Software User Manual

Brutus

Brutus™ Installation Guide

STB_Brutus-SWUM202-R

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

07/18/08

REVISION HISTORY

and

Revision	*Date*	*Change Description*
STB_Brutus-SWUM202-R	07/18/08	**Added:** • Information about Nexus to all relevant Set-topAPI procedures • “Reference Software Compile-Time Options (Nexus Platforms)” on page 22 **Removed:** • The sections “Build Server Configuration” and “Preparing Your Linux Build Server and Set-top”
STB_Brutus-SWUM201-R	06/20/08	**Updated:** • The section “Building for Support of Various Network Protocols” on page 14 **Added:** • “Live IP Channel Tuning Using UDP and RTP protocols” on page 14 • “NET DMA Support” on page 14 • “IP Playpump Support” on page 15 • “Accelerated Sockets Interface Using the NetAccel Driver” on page 15 • “IP-Based Streaming using NetAccel Driver” on page 16 • New compile-time options to Table 1 on page 20, Table 2 on page 20 and Table 3 on page 21 • A new section, “Power Management” on page 31 **Removed:** • The section “Brutus Compile Time Options”
STB_Brutus-SWUM200-R	03/12/08	Initial release.

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

TABLE OF CONTENTS

Introduction 1
Software Terms 1
Prerequisites 1
Building and Running Reference Software 3
Quick Instructions 3
The Set-Top Script 3
Configuring Linux Kernel Source 4
Untarring Source 4
Building 5
Formatting the Set-Top’s Hard Disk Drive for PVR 5
Installing and Running Brutus 6
Rebuilding and Running Again 7
Writing Brutus to Flash and Making It Start When Linux Boots 8
Installing Brutus from TFTP Server (No NFS) 9
Customizing the Brutus Installation 9
Configuration of Binaries in Release Bundle 11
Support for IP Set-Top and Cable/Satellite Are Combined in a Single Release 11
Release Bundle Binaries Are Kernel Mode 11
Kernel-Mode Linux Configuration 12
Running Big-Endian Linux 13
Building and Running IP STB Features 13
Building for Support of Various Network Protocols 14
Live IP Channel Tuning Using UDP and RTP protocols 14
NET DMA Support 14
IP Playpump Support 15
Accelerated Sockets Interface Using the NetAccel Driver 15
IP-Based Streaming using NetAccel Driver 16
Setting up an HTTP-Based Network Streamer 16
HTTP Streamer Theory of Operation 18
Setting Up an HTTP-Based Client Receiver 18
Network Recording: Recording to an IP Server w/ NetAccel 18
Streaming Over UDP/RTP Using the udpstreamer 19

Reference Software Compile-Time Options (Set-topAPI/Non-Nexus Platforms) 20
Reference Software Compile-Time Options (Nexus Platforms) 22
Brutus IP STB Customizations..24
Support for RTP and RTCP Protocols..24
Support for RTSP Protocol 25
Support for HTTP Channels 25
Support for SAP/SDP Protocol 25
Channel Map Syntax 26
Sample IP STB Channel Map Entries 27
Making Brutus IP STB Autoboot..27
Brutus IP STB Special Options..27
Simple IP Head End 28
DSL IP Video Head End and CPE 28
Simple Head End 29
Simple IP Video Server 29
Power Management...31
Using and Testing Power Management with Set-topAPI/Nexus and Brutus 31
Building...31
Running...31
Standby Process 31
Components 32

LIST OF FIGURES

Figure 1: Configuration of DSL IP Video Server Head End and CPE 28
Figure 2: Simple Head End Configuration 29

LIST OF TABLES

Table 1: Common Set-topAPI Options 20
Table 2: Various Set-topAPI Settings 20
Table 3: Various Brutus Settings 21
Table 4: Common Nexus Options 22
Table 5: Various Nexus Settings 22
Table 6: Various Brutus Settings 23
Table 7: IP Video Server Parameters 30

INTRODUCTION

This document comes as part of the Broadcom® Set-Top Reference Software Release. Refer to the Reference Software User Guide for a high-level overview of the Reference Software. Refer to the document *BroadcomReferencePlatformSetup.pdf* for instructions on how to set up the Build Machine and configure the reference platform with the CFE Boot loader and Kernel.

This installation guide contains specific instructions on how to configure, build, and install the Brutus™ application as part of the Reference Software. By building and running Brutus, you will be exercising the entire Reference Software stack. This document also describes how to build Brutus for Set-topAPI (non-Nexus) and Nexus platforms. The instructions are common for all the platforms, but Nexus-specific changes are explicitly noted throughout this document.

SOFTWARE TERMS

- **Bootloader**—The code that is first executed when the set-top powers up which is responsible for initial configuration of the hardware and loading the kernel.
- **Kernel**—The operating system which controls the set-top and allows the reference software drivers and applications to run.
- **Root file system**—On some operating systems (e.g., Linux®) it is customary to have at least a minimal disk layout for management of applications and libraries.
- **Toolchain**—The compiler and linker that runs on your build server and creates binaries that can run on the Set-Top.
- **Reference Software**—The source code delivered by Broadcom to control the features of the Broadcom Reference Platform.

PREREQUISITES

The reader should have the following:

- A Broadcom Reference Board running Linux Kernel
  - Refer to the *BroadcomReferencePlatformSetup.pdf* document for instructions.
- A Build Server and Network Setup Ready
  - Refer to the *BroadcomReferencePlatformSetup.pdf* document for instructions
- Knowledge of how to use `vi` (a Linux text editor) and be able to edit text on the set-top.
- Root access to the Linux build server and be able to switch between root and user mode when instructed.
- The capability of connecting the set-top and build server to the same Ethernet network.
- A dynamic host configuration protocol (DHCP) server running on the network that the set-top can use.
- The ability to run the `bash` shell.
  - If unsure, type **`asdf`** and press **Enter**; `bash: asdf: command not found` should appear. If it does not say `bash`: at the front, just run `bash` and try again.

Translate the following pseudo filenames to the actual names specified in our release notes:

- kernel_source.tgz
- toolchain_binary.tgz
- vmlinuz-xxxx = uclibc-based kernel where xxxx is the chip. Be aware that the exact kernel name may be different than just substituting xxxx.
- vmlinuz-initrd-xxxx = uclibc-based initrd kernel where xxxx is the chip. Be aware that the exact kernel name may be different than just substituting xxxx.
- uclibc root file system tarball binary (for installation on build server)
- uclibc root file system jffs2 image binary (for flashing on set-top)

BUILDING AND RUNNING REFERENCE SOFTWARE

QUICK INSTRUCTIONS

If you are already familiar with the process of building the reference software, you can refer to the following quick instructions. If you are not familiar, or if these instructions fail, we recommend reading this entire document.

```
# Do this on the Build Server:
bash
cd BSEAV/app/brutus/build
export LINUX=<<kernel_source>>
export PATH=$PATH:<<toolchain>>
export PLATFORM=xxxxx
export BCHP_VER=xx
make install

# Do this on the Set-Top:
mount y.y.y.y:?????/BSEAV/bin  /mnt/nfs
# For Platforms with kernel older than 2.6.18
mount -t ext2 /dev/hda1  /mnt/hd
# For Platforms with Kernel 2.6.18
mount -t ext2 /dev/sda1  /mnt/hd


# Running over NFS
cd /mnt/nfs
tar zxvf refsw-YYYYMMDD.PLATFORM-linux-uclibc.bin.tgz
settop install
settop brutus
```

These quick instructions are not meant to describe how to install Brutus to flash or a hard drive. Read the rest of this section for more detailed instructions.

If you want to run with the binary tarball, which was provided with the Reference Software Release, you can skip to the section “Formatting the Set-Top’s Hard Disk Drive for PVR” on page 5.

THE SET-TOP SCRIPT

The set-top script is used to initiate the installation and to run the Brutus and other set-top applications. It ensures that the environment is properly configured for running the install or running the specified application. It passes any extra parameters to the application listed as the first argument. For example:

```
settop brutus --help
```

runs the Brutus application and supplies the argument “--help” to list the startup options.

Configuring Linux Kernel Source

Part of the build process is building Linux drivers. Building drivers requires that the compiler have access to the Linux kernel header files. These header files must be the same ones used to build the actual Linux kernel you are running on the set-top.

The Makefile uses the Linux environment variable to override the location of the Linux kernel source. By default, it is `/opt/brcm/linux`. You can override it if needed.

Be aware that the Broadcom Linux release is configurable to a set of set-top platforms. When building your drivers, you must build using kernel source which is properly configured for your platform. Our build system assumes that $LINUX/include is not configured, so it does it for you. In order to configure the header files, you need write permission to the entire Linux directory. By default, the resulting header files are copied into the `BSEAV/linux/driver/build/$PLATFORM/linux_include` directory.

If you have already configured the Linux kernel source, you can define the LINUX_INC variable and bypass this step. Exporting `LINUX_INC=$LINUX/`include works, but only do this if the Linux kernel source is configured correctly. See the section earlier in this document for more details on building the Linux kernel.

If you are running on a shared Build Server, there are rare conditions when multiple people try to configure the same Linux kernel header files. So, you may not want to give everyone write permission to the LINUX directory. To facilitate this, the reference software build system has a convention where it looks for `$LINUX/include.mipsel-uclibc.$PLATFORM` directory. If this directory is found, the Makefile assumes that they are already configured and use them without any changes. For ease of use, the reference software has a Makefile that quickly configures a Linux kernel for a list of platforms using this convention, without having to build each kernel. See `BSEAV/linux/driver/build/makefile` and `BSEAV/linux/driver/build/common/Linux.make` for that technique.

Untarring Source

The reference software source is delivered as a tarball. You can untar it in place:

```
tar zxvf refsw-YYYYMMDD.PLATFORM.src.tgz
```

When you unpack your source, you will see three (or more) subdirectories that contain source code. Be aware that “magnum” is the name of the porting interface.

- BSEAV—Reference Software drivers, libraries, and applications
- Magnum—Porting interface for magnum platforms and some application utility code
- rockford—Magnum-based application and system code
- nexus—Nexus API and source code; available only with platforms (BCM7405, BCM7325, BCM7335, and beyond) that support the Nexus API.

These instructions assume you know where the BSEAV subdirectory is and can access it.

BUILDING

Before building, you must define the Platform environment variable.

Although your software release was for only one platform, many makefiles are common for all Broadcom platforms, so you must chose the correct Platform value that matches your release. If you pick the wrong Platform value, it starts building and then finds files missing. If this happens, just set the correct Platform value and try again. The official list of support Platform values is found in the source code in BSEAV/build/platforms.mak.

First, define the variable in your environment. Here's an example using 97401:

```
export PLATFORM=97401
```

Second, define the BCHP_VER (chip version) for the main chip on the board (i.e., BCM7401). Use all uppercase.

```
export BCHP_VER=C0
```

You should have already set your Linux and Path variables based on earlier sections of the document.

```
export LINUX=<<kernel source>>
export PATH=$PATH:<<toolchain>>
```

You may also need to define environment variables to enable compile-time options. For instance, some audio decoders are not built by default because they require special licensing. See the Build Options section for more information.

Now it's time to build:

```
cd BSEAV/app/brutus/build
make clean
make install
```

By using the Brutus application's Makefile, you will build not only Brutus, but also all needed drivers and libraries. Building everything can take over 15 minutes. If you get a large success message at the end, then everything has been installed into the BSEAV/bin location.

FORMATTING THE SET-TOP'S HARD DISK DRIVE FOR PVR

In most cases, the hard disk drive on the set-top is only needed for the PVR. If you are using a hard drive-based root file system, then you should ignore these directions because the initrd process creates four partitions, then you should use /dev/hda4 for PVR.

If you are using a Flash, NFS, or initrd root file system, the following directions show you how to manually configure your hard drive for PVR. Broadcom recommends that you partition the hard drive with one partition for the entire disk.

Here are some simple instructions:

```
fdisk /dev/hda  (Use /dev/sda for 2.6.18 and later version of the kernel.)
Command: p
Command: d 1
Command: d 2
# Delete whatever partitions happen to be there

Command: n
# Select p, then 1. Then accept all defaults in order to make a
# single maximum-sized partition.

Command: w

# See note below on ext2 usage
mke2fs -N 4000 /dev/hda1 (Use /dev/sda1 for 2.6.18 or later version of the kernel.)

# The following mount command must be performed every
# time the set-top boots.
mount -t ext2 /dev/hda1 /mnt/hd
mount -t ext2 /dev/sda1 /mnt/hd (For 2.6.18 or later version of the kernel.)
```

Broadcom recommends using the ext2 file system with a limited number of inodes. Using ext3 (which is ext2 + journaling) has poor performance with PVR. By limiting the number of inodes in the file system, you decrease FSCK time to make it very comparable.

Note: Make sure to use the explicit “-t ext2” just in case the drive was formatted as ext3 and your fstab lists it as ext3. If you use ext3, Direct I/O will not work and PVR performance suffers.

Note: Be aware that if you halted or rebooted while mounted as ext3, you will need to mount as ext3 one more time in order to clean up, then unmount and mount again with “-t ext2.”

The Brutus “set-top install” procedure looks for /mnt/hd and creates a videos directory for PVR. When it is time to make your own root file system image which autoboots Brutus, you will add the command to mount this partition.

See “Writing Brutus to Flash and Making It Start When Linux Boots” on page 8 for instructions on how to modify rc.user to automount the disk.

Installing and Running Brutus

Now you are ready to run Brutus. Because the flash file system is read-only, the easiest way to do this is to mount your Build Server using NFS and run Brutus across the network. You should have already made your source directory mountable in the previous steps. If you can not, you have to move the contents of BSEAV/bin to a directory that is mountable.

For uclibc, you may want to telnet into the box before running Brutus. You can use the console, but Ctrl-C does not work. If you telnet in and then run the following commands, you can use Ctrl-C.

If you do not have NFS, you can not do this step.You will need to install to flash using TFTP. See "Installing Brutus from TFTP Server (No NFS)" on page 9.

Follow these steps from the set-top, assuming your mount point on the set-top is /mnt/nfs. Replace `?????/BSEAV/bin` with whatever mount point you chose on your build server.

```
mount y.y.y.y:??????/BSEAV/bin /mnt/nfs

# Unpack and run Brutus over NFS
cd /mnt/nfs
tar zxvf refsw-YYYYMMDD.PLATFORM-linux-uclibc.bin.tgz
settop install
settop brutus
```

In this method, you are unpacking the tarball across NFS. The "set-top install" step configures device nodes and creates some directories and symlinks to /mnt/hd. Using "set-top brutus" load your drivers, verify that you are linking to the right shared libraries, and start the application—Brutus should start.

REBUILDING AND RUNNING AGAIN

Once you have gone through the install instructions once, you can rebuild and rerun Brutus with a smaller set of steps:

```
# On the build server
cd BSEAV/app/brutus/build
export PLATFORM=xxxxx
make

# On the Set-Top:
mount y.y.y.y:??????/BSEAV/bin /mnt/nfs
mount -t ext2 /dev/hda1  /mnt/hd

# Run Brutus over NFS
cd /mnt/nfs
settop brutus
```

By running "make" instead of "make install", you will copy the binaries into BSEAV/bin but not create a new installation tarball. On the set-top, you only need to run "set-top brutus" instead of redoing the installation procedure.

WRITING BRUTUS TO FLASH AND MAKING IT START WHEN LINUX BOOTS

These instructions assume you have already installed the file system to flash and have successfully booted Linux.

The recommended method of installing the application to flash is to mount the flash file system in Linux and copy the Brutus binary via NFS. By default, you should have mounted the flash file system read-only, so the first thing you must do is remount the file system with write permission. Writing to flash through Linux can take about 5 minutes.

The following instructions are examples only. Things that may vary include your installation location, hard drive partitions, mount point, etc.

```
# Remount the flash filesystem with write privileges
mount –o remount,rw /

# Mount NFS to load the Brutus binary
mount yyyy:??????/BSEAV/bin /mnt/nfs

# Mounting the hard-drive is required before “settop install”
mount –t ext2 /dev/hda1 /mnt/hd (Use /dev/sda1 for 2.6.18 or later version of the kernel.)

# Unpack the Brutus tarball into /home/brutus, which writes
# to flash.
mkdir /home/brutus
cd /home/brutus
tar zxvf /mnt/nfs/refsw-YYYYMMDD.PLATFORM-linux-uclibc.bin.tgz
settop install

# Make it autoboot by creating this file:
cat >/root/rc.user
mount /dev/hda1 /mnt/hd
cd /home/brutus
settop brutus &

# Press Ctrl-D to save the file

mount –o remount,ro /
sync;sync

# You can now start Brutus
settop brutus
```

INSTALLING BRUTUS FROM TFTP SERVER (NO NFS)

If you are running a Windows-only setup, then you do not have NFS available. You can still load Brutus into flash using TFTP, such as the PumpKIN TFTP server.

This process requires that you use the flash, hard drive, or NFS root file system; it does not work with the initrd kernel. The following directions assume you have already flashed the kernel and are using a read-only flashed root file system.

```
# Boot linux and login as root

# Create a ramdisk and load the tarball into it
mkdir /tmp/mnt
mount -nt tmpfs -o size=8192k,mode=777 /dev/null /tmp/mnt
cd /tmp/mnt
tftp -g -r refsw-YYYYMMDD.PLATFORM-linux-uclibc.bin.tgz y.y.y.y

# Untar from ramdisk into flash
mount –o remount,rw /
mkdir /home/brutus
cd /home/brutus
tar zxvf /tmp/mnt/refsw-YYYYMMDD.PLATFORM-linux-uclibc.bin.tgz
settop install
mount –o remount,ro /
sync;sync

# You can now start Brutus
settop brutus
```

Note that y.y.y.y is the IP address of your TFTP server. Writing Brutus to flash might take several minutes. Each file is printed separately.

CUSTOMIZING THE BRUTUS INSTALLATION

After installing on the set-top, you will need to customize your Brutus configuration.

1. Edit channels.txt and programguide.txt for your head end. See the Channel Map section in the *Brutus Usage Guide* (STB-Brutus-SWUM30x-R) for more help. Edit `brutus.cfg` with any special options. See the Configuration section of the *Brutus Usage Guide* for more help.
2. Make Brutus automatically start when the set-top reboots. This process is nonstandard because of our flash file system requirements. The flash file system should be mounted read-only, but Linux requires write access to `/etc` and `/var`. So, the default file system will mount `/etc` and `/var` with a RAM disk using an image file. Because of this, you can not modify the `/etc` tree in order to make your application autostart (unless you want to rebuild a custom file system). To accommodate this, the default `/etc/init.d/rcS` file looks for a file called `/root/rc.user`. If it exists, it will call it.

   Therefore, to make Brutus autostart create a file called `/root/rc.user,` which looks similar to the following (specifics may vary):

   `mount /dev/hda1 /mnt/hd` (Use `/dev/sda1` for 2.6.18 or later version of the kernel.)
   ```
   cd /home/brutus
   settop brutus &
   ```

The trailing `&` in the last line is critical to allow the rcS file to complete and a login prompt to appear on the console. You might need the following to make the init process continue and give you a login prompt:

```
settop brutus </dev/null >/dev/null 2>/dev/null &
```

Once you've made Brutus autostart, if you do not get a root prompt, you will need to boot with an initrd kernel and mount flash to modify the `rc.user` file.

```
#Reboot and Ctrl-C if your set-top current autoboots Linux
#Boot the initrd kernel from TFTP server

Login: root

mount /dev/mtdflash0 /mnt
cd /mnt/root
mv rc.user rc.user.bak
cd /
umount /dev/mtdflash0
# reboot
```

You may want to configure a set-top that automatically starts Brutus by default, yet also has an option to abort that start. This can be accomplished with a simple uclinux script. Here's a sample:

```
mount /dev/hda1 /mnt/hd (Use /dev/sda1 for 2.6.18 or later version of the kernel.)
echo "Press ENTER in 5 seconds to abort Brutus"
REPLY=nothing
read -t 5
if [ x$REPLY == xnothing ]; then
 cd /home/brutus
 settop brutus &
fi
```

CONFIGURATION OF BINARIES IN RELEASE BUNDLE

SUPPORT FOR IP SET-TOP AND CABLE/SATELLITE ARE COMBINED IN A SINGLE RELEASE

Previously, Broadcom had two separate releases, one for Cable/Satellite and another for IP set-top. Now, all reference software releases have all three functionalities: Cable, Satellite, and IP which are all merged into a single release. The IP set-top functionality is enabled by default. The following build options are defaulted on in order to achieve this:

```
PLAYBACK_IP_SUPPORT=y
B_HAS_PLAYPUMP_IP=y
LIVEMEDIA_SUPPORT=y
```

The Brutus application and binaries provided in the release bundle are built with the above options turned on. If you are not using the IP set-top features, the only impact you may observe is that the video and audio CDBs and ITBs are doubled in size. This does not impact behavior.

If you are not interested in the IP set-top features and want to build the Brutus application and the libraries without IP set-top support, do the following before building:

```
export PLAYBACK_IP_SUPPORT=n
```

RELEASE BUNDLE BINARIES ARE KERNEL MODE

Previously, the Brutus application and libraries provided in the release were built in the user mode configuration. Now, the Brutus application and the libraries are built in the kernel mode configuration. The kernel mode is enabled by this build option:

`KERNELMODE_SETTOPAPI=y`	(Non Nexus platforms)
`KERNELMODE=y`	(Nexus platforms)

Though the binaries provided in the release are built in kernel mode, the default build option for Brutus in the Makefiles is still user mode. If you are not interested in kernel mode binaries and want to build Brutus application and libraries in user mode, simply rebuild Brutus with the default option.

KERNEL-MODE LINUX CONFIGURATION

On all Magnum-based platforms, Broadcom supports Magnum and Set-topAPI/Nexus running in either Linux user mode or kernel mode.

When running in user mode, API calls from the application are handled like normal function calls. When running in kernel mode, API calls from the application are marshaled across the kernel/user boundary using the Proxy Set-topAPI/Nexus public API.

The Proxy API translates all Set-topAPI/Nexus public API calls into Linux I/O controls (IOCTLs). User and kernel memory mapping is handled automatically. The calling application does not see anything differently; it looks exactly like a C API. The full Set-topAPI/Nexus public API implementation is then compiled into the driver, running in kernel mode. This means that there are two implementations of the Set-topAPI/Nexus public APIs running on the system: the full implementation in kernel mode and a proxy implementation in user mode.

The benefit of a kernel mode implementation is that interrupts are handled at interrupt time and, therefore, have less latency. The detriment of a kernel mode implementation is that every Set-topAPI/Nexus public API call must be marshaled across the kernel/user boundary.

The Proxy implementation is automatically generated by Perl scripts. They parse the public header files and create the IOCTL's, user mode code, and kernel mode code.

See `BSEAV/api/build/proxy/bapi_build.sh` and the `.pl` and `.pm` Perl scripts in the same directory. For platforms with Nexus, please see `nexus/build/os/linuxkernel/ module_rules.pl`.

Building a kernel-mode version of the Set-topAPI example:

```
cd BSEAV/app/brutus/build
export PLATFORM=97401
export BCHP_VER=C0
export KERNELMODE_SETTOPAPI=y
make install
```

For Nexus platforms, replace
`export KERNELMODE_SETTOPAPI=y`
with
`export KERNELMODE=y`
to build Nexus in Kernel Mode.

After building, you find drivers, libraries, and an application in the BSEAV/bin directory. The only difference is that the driver is much larger, and the libraries are much smaller.

Running Big-Endian Linux

All the directions listed above assume little-endian Linux. To switch to big-endian, you only need to make a few changes:

1. Switch the endianess jumper on the board from little- to big-endian.
2. Flash the big-endian CFE bootloader using Broadband Studio.

   You cannot flash big-endian CFE using little-endian CFE. You must already have the board in big-endian mode, which prevents you from running little-endian CFE.
3. Install and boot big-endian Linux. Big-endian Linux has separate binaries, but not separate source.
4. Configure kernel source on the build server for big-endian Linux.

   When you copy the defconfig to .config, be sure to select the big-endian variety.
5. Set the ARCH variable when building Brutus.

ARCH defaults to mipsel-uclibc, where “el” stands for “endian little.” You should set:

```
export PLATFORM=xxxxxx
export BCHP_VER=xx
export ARCH=mips-uclibc
make clean
make install
```

Be aware that PVR .nav files are in host endianess. You must rerecord your streams or byteswap little-endian files for big-endian systems.

Building and Running IP STB Features

Rudimentary IP STB functionality is part of the standard Brutus Reference Software build, and accessing these features requires no special build options. This includes support for the following functions:

- IP STB front-end (handles IP network interface)
- Programmable IP channel map
- Record to HDD from IP network
- IP Multicast support (including IGMP v2 and v3 support)
- Hardware filtering of multicast IP addresses via perfect match registers in the Ethernet MAC
- IP Network Buffer Management: STB has optimal buffering to absorb network jitter and ensure smooth delivery of data to the decoders.

Additional capability is available via build options invoked as part of the make command for Brutus (or alternatively via the setting of environment variables, prior to the build). For an accurate listing of the capabilities of your reference platform, check the release notes (current and previous) contained in the release package.

Building for Support of Various Network Protocols

By default, released binaries are built to support RTP, RTCP, RTSP, SAP, and HTTP Protocols. This functionality is controlled by the following build flags:

- PLAYBACK_IP_SUPPORT is the global build flag that enables IP STB support in the reference software.
- The NETACCEL_SUPPORT flag enables the Accelerated socket support. This flag is automatically defined if PLAYBACK_IP_SUPPORT is defined. You can turn off this flag if you want to disable Accelerated sockets and instead use the standard Linux sockets to receive IP data.
- The LIVEMEDIA_SUPPORT flag enables the use of Live Media library to support RTP, RTCP, and RTSP features. This flag is automatically defined if PLAYBACK_IP_SUPPORT is defined.
- The B_HAS_PLAYPUMP_IP flag enables use of the legacy Net DMA interface to receiving IP data. This feature is now being deprecated in favor of accelerated sockets and this flag is no longer being enabled by default when PLAYBACK_IP_SUPPORT is on.

The BCM97401, BCM97403, and BCM97405 platforms by default define the PLAYBACK_IP_SUPPORT. For the BCM 97400 platform, it has to be manually defined.

Live IP Channel Tuning Using UDP and RTP protocols

In order to receive Live IP channels on an IP STB Client using Brutus, you must include an IP_UDP, IP_RTP channel entry in channels.txt. PROGRAM PIDS are optional, if there is PAT and PMT in the mpg. Here is an example:

Channels.txt entry example:

```
IP_UDP          0  224.1.1.10 1234
IP_RTP          0  224.1.1.10 1234
PROGRAM PIDS video=0x11 video_type=MPEG2 audio=0x14 audio_type=AAC
```

Please refer to "“Brutus IP STB Customizations” on page 24" for further details on supported protocols and traditional IP STB operations.

NET DMA Support

Recent platforms employing the 2.6.12-2.1 kernel (and beyond) support an enhanced mode of operation for IP STB. This mode is known as “NET DMA.”It intercepts IP traffic at the Ethernet driver level and provides these A/V packets to the application running in either user or kernel mode. NetDMA defines a proprietary interface to allow applications to receive data.

Going forward, however, the NetDMA proprietary interface is being deprecated in favor of the Accelerated Sockets Interface exposed by the NetAccel driver.

By default, released binaries are built with NET DMA support.

Further performance improvement is achieved through use of the Settop/Nexus API running in kernel-mode. This is done by invoking the appropriate extra option in the build, as follows:

`make KERNELMODE_SETTOPAPI=y installinstall` (Set-topAPI platforms)
or
`make KERNELMODE=y install` (Nexus platforms)

IP Playpump Support

IP Playpump is a module exposed by the Settop/Nexus public APIs that provides a mechanism to directly receive the incoming UDP or RTP packets (via the proprietary NET DMA interface). It supports basic RTP processing (packet reordering, loss detection) and then routes packets containing the compressed audio/video directly to the playback DMA hardware.

This mode is primarily useful if the an application can have a kernel module to directly read the data from the Net DMA interface, do basic processing, and feed it to the playback HW. However, this module is also being deprecated and being replaced with the IP support provided via the Playback IP module (which uses standard or accelerated sockets to receive the incoming IP data). The change aligns the IP reference implementation more closely with the customer applications in which user sockets interface to receive the incoming UDP/RTP traffic in user space and then feed it to the Playback hardware.

This option is supported by the build option B_HAS_PLAYPUMP_IP. **However, it is no longer defined by default and needs to be explicitly set in the environment.**

Accelerated Sockets Interface Using the NetAccel Driver

In the recent releases NET DMA functionality (filter engine to do flow classification) has been further enhanced to include TCP and UDP offload engines and has been renamed "NetAccel Driver." This driver supports the complete BSD Sockets Interface to allow applications to send and receive data using these accelerated offload engines. Applications need to specify new socket types (SOCK_BRCM_STREAM for TCP and SOCK_BRCM_DGRAM for UDP) to utilize this accelerated sockets interface. Please see the sample applications provided in the BSEAV/lib/netaccel/app directory for examples on using this interface for receiving & sending data.

The combination of NetAccel support and kernel-mode Set-topAPI/Nexus is the recommended mode of operation, particularly in higher data rate scenarios.

Note: The NetAccel driver is not provided in the source code form in the reference software bundle. Instead, a Linux kernel driver, bcmnetaccel.ko, is provided in the binary form. This driver is compiled with the included tool chain in BE/LE, UNI, and SMP combinations.

Please contact your customer support teams if you need this driver compiled for a different tool chain.

IP-BASED STREAMING USING NETACCEL DRIVER

As mentioned above, recent releases of reference software bundle applications, libraries, and kernel mode drivers that allow streaming of content from a local hard drive to the network clients. All of these components are provided as “binary-only” packages. The following list provides a brief description of the included binaries:

- **bcmnetaccel.ko:** This is a Linux kernel mode driver that implements the IP, UDP, and TCP offload engine. This driver allows user applications to send or receive data via this accelerated path using the BSD socket interface. This enables existing applications to seamlessly leverage the high performance offload engine capabilities provided by this module. Please contact your customer support team to further details on this interface and driver.
- **libnetaccel.so:** This is a user-space library that provides a simple high-level API to stream content from a local hard drive using HTTP, UDP, or RTP protocols. In addition, this library supports range of “server side trick modes.” The library also supports the capability to do Network/Remote Record on the server and playback the time-shifted content. This library utilizes the Accelerated Socket interface of the NetAccel driver to send and receive content over network.
- **httpstreamer:** A simple application that links with the libnetaccel.so library to provide a complete Network streaming and recording application.
- **udpstreamer:** A simple application that links with the libnetaccel.so library to provide a complete UDP and RTP streaming application.
- **http_test_server and http_test_client:** These are test applications that demonstrate the use of the Accelerated socket interface to utilize the NetAccel driver. The source code of these test applications can be provided upon request.

Setting up an HTTP-Based Network Streamer

The Httpstreamer application utilizes the bcmnetaccel.ko driver via the libnetaccel.ko library to stream out a recorded file on your hard drive out through Ethernet. It can be used with IP STB clients or a Windows- or Linux-based Client PC running VLC to receive the stream. TCP streaming out currently is only supported with a Broadcom-based IP STB as a client and PCs running VLC multimedia program.

The binaries to turn the set-top box into a IP Video Streaming Server are located in the reference software tree as binary-only modules. They are built with the toolchain and kernel that is provided in the reference release. These binaries are located in BSEAV/lib/netaccel/bin/<Chip Number>/<kernel version>/<smp?>. The binaries in this directory as associated with the kernel and software release they come bundled in. Again, please contact customer support if you need custom binaries for your environment.

Here is a quick summary of the steps required to setup an IP Streamer Server.

- Boot the set-top with the init-rd kernel image and install root file system on the hard drive using stbutil option 3.
- Once the root file system is installed on the hard drive, you will need to boot the set-top box with more kernel memory. Allocate at least 64 MB to the Linux kernel for the TCP streamer to work correctly. Use the `mem=64M` option at the CFE boot command in order to increase the kernel memory. The following is an example CFE boot string for booting a 2.6.18.x kernel from Flash (please use `/dev/hda` for 2.6.12.x kernel versions).

  ```
  boot -z -elf flash0.kernel: 'root=/dev/sda1 rw mem=64M'
  ```
- Copy the reference software binaries (which include streamer applications and drivers) onto the hard drive (say in /opt/demo directory) using tftp get or by NFS mounting from your build server.

- Copy the content that you would like to stream out. There are three ways to get this content:
  - Use the Brutus application to record content from cable/satellite/off-air channels on the local hard drive (follow instructions in the previous section for local hard drive setup for PVR mode). Brutus typically records data in the /data/videos directory. Streamer will recognize any MPEG-2 and MPEG-4 content in this file and can stream it to other Broadcom clients.

Note: PC clients cannot receive this content, as Brutus doesn't capture the PSI information while recording. However, Brutus clients can receive this content as streamer sends PSI information in HTTP tags during HTTP response packet.

  - Use the streamer to record content over the network. Here Brutus on the client machine tunes to Cable/Satellite/Off-Air channels and records the content over the network via the streamer on the server. The Brutus recording page has an option to enable network record. "Network Recording: Recording to an IP Server w/ NetAccel" on page 18 details the additional configuration needed to enable network record.
  - Tftp or copy over NFS pre-recorded streams on the local hard drive. If you do not have the indexer files (nav files) for these streams, the http streamer will generate them when it streams out the content for the first time. When a file is requested by a client, these files will be created if they are not present. This may take several minutes and the remote HTTP client session will start after 10 seconds or so have passed. **Please do not change channels during this time**. The indexes are created at that time, and it is a one-time creation process.

Note: If any of the copied streams are MPEG-4/H.264/AVC-encoded, they will need to have AVC in their file name. The current streamer does not have logic to determine the codec type and assumes an MPEG-2 codec by default.

- Now that you have copied the content on hard drive, start streamer by typing httpstreamer on the command line (you will need to export the LD_LIBRARY_PATH=. in you environment). By default, streamer starts 3 streaming threads and 0 recording threads. Please use the –r option to start any recording threads. You will need one streaming & one record thread per client for each network record client. Also, current stream supports streaming out & network recording from up to three clients. Also, per client thread additional threads are internally started by this application for sending the channel map requests and creating the nav indices. Here is the quick summary of commands to start the streamer:

```
#export LD_LIBRARY_PATH=$PWD
#./settop install <this only needs to be done once>
#./settop  <this installs all the necessary drivers>
#./httpstreamer –s 3 –r 3 & <for starting 3 streamers & 3 recorders in background>
#./Brutus -skin blue & <optionally start Brutus for local decodes>
```

- httpstreamer currently supports following options:

```
# ./httpstreamer -h
Usage: httpserver [-s N1 -r N2 -c N3]
    N1 = number of streamers (default 3)
    N2 = number of recorders (default 0)
    N3 = cpu affinity mask 1: CPU0, 2: CPU1, 3: CPU0 || CPU1 (default is 1)
```

The current streamer only supports MPEG-2 and MPEG-4 (H.264) streams encapsulated in the MPEG22 Transport format.

HTTP Streamer Theory of Operation

The httpstreamer application uses the TCP protocol to stream and is based on the “Client Pull Model.” This is similar to how a client pulls data from a local hard drive. Instead, client sends an HTTP request to the server and the server starts streaming to a client. The client throttles the server as its video CDB starts getting full by delaying reading data from the network socket. This causes the client TCP to lower the TCP window, which throttles the sender.

The HTTP streamer also supports “Server-based Trick modes.” In this model, the client sends the new play speed and current location to the server using the DLNA’s PlaySpeed string. The client closes existing TCP connection and starts a new connection with new trick play speed. The server uses an index file to seek to this location and starts streaming the requested speed. Currently, streamer only supports I-frame based trick modes for both FF & REW directions.

httpstreamer uses the TCP port number 5000 to receive HTTP requests.

Setting Up an HTTP-Based Client Receiver

Your Brutus application bundle includes the NetAccel driver. Please make sure that your reference binary includes a file in the BSEAV/bin directory named bcmnetaccel. You must insmod this file after you insmod the bcmdriver.ko or bcm<chip>.ko. This module contains processing functions for use with a Broadcom IP server.

Once you have the HTTP streamer running on a server Settop (as described in the previous section), there are two ways you can setup your client to receive the HTTP based channels from the httpstreamer server.

- Add static IP_HTTP channel entry in the channel map. Here is an example entry to receive Portugal.mpg content using HTTP channel from a streamer at IP address 192.168.1.100 port 5000:
  ```
  IP_HTTP 0 <Server’s ip address> 5000 /filename.mpg
  IP_HTTP 0 192.168.1.100 5000 /portugal.mpg
  ```
- Or you can dynamically discover the content available on the streaming server. For this option to work, you must specify the streamer server’s IP address in the Brutus.cfg file (the older ipstreamer_server.cfg is no longer being used). Brutus will query the IP streamer server set-top box for content it has and fill in the dynamic channel map. Here is an example entry in the Brutus.cfg file:
  ```
  IPSERVER_IPV4="ip address of the streaming server"
  IPSERVER_IPV4=192.168.1.100
  ```
- You can then use the Brutus GUI option to discover the content. If you run Brutus with the `–skin blue` option, you will have an **Online Content** button at the bottom of the GUI. This Icon takes you to a screen that contains a combined channel listing of channels.txt and the IP Streamer Server settop box.

Network Recording: Recording to an IP Server w/ NetAccel

As briefly mentioned in the previous section, the http streamer provides the Network/Remote Record capability. The client can tune to Cable/Satellite/Off-Air channel and record content over the network on the server and do a time-shifted playback of this content.

To use this option, run the httpstreamer with –r option on the server. You can simultaneously record up to three client sessions and play them back. **You will also need to increase the kernel memory from 64 MB to 128 MB for supporting the maximum configuration of three records and three playbacks.**

On the client, you will need to configure server details. There are three variables that need to be set in Brutus.cfg file.

```
IPSERVER_IPV4=”ip_address of server”
IPSERVER_REC_PORT=6000
IPSERVER_CLIENT_ID="A"
```

These variables need to be set in Brutus.cfg prior to starting Brutus. **Once Brutus starts, go to the record screen and check the “Network Record” checkbox.** The file will begin to be streamed out to the IP Server for recording. You can stop the recording at any time.

To playback the recorded file from the server, you will need to re-scan the channels on the Brutus (use the scan option in the Administration GUI). This re-scanning add the recorded file as a new channel to the channel map. You can tune to that channel in order to playback the recorded file.

Streaming Over UDP/RTP Using the udpstreamer

In addition to streaming over HTTP, another application, udpstreamer, is provided to allow streaming over UDP or RTP protocols. This streamer uses a Push model to pace out a streamed file. Currently, it only supports the streaming of MPEG-2 TS encapsulated files.

Please setup the content to be streamed on the server as described in the previous section. Supported options for the udpstreamer are:

```
# ./udpstreamer -h
Usage: udpstreamer -d <dst-ip> -p <port> -f <content-file> [-h -r <rate> -t <proto> -l]
Options are:
 -d <dst-ip>    # dst IP address to use
 -p <port>      # dst port to use
 -f <file>      # url to stream; /data/videos prepended
 -r <rate>      # rate (default: auto computes stream bitrate)
 -s <speed>     # forced trick play speed (default: 1, normal play)
 -t <proto>     # protocol (default = UDP = 0; RTP = 1; RTP+FEC = 2)
 -a <affinity>  # affinity (default = CPU0 = 1; CPU1 = 2; ANY = 3)
 -l             # keep resending the file (default: plays only once)
 -c <count>     # simulate random packet drops: drop every count pkt (default: no)
 -h             # prints udpstreamer usage
```

Setting Up VLC on a PC to Receive UDP/RTP/HTTP Streams

To Use VLC on a PC to receive content from various type of streams, configure it as follows:

- VLC →File →Open Network Stream: Select the **HTTP** button, and type in the URL as <Server’s IP Addr>:5000/portugalMpeg2HD.mpg
- For receiving UDP streams, select the UDP tab.

REFERENCE SOFTWARE COMPILE-TIME OPTIONS (SET-TOPAPI/NON-NEXUS PLATFORMS)

The Set-topAPI and Brutus support various compile time options. Any required or commonly used options are documented in the release notes. However, there are many minor or optional settings which change frequently. The only way to know the complete list of compile time options is to grab the makefiles.

Unless noted, all compile time options are simply on or off. We usually document this as export OPTION=y. Some are documented throughout this guide. This list is a catalog of all options which are likely to be of interest.

Table 1: Common Set-topAPI Options

Option	*Purpose*
PLATFORM	Selects the board to build for. See release notes.
BCHP_VER	Selects the chip revision of the main chip to build for. See release notes.
LINUX	Point to Linux kernel source for building drivers. Defaults to /opt/brcm/linux.
DEBUG	Set to yes or no. If yes, DBG interface is enabled. If no, DBG interface is compiled out. Defaults to yes.
ARCH	Set the compiler. Defaults to mipsel-uclibc. See BSEAV/api/build/tools.mak for options.
SMP	Set to y to build drivers for SMP Linux. Defaults to n.

Table 2: Various Set-topAPI Settings

Option	*Purpose*
ASF_SUPPORT	Compile ASF parser support. Code is only available for verified ASF licensees.
AVI_SUPPORT	AVI file format support. Needed for DivX support.
SUBTITLE_SUPPORT	Support for DivX subtitles.
RAP_WMA_SUPPORT	Compile WMA audio codec support. Code is only available for verified WMA licensees.
RAP_DDP_SUPPORT	Compile DDP audio codec support. Code is only available for verified DDP licensees.
RAP_AC3_SUPPORT	Compile AC3 audio codec support. Code is only available for verified AC3 licensees.
BHDM_DVO_ENABLE_ SUPPORT	Enable DVO output in HDMI porting interface.
STATIC_SETTOPAPI	Build the Set-topAPI as a static library only.
BPROFILE_SUPPORT	Set-topAPI's profiling engine.
PLAYBACK_IP_SUPPORT	Enable IP support in Brutus and Set-topAPI.
MACROVISION_SUPPORT	Enable Macrovision™ in VDC. Defaults off. Code is only available for verified Macrovision licensees.
BCRYPTO_xxx_SUPPORT	Enable various security modules. See BSEAV/api/build/magnum for options.

Table 3: Various Brutus Settings

Option	***Purpose***
AUDIO_SUPPORT	Enable MP3 support in Brutus. Defaults on.
PVR_SUPPORT	Enable Playback/Record support in Brutus. Defaults on.
STATIC_BRUTUS	Build and link Brutus as a static binary (no shared libraries, including no libstdc++ or libc).
KERNELMODE_SETTOPAPI	Build the Set-topAPI in the kernel mode configuration. Defaults to user mode.
PLAYBACK_IP_SUPPORT	Enables IP support in Brutus and Set-topAPI.
LIVEMEDIA_SUPPORT	Enables IP streaming support in Brutus.
DRM_SUPPORT	Compile Microsoft Digital Rights Management (DRM) 10 support. Code is only available for DRM licensees.
DIVX_DRM_SUPPORT	Compile DivX Digital Rights Management (DRM) support. Code is only available for DivX licensees.
POWERSTANDBY_SUPPORT	Set to n to disable low-power standby supports. Defaults to y on supported platforms.
DSG_SUPPORT	Enable DOCSIS Set-top Gateway (DSG) feature in Brutus.
NETACCEL_SUPPORT	Enable the usage of NetAccel Module for receiving IP_UDP, IP_RTP, & IP_HTTP channels

REFERENCE SOFTWARE COMPILE-TIME OPTIONS (NEXUS PLATFORMS)

The Broadcom reference support various compile time options. Any required or commonly used options are documented in the release notes. However, there are many minor or optional settings which change frequently. The only way to know the complete list of compile time options is to grab the makefiles.

Unless noted, all compile-time options are simply on or off. Broadcom usually documents this as export `OPTION=y`. Some are documented throughout this guide. This list is a catalog of all options which are likely to be of interest.

Table 4: Common Nexus Options

Option	***Purpose***
PLATFORM	Selects the board to build for. See release notes.
BCHP_VER	Selects the chip revision of the main chip to build for. See release notes.
LINUX	Point to Linux kernel source for building drivers. Defaults to /opt/brcm/linux.
DEBUG	Set to yes or no. If yes, DBG interface is enabled. If no, DBG interface is compiled out. Defaults to yes.
ARCH	Set the compiler. Defaults to mipsel-uclibc. See BSEAV/api/build/tools.mak for options.
SMP	Set to y to build drivers for SMP Linux. Defaults to n.

Table 5: Various Nexus Settings

Option	***Purpose***
MEDIA_ASF_SUPPORT	Compile ASF parser support. Code is only available for verified ASF licensees.
MEDIA_AVI_SUPPORT	AVI file format support. Needed for DivX support.
SUBTITLE_SUPPORT	Support for DivX subtitles.
RAP_AC3_SUPPORT	Compile AC3 audio codec support. Code is only available for verified AC3 licensees.
RAP_DDP_SUPPORT	Compile AC3+ audio codec support. Code is only available for verified DDP licensees.
RAP_MPEG_SUPPORT	Compile MPEG audio codec support.
RAP_WMA_SUPPORT	Compile WMA audio codec support. Code is only available for verified WMA licensees.
RAP_WMAPRO_SUPPORT	Compile WMA Pro audio codec support. Code is only available for verified WMA licensees.
RAP_AACSBR_SUPPORT	Compile AAC audio codec support.
RAP_SRC_SUPPORT	Compile MPEG LSF/QSF audio codec support
RAP_DDP_TO_AC3_SUPPORT	Compile Dolby Digital Plus to Dolby Digital conversion support.
HDMI_DDP_PASSTHROUGH	Enable DDP pass-through on HDMI output. By default DDP is converted to Ac3 on HDMI port.
PLAYBACK_IP_SUPPORT	Enable IP support in Brutus and Nexus.
MACROVISION_SUPPORT	Enable Macrovision in VDC. Defaults off. Code is only available for verified Macrovision licensees.

Table 5: Various Nexus Settings (Cont.)

Option	***Purpose***
BCRYPTO_xxx_SUPPORT	Enable various security modules. See BSEAV/api/build/magnum for options.

Table 6: Various Brutus Settings

Option	***Purpose***
AUDIO_SUPPORT	Enable MP3 support in Brutus. Defaults on.
PVR_SUPPORT	Enable Playback/Record support in Brutus. Defaults on.
STATIC_BRUTUS	Build and link Brutus as a static binary (no shared libraries, including no libstdc++ or libc).
KERNELMODE	Build the Nexus API in the kernel mode configuration. Defaults to user mode.
PLAYBACK_IP_SUPPORT	Enables IP support in Brutus and Nexus.
LIVEMEDIA_SUPPORT	Enables IP streaming support in Brutus.
MSDRM_PD_SUPPORT	Compile Microsoft Digital Rights Management (DRM) 10 support for Portable Devices. Code is only available for DRM licensees.
MSDRM_ND_SUPPORT	Compile Microsoft Digital Rights Management (DRM) 10 support for Network Devices. Code is only available for DRM licensees.
DIVX_DRM_SUPPORT	Compile DivX Digital Rights Management (DRM) support. Code is only available for DivX licensees.
POWERSTANDBY_SUPPORT	Set to n to disable low-power standby supports. Defaults to y on supported platforms.
NETACCEL_SUPPORT	Enable the usage of NetAccel Module for receiving IP_UDP, IP_RTP, & IP_HTTP channels

BRUTUS IP STB CUSTOMIZATIONS

After installing on the set-top, you will need to customize your Brutus IP STB configuration. This may include editing the Brutus IP STB channel map and configuring the file and setting up autoboot. These procedures are detailed in the sections “Channel Map Syntax” on page 26 and “Making Brutus IP STB Autoboot” on page 27.

These instructions are a revised and abridged version of similar documentation contained in the BrutusUsageGuide.doc. If something is unclear or requires further explanation, refer to that document.

SUPPORT FOR RTP AND RTCP PROTOCOLS

This release of IP STB may provide basic support for the Real-time Transport Protocol (check the release notes). When configured correctly, the STB is capable of accepting and decoding MPEG-2 SPTS carried in RTP. The SPTS may contain AVC or MPEG video. Basic RTP support provides the minimum set of functionality required to receive the RTP stream and decode it. As such, Basic RTP support does not support RTCP, nor does it handle RTP header extensions. Basic RTP has some additional constraints, which are detailed below.

The Basic RTP packet processing algorithm can:

- extract 4-byte RTP header
- parse CC, Padding
- skip over timestamp, SSRC,
- check CSRC == 0
- check for (and process) any RTP header extension
- check Payload Type (PT=33)
- check RTP sequence number and reorder packet, if necessary
- check for and discard any padding bytes
- pass the payload to playback for decoding

Basic RTP is similar to ProMPEG Code of Practice #3 (Forward Error Correction) but omits the FEC element of that specification.

The header needs to be parsed and then skipped, before passing the RTP payload to the playback mechanism. RTP header parsing code extracts the fields within the header and checks for required RTP profile compliance. Packet reordering is supported.

The Standard RTP packet processing algorithm can process timestamp, SSRC, and CSRC (x CC) for use with RTCP.

Standard RTP conforms fully to the IETF RFC 3550 and is implemented via the live media open-source library (under LGPL). For details on how to configure Brutus IP STB to tune to an RTP stream, see “Channel Map Syntax” .

SUPPORT FOR RTSP PROTOCOL

This release of IP STB provides basic support for the Real-Time Streaming Protocol (RTSP)—RFC-2326, signaling (control) protocol for set up, start, pause, and stop of streams. Also known as the Internet VCR protocol, it is used for retrieval of media from a media server.

When configured correctly, the STB is capable of tuning to an RTSP URL via the channel map. Some restrictions apply:

- URL tuning is limited to RTSP URL's (for now).
- For RTSP, the server offers transport options via SDP, and the client then chooses between them (e.g., UDP or RTP). The server supports RTP, UDP, or both.
- Choice of IP_UDP or IP_RTP must match the client/server capabilities (currently the STB only supports RTP when using RTSP).
- Must build with `LIVEMEDIA_SUPPORT=y` to use RTSP.

Thus to begin streaming of an RTSP stream, the user places an entry in the channels.txt, starts Brutus, and then channel changes (if not the first entry) into the RTSP stream.

SUPPORT FOR HTTP CHANNELS

The IP_HTTP channels are meant to be used in conjunction with our Broadcom IP Streamer, see "Building and Running IP STB Features" on page 13.

SUPPORT FOR SAP/SDP PROTOCOL

This release of IP STB provides support for dynamic IP channel acquisition using Session Announcement Protocols (SAP)—RFC 2974 and Session Description Protocols (SDP)—RFC4566.

SAP and SDP protocols are implemented by extending the basic SDP parsing support provided by the Live Media Library. A new library called libblive_ext.a provides SAP/SDP support. Brutus has been modified to dynamically update the channel map as SAP requests are received to add or delete an IP session. The further details on the SAP/SDP design can be found in the SAP-SDP-Design document.

In addition, libblive_ext.a also provides thread safe access to Live Media library by implementing a Scheduling thread.

Channel Map Syntax

The channel map tells Brutus IP STB which IP addresses (typically multicast IP addresses) and which UDP port numbers have programs that it can tune to.

Edit channels.txt for your IP head-end/video server configuration. The format of each entry is shown below.

Type	*Parameter*	*Parameter*	*Parameter*	*Parameter*	*Notes*
IP_UDP	disable	IP Address	UDP port number	–	–
IP_RTP[1]	disable	IP Address	UDP port number	–	1. See section on support for RTP protocol
IP_RTP	disable	URL[2]	–	–	2. See section on support for RTSP protocol
IP_HTTP	disable	IP Address	port number (5000)	/filename.mpeg	–

The first field describes the stream type. IP_UDP denotes a plain UDP stream. IP_RTP denotes a stream carried in RTP. Example RTP channel map entries are given in "Sample IP STB Channel Map Entries" on page 27.

Starting from 97398 IP STB release 4, autochannel scanning for PSI information is supported for Brutus IP STB. You have the option of providing the program PIDs and audio/video types or not. If you provide the information, Brutus uses it and "tunes" to the provided PIDs. If you do not provide the information, Brutus tries to obtain the PIDs and audio/video types from the stream. If successful, the stream is played. Otherwise, an error message is shown on the screen.

Note: Autochannel/PID scan supports transport stream only.

The syntax of the Program Pids entry is:

```
PROGRAM PIDS pcr_pid video_pid audio_pid audio_type video_type
```

If only pcr_pid is specified, it is used for video_pid and pcr_pid

- video_type values are: `MPEG, AVC.` Default is MPEG.
- audio_type values are: `AC3, AAC, DTS, MPEG.` Default is MPEG.

You can also use the numeric Set-topAPI values as well. If PCR is not specified, it defaults to video.

SAMPLE IP STB CHANNEL MAP ENTRIES

```
   # IP channels (channel 4 and 5 are auto channel/PID scan, 6 and 7 are RTP
streams)
   # Channel 1
   IP_UDP          0  224.1.1.10  1234
   PROGRAM PIDS 310 310 410 0x81 video_type=0x1b
   # Channel 2
   IP_UDP          0  224.1.1.11  1234
   PROGRAM PIDS 0x1022 0x1022 0x8b9 0x81 video_type=0x1b
   # Channel 3
   IP_UDP          0  224.1.1.12  1234
   PROGRAM PIDS 0x1023 0x1023 0x1022 0x3 video_type=0x2

   # channel 4 - auto channel/PIDs scan
   IP_UDP          0  224.1.1.13  1234
   # channel 5 - auto channel/PIDs scan
   IP_UDP          0  224.1.1.14  1234

   # Channel 6 - an RTSP "channel"
   IP_RTP          0 rtsp://192.168.0.38:554/Video/MyVTR/mystream.ts
   # Channel 7
   IP_RTP          0  224.1.1.16  1234
```

MAKING BRUTUS IP STB AUTOBOOT

Make Brutus IP STB automatically start when the set-top reboots. To make Brutus IP STB autostart, create a file called `/root/rc.user,` which looks something like the following (specifics may vary):

```
cd /home/brutus
settop brutus &
```

The trailing "&" in the last line is critical to allow the rcS file to complete and a login prompt to appear on the console. You might need the following to make the init process continue on and give you a login prompt:

```
settop brutus </dev/null >/dev/null 2>/dev/null &
```

BRUTUS IP STB SPECIAL OPTIONS

Edit `brutus.cfg` with any special options. See the configuration section of the *Brutus Usage Guide* (STB_Brutus-SWUM300-R) for more help.

SIMPLE IP HEAD END

DSL IP VIDEO HEAD END AND CPE

This is the recommended DSL IP Video Server Head End. The DSLAM is multicast aware and dynamically routes multicast data to the STB on demand, in response to the IGMP messages from the STB. This is how channel change is achieved on an IP STB.

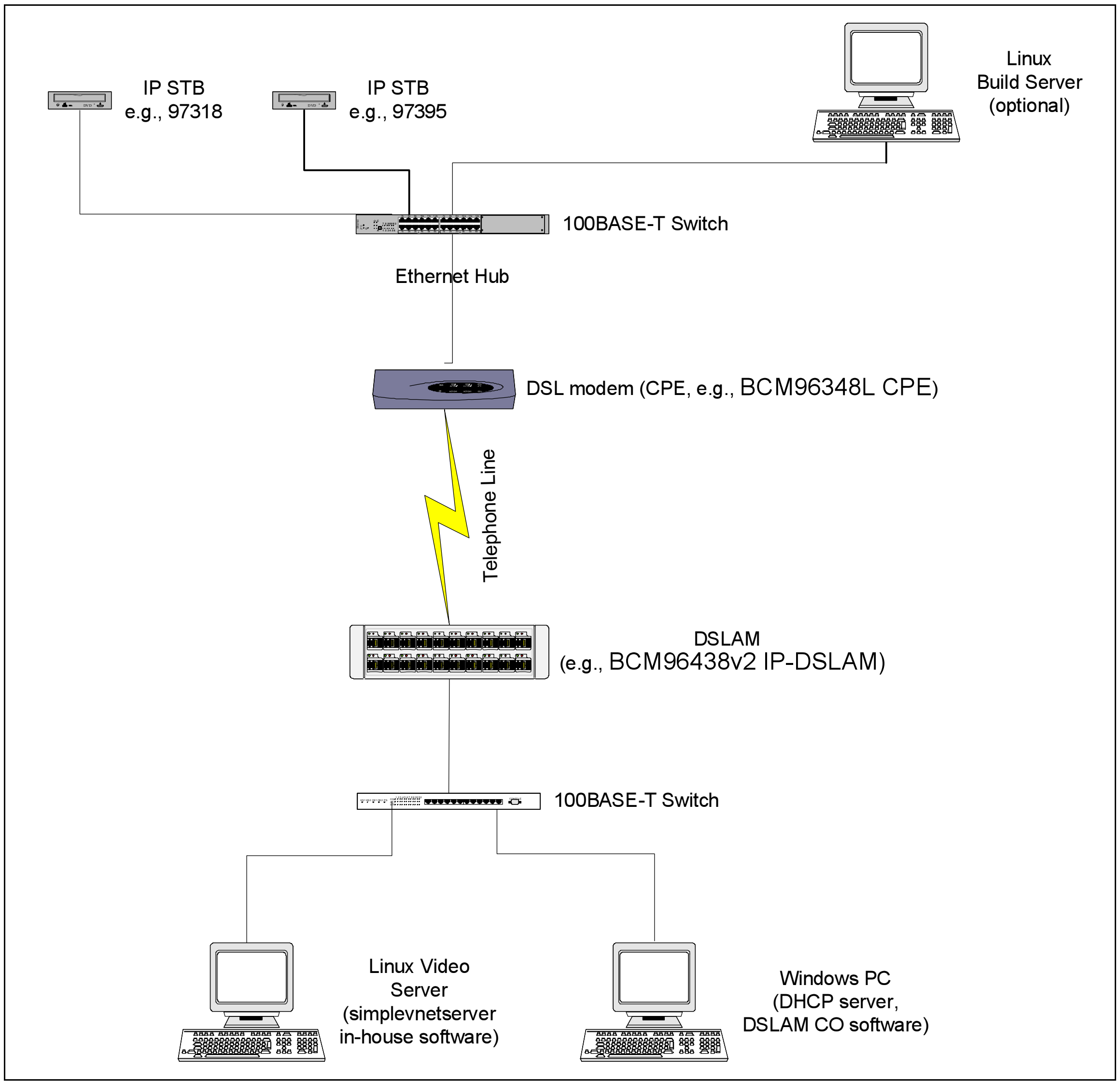

Figure 1: Configuration of DSL IP Video Server Head End and CPE

SIMPLE HEAD END

This is the minimum supported configuration. But note that excessive aggregate bit rate of content streamed from the video server onto the STB ethernet interface can cause video degradation.

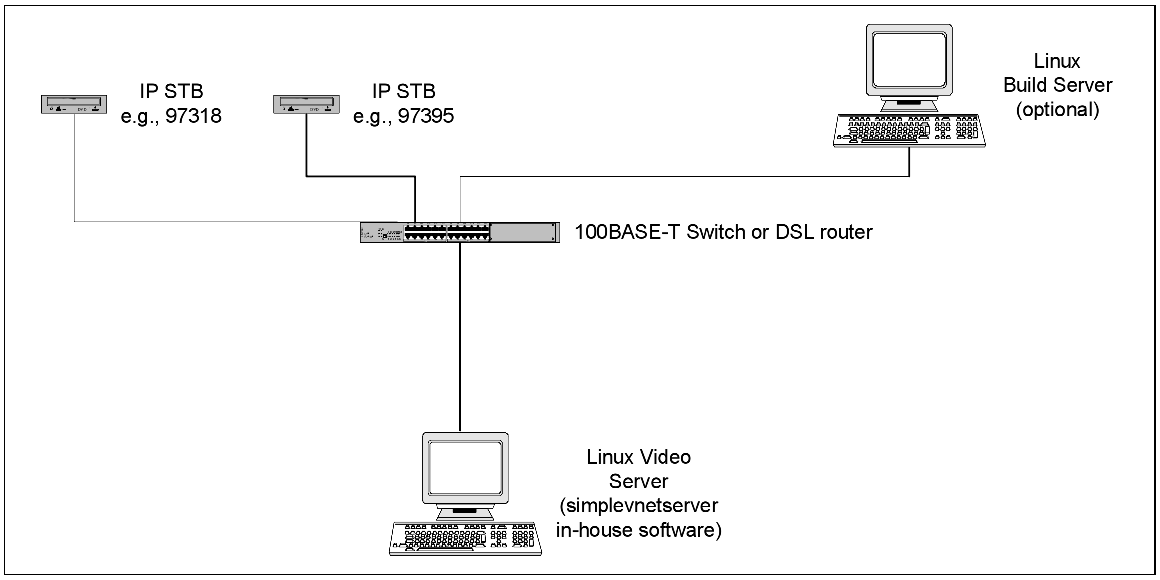

Figure 2: Simple Head End Configuration

Broadcom used a Netgear 10/100 switch and added a LinkSys DSL router for the DHCP server. If you have networking issues, you should try to get as close as possible to this configuration. If you still run into networking issues, try unicast to the STB to solve the problem.

SIMPLE IP VIDEO SERVER

You can use any IP video server that supports AVC/MPEG-2 TS/UDP/IP. However, we have supplied a Broadcom-developed IP video server that supports AVC video embedded in MPEG-2 TS, as well as MPEG-2 video TS. You can use this application to create an IP Video Server based on a RH Linux PC. Install the simplevnetserver binary by simply copying it to your RH Linux PC.

For an example on using simplevnetserver, we have a stream called “crawibc16m.ts” with a bit rate of 8.5 Mbps. To multicast it on IP address 224.0.0.10 and use UDP port 1234, the command line looks like this:

```
./simplevnetserver -f crawibc16m.ts -ipaddr 224.0.0.10 -port 4d2 -r 8500000
```

You must provide all of these parameters; none are optional.

Table 7: IP Video Server Parameters

Argument	***Description***
`-f`	Specify the TS filename
`-ipaddr`	Specify the IP address you wish to send with
`-port`	Specify the UDP port you wish to send with
`-r`	Specify the bit rate of the TS

Note: The UDP port number is in hexadecimal (without the leading 0x).

Note: Simplevnetserver does not support RTP.

POWER MANAGEMENT

USING AND TESTING POWER MANAGEMENT WITH SET-TOPAPI/NEXUS AND BRUTUS

The Power Management features provided with the Brutus Reference Software demonstrate *passive standby*, the shutdown of the set-top to a minimum power state, with all functionality deactivated, except for the ability to wake up in response to the IR remote control, the front-panel “Power” button, or a timer. *Active standby* may be implemented using the PWR porting interface and other provided software detailed below to power down selected set-top functionality, while keeping other functions operational, such as recording a program, or downloading a program guide. For more information on design of the power management features and software for Broadcom set-top and DTV chips, please refer to the *Power Management Software Application Note* (STB-DTV-AN20x-R).

Building

On platforms with power management support, POWERSTANDBY_SUPPORT defaults to `y`. This builds the power_standby utility and adds settop.power to the install image. Build with `POWERSTANDBY_SUPPORT=n` to disable those build steps.

Running

When running on the set-top box,

```
export POWER_STANDBY=y
```

When this environment variable is set, pressing the power button on the remote quits Brutus and launches the separate power_standby utility.

STANDBY PROCESS

Setting the environment variable POWER_STANDBY causes the settop command to be invoked via settop.power instead. When the program launched by the script exits, the following occurs:

1. All settop kernel modules are unloaded.
2. bcmkni.ko and brcmpmdrv.ko are loaded.
3. **`power_standby -passive`** is called.

Once the board is woken up:

4. brcmpmdrv.ko and bcmkni,ko are unloaded.
5. **`settop $*`** is called again to restart the application.

This mechanism does result in the application being relaunched in an infinite loop.

COMPONENTS

Power Management support consists of the following components:

- BSEAV/linux/driver/power–Source for brcmpmdrv
- BSEAV/app/power/–Application demonstrating use of brcmpmdrv
- BSEAV/app/brutus/build/settop.power–settop sub-script for use when `POWER_STANDBY=y`
- magnum/portinginterface/pwr/–PWR porting interface
- uclinux-rootfs/user/brcm-pm/pmlib.c, pmlib.h–source for controlling the power status of devices controlled by the Linux OS.
- rockford/applications/power_standby–Source for power_standby.

brcmpmdrv is the kernel power management driver. This driver depends on bcmkni.ko. brcmpmdrv has a matching application and launch scripts. This driver does the bulk of the work for passive standby, including receiving input from the keypad or the remote to wake up.

pmlib is a library to shut down SATA, Ethernet, USB, and the second CPU, using entries in the /sys directory. It can also change the CPU clock speed divisor and DDR refresh rate. It has an accompanying pmtest utility in the same directory to demonstrate its use. The pmlib source files are duplicated with, and used by power_standby.

power_standby is a stand-alone utility that integrates the functionality of the individual component test applications in /uclinux-rootfs/user/brcm-pm and in /BSEAV/app/power. It invokes pmlib to shut down the Linux drivers for SATA, Ethernet, USB, the second thread processor in the case of SMP Linux, and increases the CPU divisor. It then calls BPWR to disable the clocks on any cores which should be shut off. power_standby then calls the ioctl for brcmpmdrv to enter passive mode and waits to return. Once the ioctl has returned, power_standby re-enables the clocks via BPWR, re-sets the CPU divisor, wakes the second CPU, USB, Ethernet, and SATA, and exits. power_standby is used by the settop.power script to integrate passive standby power management with Brutus.

Broadcom Corporation

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