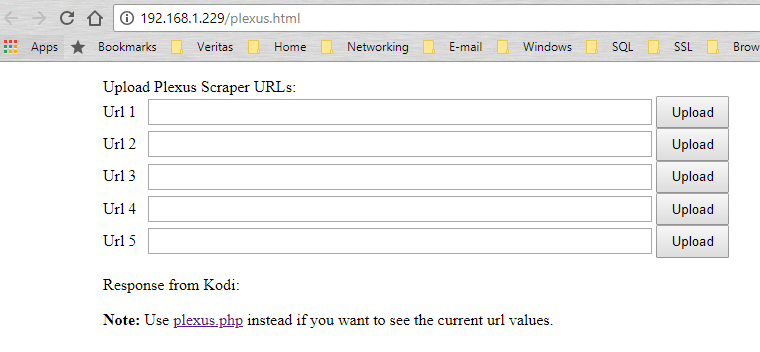
**Using PHP with the Kodi web server**  
  
Kodi version:

|  |
| --- |
| $ grep -i 'starting kodi' /storage/.kodi/temp/kodi.log | tail -1 07:09:04 11.082834 T:1967837184 NOTICE: Starting Kodi (15.2 Git:02e7013). Platform: Linux ARM 32-bit |

OpenELEC version:

|  |
| --- |
| $ cat /etc/motd ############################################## # OpenELEC # # http://openelec.tv # ############################################## OpenELEC (official) Version: 6.0.3 $ uname -a Linux OpenELEC 4.1.18 #1 SMP Mon Feb 29 20:48:22 CET 2016 armv7l GNU/Linux) |

**Introduction**OpenELEC Kodi actually includes two separate web servers; one web server is built into the Kodi exe (kodi.bin), and the other is “/sbin/httpd”. The web interface feature of Kodi allows anyone to write their own web interface addon and plug it into Kodi. That’s what I’ve done with my own web interface addon **webinterface.webif**. Once this web interface is chosen in the Kodi settings (SYSTEM -> Settings ->Services -> Web server -> Web interface”), in addition to serving web pages from the addon location, any Kodi RPC requests (<https://kodi.wiki/view/JSON-RPC_API>) are also sent to the addon, and you can get your addon to do whatever you want in response. For example, my addon **webinterface.webif** responds to an RPC request to update the “settings.xml” file used by another of my addons **plugin.program.plexusscraper**.   
  
To aid me when sending the RPC request to my webinterface.webif addon, I developed “plexus.hml”. This enabled me to update up to 5 url values in the “settings.xml” file of my addon plugin.program.plexusscraper:  
  
  
To improve on this, I thought it would be good if this page displayed the existing url values when it was displayed. To do this, we need to parse the contents of the “settings.xml” file, so I knew we would need some server-side processing. I decided to do this using PHP, as the configuration file “/etc/httpd.conf” already mentioned “php-cgi”:  
  
\*.php:/usr/bin/php-cgi

However, there were a couple of problems I noticed immediately:  
1) The file “/usr/bin/php-cgi” did not exist.  
2) The file “/etc/httpd.conf”, and locations referenced in the file, were all read-only, even though I was the root user.  
  
A further problem, that I was to notice later, was that the “httpd” web server, which uses the “/etc/httpd.conf” file, is nothing to do with the Kodi built-in web server that is used by a web interface addon. And the result of trying to display a .php file using the Kodi built-in web server, indicates immediately that php is not supported.   
  
So, to get this to work, I first needed to build a suitable Arm “php-cgi” executable, and then I needed to work out how to edit the read-only filesystem so I can use the separate “httpd” web server. Our Kodi RPC requests will still need to be sent to the Kodi built-in web server, so we’ll also need to use two running web servers; one to handle the php pages, and the other to handle the Kodi RPC requests.  
  
  
**Building php-cgi for OpenELEC Kodi**I first considered cross-compiling PHP on my PC, but I anticipated spending a lot of time trying to get this to work. Instead, I opted to build the PHP sources (<http://php.net/get/php-5.6.36.tar.xz/from/a/mirror>) directly on Raspbian OS on the Pi. The build steps are very simple:

|  |
| --- |
| sudo apt-get install libxml2-dev ./configure make |

There’s no need to run “make install” afterwards. Simply find the “php-cgi” executable and copy it to the OpenELEC Kodi.  
  
**Note:** I had to update the repository list in “/etc/apt/sources.list” in order to install libxml2-dev. I edited the file as follows:

|  |
| --- |
| $ cat /etc/apt/sources.list #deb http://mirrordirector.raspbian.org/raspbian/ stretch main contrib non-free rpi deb http://ftp.plusline.de/debian/ stretch main contrib non-free rpi deb http://ftp.au.debian.org/debian stretch main contrib non-free rpi # Uncomment line below then 'apt-get update' to enable 'apt-get source' #deb-src http://archive.raspbian.org/raspbian/ stretch main contrib non-free rpi |

Here are the details of my built php-cgi executable:

|  |
| --- |
| pi@raspberrypi:~ $ file php-cgi php-cgi: ELF 32-bit LSB executable, ARM, EABI5 version 1 (SYSV), dynamically linked, interpreter /lib/ld-linux-armhf.so.3, for GNU/Linux 3.2.0, BuildID[sha1]=e78ff835b2e965bbd3f1de276d2ced33264d236e, not stripped  $ ./php-cgi -v PHP 5.6.36 (cgi-fcgi) (built: Apr 29 2018 20:02:54) Copyright (c) 1997-2016 The PHP Group Zend Engine v2.6.0, Copyright (c) 1998-2016 Zend Technologies |

**Note:** The following approach looks potentially useful for simple cross-compiling projects: <https://medium.com/@au42/the-useful-raspberrypi-cross-compile-guide-ea56054de187>

**PHP command-line testing**After copying php-cgi to OpenELEC, it’s very simple to test it on the command-line, i.e. without any web server:

|  |
| --- |
| OpenELEC:~ # cat ipaddr.php  <?php  include 'utils.php';  echo getIPAddress();  ?> OpenELEC:~ #  OpenELEC:~ # cat utils.php  <?php  /\*  \*\* Given the name of a network interface, e.g. "eth0",  \*\* return the corresponding network ip address.  \*/  function getIPAddressFromInterface($network\_interface) {  $ip=exec("ifconfig | sed -n '/eth0/,/^$/p' | grep -Eo 'inet (addr:)?([0-9]\*\.){3}[0-9]\*' | grep -Eo '([0-9]\*\.){3}[0-9]\*'");  return $ip;  }  /\*  \*\* Return the server ip address by examining the network interfaces.  \*\*  \*\* We'll look only at network interface "eth0", but this function  \*\* should be easy to extend to examine other interfaces, e.g. "wan0".  \*/  function getIPAddress() {  $ip=getIPAddressFromInterface("eth0");  return $ip;  }  ?>  OpenELEC:~ # OpenELEC:~ # ./php-cgi ipaddr.php  X-Powered-By: PHP/5.6.36  Content-type: text/html; charset=UTF-8  192.168.1.229OpenELEC:~ #  OpenELEC:~ # ./php-cgi -q ipaddr.php  192.168.1.229OpenELEC:~ # |

**Note:** Fortunately for me, although the php-cgi has many shared library dependencies, my executable built on Raspbian OS works successfully on my OpenELEC Kodi. This is because all the shared library versions are successfully being satisified:

|  |
| --- |
| OpenELEC:~ # ldd php-cgi  linux-vdso.so.1 (0x7eb27000)  /lib/libarmmem-a7.so (0x76ee5000)  libcrypt.so.1 => /lib/libcrypt.so.1 (0x76ea6000)  libresolv.so.2 => /lib/libresolv.so.2 (0x76e82000)  librt.so.1 => /lib/librt.so.1 (0x76e6b000)  libm.so.6 => /lib/libm.so.6 (0x76df1000)  libdl.so.2 => /lib/libdl.so.2 (0x76dde000)  libnsl.so.1 => /lib/libnsl.so.1 (0x76dba000)  libxml2.so.2 => /usr/lib/libxml2.so.2 (0x76caa000)  libc.so.6 => /lib/libc.so.6 (0x76b74000)  libpthread.so.0 => /lib/libpthread.so.0 (0x76b4b000)  /lib/ld-linux-armhf.so.3 (0x76ef8000)  libz.so.1 => /usr/lib/libz.so.1 (0x76b3a000)  libgcc\_s.so.1 => /usr/lib/libgcc\_s.so.1 (0x76b0d000)  OpenELEC:~ # |

This meant that we didn’t need to consider using any clever methods to build the php-cgi, such as using <http://www.magicermine.com/> to build a portable program packaged together with the required shared libraries.

**Note:** Using php on the command-line can apparently be very useful for general scripting needs. See <https://www.ibm.com/developerworks/library/os-php-command/index.html>

**Updating the OpenELEC read-only filesystem**The read-only files we need to update, including “/etc/httpd.conf”, are contained in the “SYSTEM” squashfs file, found on the vfat partition of the OpenELEC microSD card.   
  
1) Copy the “SYSTEM” squashfs file from the microSD card to our Linux PC:

|  |
| --- |
| jcdc@linux-netbook:~$ df -Th Filesystem     Type      Size  Used Avail Use% Mounted on ... /dev/sdb1      vfat      512M  117M  396M  23% /media/3DE8-395B /dev/sdb2      ext4       13G  850M   12G   7% /media/425d2911-77eb-4bd4-ab00-1e4f22b534be  jcdc@linux-netbook:/files/tmp$ cp /media/3DE8-395B/SYSTEM . jcdc@linux-netbook:/files/tmp$ ls -ltrh total 108M -rw-r--r--  1 jcdc jcdc 108M May  2 18:57 SYSTEM |

2) Install the squashfs tools in our Linux PC:

|  |
| --- |
| sudo apt-get install squashfs-tools |

3) Extract the “SYSTEM” squashfs file into a temporary directory on our Linux PC:

|  |
| --- |
| sudo unsquashfs SYSTEM |

Here’ what we have afterwards:

|  |
| --- |
| jcdc@linux-netbook:/files/tmp$ ls -ltrh total 108M drwxrwxr-x 16 root root 4.0K Mar  1  2016 squashfs-root -rw-r--r--  1 jcdc jcdc 108M May  2 18:57 SYSTEM |

**Note:** The files we can edit are inside this “squashfs-root” directory.  
  
4) Edit the “/etc/httpd.conf” file in the “squashfs-root” directory.

|  |
| --- |
| … # Point to our webinterface as the server files root. JeremyC 02-05-2018.  #H:/usr/www # define the server root. It will override -h  H:/storage/.kodi/addons/webinterface.webif  … # Point to our v5 php-cgi we built on Rasbian OS. JeremyC 02-05-2018.  #\*.php:/usr/bin/php-cgi  \*.php:/storage/.kodi/addons/webinterface.webif/bin/php-cgi |

5) Copy our php-cgi, built on Raspbian OS, to “/storage/.kodi/addons/webinterface.webif/bin/php-cgi” on the Pi.  
  
6) Create file “/usr/local/lib/php.ini”.  
Add the following:

|  |
| --- |
| OpenELEC:~ # cat /usr/local/lib/php.ini  cgi.force\_redirect = 0 |

This is required so that we can display a php web page using one web server (httpd running on port 8080) and then send Ajax requests from this web page to the Kodi web server (running on portt 80). This is needed because of the “Same-origin security policy”, see <https://en.wikipedia.org/wiki/Same-origin_policy>.   
  
**Note:** Here’s what you see coming back from php without this setting in php.ini:

|  |
| --- |
| <p>This PHP CGI binary was compiled with force-cgi-redirect enabled. This  means that a page will only be served up if the REDIRECT\_STATUS CGI variable is  set, e.g. via an Apache Action directive.</p>  <p>For more information as to <i>why</i> this behaviour exists, see the <a href="http://php.net/security.cgi-bin">manual page for CGI security</a>.</p>  <p>For more information about changing this behaviour or re-enabling this webserver,  consult the installation file that came with this distribution, or visit  <a href="http://php.net/install.windows">the manual page</a>.</p> |

**Note:** The location of where the php.ini file needs to be was determine by running the “phpinfo.php” script on the command. This script simply contains the following:

|  |
| --- |
| OpenELEC:~/.kodi/addons/webinterface.webif # cat phpinfo.php  <?php  phpinfo();  ?> |

7) Rename the old “SYSTEM” squashfs file and build a new one.  
First we determine block size and compression method of the original “SYSTEM” squashfs file:

|  |
| --- |
| jcdc@linux-netbook:/files/tmp$ sudo unsquashfs -s SYSTEM Found a valid SQUASHFS 4:0 superblock on SYSTEM. Creation or last append time Tue Mar  1 09:41:38 2016 Filesystem size 109844.15 Kbytes (107.27 Mbytes) Compression lzo Block size 131072 Filesystem is exportable via NFS Inodes are compressed Data is compressed Fragments are compressed Always\_use\_fragments option is not specified Xattrs are not stored Duplicates are removed Number of fragments 511 Number of inodes 7934 Number of ids 2 |

Now we can rename the old “SYSTEM” file and build a new one:

|  |
| --- |
| jcdc@linux-netbook:/files/tmp$ mv SYSTEM SYSTEM.old  jcdc@linux-netbook:/files/tmp$ sudo mksquashfs squashfs-root SYSTEM -comp lzo -b 131072 |

Afterwards:

|  |
| --- |
| jcdc@linux-netbook:/files/tmp$ ls -ltrh total 215M drwxrwxr-x 16 root root 4.0K Mar  1  2016 squashfs-root -rw-r--r--  1 jcdc jcdc 108M May  2 18:57 SYSTEM.old -rw-r--r--  1 root root 108M May  2 19:21 SYSTEM |

Copy our new “SYSTEM” file back to the OpenELEC microSD card:

|  |
| --- |
| jcdc@linux-netbook:/files/tmp$ cp SYSTEM /media/3DE8-395B/SYSTEM |

**Creating a crontab entry for the httpd webserver**  
See the instructions in file “/storage/.kodi/addons/webinterface.webif/bin/monitorhttpd.sh” in order to call this script from cron every 5 minutes to ensure that the httpd server is always available.   
  
The monitoring script monitorhttpd.sh is very simple:

|  |
| --- |
| # Simply start httpd if it isn't running.  if ! ps afx | grep -v grep | grep -q -w httpd ; then  /sbin/httpd -p 8080 -c /storage/.kodi/addons/webinterface.webif/resources/httpd.conf  Fi |

**Note**: We are running the httpd web server on port 8080, leaving the Kodi built-in web server running on port 80.  
  
1) Add a crontab entry to call monitorhttpd.sh (while logged into OpenELEC as the root user):

|  |
| --- |
| 1) VISUAL=vi crontab -e  2) Add your new crontab line: \*/5 \* \* \* \* /storage/.kodi/addons/webinterface.webif/bin/monitorhttpd.sh  3) Save the changes and verify: crontab -l |

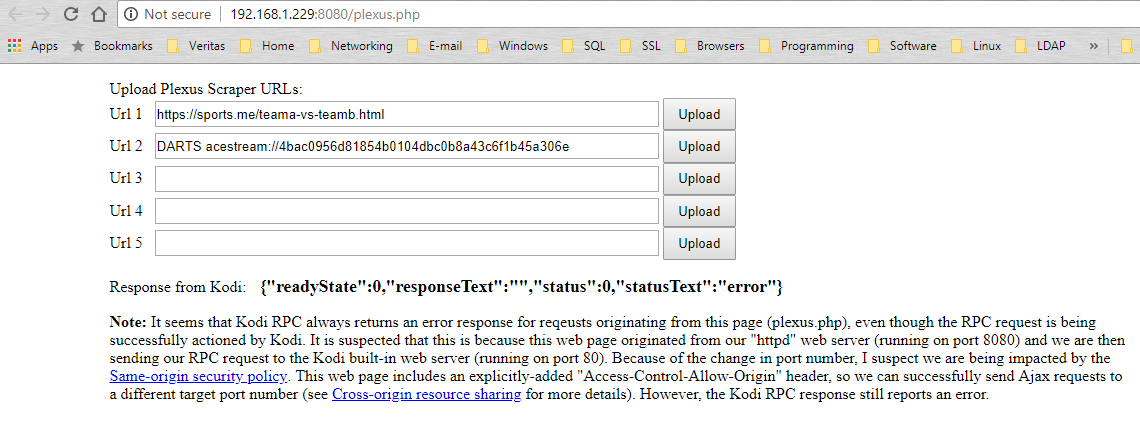
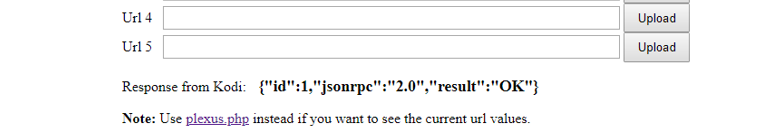
**Note**: Cron needs to be enabled in the OpenELEC settings:

“SYSTEM > OpenELEC > Services > Cron > Enable Cron”

2) Update autostart.sh.  
So that the httpd web server is immediately available when Kodi starts up, we also need to call our script from “/storage/.config/autostart.sh”:

|  |
| --- |
| OpenELEC:~/.kodi/addons/webinterface.webif # cat /storage/.config/autostart.sh  (  /storage/.kodi/addons/webinterface.webif/bin/monitorhttpd.sh;  /storage/genrssxml.py;  ) & |

More on Kodi and “autostart.sh” can be found here: <https://wiki.libreelec.tv/autostart.sh>

**Conclusion**With support now added for php web pages, here’s what my “plexus.php” page looks like:  
  
**Note:** The only remaining issue is this “error” response coming back from the RPC call to Kodi. The RPC call is definitely being actioned successfully by our webinterface.webif addon, but I suspect the “Same-origin security policy” is still causing this error response. Contrast this with the RPC response when we use the non-php equivalent page “plexus.html”:  
  
  
Here we can see a success “OK” result coming back in the RPC in response. It’s a shame we can’t get this response from our “plexus.php” file, but it doesn’t seem to negatively affect the RPC call.  
  
  
*JeremyC 05-05-2018***END**