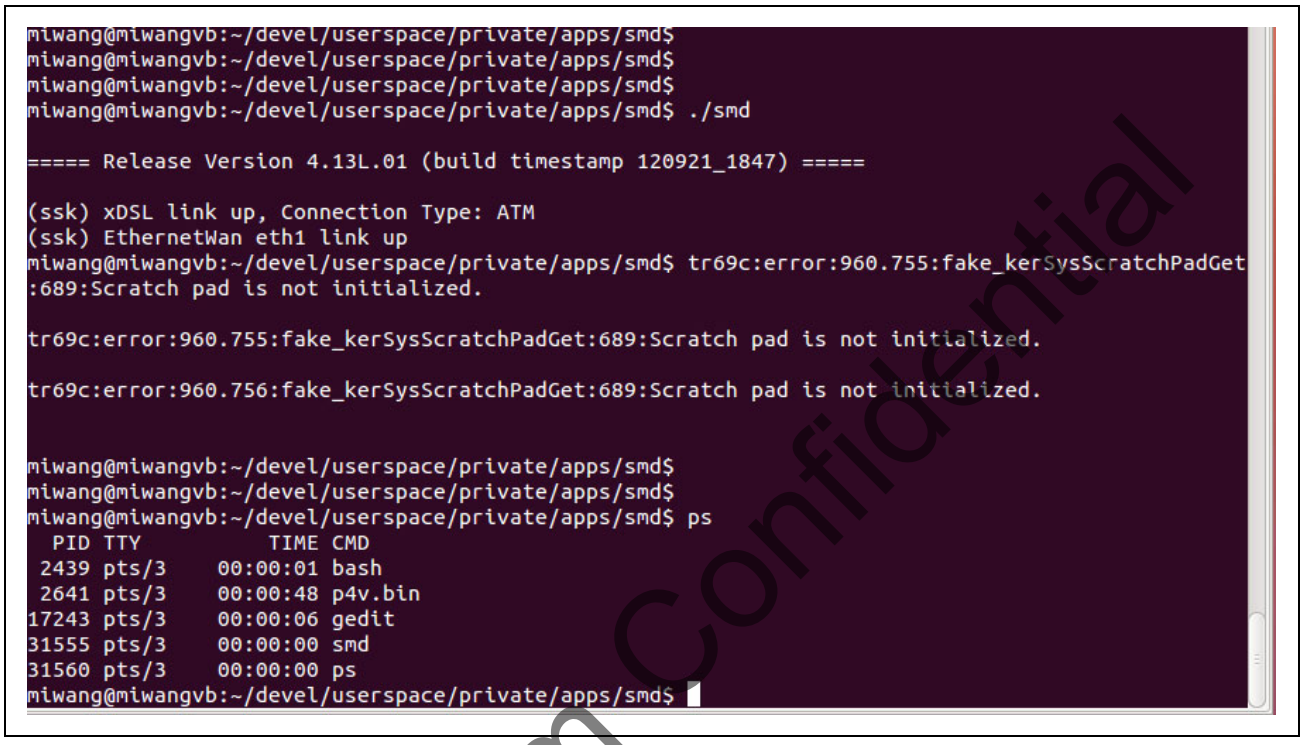
It is strongly recommended that you do NOT run Desktop Linux as the root user. Although the software is designed to only operate on files in the build directory, a bug in the code could cause the software to operate on the files under /. If you are running as root, you could overwrite critical system files which make your desktop system unusable.

Typical Linux desktop systems will disable core dumping when an application crashes. However, since we are doing software development, it is useful for applications to generate a core file when they crash. You can check for this feature by typing **ulimit -c**. If it says “unlimited”, then that means core dumps (of unlimited size) are enabled. If it does not say “unlimited”, you can set it by typing **ulimit -c unlimited** in the same shell that you will start Smd in.

Starting Smd

Once all the software has been compiled, cd into userspace/private/apps/smd and type **./smd**. Figure 1 shows what this looks like:

Figure 1: Starting Smd



In previous releases, Smd will stay in the foreground, but starting with the 4.14L.01 release, the initial Smd will fork a child Smd, which becomes the main Smd, and the initial Smd will exit. So it may appear that Smd is not running, but if you type **ps**, you will see there is a Smd running in the background. The output of Smd and all other CMS applications, including cmsLog_xxx output, will come out in this window.

Some error messages may appear when you start Smd. It is usually safe to ignore them. If it was really an important error, ssk or Smd will fail to initialize and will exit.

Since Smd now runs in the background, it is easy to forget that it is still running. If you try to start a new instance of Smd while an old instance is still running, you will see the following error message:

```
./smd:error:992.856:initInetServerSocket:1637:bind errno=98 port=44401 fd=5
./smd:error:992.856:oalSysmon_init:119:Could not open sysmon socket, init failed.
./smd:error:992.856:main:178:initialization failed (9002), exit.
```

In this case, stop the old Smd and then start the new Smd again.

Stopping Smd (and all CMS Software)

Smd is the main controlling application in the CMS system. So if you exit Smd, all other CMS applications will also exit. The easiest way to exit Smd is with “killall smd”. Most modern Linux distributions have the “killall” command. If your distribution does not have the killall command, you will have to type **ps** to find the pid of Smd, and then type **kill -TERM <pid_of_smd>**.

Accessing the WebUI

To access the WebUI on Desktop Linux:

1. Start Smd.
2. Open a browser and go to **http://127.0.0.1:44480**. The browser must be on the same desktop system that is running Smd.
3. Log into the WebUI as usual. Httpd will treat your access as a LAN side access.

When you modify settings and create PVCs on the desktop WebUI, all the code in the cgi_xxx files, the DAL, MDM, RCL, STL, and RUT will be executed. The only things that are not executed are the actual low-level system commands that configure the PVC or get statistics from the PVC (because there is no DSL hardware on your desktop). However, the code that does run on the desktop is a high percentage of the total code needed to implement a feature, so being able to quickly and easily debug that code on the desktop should save developers a lot of time.

Accessing the CMS CLI with Consoled

You can use consoled to debug and develop CMS CLI functions.

1. Start Smd.
2. Either on the same terminal window as Smd or in a different window, cd into userspace/private/apps/consoled and type **./consoled**.
3. Log in as usual and use the CLI.

Using TR69C

To use tr69c on the desktop:

1. Start Smd.
2. Use the WebUI and go to the Advanced Setup, and then the LAN page.
3. Change the IP address to the IP address of the Ethernet interface (eth0) on your desktop Linux system. You can leave the DHCP server enabled as it does not run on the desktop Linux.
4. Go to Management and then TR-069 Client. For the ACS URL, type the <URL> of the ACS server. The ACS must be reachable from the eth0 interface of your desktop Linux system. If you are using an ACS simulator, it can be on the network, or it could be on the same system as the desktop Linux.
5. From the WAN Interface used by TR-069 client list, select LAN.

Now the tr69c on the desktop Linux will use the eth0 on the desktop Linux system as its interface to contact the ACS server. The tr69c application works as if it was running on the modem, with the exception that it will not actually configure the system.

Using Desktop Linux on 64-bit Linux Systems

Support for running Desktop Linux on 64-bit Linux systems was added in 4.16L.02. Even though the software is running on a 64-bit system, all the software is compiled in 32-bit mode. So you must install all the 32-bit compatibility libraries on your 64-bit system.

Inspecting the Configuration File

When running on Desktop Linux, the configuration file is stored as a plain text file called mdm.config in the top level directory of your snapshot (at the same level as the top-level Makefile). It may be useful to view this file to see what was written to the config.

Starting with the 4.01L06 release, by default, the configuration file is compressed. Therefore, it is not possible to see the contents of the configuration file using a simple text editor.

To disable configuration file compression:

1. Type **make menuconfig**.
2. Load the profile you are using for Desktop Linux, for example 963268GW_DESKTOP.
3. Go to Special Feature selection and clear the Enable Compressed Config File option.
4. Rebuild the image for this change to take effect.

(You could also inspect the configuration file by typing **dumpmdm** on the CMS CLI or by accessing http://127.0.0.1:44480/dumpmdm.cmd.)

Writing Desktop Linux Friendly Code

This section is intended for developers who are working on code which already runs on Desktop Linux, or for developers who are interested in making their code Desktop Linux friendly.

When any userspace code is being built for Desktop Linux, the compiler symbol DESKTOP_LINUX is defined. So if you want to include or exclude code for Desktop Linux, you can use #ifdef or #ifndef to control compilation.

Generally, there are two issues that need to be addressed, file access and ioctl's.

File access: when running on the real hardware, the root of the file system is "/". But when running on Desktop Linux, the root of the file system is in targets/963268GW_DESKTOP/fs.install. To make it easier to generate the pathname on Desktop Linux, use cmsUtl_getRunTimePath(). This function will fill in your buffer with the appropriate path for Desktop Linux or real hardware. Grep through the source code in userspace/private/libs/libcms_core for example usage.

Ioctls: when running on Desktop Linux, the ioctl's in the CMS code cannot go into the kernel of the Desktop system. However, some CMS code depend on correct values being returned or some real action being taken. These critical ioctl's can be "wrapped" so that a fake ioctl function is called on Desktop Linux. The fake ioctl can return hard-coded values or simulate hardware devices, such as the flash for saving configuration files. See the code in userspace/public/libs/cms_boardctl for an example of fake ioctl's.

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Phone: 949-926-5000
Fax: 949-926-5203
E-mail: info@broadcom.com
Web: www.broadcom.com