

Coova-Chilli_1.4_HHLB

Source

Coova-Chilli on OpenWRT and LEDE 17.01 is based on the 1.3.0+20141128 version.

If you just clone the git repository from coova-chilli the Makefile and patches applies to **coova-chilli-1.3.0+20141128**

This page describes how to upgrade to the latest **1.4**.

In order to compile the latest **1.4** on **LEDE 17.01** some requirements are needed.

Cleaning

1. Under the directory **dl** be sure that no coova-chilli packages are already there.
 - a. Remove if any files: **coova-chilli-1.4.tar.gz** or **coova-chilli-1.3.0+20141128.tar.gz**
2. Under the **build_dir/target-mipsel_24kc_musl-1.1.16** directory (Specific to your target).
 - a. Remove if any directories: **coova-chilli-1.4** or **coova-chilli-1.3.0+20141128**

Layout

Under the **feeds/packages/net/coova-chilli** directory is the **Makefile**, **Config.in**, **patches** and **files** directories specific to your OpenWRT/LEDE coova-chilli build target platform.

This directory is also referenced as a symbolic link in **package/feeds/packages/coova-chilli**.

Makefile

Open the **package/feeds/packages/coova-chilli/Makefile** and patch with the following content:

Following the compile definitions from the file <https://github.com/coova/coova-chilli/blob/master/configure.ac>

```
#                                     -*- Autoconf -*-
# Process this file with autoconf to produce a configure script.

AC_PREREQ([2.59])
AC_INIT([coova-chilli],[1.4],[https://github.com/coova/coova-chilli/issues])
AC_CONFIG_SRCDIR([src/chilli.c])

AM_INIT_AUTOMAKE

AC_CONFIG_HEADERS([config.h])

AC_CONFIG_MACRO_DIR([m4])

# Checks for programs.
AC_PROG_CXX
AC_PROG_AWK
AC_PROG_CC
AC_PROG_CC_C_O
AC_PROG_CC_C99
AC_PROG_CPP
AC_PROG_INSTALL
AC_PROG_LN_S
AC_PROG_MAKE_SET

#JJAKO Check for libtool
LT_INIT
AC_ARG_PROGRAM

# Checks for libraries.

if test -r /dev/bpf -o -h /dev/bpf ; then
  V_DEV=bpf
  AC_DEFINE(HAVE_CLONING_BPF,1,[define if you have a cloning BPF device])
elif test -r /dev/bpf0 ; then
  V_DEV=bpf
elif test -r /usr/include/net/pfilt.h ; then
  V_DEV=pf
elif test -r /dev/enet ; then
  V_DEV=enet
```

```

elif test -r /dev/nit ; then
    V_DEV=snit
elif test -r /usr/include/sys/net/nit.h ; then
    V_DEV=nit
elif test -r /usr/include/linux/socket.h ; then
    V_DEV=linux
elif test -r /usr/include/net/raw.h ; then
    V_DEV=snoop
elif test -r /usr/include/odmi.h ; then
    V_DEV=bpf
elif test -c /dev/bpf0 ; then # check again in case not readable
    V_DEV=bpf
elif test -r /usr/include/sys/dlpi.h ; then
    V_DEV=dlpi
elif test -c /dev/enet ; then # check again in case not readable
    V_DEV=enet
elif test -c /dev/nit ; then # check again in case not readable
    V_DEV=snit
else
    V_DEV=null
fi
AC_MSG_RESULT($V_DEV)

# Do capture-mechanism-dependent tests.
case "$V_DEV" in
linux)
    AC_LBL_TPACKET_STATS
    AC_LBL_LINUX_TPACKET_AUXDATA_TP_VLAN_TCI
    ;;
esac

# Checks for header files.
AC_HEADER_STDC
AC_HEADER_RESOLV
AC_CHECK_HEADERS([arpa/inet.h errno.h fcntl.h dirent.h \
    inttypes.h limits.h \
    netdb.h netinet/in.h netinet/tcp.h \
    stdint.h stddef.h stdlib.h string.h strings.h \
    sys/ioctl.h sys/socket.h linux/sysinfo.h sys/sysinfo.h \
    sys/param.h sys/time.h time.h \
    sys/ipc.h sys/msg.h signal.h \
    sys/wait.h sys/un.h ifaddrs.h \
    sys/stat.h sys/types.h regex.h \
    syslog.h poll.h sys/epoll.h \
   unistd.h endian.h libgen.h \
    asm/types.h pwd.h grp.h wchar.h stdarg.h])

AC_CHECK_HEADERS([resolv.h net/route.h net/if.h net/if_arp.h net/if_tun.h net/ethernet.h], [], [],
[
#include <stdio.h>
#ifdef STDC_HEADERS
# include <stdlib.h>
# include <stddef.h>
#else
# ifdef HAVE_STDLIB_H
# include <stdlib.h>
# endif
#endif
#ifdef HAVE_SYS_TYPES_H
# include <sys/types.h>
#endif
#ifdef HAVE_SYS_SOCKET_H
# include <sys/socket.h>
#endif
#ifdef HAVE_NETINET_IN_H
# include <netinet/in.h>
#endif
])

AC_CHECK_HEADER(inttypes.h,[AC_DEFINE([JSON_C_HAVE_INTTYPES_H],[1],[Public define for json_inttypes.h])])
AC_CONFIG_HEADER(json/json_config.h)

```

```

AC_CHECK_DECLS([INFINITY], [], [], [[#include <math.h>]])
AC_CHECK_DECLS([nan], [], [], [[#include <math.h>]])
AC_CHECK_DECLS([isnan], [], [], [[#include <math.h>]])
AC_CHECK_DECLS([isinf], [], [], [[#include <math.h>]])
AC_CHECK_DECLS([_isnan], [], [], [[#include <float.h>]])
AC_CHECK_DECLS([_finite], [], [], [[#include <float.h>]])

# Checks for typedefs, structures, and compiler characteristics.
AC_C_CONST
AC_TYPE_UID_T
AC_C_INLINE
AC_TYPE_INT16_T
AC_TYPE_INT32_T
AC_TYPE_INT8_T
AC_TYPE_MODE_T
AC_TYPE_PID_T
AC_TYPE_SIZE_T
AC_TYPE_SSIZE_T
AC_TYPE_UINT16_T
AC_TYPE_UINT32_T
AC_TYPE_UINT64_T
AC_TYPE_UINT8_T
AC_CHECK_TYPES([ptrdiff_t])

# Checks for library functions.
AC_FUNC_CHOWN
AC_FUNC_FORK
AC_PROG_GCC_TRADITIONAL
AC_FUNC_MEMCMP
AC_FUNC_SELECT_ARGTYPES
AC_FUNC_MKTIME
AC_FUNC_MMAP
AC_CHECK_FUNCS([bzero clock_gettime dup2 gethostbyname getprotoent gettimeofday inet_ntoa \
memchr memmove memset mkdir munmap regcomp select setenv socket strcasecmp \
strchr strcspn strdup strerror strncasecmp strndup strchr strspn strstr strtol getline dirname \
glob getaddrinfo getnameinfo getifaddrs sysinfo strlcpy tzset snprintf vsnprintf vasprintf])
AC_CHECK_LIB(resolv, res_init)

AC_ARG_ENABLE(chilliquery, [AS_HELP_STRING([--disable-chilliquery],[Disable chilli_query]),
    enable_chilliquery=$enableval, enable_chilliquery=yes)

if test x"$enable_chilliquery" = xyes; then
    AC_DEFINE(ENABLE_CHILLIQUERY,1,[Define to enable chilli_query])
fi

AM_CONDITIONAL(WITH_CHILLIQUERY, [test x"$enable_chilliquery" != xno])

AC_ARG_ENABLE(leakybucket, [AS_HELP_STRING([--disable-leakybucket],[disable use of leaky bucket shaping]),
    enable_leakybucket=$enableval, enable_leakybucket=yes)

if test x"$enable_leakybucket" = xyes; then
    AC_DEFINE(ENABLE_LEAKYBUCKET,1,[Define to enable Chilli Leaky Bucket shaping])
fi

AC_ARG_ENABLE(uamanyip, [AS_HELP_STRING([--disable-uamanyip],[disable use of uamanyip]),
    enable_uamanyip=$enableval, enable_uamanyip=yes)

if test x"$enable_uamanyip" = xyes; then
    AC_DEFINE(ENABLE_UAMANYIP,1,[Define to enable uamanyip])
fi

AC_ARG_ENABLE(uamuiport, [AS_HELP_STRING([--disable-uamuiport],[disable use of uamuiport]),
    enable_uamuiport=$enableval, enable_uamuiport=yes)

if test x"$enable_uamuiport" = xyes; then
    AC_DEFINE(ENABLE_UAMUIPORT,1,[Define to enable uamuiport])
fi

AC_ARG_ENABLE(accounting-onoff, [AS_HELP_STRING([--disable-accounting-onoff],[disable use of Accounting-On and Accounting-Off]),

```

```

    enable_accounting_onoff=$enableval, enable_accounting_onoff=yes)

if test x"$enable_accounting_onoff" = xyes; then
    AC_DEFINE(ENABLE_ACCOUNTING_ONOFF,1,[Define to enable Accounting-On and Accounting-Off])
fi

AC_ARG_ENABLE(tap, [AS_HELP_STRING([--disable-tap],[Disable support for tap interface (tun only)])],
    enable_tap=$enableval, enable_tap=yes)

if test x"$enable_tap" = xyes; then
    AC_DEFINE(ENABLE_TAP,1,[Define to enable Chilli tap support])
fi

AC_ARG_ENABLE(tcpreset, [AS_HELP_STRING([--disable-tcpreset],[Disable support for TCP reset of filtered
connections])],
    enable_tcpreset=$enableval, enable_tcpreset=yes)

if test x"$enable_tcpreset" = xyes; then
    AC_DEFINE(ENABLE_TCPRESET,1,[Define to enable TCP reset support])
fi

AC_ARG_ENABLE(radproxy, [AS_HELP_STRING([--disable-radproxy],[Disable support RADIUS (EAP) Proxy])],
    enable_radproxy=$enableval, enable_radproxy=yes)

if test x"$enable_radproxy" = xyes; then
    AC_DEFINE(ENABLE_RADPROXY,1,[Define to enable Chilli RADIUS (EAP) Proxy support])
fi

AC_ARG_ENABLE(json, [AS_HELP_STRING([--enable-json],[Enable support for JSON])],
    enable_json=$enableval, enable_json=no)

if test x"$enable_json" = xyes; then
    AC_DEFINE(ENABLE_JSON,1,[Define to enable Chilli JSON])
    AC_CHECK_LIB([json-c], [json_object_new_object],
        [AC_SUBST([LIBJSON], ["-ljson-c"]),
         [AC_CHECK_LIB([json], [json_object_new_object],
          [AC_SUBST([LIBJSON], ["-ljson"]),
           [AC_MSG_FAILURE(
            [--enable-json was given, but test for libjson failed])]),])])
fi

AC_ARG_ENABLE(debug, [AS_HELP_STRING([--disable-debug],[Disable debugging messages])],
    enable_debug=$enableval, enable_debug=yes)

if test x"$enable_debug" = xyes; then
    AC_DEFINE(ENABLE_DEBUG,1,[Define to enable debugging])
fi

AC_ARG_ENABLE(dhcpradius, [AS_HELP_STRING([--disable-dhcpradius],[Disable support DHCP/RADIUS integration])],
    enable_dhcpradius=$enableval, enable_dhcpradius=yes)

if test x"$enable_dhcpradius" = xyes; then
    AC_DEFINE(ENABLE_DHCPRADIUS,1,[Define to enable DHCP/RADIUS integration])
fi

AC_ARG_ENABLE(wpad, [AS_HELP_STRING([--enable-wpad],[Enable support WPAD])],
    enable_wpad=$enableval, enable_wpad=no)

if test x"$enable_wpad" = xyes; then
    AC_DEFINE(ENABLE_WPAD,1,[Define to enable WPAD])
fi

AC_ARG_ENABLE(gardenaccounting, [AS_HELP_STRING([--enable-gardenaccounting],[Enable walled garden
accounting])],
    enable_gardenaccounting=$enableval, enable_gardenaccounting=no)

if test x"$enable_gardenaccounting" = xyes; then
    AC_DEFINE(ENABLE_GARDENACCOUNTING,1,[Define to enable walled garden accounting])
fi

AC_ARG_ENABLE(gardenext, [AS_HELP_STRING([--enable-gardenext],[Enable extended walled garden features])],

```

```

    enable_gardenext=$enableval, enable_gardenext=no)

if test x"$enable_gardenext" = xyes; then
    AC_DEFINE(ENABLE_GARDENEXT,1,[Define to enable extended walled garden features])
fi

AC_ARG_ENABLE(inspect, [AS_HELP_STRING([--enable-inspect],[Enable inspect feature in cmdsock])],
    enable_inspect=$enableval, enable_inspect=no)

if test x"$enable_inspect" = xyes; then
    AC_DEFINE(ENABLE_INSPECT,1,[Define to enable inspect feature in cmdsock])
fi

AC_ARG_ENABLE(coa, [AS_HELP_STRING([--disable-coa],[Disable CoA RADIUS support])],
    enable_coa=$enableval, enable_coa=yes)

if test x"$enable_coa" = xyes; then
    AC_DEFINE(ENABLE_COA,1,[Define for CoA RADIUS support])
fi

AC_ARG_ENABLE(dhcpt, [AS_HELP_STRING([--enable-dhcpt],[Enable support for DHCP option setting])],
    enable_dhcpt=$enableval, enable_dhcpt=no)

if test x"$enable_dhcpt" = xyes; then
    AC_DEFINE(ENABLE_DHCP,1,[Define to enable DHCP option setting])
fi

AC_ARG_ENABLE(debug2, [AS_HELP_STRING([--enable-debug2],[Enable verbose debugging])],
    enable_debug2=$enableval, enable_debug2=no)

if test x"$enable_debug2" = xyes; then
    AC_DEFINE(ENABLE_DEBUG2,1,[Define to enable verbose debugging])
fi

AC_ARG_ENABLE(sessgarden, [AS_HELP_STRING([--enable-sessgarden],[Enable support for session-based walled garden])],
    enable_sessgarden=$enableval, enable_sessgarden=no)

if test x"$enable_sessgarden" = xyes; then
    AC_DEFINE(ENABLE_SESSGARDEN,1,[Define to enable Chilli session walled garden])
fi

AC_ARG_ENABLE(sessproxy, [AS_HELP_STRING([--enable-sessproxy],[Enable support for per session postauth proxy])],
    enable_sessproxy=$enableval, enable_sessproxy=no)

if test x"$enable_sessproxy" = xyes; then
    AC_DEFINE(ENABLE_SESSPROXY,1,[Define to enable per session postauth proxy])
fi

AC_ARG_ENABLE(sessdhcp, [AS_HELP_STRING([--enable-sessdhcp],[Enable support for per session DHCP relay])],
    enable_sessdhcp=$enableval, enable_sessdhcp=no)

if test x"$enable_sessdhcp" = xyes; then
    AC_DEFINE(ENABLE_SESSDHCP,1,[Define to enable per session DHCP relay])
fi

AC_ARG_ENABLE(sessdns, [AS_HELP_STRING([--enable-sessdns],[Enable support for per session DNS enforcement])],
    enable_sessdns=$enableval, enable_sessdns=no)

if test x"$enable_sessdns" = xyes; then
    AC_DEFINE(ENABLE_SESSDNS,1,[Define to enable per session DNS enforcement])
fi

AC_ARG_ENABLE(chillixml, [AS_HELP_STRING([--enable-chillixml],[Enable use of chillixml])],
    enable_chillixml=$enableval, enable_chillixml=no)

if test x"$enable_chillixml" = xyes; then
    AC_DEFINE(ENABLE_CHILLIXML,1,[Define to enable Chilli XML])
fi

```

```

AC_ARG_ENABLE(proxyvsa, [AS_HELP_STRING([--enable-proxyvsa],[Enable support for VSA attribute proxy]]),
    enable_proxyvsa=$enableval, enable_proxyvsa=no)

if test x"$enable_proxyvsa" = xyes; then
    AC_DEFINE(ENABLE_PROXYVSA,1,[Define to enable VSA proxy])
fi

AC_ARG_ENABLE(ipwhitelist, [AS_HELP_STRING([--enable-ipwhitelist],[Enable file based IP white list]]),
    enable_ipwhitelist=$enableval, enable_ipwhitelist=no)

if test x"$enable_ipwhitelist" = xyes; then
    AC_DEFINE(ENABLE_IPWHITELIST,1,[Define to support file based whitelists])
fi

AC_ARG_ENABLE(uamdomainfile, [AS_HELP_STRING([--enable-uamdomainfile],[Enable loading of mass uamdomains from
file]]),
    enable_uamdomainfile=$enableval, enable_uamdomainfile=no)

if test x"$enable_uamdomainfile" = xyes; then
    AC_DEFINE(ENABLE_UAMDOMAINFILE,1,[Define to support loading of uamdomains (with regex) from file])
fi

AC_ARG_ENABLE(redirdnsreq, [AS_HELP_STRING([--enable-redirdnsreq],[Enable the sending of a DNS query on
redirect]]),
    enable_redirdnsreq=$enableval, enable_redirdnsreq=no)

if test x"$enable_redirdnsreq" = xyes; then
    AC_DEFINE(ENABLE_REDIRDNSREQ,1,[Define to DNS query on redirect to pick up dynamic walled garden])
fi

AC_ARG_ENABLE(ieee8021q, [AS_HELP_STRING([--disable-ieee8021q],[disable support for IEEE 802.1Q]]),
    enable_ieee8021q=$enableval, enable_ieee8021q=yes)

if test x"$enable_ieee8021q" = xyes; then
    AC_DEFINE(ENABLE_IEEE8021Q,1,[Define to enable Chilli IEEE 802.1Q])
fi

AC_ARG_ENABLE(largelimits, [AS_HELP_STRING([--enable-largelimits],[Enable larger limits for use with non-
embedded systems]]),
    enable_largelimits=$enableval, enable_largelimits=no)

if test x"$enable_largelimits" = xyes; then
    AC_DEFINE(ENABLE_LARGELIMITS,1,[Enable larger limits for use with non-embedded systems])
fi

AC_ARG_WITH([openssl],
    [AS_HELP_STRING([--with-openssl], [enable support for openssl])],[],[with_openssl=no])

OPENSSL=
AS_IF([test x"$with_openssl" != xno],
    [AC_CHECK_LIB([crypto], [CRYPTO_malloc],
        [AC_SUBST([LIBSSL], ["-lssl -lcrypto"])
        AC_DEFINE([HAVE_OPENSSL], [1],
            [Define if you have openssl])
        ],
        [AC_MSG_FAILURE(
            [--with-openssl was given, but test for openssl failed])],
        [-lssl -lcrypto])])

AM_CONDITIONAL(WITH_OPENSSL, [test x"$with_openssl" != xno])

AC_ARG_WITH([matrixssl],
    [AS_HELP_STRING([--with-matrixssl], [enable support for matrixssl])],[],[with_matrixssl=no])

AS_IF([test x"$with_matrixssl" != xno],
    [AC_CHECK_LIB([matrixssl], [matrixSslOpen],
        [AC_SUBST([LIBSSL], ["-lmatrixssl"])
        AC_DEFINE([HAVE_MATRIXSSL], [1],
            [Define if you have matrixssl])
        ],
        ],

```

```

        [AC_MSG_FAILURE(
            [--with-matrixssl was given, but test for matrixssl failed]]),
        [-lmatrixssl]))

AS_IF([test x"$with_matrixssl" != xno],
    [AC_CHECK_HEADERS([matrixSsl.h matrixSsl/matrixSsl.h])])

AM_CONDITIONAL(WITH_MATRIXSSL, [test x"$with_matrixssl" != xno])

AC_ARG_WITH([cyassl],
    [AS_HELP_STRING([--with-cyassl], [enable support for cyassl]),[],[with_cyassl=no])

AS_IF([test x"$with_cyassl" != xno],
    [AC_CHECK_LIB([cyassl], [CyaSSL_Init],
        [AC_SUBST([LIBSSL], ["-lcyassl"])
        AC_DEFINE([HAVE_CYASSL], [1],
            [Define if you have cyassl])
        ],
        [AC_MSG_FAILURE(
            [--with-cyassl was given, but test for cyassl failed]]),
        [-lcyassl])])

AS_IF([test x"$with_cyassl" != xno],
    [AC_CHECK_HEADERS([cyassl/ssl.h ssh.h])])

AM_CONDITIONAL(WITH_CYASSL, [test x"$with_cyassl" != xno])

AM_CONDITIONAL(WITH_SSL, [test x"$with_openssl" != xno || test x"$with_matrixssl" != xno || test
x"$with_cyassl" != xno])

AC_ARG_WITH([matrixssl-cli],
    [AS_HELP_STRING([--with-matrixssl-cli], [enable matrixssl client use]),[],[with_matrixssl_cli=no])

AM_CONDITIONAL(WITH_MATRIXSSL_CLI, [test x"$with_matrixssl_cli" != xno])

AC_ARG_WITH([nfqueue],
    [AS_HELP_STRING([--with-nfqueue], [enable support for netfilter_queue]),[],[with_nfqueue=no])

AS_IF([test x"$with_nfqueue" != xno],
    [AC_CHECK_LIB([netfilter_queue], [nfq_open],
        [AC_SUBST([LIBNETFILTER_QUEUE], ["-lnetfilter_queue -lnfnetlink -lmnl"])
        AC_DEFINE([HAVE_NETFILTER_QUEUE], [1],
            [Define if you have netfilter_queue])
        ],
        [AC_MSG_FAILURE(
            [--with-nfqueue was given, but test for netfilter_queue failed]]),
        [-lnetfilter_queue -lnfnetlink -lmnl])])

AM_CONDITIONAL(WITH_NETFILTER_QUEUE, [test x"$with_nfqueue" != xno])

AC_ARG_WITH([avl],
    [AS_HELP_STRING([--with-avl], [enable support for avl library]),[],[with_avl=no])

AS_IF([test x"$with_avl" != xno],
    [AC_DEFINE([HAVE_AVL], [1], [Define to use avl library])])

AM_CONDITIONAL(WITH_AVL, [test x"$with_avl" != xno])

AC_ARG_WITH([nfcoova],
    [AS_HELP_STRING([--with-nfcoova], [enable support for coova netfilter module]),[],[with_nfcoova=no])

AS_IF([test x"$with_nfcoova" != xno],
    [AC_DEFINE([HAVE_NETFILTER_COOVA], [1], [Define to use coova kernel module])])

AM_CONDITIONAL(WITH_NETFILTER_COOVA, [test x"$with_nfcoova" != xno])

AC_ARG_WITH([sfhash],
    [AS_HELP_STRING([--without-sfhash], [disable SuperFastHash use]),[],[with_sfhash=yes])

AS_IF([test x"$with_sfhash" != xno],
    [AC_DEFINE([HAVE_SFHASH], [1], [Define to use SuperFastHash])])

```

```

AM_CONDITIONAL(WITH_SFHASH, [test x"$with_sfhash" != xno])

AC_ARG_WITH([lookup3],
  [AS_HELP_STRING([--with-lookup3], [enable Jenkins lookup3 use]),[],[with_lookup3=no])

AS_IF([test x"$with_lookup3" != xno],
  [AC_DEFINE([HAVE_LOOKUP3], [1], [Define to use Jenkins lookup3])])

AM_CONDITIONAL(WITH_LOOKUP3, [test x"$with_lookup3" != xno])

AC_ARG_WITH([patricia],
  [AS_HELP_STRING([--with-patricia], [enable Patricia use]),[],[with_patricia=no])

AS_IF([test x"$with_patricia" != xno],
  [AC_DEFINE([HAVE_PATRICIA], [1], [Define to include Patricia])])

AM_CONDITIONAL(WITH_PATRICIA, [test x"$with_patricia" != xno])


AC_ARG_ENABLE([authedallowed],
  [AS_HELP_STRING([--enable-authedallowed], [enable Authorized Garden]),[],[with_authedallowed=no])

AS_IF([test x"$with_authedallowed" != xno],
  [AC_DEFINE([ENABLE_AUTHEDALLOWED], [1], [Define to include Authenticated Garden])])

AM_CONDITIONAL(WITH_AUTHEDALLOWED, [test x"$with_authedallowed" != xno])


AC_ARG_WITH([ipv6],
  [AS_HELP_STRING([--without-ipv6], [enable IPv6]),[],[with_ipv6=yes])

AS_IF([test x"$with_ipv6" != xno],
  [AC_DEFINE([ENABLE_IPV6], [1], [Define to use IPv6])])

AM_CONDITIONAL(WITH_IPV6, [test x"$with_ipv6" != xno])

AC_ARG_WITH([pcap],
  [AS_HELP_STRING([--with-pcap], [enable support for pcap]),[],[with_pcap=no])

AS_IF([test x"$with_pcap" != xno],
  [AC_CHECK_LIB([pcap], [pcap_open_live],
    [AC_SUBST([LIBPCAP], ["-lpcap"])
    AC_DEFINE([USING_PCAP], [1],
      [Define if you have pcap enabled])
    ],
    [AC_MSG_FAILURE(
      [--with-pcap was given, but test for pcap failed]),
    [-lpcap]])])

AM_CONDITIONAL(WITH_PCAP, [test x"$with_pcap" != xno])

AC_ARG_WITH([curl],
  [AS_HELP_STRING([--with-curl], [enable support for curl]),[],[with_curl=no])

AS_IF([test x"$with_curl" != xno],
  [AC_CHECK_LIB([curl], [curl_global_init],
    [AC_SUBST([LIBCURL], ["-lcurl -lz -lssl -lcrypto -lcares"])
    AC_DEFINE([USING_CURL], [1],
      [Define if you have curl enabled])
    ],
    [AC_MSG_FAILURE(
      [--with-curl was given, but test for curl failed]),
    [-lz -lssl -lcrypto -lcares]])])

AM_CONDITIONAL(WITH_CURL, [test x"$with_curl" != xno])

AC_ARG_WITH([mmap],
  [AS_HELP_STRING([--with-mmap], [enable support for mmap]),[],[with_mmap=no])

AS_IF([test x"$with_mmap" != xno],

```



```

[AC_DEFINE([USING_MMMap], [1], [Define if you have mmap enabled])])

AC_ARG_WITH([poll],
  [AS_HELP_STRING([--with-poll], [enable support for poll]),[],[with_poll=no])

AS_IF([test x"$with_poll" != xno],
  [AC_DEFINE([USING_POLL], [1], [Define if you have poll() enabled])])

AC_ARG_WITH([ipc-msg],
  [AS_HELP_STRING([--with-ipc-msg], [enable support for msgsnd/msgrcv SV IPC]),[],[with_ipcmMsg=no])

AS_IF([test x"$with_ipcmMsg" != xno],
  [AC_DEFINE([USING_IPC_MSG], [1], [Define to use SV IPC message queue])])

AC_ARG_ENABLE(binstatfile, [AS_HELP_STRING([--enable-binstatfile], [Enable support for binary status
file])],
  enable_binstatfile=$enableval, enable_binstatfile=no)

if test x"$enable_binstatfile" = xyes; then
  AC_DEFINE(ENABLE_BINSTATFILE,1,[Define to enable binary status file])
fi

AC_ARG_ENABLE(statusfile, [AS_HELP_STRING([--enable-statusfile], [Enable support for status file])],
  enable_statfile=$enableval, enable_statfile=no)

if test x"$enable_statfile" = xyes || test x"$enable_binstatfile" = xyes; then
  AC_DEFINE(ENABLE_STATFILE,1,[Define to enable status file])
fi

AC_ARG_ENABLE(chilliproxy, [AS_HELP_STRING([--enable-chilliproxy], [Enable support for HTTP AAA Proxy])],
  enable_chilliproxy=$enableval, enable_chilliproxy=no)

if test x"$enable_chilliproxy" = xyes; then
  AC_DEFINE(ENABLE_CHILLIPROXY,1,[Define to enable HTTP AAA Proxy])
fi

AM_CONDITIONAL(WITH_CHILLIPROXY, [test x"$enable_chilliproxy" = xyes])

AC_ARG_ENABLE(multiroute, [AS_HELP_STRING([--enable-multiroute], [Enable support for multiple routes])],
  enable_multiroute=$enableval, enable_multiroute=no)

if test x"$enable_multiroute" = xyes; then
  AC_DEFINE(ENABLE_MULTIRoute,1,[Define to enable multiple routes])
fi

AM_CONDITIONAL(WITH_MULTIRoute, [test x"$enable_multiroute" = xyes])

AC_ARG_ENABLE(multilan, [AS_HELP_STRING([--enable-multilan], [Enable support for multiple LANs])],
  enable_multilan=$enableval, enable_multilan=no)

if test x"$enable_multilan" = xyes; then
  AC_DEFINE(ENABLE_MULTILAN,1,[Define to enable multiple LANs])
fi

AM_CONDITIONAL(WITH_MULTILAN, [test x"$enable_multilan" = xyes])

AC_ARG_ENABLE(chilliradsec, [AS_HELP_STRING([--enable-chilliradsec], [Enable support for RadSec AAA Proxy])],
  enable_chilliradsec=$enableval, enable_chilliradsec=no)

if test x"$enable_chilliradsec" = xyes; then
  AC_DEFINE(ENABLE_CHILLIRADSEC,1,[Define to enable RadSec AAA Proxy])
fi

AM_CONDITIONAL(WITH_CHILLIRADSEC, [test x"$enable_chilliradsec" = xyes])

AC_ARG_ENABLE(chilliredir, [AS_HELP_STRING([--enable-chilliredir], [Enable support for Redir server])],
  enable_chilliredir=$enableval, enable_chilliredir=no)

if test x"$enable_chilliredir" = xyes; then
  AC_DEFINE(ENABLE_CHILLIREDIR,1,[Define to enable Redir server])
fi

```

```

AM_CONDITIONAL(WITH_CHILLIREDIR, [test x"$enable_chilliredir" = xyes])

AC_ARG_ENABLE(chilliscript, [AS_HELP_STRING([--enable-chilliscript],[Enable support for chilli_script
helper])],
    enable_chilliscript=$enableval, enable_chilliscript=no)

if test x"$enable_chilliscript" = xyes; then
    AC_DEFINE(ENABLE_CHILLIScript,1,[Define to enable chilli_script helper])
fi

AM_CONDITIONAL(WITH_CHILLIScript, [test x"$enable_chilliscript" = xyes])

AC_ARG_ENABLE(cluster, [AS_HELP_STRING([--enable-cluster],[Enable support for clustering])],
    enable_cluster=$enableval, enable_cluster=no)

if test x"$enable_cluster" = xyes; then
    AC_DEFINE(ENABLE_CLUSTER,1,[Define to enable cluster])
fi

AM_CONDITIONAL(WITH_CLUSTER, [test x"$enable_cluster" = xyes])

AC_ARG_ENABLE(sessionstate, [AS_HELP_STRING([--enable-sessionstate],[Enable extended use of the CoovaChilli-
Session-State attribute])],
    enable_sessionstate=$enableval, enable_sessionstate=no)

if test x"$enable_sessionstate" = xyes; then
    AC_DEFINE(ENABLE_SESSIONSTATE,1,[Define to enable extended use of the CoovaChilli-Session-State attribute])
fi

AC_ARG_ENABLE(sessionid, [AS_HELP_STRING([--enable-sessionid],[Enable the use of the CoovaChilli-Session-Id
attribute])],
    enable_sessionid=$enableval, enable_sessionid=no)

if test x"$enable_sessionid" = xyes; then
    AC_DEFINE(ENABLE_SESSIONID,1,[Define to enable the use of the CoovaChilli-Session-Id attribute])
fi

AC_ARG_ENABLE(apsessionid, [AS_HELP_STRING([--enable-apsessionid],[Enable the use of the CoovaChilli-AP-Session-
Id attribute])],
    enable_apsessionid=$enableval, enable_apsessionid=no)

if test x"$enable_apsessionid" = xyes; then
    AC_DEFINE(ENABLE_APSESSIONID,1,[Define to enable the use of the CoovaChilli-AP-Session-Id attribute])
fi

AC_ARG_ENABLE(coovachilliconfig, [AS_HELP_STRING([--disable-coovachilliconfig],[Disable the use of the
CoovaChilli-Config attribute])],
    enable_coovachilliconfig=$enableval, enable_coovachilliconfig=yes)

if test x"$enable_coovachilliconfig" = xyes; then
    AC_DEFINE(ENABLE_COOVACHILLICONFIG,1,[Define to enable the use of the CoovaChilli-Config attribute])
fi

AC_ARG_ENABLE(mdns, [AS_HELP_STRING([--enable-mdns],[Enable support for MDNS])],
    enable_mdns=$enableval, enable_mdns=no)

if test x"$enable_mdns" = xyes; then
    AC_DEFINE(ENABLE_MDNS,1,[Define to enable mDNS])
fi

AM_CONDITIONAL(WITH_MDNS, [test x"$enable_mdns" = xyes])

AC_ARG_ENABLE(netbios, [AS_HELP_STRING([--enable-netbios],[Enable support for NetBIOS])],
    enable_netbios=$enableval, enable_netbios=no)

if test x"$enable_netbios" = xyes; then
    AC_DEFINE(ENABLE_NETBIOS,1,[Define to enable NetBIOS])
fi

AM_CONDITIONAL(WITH_NETBIOS, [test x"$enable_netbios" = xyes])

```

```

AC_ARG_ENABLE(ieee8023, [AS_HELP_STRING([--enable-ieee8023],[Enable support for IEEE 802.3 Ethernet]]),
    enable_ieee8023=$enableval, enable_ieee8023=no)

if test x"$enable_ieee8023" = xyes; then
    AC_DEFINE(ENABLE_IEEE8023,1,[Define to enable IEEE 802.3])
fi

AM_CONDITIONAL(WITH_IEEE8023, [test x"$enable_ieee8023" = xyes])

AC_ARG_ENABLE(pppoe, [AS_HELP_STRING([--enable-pppoe],[Enable support for PPPoE]]),
    enable_pppoe=$enableval, enable_pppoe=no)

if test x"$enable_pppoe" = xyes; then
    AC_DEFINE(ENABLE_PPPoE,1,[Define to enable PPPoE])
fi

AM_CONDITIONAL(WITH_PPPoE, [test x"$enable_pppoe" = xyes])

AC_ARG_ENABLE(l2tpppp, [AS_HELP_STRING([--enable-l2tpppp],[Enable support for L2TP/PPP Tunneling]]),
    enable_l2tpppp=$enableval, enable_l2tpppp=no)

if test x"$enable_l2tpppp" = xyes; then
    AC_DEFINE(ENABLE_L2TP_PPP,1,[Define to enable L2TP/PPP])
fi

AM_CONDITIONAL(WITH_L2TP_PPP, [test x"$enable_l2tpppp" = xyes])

AC_ARG_ENABLE(eapol, [AS_HELP_STRING([--enable-eapol],[Enable support for EAPOL]]),
    enable_eapol=$enableval, enable_eapol=no)

if test x"$enable_eapol" = xyes; then
    AC_DEFINE(ENABLE_EAPOL,1,[Define to enable EAPOL])
fi

AM_CONDITIONAL(WITH_EAPOL, [test x"$enable_eapol" = xyes])

AC_ARG_ENABLE(miniportal, [AS_HELP_STRING([--enable-miniportal],[Enable support Coova miniportal]]),
    enable_miniportal=$enableval, enable_miniportal=no)

if test x"$enable_miniportal" = xyes ; then
    AC_DEFINE(ENABLE_MINIportal,1,[Define to enable Coova miniportal])
fi

AM_CONDITIONAL(WITH_MINIportal, [test x"$enable_miniportal" = xyes])

AC_ARG_ENABLE(miniconfig, [AS_HELP_STRING([--enable-miniconfig],[Enable support minimal cmdline config]]),
    enable_miniconfig=$enableval, enable_miniconfig=no)

if test x"$enable_miniconfig" = xyes ; then
    AC_DEFINE(ENABLE_MINICONFIG,1,[Define to enable minimal cmdline config])
fi

AM_CONDITIONAL(WITH_MINICONFIG, [test x"$enable_miniconfig" = xyes])

AC_ARG_ENABLE(ewtapi, [AS_HELP_STRING([--enable-ewtapi],[Enable support for CoovaEWT API]]),
    enable_ewtapi=$enableval, enable_ewtapi=no)

if test x"$enable_ewtapi" = xyes; then
    AC_DEFINE(ENABLE_EWTAPI,1,[Define to enable CoovaEWT API])
fi

AC_ARG_ENABLE(libjson, [AS_HELP_STRING([--enable-libjson],[Enable support for libjson]]),
    enable_libjson=$enableval, enable_libjson=no)

AM_CONDITIONAL(WITH_EWTAPI, [test x"$enable_ewtapi" = xyes])
AM_CONDITIONAL(WITH_JSONLIB, [test x"$enable_ewtapi" = xyes || test x"$enable_libjson" = xyes])

AC_ARG_ENABLE(ssdp, [AS_HELP_STRING([--enable-ssdp],[Enable support for Simple Service Discovery Protocol]]),
    enable_ssdp=$enableval, enable_ssdp=no)

```

```

if test x"$enable_ssd" = xyes ; then
    AC_DEFINE(ENABLE_SSDP,1,[Define to enable Simple Service Discovery Protocol])
fi

AC_ARG_ENABLE(layer3, [AS_HELP_STRING([--enable-layer3],[Enable support for Layer3 only operation])],
    enable_layer3=$enableval, enable_layer3=no)

if test x"$enable_layer3" = xyes ; then
    AC_DEFINE(ENABLE_LAYER3,1,[Define to enable Layer3 only support])
fi

AC_ARG_ENABLE(modules, [AS_HELP_STRING([--enable-modules],[Enable dynamically loadable modules
(experimental)])],
    enable_modules=$enableval, enable_modules=no)

if test x"$enable_modules" = xyes ; then
    AC_DEFINE(ENABLE_MODULES,1,[Define to enable dynamically loadable modules])
fi

AM_CONDITIONAL(WITH_MODULES, [test x"$enable_modules" = xyes])

AC_ARG_ENABLE(extadmvs, [AS_HELP_STRING([--enable-extadmvs],[Enable admin-user VSA attribute support])],
    enable_extadmvs=$enableval, enable_extadmvs=no)

if test x"$enable_extadmvs" = xyes; then
    AC_DEFINE(ENABLE_EXTADMVS,1,[Define to enable admin-user VSA support])
fi

AC_ARG_ENABLE(redirinject, [AS_HELP_STRING([--enable-redirinject],[Enable Redir content-injection support])],
    enable_redirinject=$enableval, enable_redirinject=no)

if test x"$enable_redirinject" = xyes; then
    AC_DEFINE(ENABLE_REDIRINJECT,1,[Define to Redir content-injection support])
fi

AM_CONDITIONAL(WITH_REDIRINJECT, [test x"$enable_redirinject" = xyes])

AC_ARG_ENABLE(netnat, [AS_HELP_STRING([--enable-netnat],[Enable net interface nat (experimental)])],
    enable_netnat=$enableval, enable_netnat=no)

if test x"$enable_netnat" = xyes ; then
    AC_DEFINE(ENABLE_NETNAT,1,[Define to enable network interface nat])
fi

AM_CONDITIONAL(WITH_NETNAT, [test x"$enable_netnat" = xyes])

AC_ARG_ENABLE(useragent, [AS_HELP_STRING([--enable-useragent],[Enable recording user-agent])],
    enable_useragent=$enableval, enable_useragent=no)

if test x"$enable_useragent" = xyes ; then
    AC_DEFINE(ENABLE_USERAGENT,1,[Define enable the recording of user-agent])
fi

AC_ARG_ENABLE(acceptlanguage, [AS_HELP_STRING([--enable-acceptlanguage],[Enable recording Accept-Language])],
    enable_acceptlanguage=$enableval, enable_acceptlanguage=no)

if test x"$enable_acceptlanguage" = xyes ; then
    AC_DEFINE(ENABLE_ACCEPTLANGUAGE,1,[Define enable the recording of accept-language])
fi

AC_ARG_ENABLE(location, [AS_HELP_STRING([--enable-location],[Enable Location Awareness])],
    enable_location=$enableval, enable_location=no)

if test x"$enable_location" = xyes ; then
    AC_DEFINE(ENABLE_LOCATION,1,[Define enable Location Awareness])
fi

AM_CONDITIONAL(WITH_LOCATION, [test x"$enable_location" = xyes])

AC_ARG_ENABLE(forcedns, [AS_HELP_STRING([--enable-forcedns],[Enable the forcing (NAT) of DNS])],

```

```

enable_forcedns=$enableval, enable_forcedns=no)

if test x"$enable_forcedns" = xyes ; then
    AC_DEFINE(ENABLE_FORCEDNS,1,[Enable the forcing (NAT) of DNS])
fi

AC_CHECK_LIB(rt, clock_gettime)

AM_CONDITIONAL([HAVE_STRLCPY], [test "x$ac_cv_func_strlcpy" = xyes])

AC_ARG_ENABLE(config,
[ --enable-config=file],
[ if test "x$enableval" != "x" ; then
    config_file=$enableval
  else
    echo "Error! Needs an argument"
    exit -1
  fi
])

if test x"$config_file" != x ; then
    AC_DEFINE_UNQUOTED(ENABLE_CONFIG,"$config_file",[none])
fi

AM_CONDITIONAL(WITH_CONFIG, [test x"$enable_config" = xyes])

AC_SUBST(sysconfdir)
AC_CONFIG_FILES([Makefile
    bstring/Makefile
    conf/Makefile
    debian/Makefile
    distro/Makefile
    doc/Makefile
    json/Makefile
    miniportal/Makefile
    src/Makefile
    src/mssl/Makefile
    www/Makefile
    distro/suse/coova-chilli.spec
    distro/redhat/coova-chilli.spec])

AC_OUTPUT

```

We expose more compile options in the **Makefile**.

```

#
# Copyright (C) 2007-2017 OpenWrt.org
#
# This is free software, licensed under the GNU General Public License v2.
# See /LICENSE for more information.
#

include $(TOPDIR)/rules.mk

PKG_NAME:=coova-chilli
PKG_VERSION:=1.4
PKG_MAINTAINER:=Xavier Maysonnave <x.maysonnave@auroville.org.in>
PKG_LICENSE:=GPL-2.0+
PKG_LICENSE_FILES:=COPYING
PKG_RELEASE:=4

PKG_SOURCE_PROTO:=git
PKG_SOURCE_URL:=git://github.com/coova/coova-chilli
PKG_SOURCE_SUBDIR:=$(PKG_NAME)-$(PKG_VERSION)
PKG_SOURCE_VERSION:=1.4
PKG_SOURCE:=$(PKG_NAME)-$(PKG_VERSION).tar.gz

PKG_INSTALL:=1

```

```
PKG_CONFIG_DEPENDS := \
COOVACHILLI_DISABLE_CHILLIQUERY \
COOVACHILLI_DISABLE_LEAKYBUCKET \
COOVACHILLI_DISABLE_UAMANYIP \
COOVACHILLI_DISABLE_UAMUIPORT \
COOVACHILLI_DISABLE_ACCOUNTING_ONOFF \
COOVACHILLI_DISABLE_TAP \
COOVACHILLI_DISABLE_TCPRESET \
COOVACHILLI_DISABLE_RADPROXY \
COOVACHILLI_ENABLE_JSON \
COOVACHILLI_DISABLE_DEBUG \
COOVACHILLI_DISABLE_DHCPADIUS \
COOVACHILLI_ENABLE_WPAD \
COOVACHILLI_ENABLE_GARDENACCOUNTING \
COOVACHILLI_ENABLE_GARDENEXT \
COOVACHILLI_ENABLE_INSPECT \
COOVACHILLI_DISABLE_COA \
COOVACHILLI_ENABLE_DHCPOPT \
COOVACHILLI_ENABLE_DEBUG2 \
COOVACHILLI_ENABLE_SESSGARDEN \
COOVACHILLI_ENABLE_SESSPROXY \
COOVACHILLI_ENABLE_SESSDHCP \
COOVACHILLI_ENABLE_SESSDNS \
COOVACHILLI_ENABLE_CHILLIXML \
COOVACHILLI_ENABLE_PROXYVSA \
COOVACHILLI_ENABLE_IPWHITELIST \
COOVACHILLI_ENABLE_UAMDOMAINFILE \
COOVACHILLI_ENABLE_REDIRDNSREQ \
COOVACHILLI_DISABLE_IEEE8021Q \
COOVACHILLI_ENABLE_LARGELIMITS \
COOVACHILLI_WITH_NFQUEUE \
COOVACHILLI_WITH_AVL \
COOVACHILLI_WITH_NFCOOVA \
COOVACHILLI_WITHOUT_SFHASH \
COOVACHILLI_WITH_LOOKUP3 \
COOVACHILLI_WITH_PATRICIA \
COOVACHILLI_ENABLE_AUTHEDALLOWED \
COOVACHILLI_WITHOUT_IPV6 \
COOVACHILLI_WITH_PCAP \
COOVACHILLI_WITH_CURL \
COOVACHILLI_WITH_MMAP \
COOVACHILLI_WITH_POLL \
COOVACHILLI_WITH_IPC_MSG \
COOVACHILLI_ENABLE_BINSTATUSFILE \
COOVACHILLI_ENABLE_STATUSFILE \
COOVACHILLI_ENABLE_CHILLIPROXY \
COOVACHILLI_ENABLE_MULTIRROUTE \
COOVACHILLI_ENABLE_MULTILAN \
COOVACHILLI_ENABLE_CHILLIRADSEC \
COOVACHILLI_ENABLE_CHILLIREDIR \
COOVACHILLI_ENABLE_CHILLISCRIP \
COOVACHILLI_ENABLE_CLUSTER \
COOVACHILLI_ENABLE_SESSIONSTATE \
COOVACHILLI_ENABLE_SESSIONID \
COOVACHILLI_ENABLE_APSESSIONID \
COOVACHILLI_DISABLE_COOVACHILLICONFIG \
COOVACHILLI_ENABLE_MDNS \
COOVACHILLI_ENABLE_NETBIOS \
COOVACHILLI_ENABLE_IEEE8023 \
COOVACHILLI_ENABLE_PPPOE \
COOVACHILLI_ENABLE_L2TPPPP \
COOVACHILLI_ENABLE_EAPOL \
COOVACHILLI_ENABLE_MINIPORTAL \
COOVACHILLI_ENABLE_MINICONFIG \
COOVACHILLI_ENABLE_EWTAPI \
COOVACHILLI_ENABLE_LIBJSON \
COOVACHILLI_ENABLE_SSDP \
COOVACHILLI_ENABLE_LAYER3 \
COOVACHILLI_ENABLE_MODULES \
COOVACHILLI_ENABLE_EXTADMVSA \
```

```

COOVACHILLI_ENABLE_REDIRINJECT \
COOVACHILLI_ENABLE_NETNAT \
COOVACHILLI_ENABLE_USERAGENT \
COOVACHILLI_ENABLE_ACCEPTLANGUAGE \
COOVACHILLI_ENABLE_LOCATION \
COOVACHILLI_ENABLE_FORCEDNS \
COOVACHILLI_DISABLE_SSL \
COOVACHILLI_ENABLE_OPENSSL \
COOVACHILLI_ENABLE_MATRIXSSL \
COOVACHILLI_ENABLE_MATRIXSSL_CLI \
COOVACHILLI_ENABLE_CYASSL

include $(INCLUDE_DIR)/package.mk
include $(INCLUDE_DIR)/kernel.mk

define Package/coova-chilli
    SUBMENU:=Captive Portals
    SECTION:=net
    CATEGORY:=Network
    DEPENDS:= \
        +kmod-tun \
        +librt \
        +COOVACHILLI_ENABLE_MATRIXSSL:libmatrixssl \
        +COOVACHILLI_ENABLE_CYASSL:libcyassl \
        +COOVACHILLI_ENABLE_OPENSSL:libopenssl \
        +COOVACHILLI_ENABLE_LIBJSON:libjson-c \
        +COOVACHILLI_WITH_NFQUEUE:libnetfilter-queue \
        +COOVACHILLI_WITH_NFQUEUE:libnfnetlink \
        +COOVACHILLI_WITH_NFQUEUE:libmnl \
        +COOVACHILLI_WITH_PCAP:libpcap \
        +COOVACHILLI_WITH_CURL:libcares
    TITLE:=Wireless LAN HotSpot controller (Coova Chilli Version)
    URL:=http://www.coova.org/CoovaChilli
    MENU:=1
endef

define Package/coova-chilli/description
    CoovaChilli is an open source access controller for wireless LAN
    access points and is based on ChilliSpot. It is used for authenticating
    users of a wireless (or wired) LAN. It supports web based login (UAM)
    which is today's standard for public HotSpots and it supports Wireless
    Protected Access (WPA) which is the standard of the future.
    Authentication, authorization and accounting (AAA) is handled by your
    favorite radius server.
endef

define Package/coova-chilli/config
    source "$(SOURCE)/Config.in"
endef

define KernelPackage/ipt-coova
    URL:=http://www.coova.org/CoovaChilli
    SUBMENU:=Netfilter Extensions
    DEPENDS:=coova-chilli +kmod-ipt-core +libxtables
    TITLE:=Coova netfilter module
    FILES:=$(PKG_BUILD_DIR)/src/linux/xt_*.$(LINUX_KMOD_SUFFIX)
    AUTOLOAD:=$(call AutoProbe,xt_coova)
endef

define KernelPackage/ipt-coova/description
    Netfilter kernel module for CoovaChilli
    Includes:
    - coova
endef

DISABLE_NLS=

TARGET_CFLAGS += $(FPIC)

CONFIGURE_VARS += \
    ARCH="$(LINUX_KARCH)" \

```

```

        KERNEL_DIR="$(LINUX_DIR)"

MAKE_FLAGS += \
    ARCH="$(LINUX_KARCH)" \
    KERNEL_DIR="$(LINUX_DIR)"

MAKE_INSTALL_FLAGS += \
    ARCH="$(LINUX_KARCH)" \
    KERNEL_DIR="$(LINUX_DIR)" \
    INSTALL_MOD_PATH="$(PKG_INSTALL_DIR)"

define Build/Prepare
$(call Build/Prepare/Default)
    ( cd $(PKG_BUILD_DIR) ; \
        [ -f ./configure ] || { \
            ./bootstrap ; \
        } \
    )
endef

define Build/Configure
$(call Build/Configure/Default, \
$(if $(CONFIG_COOVACHILLI_DISABLE_CHILLIQUERY),--disable-chilliquery) \
$(if $(CONFIG_COOVACHILLI_DISABLE_LEAKYBUCKET),--disable-leakybucket) \
$(if $(CONFIG_COOVACHILLI_DISABLE_UAMANYIP),--disable-uamanyip) \
$(if $(CONFIG_COOVACHILLI_DISABLE_UAMUIPORT),--disable-uamuiport) \
$(if $(CONFIG_COOVACHILLI_DISABLE_ACCOUNTING_ONOFF),--disable-accounting-onoff) \
$(if $(CONFIG_COOVACHILLI_DISABLE_TAP),--disable-tap) \
$(if $(CONFIG_COOVACHILLI_DISABLE_TCPRESET),--disable-tcpreset) \
$(if $(CONFIG_COOVACHILLI_DISABLE_RADPROXY),--disable-radproxy) \
$(if $(CONFIG_COOVACHILLI_ENABLE_JSON),--enable-json) \
$(if $(CONFIG_COOVACHILLI_DISABLE_DEBUG),--disable-debug) \
$(if $(CONFIG_COOVACHILLI_DISABLE_DHCPRADIUS),--disable-dhcpradius) \
$(if $(CONFIG_COOVACHILLI_ENABLE_WPAD),--enable-wpad) \
$(if $(CONFIG_COOVACHILLI_ENABLE_GARDENACCOUNTING),--enable-gardenaccounting,) \
$(if $(CONFIG_COOVACHILLI_ENABLE_GARDENEXT),--enable-gardenext) \
$(if $(CONFIG_COOVACHILLI_ENABLE_INSPECT),--enable-inspect) \
$(if $(CONFIG_COOVACHILLI_DISABLE_COA),--disable-coa) \
$(if $(CONFIG_COOVACHILLI_ENABLE_DHCPD),--enable-dhcpd) \
$(if $(CONFIG_COOVACHILLI_ENABLE_DEBUG2),--enable-debug2) \
$(if $(CONFIG_COOVACHILLI_ENABLE_SESSGARDEN),--enable-sessgarden) \
$(if $(CONFIG_COOVACHILLI_ENABLE_SESSPROXY),--enable-sessproxy) \
$(if $(CONFIG_COOVACHILLI_ENABLE_SESSDHCP),--enable-sessdhcp) \
$(if $(CONFIG_COOVACHILLI_ENABLE_SESSDNS),--enable-sessdns) \
$(if $(CONFIG_COOVACHILLI_ENABLE_CHILLIXML),--enable-chillixml) \
$(if $(CONFIG_COOVACHILLI_ENABLE_PROXYVSA),--enable-proxyvsa,) \
$(if $(CONFIG_COOVACHILLI_ENABLE_IPWHITELIST),--enable-ipwhitelist) \
$(if $(CONFIG_COOVACHILLI_ENABLE_UAMDOMAINFILE),--enable-uamdomainfile) \
$(if $(CONFIG_COOVACHILLI_ENABLE_REDIRDNSREQ),--enable-redirdnsreq) \
$(if $(CONFIG_COOVACHILLI_DISABLE_IEEE8021Q),--disable-ieee8021q) \
$(if $(CONFIG_COOVACHILLI_ENABLE_LARGE_LIMITS),--enable-large-limits) \
$(if $(CONFIG_COOVACHILLI_ENABLE_OPENSSL),--with-openssl) \
$(if $(CONFIG_COOVACHILLI_ENABLE_MATRIXSSL),--with-matrixssl) \
$(if $(CONFIG_COOVACHILLI_ENABLE_MATRIXSSL_CLI),--with-matrixssl-cli) \
$(if $(CONFIG_COOVACHILLI_ENABLE_CYASSL),--with-cyassl) \
$(if $(CONFIG_COOVACHILLI_WITH_NFQUEUE),--with-nfqueue) \
$(if $(CONFIG_COOVACHILLI_WITH_AVL),--with-avl) \
$(if $(CONFIG_COOVACHILLI_WITH_NFCOOVA),--with-nfcoova) \
$(if $(CONFIG_COOVACHILLI_WITHOUT_SFHASH),--without-sfhash) \
$(if $(CONFIG_COOVACHILLI_WITH_LOOKUP3),--with-lookup3) \
$(if $(CONFIG_COOVACHILLI_WITH_PATRICIA),--with-patricia) \
$(if $(CONFIG_COOVACHILLI_ENABLE_AUTHEDALLOWED),--enable-authedallowed) \
$(if $(CONFIG_COOVACHILLI_WITHOUT_IPV6),--without-ipv6) \
$(if $(CONFIG_COOVACHILLI_WITH_PCAP),--with-pcap) \
$(if $(CONFIG_COOVACHILLI_WITH_CURL),--with-curl) \
$(if $(CONFIG_COOVACHILLI_WITH_MMAP),--with-mmap) \
$(if $(CONFIG_COOVACHILLI_WITH_POLL),--with-poll) \
$(if $(CONFIG_COOVACHILLI_WITH_IPC_MSG),--with-ipc-msg) \
$(if $(CONFIG_COOVACHILLI_ENABLE_BINSTATUSFILE),--enable-binstatusfile) \
$(if $(CONFIG_COOVACHILLI_ENABLE_STATUSFILE),--enable-statusfile) \
$(if $(CONFIG_COOVACHILLI_ENABLE_CHILLIPROXY),--enable-chilliproxy) \

```



```

$(if $(CONFIG_COOVACHILLI_ENABLE_MULTIRROUTE),--enable-multiroute) \
$(if $(CONFIG_COOVACHILLI_ENABLE_MULTILAN),--enable-multilan) \
$(if $(CONFIG_COOVACHILLI_ENABLE_CHILLIRADSEC),--enable-chilliradsec) \
$(if $(CONFIG_COOVACHILLI_ENABLE_CHILLIREDIR),--enable-chilliredir) \
$(if $(CONFIG_COOVACHILLI_ENABLE_CHILLISCRIP),--enable-chilliscrpt) \
$(if $(CONFIG_COOVACHILLI_ENABLE_CLUSTER),--enable-cluster) \
$(if $(CONFIG_COOVACHILLI_ENABLE_SESSIONSTATE),--enable-sessionstate) \
$(if $(CONFIG_COOVACHILLI_ENABLE_SESSIONID),--enable-sessionid) \
$(if $(CONFIG_COOVACHILLI_ENABLE_APSESSIONID),--enable-apsessionid) \
$(if $(CONFIG_COOVACHILLI_DISABLE_COOVACHILLICONFIG),--disable-coovachilliconfig) \
$(if $(CONFIG_COOVACHILLI_ENABLE_MDNS),--enable-mdns) \
$(if $(CONFIG_COOVACHILLI_ENABLE_NETBIOS),--enable-netbios) \
$(if $(CONFIG_COOVACHILLI_ENABLE_IEEE8023),--enable-ieee8023) \
$(if $(CONFIG_COOVACHILLI_ENABLE_PPPOE),--enable-pppoe) \
$(if $(CONFIG_COOVACHILLI_ENABLE_L2TPPPP),--enable-l2tp PPP) \
$(if $(CONFIG_COOVACHILLI_ENABLE_EAPOL),--enable-eapol) \
$(if $(CONFIG_COOVACHILLI_ENABLE_MINIportal),--enable-miniportal) \
$(if $(CONFIG_COOVACHILLI_ENABLE_MINICONFIG),--enable-miniconfig) \
$(if $(CONFIG_COOVACHILLI_ENABLE_EWTAPI),--enable-ewtapi) \
$(if $(CONFIG_COOVACHILLI_ENABLE_LIBJSON),--enable-libjson) \
$(if $(CONFIG_COOVACHILLI_ENABLE_SSDP),--enable-ssdp) \
$(if $(CONFIG_COOVACHILLI_ENABLE_LAYER3),--enable-layer3) \
$(if $(CONFIG_COOVACHILLI_ENABLE_MODULES),--enable-modules) \
$(if $(CONFIG_COOVACHILLI_ENABLE_EXTADMVSA),--enable-extadmvs) \
$(if $(CONFIG_COOVACHILLI_ENABLE_REDIRINJECT),--enable-redirinject) \
$(if $(CONFIG_COOVACHILLI_ENABLE_NETNAT),--enable-netnat) \
$(if $(CONFIG_COOVACHILLI_ENABLE_USERAGENT),--enable-useragent) \
$(if $(CONFIG_COOVACHILLI_ENABLE_ACCEPTLANGUAGE),--enable-acceptlanguage) \
$(if $(CONFIG_COOVACHILLI_ENABLE_LOCATION),--enable-location) \
$(if $(CONFIG_COOVACHILLI_ENABLE_FORCEDNS),--enable-forcedns) \
$(if $(CONFIG_PACKAGE_kmod-ipt-coova),--with-nfcoova) \
)
endif

define Package/coova-chilli/conffiles
/etc/config/chilli
endif

define Package/coova-chilli/install
$(INSTALL_DIR) $(1)/etc
$(INSTALL_CONF) $(PKG_INSTALL_DIR)/etc/chilli.conf $(1)/etc/
$(INSTALL_DIR) $(1)/etc/chilli
$(CP) $(PKG_INSTALL_DIR)/etc/chilli/* $(1)/etc/chilli/
$(INSTALL_DIR) $(1)/etc/hotplug.d/iface
$(INSTALL_DATA) ./files/chilli.hotplug $(1)/etc/hotplug.d/iface/30-chilli
$(INSTALL_DIR) $(1)/usr/sbin
$(INSTALL_BIN) $(PKG_INSTALL_DIR)/usr/sbin/chilli* $(1)/usr/sbin/
$(INSTALL_DIR) $(1)/usr/lib/
$(CP) $(PKG_INSTALL_DIR)/usr/lib/lib*.so.* $(1)/usr/lib/
$(if $(CONFIG_PACKAGE_kmod-ipt-coova), \
    $(INSTALL_DIR) $(1)/usr/lib/iptables; \
    $(CP) $(PKG_INSTALL_DIR)/usr/lib/iptables/lib*.so $(1)/usr/lib/iptables/ \
)
$(INSTALL_DIR) $(1)/etc/init.d
$(INSTALL_BIN) files/chilli.init $(1)/etc/init.d/chilli
$(INSTALL_DIR) $(1)/etc/config
$(INSTALL_DATA) files/chilli.config $(1)/etc/config/chilli
$(INSTALL_DIR) $(1)/lib/firewall
$(CP) files/chilli.firewall $(1)/lib/firewall/chilli.sh
endif

$(eval $(call BuildPackage,coova-chilli))
$(eval $(call KernelPackage,ipt-coova))

```

Patches

Under the [package/feeds/packages/coova-chilli/patches](#) directory.

Remove useless patches

Remove the patches applicable for **coova-chilli-1.3.0+20141128**

1. **100-fix-sysinfo-redeclaration.patch**
2. **201-fix_dereferencing_pointers.patch**
3. **300-fix-compile-with-cyassl.patch**
4. **400-fix-compile-with-musl.patch**

Patch the **200-fix_compile_kmod.patch**

Only the file **200-fix_compile_kmod.patch** should remain in the patches directory.

Open it and patch with the following content:

```
--- a/src/linux/Makefile      2016-12-15 22:58:43.000000000 +0530
+++ b/src/linux/Makefile.new  2017-04-17 14:34:42.982149206 +0530
@@ -22,11 +22,11 @@
     $(CC) $(CFLAGS) -shared -o $@ $^;

lib%.o: lib%.c
-    $(CC) $(CFLAGS) -fPIC -O2 -Wall -D_INIT=lib$_init -c -o $@ $<;
+    $(CC) $(CFLAGS) -D_INIT=lib$_init -c -o $@ $<;

install: modules_install libxt_coova.so
-    mkdir -p $(DESTDIR)/lib/xtables/
-    cp libxt_coova.so $(DESTDIR)/lib/xtables/
+    mkdir -p $(DESTDIR)/usr/lib/iptables/
+    cp libxt_coova.so $(DESTDIR)/usr/lib/iptables/

distdir:
```

Config.in

OpenWRT and LEDE 17.01 doesn't expose all the compile options of Coova-Chilli.

Here is an enhancement of compile options exposed to the OpenWRT/LEDE builder.

Following the compile definitions from <https://github.com/coova/coova-chilli/blob/master/configure.ac>

Open the **package/feeds/packages/coova-chilli/Config.in** file :

```

# CoovaChilli advanced configuration

menu "Configuration"
    depends on PACKAGE_coova-chilli

config COOVACHILLI_PROXY
    bool "Enable support for chilli proxy. Required for AAA Proxy through http"
    default n

config COOVACHILLI_REDIR
    bool "Enable support for redir server. Required for uamregex"
    default n

config COOVACHILLI_MINIPORTAL
    bool "Enable support Coova miniportal"
    default n

config COOVACHILLI_USERAGENT
    bool "Enable recording user-agent"
    default n

config COOVACHILLI_DNSLOG
    bool "Enable support to log DNS name queries"
    default n

config COOVACHILLI_UAMDOMAINFILE
    bool "Enable loading of mass uamdomains from file"
    default n

config COOVACHILLI_LARGELIMITS
    bool "Enable larger limits for use with non-embedded systems"
    default n

choice
    prompt "SSL library"
    default COOVACHILLI_NOSSL

config COOVACHILLI_NOSSL
    bool "No SSL support"

config COOVACHILLI_MATRIXSSL
    bool "MatrixSSL"

config COOVACHILLI_CYASSL
    bool "CyaSSL"

config COOVACHILLI_OPENSSL
    bool "OpenSSL"

endchoice

endmenu

```

Patch with the following content:

```

# CoovaChilli advanced configuration

menu "Configuration"
    depends on PACKAGE_coova-chilli

config COOVACHILLI_DISABLE_CHILLIQUERY
    bool "Disable chilli_query"
    default n

config COOVACHILLI_DISABLE_LEAKYBUCKET
    bool "Disable use of leaky bucket shaping"
    default n

```

```
config COOVACHILLI_DISABLE_UAMANYIP
    bool "Disable use of uamanyip"
    default n

config COOVACHILLI_DISABLE_UAMUIPORT
    bool "Disable use of uamuiport"
    default n

config COOVACHILLI_DISABLE_ACCOUNTING_ONOFF
    bool "Disable use of Accounting-On and Accounting-Off"
    default n

config COOVACHILLI_DISABLE_TAP
    bool "Disable support for tap interface (tun only)"
    default n

config COOVACHILLI_DISABLE_TCPRESET
    bool "Disable support for TCP reset of filtered connections"
    default n

config COOVACHILLI_DISABLE_RADPROXY
    bool "Disable support RADIUS (EAP) Proxy"
    default n

config COOVACHILLI_ENABLE_JSON
    bool "Enable support for JSON"
    default n

config COOVACHILLI_DISABLE_DEBUG
    bool "Disable debugging messages"
    default n

config COOVACHILLI_DISABLE_DHCPRADIUS
    bool "Disable support DHCP/RADIUS integration"
    default n

config COOVACHILLI_ENABLE_WPAD
    bool "Enable support WPAD"
    default n

config COOVACHILLI_ENABLE_GARDENACCOUNTING
    bool "Enable walled garden accounting"
    default n

config COOVACHILLI_ENABLE_GARDENEXT
    bool "Enable extended walled garden features"
    default n

config COOVACHILLI_ENABLE_INSPECT
    bool "Enable inspect feature in cmdsock"
    default n

config COOVACHILLI_DISABLE_COA
    bool "Disable CoA RADIUS support"
    default n

config COOVACHILLI_ENABLE_DHCPOPT
    bool "Enable support for DHCP option setting"
    default n

config COOVACHILLI_ENABLE_DEBUG2
    bool "Enable verbose debugging"
    default n

config COOVACHILLI_ENABLE_SESSGARDEN
    bool "Enable support for session-based walled garden"
    default n

config COOVACHILLI_ENABLE_SESSPROXY
    bool "Enable support for per session postauth proxy"
    default n
```

```
config COOVACHILLI_ENABLE_SESSDHCP
    bool "Enable support for per session DHCP relay"
    default n

config COOVACHILLI_ENABLE_SESSDNS
    bool "Enable support for per session DNS enforcement"
    default n

config COOVACHILLI_ENABLE_CHILLIXML
    bool "Enable use of chillixml"
    default n

config COOVACHILLI_ENABLE_PROXYVSA
    bool "Enable support for VSA attribute proxy"
    default n

config COOVACHILLI_ENABLE_IPWHITELIST
    bool "Enable file based IP white list"
    default n

config COOVACHILLI_ENABLE_UAMDOMAINFILE
    bool "Enable loading of mass uamdomains from file"
    default n

config COOVACHILLI_ENABLE_REDIRDNSREQ
    bool "Enable the sending of a DNS query on redirect"
    default n

config COOVACHILLI_DISABLE_IEEE8021Q
    bool "Disable support for IEEE 802.1Q"
    default n

config COOVACHILLI_ENABLE_LARGE_LIMITS
    bool "Enable larger limits for use with non-embedded systems"
    default n

config COOVACHILLI_WITH_NFQUEUE
    bool "Enable support for netfilter_queue"
    default n

config COOVACHILLI_WITH_AVL
    bool "Enable support for AVL library"
    default n

config COOVACHILLI_WITH_NFCOOVA
    bool "Enable support for coova netfilter module"
    default n

config COOVACHILLI_WITHOUT_SFHASH
    bool "Disable SuperFastHash use"
    default n

config COOVACHILLI_WITH_LOOKUP3
    bool "Enable Jenkins lookup3 use"
    default n

config COOVACHILLI_WITH_PATRICIA
    bool "Enable Patricia use"
    default n

config COOVACHILLI_ENABLE_AUTHEDALLOWED
    bool "Enable Authorized Garden"
    default n

config COOVACHILLI_WITHOUT_IPV6
    bool "Enable IPv6"
    default n

config COOVACHILLI_WITH_PCAP
    bool "Enable support for pcap"
```

```
        default n

config COOVACHILLI_WITH_CURL
    bool "Enable support for curl"
    default n

config COOVACHILLI_WITH_MMAP
    bool "Enable support for mmap"
    default n

config COOVACHILLI_WITH_POLL
    bool "Enable support for poll"
    default n

config COOVACHILLI_WITH_IPC_MSG
    bool "Enable support for msgsnd/msgrcv SV IPC"
    default n

config COOVACHILLI_ENABLE_BINSTATUSFILE
    bool "Enable support for binary status file"
    default n

config COOVACHILLI_ENABLE_STATUSFILE
    bool "Enable support for status file"
    default n

config COOVACHILLI_ENABLE_CHILLIPROXY
    bool "Enable support for HTTP AAA Proxy"
    default n

config COOVACHILLI_ENABLE_MULTIRROUTE
    bool "Enable support for multiple routes"
    default n

config COOVACHILLI_ENABLE_MULTILAN
    bool "Enable support for multiple LANs"
    default n

config COOVACHILLI_ENABLE_CHILLIRADSEC
    bool "Enable support for RadSec AAA Proxy"
    default n

config COOVACHILLI_ENABLE_CHILLIREDIR
    bool "Enable support for Redir server"
    default n

config COOVACHILLI_ENABLE_CHILLIScript
    bool "Enable support for chilli_script helper"
    default n

config COOVACHILLI_ENABLE_CLUSTER
    bool "Enable support for clustering"
    default n

config COOVACHILLI_ENABLE_SESSIONSTATE
    bool "Enable extended use of the CoovaChilli-Session-State attribute"
    default n

config COOVACHILLI_ENABLE_SESSIONID
    bool "Enable the use of the CoovaChilli-Session-Id attribute"
    default n

config COOVACHILLI_ENABLE_APSESSIONID
    bool "Enable the use of the CoovaChilli-AP-Session-Id attribute"
    default n

config COOVACHILLI_DISABLE_COOVACHILLICONFIG
    bool "Disable the use of the CoovaChilli-Config attribute"
    default n

config COOVACHILLI_ENABLE_MDNS
```

```
    bool "Enable support for MDNS"
    default n

config COOVACHILLI_ENABLE_NETBIOS
    bool "Enable support for NetBIOS"
    default n

config COOVACHILLI_ENABLE_IEEE8023
    bool "Enable support for IEEE 802.3 Ethernet"
    default n

config COOVACHILLI_ENABLE_PPPOE
    bool "Enable support for PPPoE"
    default n

config COOVACHILLI_ENABLE_L2TPPPP
    bool "Enable support for L2TP/PPP Tunneling"
    default n

config COOVACHILLI_ENABLE_EAPOL
    bool "Enable support for EAPOL"
    default n

config COOVACHILLI_ENABLE_MINIPORTAL
    bool "Enable support Coova miniportal"
    default n

config COOVACHILLI_ENABLE_MINICONFIG
    bool "Enable support minimal cmdline config"
    default n

config COOVACHILLI_ENABLE_EWTAPI
    bool "Enable support for CoovaEWT API"
    default n

config COOVACHILLI_ENABLE_LIBJSON
    bool "Enable support for libjson"
    default n

config COOVACHILLI_ENABLE_SSDP
    bool "Enable support for Simple Service Discovery Protocol"
    default n

config COOVACHILLI_ENABLE_LAYER3
    bool "Enable support for Layer3 only operation"
    default n

config COOVACHILLI_ENABLE_MODULES
    bool "Enable dynamically loadable modules (experimental)"
    default n

config COOVACHILLI_ENABLE_EXTADMVSA
    bool "Enable admin-user VSA attribute support"
    default n

config COOVACHILLI_ENABLE_REDIRINJECT
    bool "Enable Redir content-injection support"
    default n

config COOVACHILLI_ENABLE_NETNAT
    bool "Enable net interface nat (experimental)"
    default n

config COOVACHILLI_ENABLE_USERAGENT
    bool "Enable recording user-agent"
    default n

config COOVACHILLI_ENABLE_ACCEPTLANGUAGE
    bool "Enable recording Accept-Language"
    default n
```

```

config COOVACHILLI_ENABLE_LOCATION
    bool "Enable Location Awareness"
    default n

config COOVACHILLI_ENABLE_FORCEDNS
    bool "Enable the forcing (NAT) of DNS"
    default n

choice
    prompt "SSL Library"
    default COOVACHILLI_DISABLE_SSL

config COOVACHILLI_DISABLE_SSL
    bool "Disable SSL support"

config COOVACHILLI_ENABLE_OPENSSL
    bool "Enable support for OpenSSL"

config COOVACHILLI_ENABLE_MATRIXSSL
    bool "Enable support for MatrixSSL"

config COOVACHILLI_ENABLE_CYASSL
    bool "Enable support for CyaSSL"

endchoice

config COOVACHILLI_ENABLE_MATRIXSSL_CLI
    bool "Enable MatrixSSL client use"
    depends on COOVACHILLI_ENABLE_MATRIXSSL
    default n

endmenu

```

Coova-Chilli arguments

<https://github.com/coova/coova-chilli/blob/master/src/cmdline.ggo>

Check the boolean settings section in *package/feeds/packages/coova-chilli/files/chilli.init* with the content of the previous file.

All the variables with the mention **flag on** or **flag off** match the boolean settings section.

```

option "dhcpbroadcast" - "Always broadcast DHCP responses" flag off
option "dynip"         - "Dynamic IP address pool"           string no
option "nodynip"       - "No Dynamic IP assignment"         flag off

```

dhcpbroadcast and **nodynip** are attributes of boolean settings while **dynip** is not.

Here is the *chilli.init* who matches the 1.4 version of Coova-Chilli:

```

#!/bin/sh /etc/rc.common

START=30
STOP=90

config_cb() {
    chilli_inst=$2
    if [ "$chilli_inst" != "" ]
    then
        rm -f /var/run/chilli_${chilli_inst}.*
        chilli_conf=/var/run/chilli_${chilli_inst}.conf
        eval "start_chilli_${chilli_inst}=1"
    fi
}

option_cb() {
    case "$1" in
        # UCI settings
        network)

```



```

. /lib/functions/network.sh
local ifname
network_get_device ifname $2
echo "dhcpif=\"${ifname}\"" >> $chilli_conf
;;

disabled)
eval "start_chilli_$chilli_inst=0"
;;

# boolean settings
dhcpbroadcast|nodynip|vlanlocation|locationstopstart|
locationcopycalled|locationimmediateupdate|locationopt82|mmapstring|
coanoipcheck|noradallow|proxymacaccept|proxyonacct|
dhcpradius|dhcpradius|noc2c|eapolenable|nosystemdns|
uamanydns|uamanyip|uamnatanyip|nouamsuccess|nowispr1|nowispr2|
uamauthedallowed|domaindnslocal|radsec|macauth|macreauth|
macauthdeny|macallowlocal|strictmacauth|strictdhcp|ieee8021q|only8021q|
radiusoriginalurl|swapoctets|statusfilesave|postauthproxyssl|wpaguests|
openidauth|papalwaysok|mschapv2|chillixml|acctupdate|dnssparanoia|
seskeepalive|usetap|noarpenries|framedservice|scalewin|nochallenge|
redir|injectwispr|redirurl|routetonetone|nousergardendata|uamgardendata|
uamotherdata|uamallowpost|redirssl|uamuissl|layer3|patricia|redirdnsreq|
dhcprnotidle|ipv6|ipv6only)
[ "$2" = "true" -o "$2" = "1" ] && echo "$1" >> $chilli_conf
;;

*)
echo "$1=\"${2}\"" >> $chilli_conf
;;

esac
}

start_chilli() {
local cfg="$1"
local start_chilli=$(eval "echo \${start_chilli}_$cfg")
[ "$start_chilli" = "0" ] && return
local base=/var/run/chilli_${cfg}
chilli -c ${base}.conf \
--pidfile ${base}.pid \
--cmdsocket ${base}.sock \
--unixipc ${base}.ipc &
}

start() {
config_load chilli
config_foreach start_chilli chilli
}

stop() {
for pID in $( pgrep chilli )
do
kill $pID
done
rm -f /var/run/chilli*
}

```

Reference

[\[Chilli\] Compile options list](#)

<https://wiki.openwrt.org/doc/howto/wireless.hotspot.coova-chilli>