Application Note

CPE

CMS Configuration File

CPE-AN700-R

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

February 12, 2014

Revision History

Revision	*Date*	*Change Description*
CPE-AN700-R	02/12/14	Initial release

Broadcom Corporation
5300 California Avenue
Irvine, CA 92617

Table of Contents

List of Figures

About This Document

Purpose and Audience

This document describes various topics related to the configuration file in the Broadcom BCM96XXX Broadband Router. Refer to the *Configuration Management System Development and Porting Guide* (see Reference [1] on page 6) for more information about the configuration file.

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] Configuration Management System Development and Porting Guide	963XX-PG2xx-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/).

Saving the Config File to a PC

Once you have configured the Broadband Router to the desired settings, you can save the config file to a PC connected to the Broadband Router. There are three ways of doing this: via WebUI, CLI menu, or CLI command.

WebUI

Only a local “admin” user or a remote “support” user is allowed to perform the configuration backup under backup management function. Click **Management**, **Settings**, then click **Backup Settings** to save the configuration data (default file name is “backupsettings.cfg”).

CLI Menu

In the Main Menu, enter **9** (Management Menu), **1** (Settings), **1** (Backup), and then follow the prompt to enter **Tftp Server IP address** and **Remote File Name (backupsettings.cfg)**.

CLI Command

From the CLI command prompt, type:

```
tftp -p -f file-name tftp-server-ip
```

where `tftp-server-ip` is the IP address of the tftp server that will receive the config file.

Uploading a Saved Config File from a PC

You can upload a saved config file to the Broadband Router. Once the Broadband Router has validated the config file, it will overwrite the existing config file with the uploaded config file and reboot the router. There are four ways to upload a saved config file: WebUI, CLI menu, CLI command, and FTP server.

WebUI

Only a local “admin” user or a remote “support” user is allowed to perform the configuration update function. Click **Management**, **Settings**, **Update**. Enter the <remote configuration file (and path)> or click **Browse** to select the configuration file. Click **Update Settings** to start the upload.

CLI Menu

In the Main Menu, enter **9** (Management Menu), **1** (Settings), **2** (Update), and then follow the prompt to enter **Tftp Server IP address** and **Update Setting File Name (backupsettings.cfg)**.

CLI Command

From the CLI command prompt, type:

```
tftp -g -t c -f file-name tftp-server-ip
```

where `tftp-server-ip` is the IP address of the tftp server that will receive the config file.

FTP Server

If the FTP server option is built into the system image, you can upload a configuration file to the Broadband Router's internal FTP server. From the PC, connect via FTP into the Broadband Router. Log in using the admin/admin or support/support account. From the FTP command prompt, type:

```
put file-name
```

where `file-name` is the name of the saved config file.

Building a Default Config File into the System Image

When there is no configuration file saved in the config file section of the flash, the Broadband Router will boot in the most basic configuration (no WAN interfaces or services configured.) In this scenario, you may want the modem to use a preconfigured config file. This section describes the procedure to embed a default config file in the system image. The default config file is stored on the read-only file system of the image. It is not stored in the area of the flash memory dedicated to storing the config file.

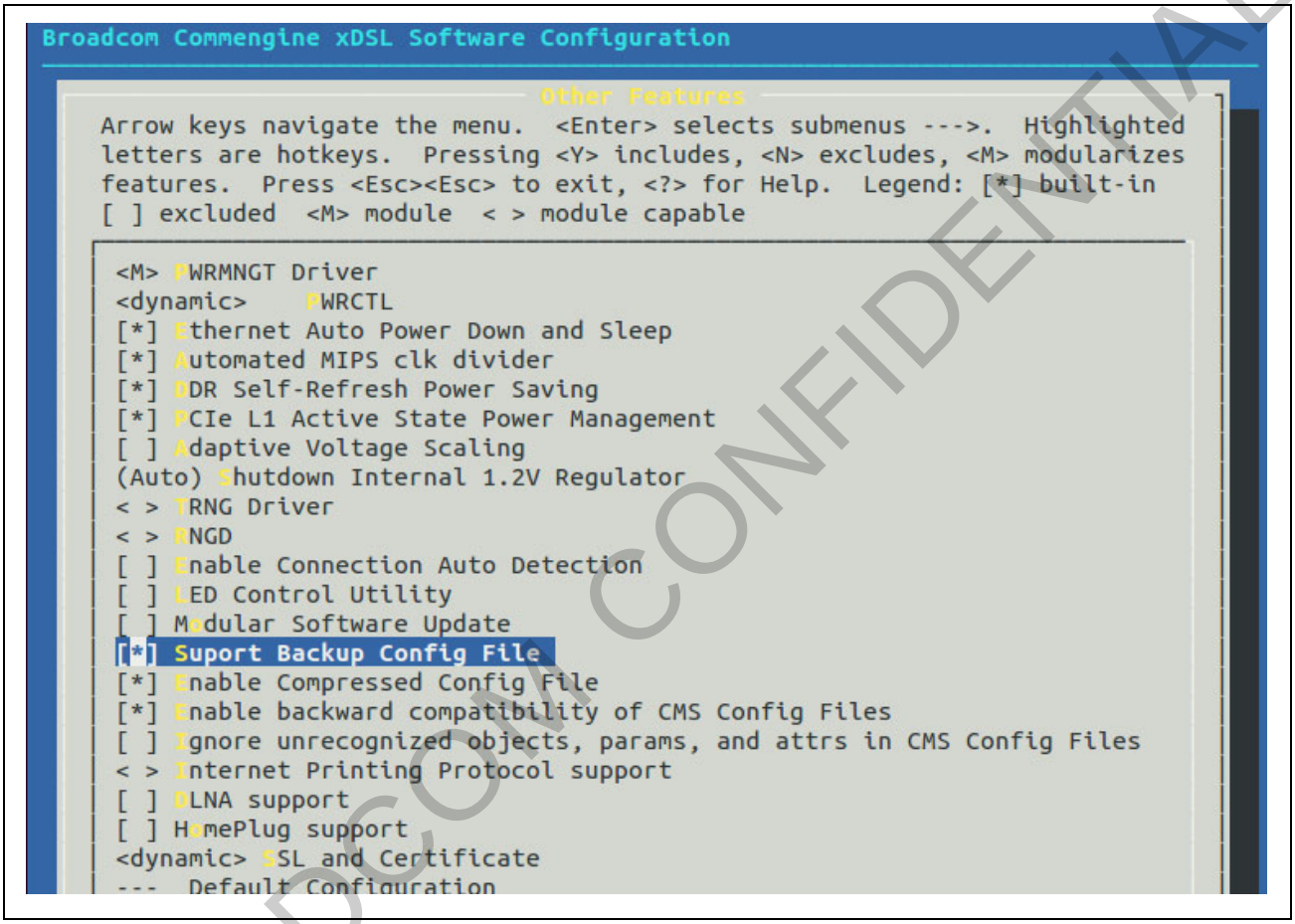
1. Configure the Broadband Router to the desired configuration.
2. Save the configuration using one of the methods described in "Saving the Config File to a PC" on page 7.
3. Move the saved config file to your development directory, under "targets/defaultcfg".
4. Using make menuconfig, load the profile you are using, go to the "Other Features" section, and enter <name of the saved config file> in the "Default Configuration" item.
5. Build the image.

Note: Default configuration settings can also be added or modified during system start-up time via c code. This allows the default settings to be customized depending on the properties of the system. See userspace/private/libs/cms_core/mdm_init.c and mdm_initxxx.c.

Backup Config File

If the power to the system is lost in the middle of a config file write, the config file will be corrupted. The system can be hardened against this scenario by writing out two copies of the config file. This option can be enabled with make menuconfig, in the “Other Features” section, “Support Backup Config File”.

Figure 1: Support Backup Config File



Note: On NOR-based systems, the CFE must also be configured to reserve sectors for the backup config file (referred to as “backup PSI”). No additional configuration is needed on NAND-based systems.

Starting with the 4.14L.04 release, when you enable “Backup Config File”, you must choose from one of two modes:

- Mirror Mode—every time the configuration is saved, an exact copy of the configuration is written to both the primary and backup config file. When you invalidate the configuration using the “restore to default” command, both copies will be invalidated.

- Device Default Mode—in this mode, when the configuration is saved, it is only saved to the primary config file. The "restore to default" command will only erase the primary config file. The backup configuration file contains a copy of the configuration that contains device-specific defaults. This device-specific backup configuration file is used only if the primary config file is corrupted or invalidated. The device-specific configuration file allows the system to save device-specific configuration info, such as mac address, device name, network passwords, calibration data, and so on.

  The device-specific configuration file is usually written at the factory when the system is programmed with device-specific information. To write to the backup (device-specific) config file, first configure the system with the device-specific info as you normally would and save the configuration. Once the configuration is saved to the primary config file, then go to the busybox shell and run the following command:

  ```
  psictl copyprimarytobackup
  ```

  This command will copy the contents of the primary config file to the backup config file. (Remember, in Device Default Mode, the configuration info is not saved to the backup config file. You need to use this special command to transfer the contents from primary to backup.) psictl contains other useful debug commands. Type **psictl help** for usage.

  A Perl script showing an example of how this feature works is available under hostTools/examples/psi_example.pl.

Config File and Data Model Compatibility

When the system boots, ssk tries to populate the Memory Data Model (MDM) with the contents of the config file. If the config file contains an object, parameter, or attribute which is not in the Data Model of the currently running image, the config file is assumed to be invalid or corrupted, and the entire config file is rejected. A message similar to this one will appear on the console:

```
ssk:error:10.917:mdm_tagBeginCallbackFunc:637:Unrecognized tag
X_BROADCOM_COM_IPv6LANHostConfigManagement
ssk:error:10.951:mdm_loadConfig:124:primary config file from flash is
unrecognized or invalid.
```

This section discusses the causes of config file/Data Model incompatibility and how to avoid or fix this problem. The first three subsections describe the causes, and the subsequent four subsections describe avoidance strategies and fixes.

Same Release but Different PROFILE Options (Cause 1)

Assume a developer is using the same release, for example 4.12L.04. First, he/she compiles an image with the VoIP option enabled (VoIP image). He boots the VoIP image, configures the voice parameters, and saves the configuration to the config file. He then compiles another image with the VoIP option disabled (data-only image). When he boots the data-only image, ssk will try to populate the MDM with the contents of the saved config file. However, this config file contains VoIP parameters which are not in the Data Model of the data-only image. Ssk (actually, the code in mdm_configin.c) does not know what to do with these unrecognized voice parameters, so

the default behavior is for ssk to assume that the entire config file is invalid and not use any part of the config file. Starting with 4.12L.07 (and 4.12L.05BL), there is an option called “Ignore unrecognized objects, params, and attrs”, which changes the default behavior. See “Ignore Unrecognized Objects, Params, and Attrs Option (Fix 2)” on page 14.

Going from a VoIP image to a data-only image is only one example of a PROFILE option change that could cause this problem. This problem can also be seen when going from an IPv6-capable image to an IPv4-only image, or from a USB enabled image to a non-USB enabled image. In general, anytime the first image has more capabilities and has written a parameter associated with that capability, and a second image without that capability is booted on the same system, a config file and Data Model incompatibility will occur.

Note: There is no problem when going from a less capable image to a more capable image (for example, data-only image to a VoIP image), because if a data-only image is run first, then only data-only related parameters will be written to the config file. Later, when the VoIP image is booted, the MDM will be populated with the data-only parameters. Since no VoIP-related parameters were written to the config file by the data-only image, the VoIP-related parameters in the MDM will remain at their default values.

Newer Release That Does Not Recognize a Parameter from Older Release (Cause 2)

During development, the Data Model may need to be changed to fix a bug, modify an existing feature, implement a new feature, or some similar task. Changes to the Data Model can be categorized into two classes, backward compatible changes and backward incompatible changes. Backward compatible changes do not cause config file/Data Model compatibility problems, while backward incompatible changes do.

Adding a new object or parameter is a backward compatible change. To understand this, assume that in 4.12L.02, there was an object A with parameter x. Now in 4.12L.04, we added a new parameter y to object A. So when the system was running 4.12L.02, it wrote a config file containing object A parameter x. When 4.12L.04 is booted on the system, ssk will initialize object A parameter x in the MDM with contents of the config file. Because the config file does not contain object A parameter y (since this parameter did not even exist in 4.12L.02), object A parameter y will remain at its default value in the MDM.

Renaming or deleting an existing parameter which was never written to the config file (for example, when the parameter is read-only), or was marked with the “NeverWriteToConfigFile=true” attribute, is also a backward compatible change.

Renaming or deleting an existing object which was never written to the config file is also a backward compatible change. In order for an object to not get written to the config file, it must not have any attributes that need to be saved to the config file, such as instance numbers, TR69 notification attributes, and so on. It must also not contain any parameters or child objects which need to be written to the config file. Higher level and multiple instance objects usually will get written to the config file for some reason. Lower level objects (leaf objects) might not get written to the config file. Use any of the CMS config file dumping utilities, dumpcfg for example, to check.

All other changes to the Data Model, such as renaming or deleting an object or parameter which could be written to the config file, are backward incompatible and should be avoided. Sometimes making a backward incompatible change is unavoidable. It is still possible to add code to mdm_configin.c to convert the old object or parameter name to the new object name or parameter name, or to simply ignore the deleted object or parameter name. See "Implement Conversion Code (Fix 1)" on page 13. Or the old deleted/renamed parameters and objects can simply be ignored with the "Ignore unrecognized objects, params, and attrs" option (see "Ignore Unrecognized Objects, Params, and Attrs Option (Fix 2)" on page 14).

Older Release which Does Not Recognize a Parameter from Newer Release (Cause 3)

Consider the following scenario: a system is running a newer release image, for example 4.12L.07, and this newer release image writes out its objects and parameters to a config file. Then an older release image, for example 4.12L.02 or even 4.06L.02 or 4.02L.02, is loaded on the system. There have been requests that the older releases be made "forward compatible" so that they can understand a config file written by a release image/Data Model from a future release.

In general, this is very difficult to do, since the older image cannot anticipate what the format or contents of a newer config file (that will be defined in the future) will be. Traditionally, this scenario is not supported by the BCA CPE reference software.

However, if an image is compiled with the "Ignore unrecognized objects, params, and attrs" option, then if it sees a config file from the future that has the same format but only contains unrecognized objects, parameters, or attributes, then it will simply ignore those objects, parameters, and attributes, but will still use the objects, parameters, and attributes that it does recognize to initialize the MDM. This solution is only applicable to current releases going forward. It cannot apply to existing systems already deployed in the field running releases prior to the introduction of this feature.

Design Data Model Carefully (Avoidance 1)

The best way to avoid config file/Data Model compatibility problems is to design the addition of new objects, parameters, features to the Data Model carefully and thoughtfully. Instead of just satisfying the minimum current needs, try to plan ahead and imagine what the Data Model might need to support in the next 5 to 8 years. Even if the code does not support all the features that the Data Model is capable of, it is better to add the capability in the Data Model first and then let the implementation catch up later as requirements change. (Code is easier to change and add to than the Data Model.)

Two common mistakes in this area are:

- Creating a single instance object because only one instance is specified in the requirements for this feature. But in the future, multiple instances could be needed. It is much better to make the object multiple instances at the beginning, even though for now, only one instance is needed.
- Not considering IPv6. IP addresses and other IP address-related fields should always be defined as a string and not as a 32-bit integer. Strings can have arbitrary length, so they can easily support an IPv6 address, while a 32-bit integer cannot support an IPv6 address.

Also, look at the existing standards, such as TR-181, to see if the desired objects and parameters are already defined there. Whenever possible, use the objects and parameters defined in public specifications.

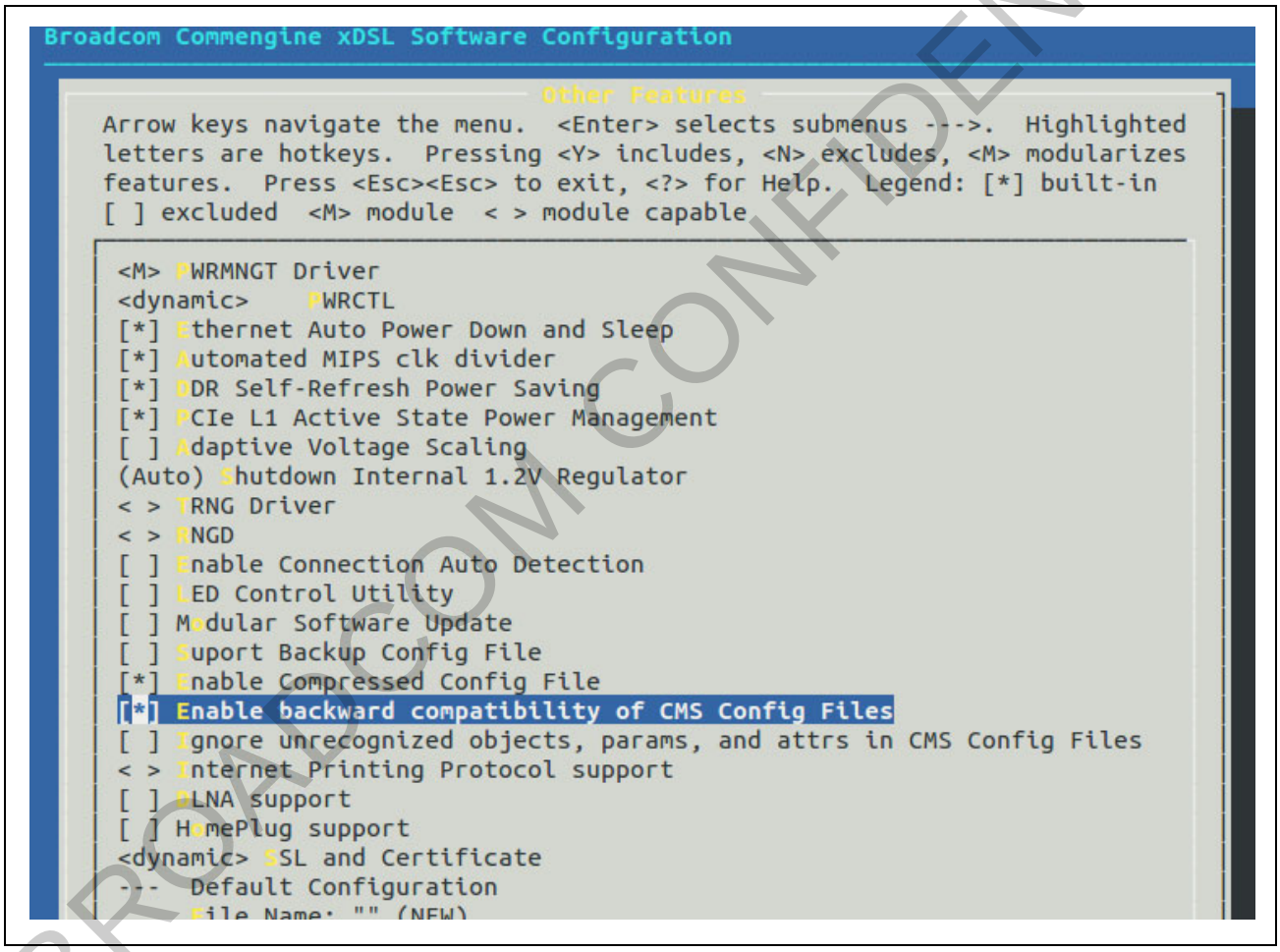
Only Make Backward Compatible Changes (Avoidance 2)

Backward compatible and non-backward compatible changes were discussed in “Newer Release That Does Not Recognize a Parameter from Older Release (Cause 2)” on page 11.

Implement Conversion Code (Fix 1)

In the BCA CPE reference software, whenever a backward incompatible change is made, conversion code is also written in mdm_configin.c to convert the previous object or parameter name to the new object or parameter name. This feature must be enabled in make menuconfig, in the “Other Features” section, “Enable backward compatibility of CMS Config Files”. See Figure 2.

Figure 2: Conversion Code

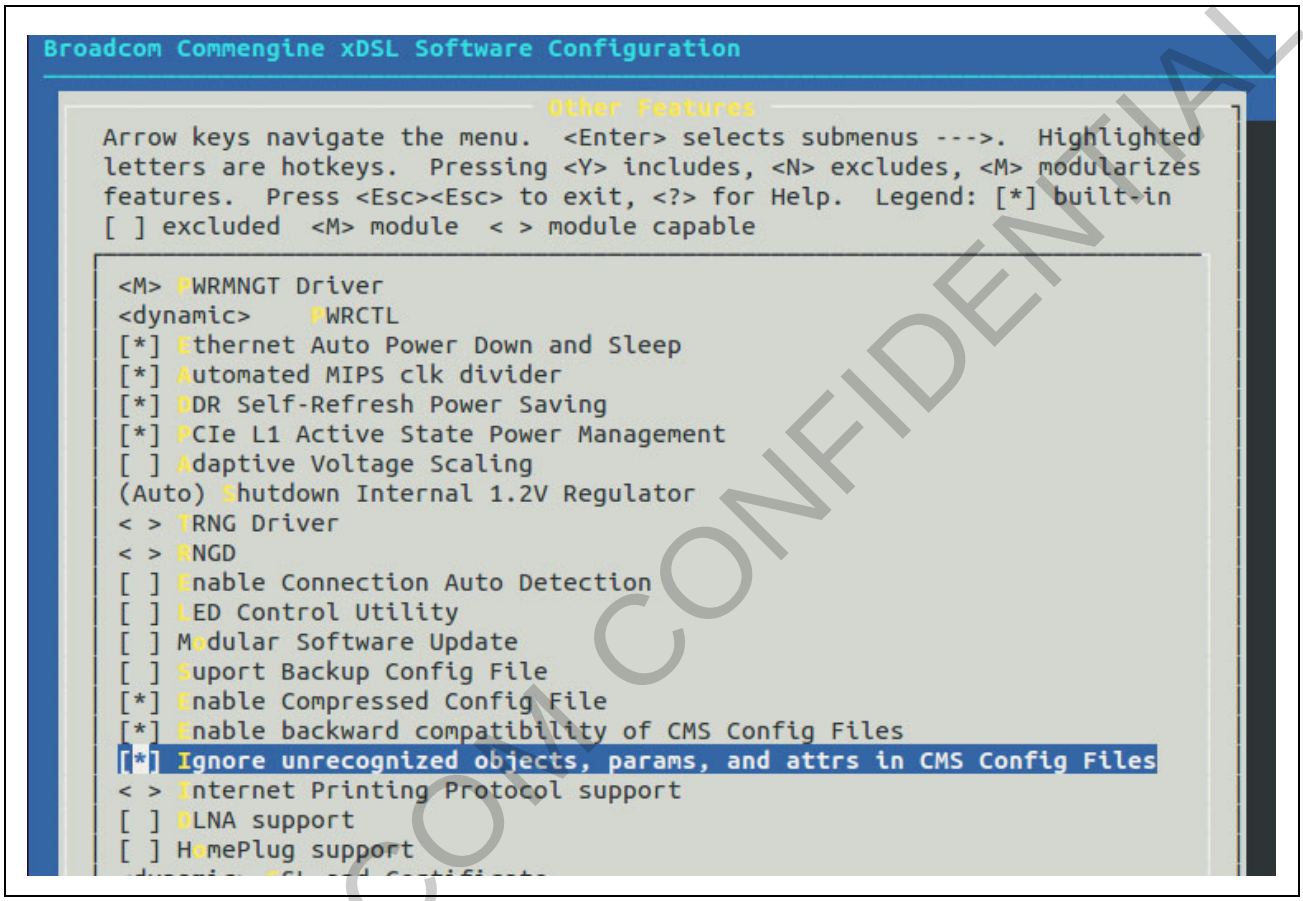


The conversion code is in mdm_configin.c. Follow the code inside the #ifdef CMS_CONFIG_COMPAT to see how X_BROADCOM_COM_NTPEnable was converted to NTPEnable.

Ignore Unrecognized Objects, Params, and Attrs Option (Fix 2)

Starting with the 4.12L.07 and 4.12L.05BL releases, there is a make menuconfig option which tells mdm_configin.c to ignore unrecognized objects, parameters, and attributes. This can be enabled in make menuconfig, in the “Other Features” section, “Ignore unrecognized objects, params, and attrs”. See Figure 3.

Figure 3: Ignore Unrecognized Objects, Params, and Attrs



Once an object, parameter, or attribute has been ignored, it will not be preserved when the config file is written again. For example, you run a VoIP image, configure the VoIP parameters, and write out the config file. Then you run a data-only image, which has the “Ignore unrecognized objects, params, and attrs” option enabled. This data-only image will ignore the VoIP-related objects and parameters and still use the data-related objects and parameters to initialize the MDM. Suppose you make some other configuration changes and save config file. At this point, the config file that is written will contain only the data-related parameters. The VoIP-related objects, parameters, and attributes that were ignored from the previous config file will not be written to the config file. So if you now boot the VoIP image you initially used, the VoIP-related configuration will be lost.

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