English • Deutsch • Español • Français • Italiano • ??? • Polski • Português • ??????? • Svenska • ???(????)? • ???(??)? •

This guide will take you through the steps to setup a router running DD-WRT to use a cellular phone or other usb modem as it's WAN connection to the internet.

I developed and tested on an Asus WL-500W but this should work with other models that have USB ports and can run DD-WRT v24-sp1 Mega. I have also tested on a Linsys WRTSL54GS to the point of the phone being connected, recognized as an ACM device, and the device node being created.

This is not the most elegant solution but it is working great for me so far. If the pppd scripts for establishing a pppoe connection could be modified setting this up would be less complex and more centralized.

Even though I used this with a Samsung CDMA cellular phone, it should work for any phone or modem that will work with the ACM driver. For those looking for a new cell phone almost all of Samsungs current phones will use the acm driver.

Contents

Disclaimer

As always if you attempt this you may permanently damage your router and/or your phone or modem and any other USB devices which are connected... Procede at your own risk, I assume no responsibility.

UPDATE:

K26 NEWD-2 Mega/Big builds starting with 14414 now have the 3G/UTMS WAN connection option built into the firmware.

See the following thread for more information:

http://www.dd-wrt.com/phpBB2/viewtopic.php?t=69970

See also http://www.dd-wrt.comhttp://wiki.dd-wrt.com/wiki/index.php/Mobile_Broadband for a list of directly supported device in K26 and K3.x builds.

Before we begin

- To follow this how-to you will need to be able to use the command shell through a telnet or ssh session.
- I will not go into too much detail for procedures that are already available as tutorials in the wiki.
- Beginners may want to do some reading and make sure that they understand all of the steps involved BEFORE starting. I always believe it is best to know why you are doing something instead of just typing commands blindly since if you screw something up you won't understand what you did and how to fix it.
- If you have any experience with *nix os's and a command shell it will be a great help.
- Having a full fledged linux system available with pppd, chat, and wvdial installed will provide you with man pages to help you out otherwise the man pages can be found on the internet.
- Access to a computer running linux that has ACM support and wvdial installed can be used to get the init strings for your phone/modem in many cases.
- I precede all commands that are to be issued in the shell with a #.

In the router's web gui I left the WAN connection set on *Automatic Configuration- DHCP* on the *Setup -> Basic Setup* page. I don't know if it matters if it is set to one of the other settings but there is a possibility that setting it to *pppoe* or *dissabled* could cause a problem.

If you use a cell phone, it must be capable of being used as a modem. Most smartphones that run Windows Mobile are not capable of this without third party software installed on the phone. The Internet Connection Sharing application on the phone will not work with this method, I am working on a solution for this as well though.

What you need

- Router with a USB port and capable of running DD-WRT v24-sp1 Mega

(as of 03/2010 some version of sp2 don't support jffs)

- USB Cellular Phone or other modem compatible with ACM
- ACM kernel module
- chat ipkg

What to do:

Flash the router

If you haven't allready done so, flash your router to DD-WRT v24-sp1 Mega. Be sure to use the proper procedure for your router model.

Enable USB and JFFS

In the web gui go to **Services -> Services** and enable USB support. I enabled all of the USB support as I plan to also use the router with storage devices and printers. For more on USB support see USB in the wiki.

In the web gui enable JFFS2 support in **Administration -> Management**. See JFFS File System

I plan to eventually use a USB thumb drive for my JFFS partition but I will skip that step here as it is not necessary for this how to. If you are interested in doing the same refer to the tutorials in the wiki.

After enabling USB and jffs support reboot the router for good measure.

ACM Support

Log into the router through a telnet or ssh session if you haven't already done so.

Getting the kernel module

Next we need the ACM kernel module (driver) for the phone to be used as a modem. I used the driver that is available in the following forum post http://www.dd-wrt.com/phpBB2/viewtopic.php?t=43358

Download the archive and extract the file *acm.o*

Now copy the file to the jffs directory on the router using scp, wget, sshfs, samba share, a thumb drive..

whatever, see the other Tutorials on this wiki if you need help. I place to put the module is in `/jffs/lib/modules/` but it can be anywhere you want in `/jffs/` you just need to remember the location.

Loading the kernel module

```
# insmod /jffs/lib/modules/acm.o
```

Change the directory above if you placed the module somewhere else.

Now check to see if the ACM kernel module loaded.

```
# dmesg | grep acm
```

Should return something like:

```
usb.c: registered new driver acm
acm.c: v0.21:USB Abstract Control Model driver for USB modems and ISDN
adapters
```

Put your phone into modem mode and connect it to the routers USB port. Wait a few seconds and see if the modem is recognized:

```
# dmesg
```

In the last few lines you should see something like:

```
hub.c: new USB device 01:03.0-2, assigned address 4
ttyACM0: USB ACM device
```

This tells us that the modem device was found and assigned a device node.

Modem Init Strings

For the next part we need the init strings for the modem. You might find them in the user manual or available online. Another way to get the init strings is to use an existing linux box that has *wvdial* installed and ACM support. If you don't have an existing linux install you can boot from an Ubuntu 8.10 live cd. It has ACM

support by default and includes *wvdial*. After connecting the modem, you can check dmesg here also to see that it was recognized, go to a shell or terminal session and use *wvdialconf* to generate a config file for your modem.

```
# wvdialconf ~/modem_config
```

This will generate a file called *modem_config* in your home directory. This is just a text file that is a few lines long. You can read it however you like or just use *cat* to display it in the terminal. The things in the file that we are interested in are the maximum baud rate and init strings.

Dialing and Establishing a Conection

To dial out and establish a PPP connection we need two tools... *pppd* and *chat*. *pppd* establishes the ppp connection and starts the network interface. *Chat* is called by *pppd* to do the actual dialing of the modem. *pppd* is already installed by default as it is used for PPPoE connections. However, we do need to install *chat*, and it is available as an ipkg from the openwrt white russian repositories.

We also need to create two scripts, one for *pppd* and one for *chat*. Both of the scripts we create will need to be placed in the */jffs/etc/ppp/peers/* directory. If you choose to use a different directory structure make sure to adjust any commands or scripts to fit your situation. The scripts are simple text files and can be created with any text editor, you can use vi while logged into the router and save them right to the directory where they need to be or you can create them on your computer and copy them over.

First lets start by making a place to put the scripts. I chose to use the default directory structure that would normally be used but place it under */jffs/*

```
# mkdir /jffs/etc
# mkdir /jffs/etc/ppp
# mkdir /jffs/etc/ppp/peers
```

Chat

Installing Chat

If you already have an existing internet connection to the router see Ipkg to download and install the package. As my router could not get connected to the internet yet, I downloaded the *chat* ipkg on my laptop and copied it to */jffs/* on the router and issued the following command.

```
# ipkg install /jffs/chat_2.4.3-7_mipsel.ipk
```

Writing a Chat Script

Now to write the script to tell *chat* the phone number to dial and how to do it.

A chat script is a series of expect/reply strings. You can actually pass everything in my simple script as options on the command line but the script makes things much more tidy.

Here is my *chat* script as an example:

```
'' ATZ
OK 'ATQ0 V1 E1 S0=0 &C1 &D2 +FCLASS=0'
TIMEOUT 12
OK ATD#777
TIMEOUT 22
CONNECT
```

I saved the script as */jffs/etc/ppp/peers/isp_chat* (I used my actual isp's name instead of 'isp'). I will refer to the file as though it is named as this. Adjust things if you name it something else.

The twin single quotes starting the first line expects nothing, null, and sends *ATZ*, this is the first init string for the modem and is a common one for many modems. The second line expects to see *OK* and then sends the second init string to the modem. Both of these init strings were from taken from the config file that I generated using wvdialconf on my laptop. Note the single quotes around the second init string, this is necessary for the white spaces to be interpreted properly, otherwise chat would send the *ATQ0* and think that *V1* is the next expect string. The fourth line expects *OK* and then sends the command to dial, *ATD*, in this case the number being dialed is *#777*, this is a common number for many CDMA cellular carriers. Note there is no space between the *ATD* command and the number to be dialed and the # is actually part of the phone number used in this example. The last line expects *CONNECT* and sends nothing ending the script and returning a *0* termination code to *pppd*. I added the timeouts after spending a couple of hours trying to figure out why I could manually enter the commands using a dumb terminal like microcom, but *chat* would always hang or error out when it came to the dial command. After inserting the timeouts I haven't had any problems. If you need help writing the script refer to the *chat* man page or the internet. Just a note, I can successfully get a connection without the second init string with the phone I am using. The second init just sets things like the speed, buffers, compression, etc. so you may be able to get by with just the *ATZ* if you can not obtain the init strings for your modem. You can make the chat script far more complex but I saw no need in my situation.

Writing the PPPd script

Once again you could pass all of the commands in this script as command line options when you start *pppd*. In fact I did just that while testing and figuring this all out. Save the script as */jffs/etc/ppp/peers/isp*

Here is my script file:

```
demand
```

```
persist
idle 300
maxfail 0
lock
/dev/usb/acm/0
460800
noauth
defaultroute
user xxxxxxxxxx
password xxxxxxxxxxx
connect '/jffs/usr/sbin/chat -t3 -f /jffs/etc/ppp/peers/isp_chat'
```

The first line tells *pppd* to dial on demand. Persist tells *pppd* to redial if the connection is terminated. Idle sets a timeout to disconnect the connection after 300 seconds of inactivity. I passed maxfail 0 to tell *pppd* not to error out in case someone disconnects the phone from the router to actually use it as a phone. You might not want to use this option if this does not apply to your situation. Lock creates a lock file so that *pppd* has exclusive access to the device. */dev/usb/acm/0* is the device node that was created for the modem when it was plugged in, this might be different in your situation. The dmesg output from when you plugged in the phone/modem should lead you in the right direction to find out the node. 460800 is the port speed, I took this from the config file that was created by *wvdialconf*. Noauth tells *pppd* to skip authorization. Defaultroute creates a default route to the network interface (ppp0) in the routing table. User is your username for the isp. Password is the password for the isp.

The last line tells *pppd* to issue the shell command that is in the single quotes. Here we are invoking *chat* to dial the modem. The options set the timeout to 3 and load the *chat* script file that you made in the previous section.

Testing the Connection

Dissable or disconnect any other internet connections before testing as they could give you a false possitive.

Start the connection:

```
# pppd file /jffs/etc/ppp/peers/isp
```

Test to see if it connected:

```
# ifconfig
```

should show you ppp0 is up

```
# ps
```

should show *pppd* is running

```
# ping -c3 yahoo.com
```

should get a response

If the three tests are successfull move on to #Starting things on boot.

Troubleshooting

If you can not connect at this point here are a few tips. Remember to remove these changes once you have solved any issues.

- add *nodetach* or *updetach* to the start of the *pppd* script so that you can see any output. *nodetach* will keep *pppd* from moving to the background and *updetach* will allow *pppd* to move to the background only after the network interface (ppp0) is brought up.
- along with the above add a *-V* to the *chat* command in the *pppd* script, /jffs/usr/sbin/chat -V -t3 -f /jffs/etc/ppp/peers/isp_chat. This tells *chat* to be verbose and send the output to *stderr* (your shell session) for you to see.
- if the problem seems to be with *chat* communicating with or dialing the modem you may want to use ipkg to install the microcom ipkg from the white russian or kamikaze openwrt repositories. You can use it to send commands directly to the modem one at a time to see where the problem is occuring.

Using Microcom

This section needs to be finished...

Starting things on boot

Once you can successfully connect and get a ping response from a shell on the router, we need to make sure that the kernel module is loading and that *pppd* is being started on when the router boots. You can use nvram set commands if you know how to do it otherwise go back to the router's web gui. Go to **Administration** -> **Commands**. In the text box type the following:

```
insmod /jffs/acm.o
pppd file /jffs/etc/ppp/peers/isp
```

Then click the **Save Startup** button.

We're not done yet!

Fixing DNS and Internet Routing to the LAN/WLAN

At this point we have two issues left to resolve:

- routing to the lan/wlan and our new internet connection
- DNS

DNS

Normally we could use the *usepeerdns* option in our *pppd* script. *pppd* would then request and receive two DNS server addresses when it connects to the remote peer. It would then write them to */etc/ppp/resolv.conf* and pass them on to the */etc/ppp/ip-up* script. Unfortunately DD-WRT keeps DNS addresses in */tmp/resolv.conf* which also has a symbolic link at */etc/resolv.conf* and there is also no */etc/ppp/ip-up* script. Normally we could make a symbolic link at */etc/ppp/resolv.conf* and have it point to */tmp/resolv.conf* but */etc* can not be written to, so we can not create the directory or the symbolic link. I don't know of any way to have *pppd* write to a different file.

The only way that I have found to solve the DNS issue is to set static DNS server addresses. You can do this on the **Setup** -> **Basic Setup** page of the web gui.

Internet Routing

The last thing to do is to set up the routing to the LAN and WLAN. There are two ways to accomplish this and they each have pros and cons.

Add two rules to iptables

```
# iptables -A FORWARD -i br0 -o ppp0 -j ACCEPT
# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

The best way to add these rules is through the web gui so that they are written to nvram so you don't have to manually add them every time the router is rebooted. The problem with this solution is that if you use the web gui to set up any advanced routing involving the "WAN" interface, it will not recognize ppp0 as the WAN interface. Also the bandwidth from ppp0 will not show up on the WAN bandwidth graph. The advange to using this method is that if you are using your dial up connection in addition to or as a backup for a normal WAN connection on the WAN ethernet port of the router you don't mess up it's connection.

Change WAN values in nvram

This is the solution that I used. What we do is to change the WAN references in nvram to point to ppp0 instead of the default of eth1. You can find the values that need changed by:

```
# nvram show | grep wan
```

wan_unit=0
wan_get_dns=
wan_lease=0
http_wanport=8080
wan_gateway=0.0.0.0
wan_hwname=
wan_domain=
wan_netmask=0.0.0.0
wan_ifname2=eth1
block_wan=1
pppoe_wan_ifname=eth1
dr_wan_rx=0
wan_dns=
dhcp_wins=wan
wan_proto=dhcp
wanup=0
wan_hwaddr=00:23:54:07:FD:E9
wan_default=eth1
wan_ifnames=eth1
dr_wan_tx=0
wan_primary=1
openvpn_onwan=0
dhcp_domain=wan
wan_gateway_buf=0.0.0.0
wan_vdsl=0
upnp_wan_proto=
wan_ipaddr=0.0.0.0
wan_wins=0.0.0.0
wan_mtu=1500

size: 21126 bytes (11642 left)
sshd_wanport=22
wan_get_domain=
wan_ifname=eth1
wan_hostname=

Now we change all of the eth1 values to ppp0 and commit them to nvram:

```
# nvram set wan_ifname=ppp0
# nvram set wan_ifnames=ppp0
# nvram set wan_default=ppp0
# nvram set pppoe_wan_ifname=ppp0
# nvram set wan_ifname2=ppp0
# nvram commit
```

Reboot the router and you should be good to go.

The only quirk that I have noticed is that when the router boots the phone needs to be connected or pppd does not start. Once the router is booted though, I can disconnect and reconnect the phone from the router as much as I want.