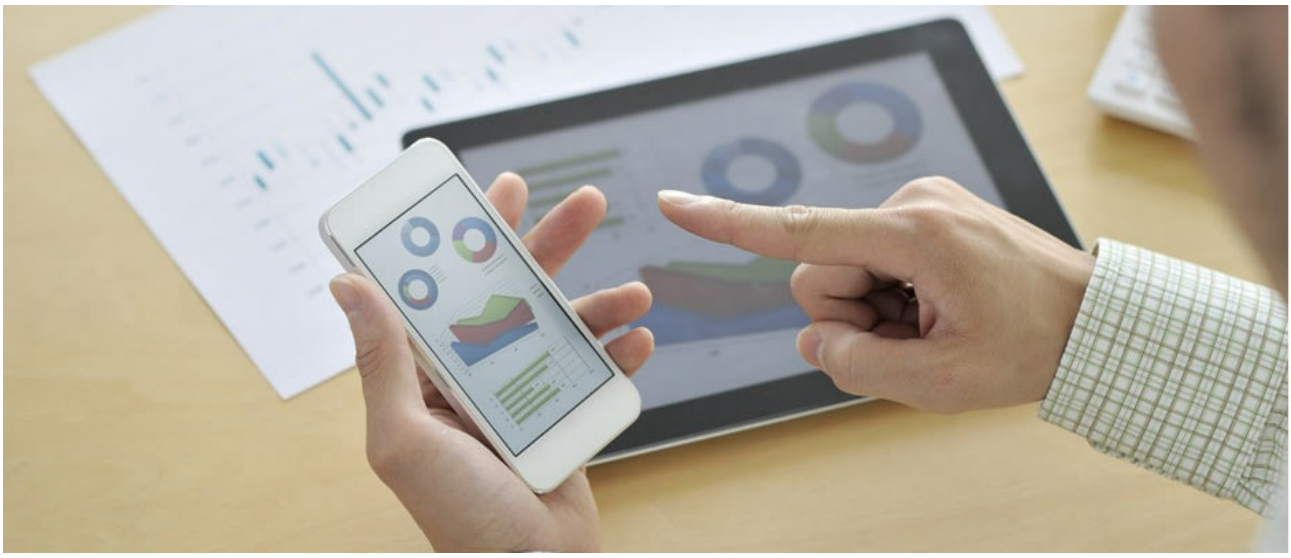
scientiamobile

"The first step in a great mobile experience"



INFUZE NGINX MODULE USER GUIDE

Support

The ScientiaMobile Support Forum is open to all WURFL users, both commercial license holders and evaluation users. It represents the combined knowledge base for the WURFL community. Commercial licensees are invited to post questions in the forum using the account to which their licenses are associated. This may mean faster handling of those posts by ScientiaMobile's personnel.

For commercial license holders, there are tiered support levels to address a variety of business support needs. After logging into your account, commercial licensees with support options can access the Enterprise Support portal to post tickets. These tickets will receive expedited attention.

To inquire about support plans, use our License Inquiry or our General Inquiry form.

Update Notifications

If you would like to be notified of our API updates, major data updates, and other technical changes, please subscribe to our ScientiaMobile Announcements list

---

scientiamobile

www.scientiamobile.com
Tel +1.703.310.6650
E-mail: sales@scientiamobile.com

WURFL InFuze Module for NGINX: User Guide

This document is aimed at developers and system administrators who intend to install and configure the WURFL InFuze Module for Nginx on Unix, Linux, and other Unix-based systems.

If you are using NGINX Plus, please see the documentation for our NGINX Plus module here.

Note: Unlike NGINX Plus, you will still need to compile and link the NGINX module.

Installing libwurfl

In order for the Module to work it is **ESSENTIAL** that the libwurfl library is installed on your system. libwurfl is provided in your Customer Vault/FileX.

If you have not already installed libwurfl, instructions can be found here. Release notes for each API can be found here.

!!! WARNING !!!

SINCE MODULE VERSION 1.8.4.1 YOU NEED TO ADD A wurfl_enable on; COMMAND TO THE http section OF YOUR CONFIGURATION FILE,
OTHERWISE WURFL MODULE WILL NOT BE LOADED. THIS CHANGE IS PART OF A FEATURE ALIGNMENT, NECESSARY FOR ENTERPRISE USERS OF NGINX.

Installing NGINX

Unlike other web server applications (e.g. Apache), NGINX web server does not support dynamically linked or third-party modules. You are required to download and compile the entire source code, including the WURFL Module extension for NGINX.

Install Prerequisites on Ubuntu/Debian

Run the following commands to ensure that your system's package database and installed programs are up to date, and then install the dependent packages:

```
sudo apt-get update
sudo apt-get upgrade --show-upgraded
sudo apt-get install libpcre3-dev build-essential libssl-dev
sudo apt-get install libpcap-dev libnet1-dev libyaml-0-2 libyaml-dev
sudo apt-get install zlib1g zlib1g-dev libcap-ng-dev libcap-ng0
```

Install Prerequisites on RedHat/Fedora/CentOS

Before you compile NGINX, you will need to install the following dependent packages:

```
yum groupinstall "Development Tools"
yum install libpcap pcre-devel zlib-devel openssl-devel
```

Download and Compile NGINX

You may already have an instance of NGINX running, in which case we recommend that you stop. You will need to recompile NGINX to include nginx-mod_wurfl, a WURFL module extension for NGINX.

> *Tip: We also provide an installation script named "wurfl_nginx_mod_setup.sh" which automatically installs the NGINX server and the WURFL NGINX Module. We strongly advise use of this script to easily build NGINX with the WURFL Module.*

Build WURFL module embedded in NGINX or as a NGINX Dynamic Module

Since NGINX 1.9.11, dynamic modules are supported. Starting from WURFL NGINX module version 1.8.1.0 you can choose to build it as a shared object.
Edit the wurfl_nginx_mod_setup.sh file, uncomment the line

```
##./configure --with-debug --with-threads --with-cc-opt=-Wno-error --add-dynamic-module=../src
```

and comment the line

```
./configure --with-debug --with-threads --with-cc-opt=-Wno-error --add-module=../src
```

Specify NGINX version as a first script parameter, and WURFL version as the second script parameter. For example:

```
chmod +x ./wurfl_nginx_mod_setup.sh
./wurfl_nginx_mod_setup.sh 1.10.3 1.9.0.0
```

Alternatively, you may install the module yourself following the steps below:

For this example, the source files will be downloaded to the /opt/nginx/ directory (you must manually create the 'nginx' directory inside /opt/ - mkdir nginx). Download and Extract the nginx-mod_wurfl (nginx-mod_wurfl-1.8.0.0.tar) file to the /opt/nginx/ folder, which consists of the source code of module extension, by running the following commands:

```
cd /opt/nginx
tar xvf nginx-mod_wurfl-1.9.0.0.tar
```

Check out the NGINX download page for the URL of the latest stable release, and then issue the following commands to obtain it (substituting a newer link if necessary):

```
wget http://nginx.org/download/nginx-1.10.3.tar.gz
tar zxvf nginx-1.10.3.tar.gz
cd nginx*
```

Before compiling the NGINX server, be sure to append --add-module option if you want to build mod_wurfl embedded in NGINX

```
./configure --with-debug --with-threads --with-cc-opt=-Wno-error --add-module=../src
```

or --add-dynamic-module option if you want to build mod_wurfl as a NGINX Dynamic Module (NGINX version >= 1.9.11, WURFL NGINX module version >= 1.8.1.0)

```
./configure --with-debug --with-threads --with-cc-opt=-Wno-error --add-dynamic-module=../src
```

> *Tip: By default, NGINX will be installed in /usr/local/nginx, which, although a good place, means that the main NGINX binary will be found in /usr/local/nginx/sbin/nginx. Have a look at the Install Options page of*

> *the NGINX wiki for full details.*

To build and install NGINX with the above configuration, use the following command sequence:

```
make
sudo make install
```

> *Tip: For significant performance enhancement, you might want to consider reinstalling PCRE with "just-in-time compilation", and then adding --with-pcre-jit option to NGINX builds.*

> *Tip: If you choosed to build mod_wurfl as a NGINX Dynamic Module, you should find the ngx_http_wurfl_module.so file in /usr/local/nginx/sbin/nginx/modules .*

Configuration Guide

Below is an example nginx.conf configuration file for WURFL setup. Please refer to the Module Command Table below which explains each element in detail (Table 1), their parameters, constraints, and default recommended settings.

```
...

# -- Uncomment this if you are using NGINX Plus
# -- or you compiled WURFL module with --add-dynamic-module (WURFL API version 1.8.1.0 or above / NGINX OSS 1.9.11 or above).
#load_module modules/ngx_http_wurfl_module.so;

...

http {
    ...

    # -- Command to enable WURFL module. Since WURFL module version 1.8.4.1, WURFL module is DISABLED BY DEFAULT.
    # -- Used to switch on/off WURFL module; if set 'off', NGINX will ignore all wurfl_* configuration.
    # -- If WURFL module is disabled, NGINX variables used to handle WURFL detection results (i.e: $wurfl_id , $wurfl_cap_is_smarttv...)
    are still valid but their values will be "empty string".
    # -- Valid values are on/off. Default value is off.
    wurfl_enable    on;

    # -- WURFL root definition, one per config. User MUST specify this path in order to make WURFL engine correctly start.
    wurfl_root      /usr/share/wurfl/wurfl.zip;

    # -- WURFL Updater allows seamless update of WURFL engine with new data downloaded from Scientiamobile.
    # -- Updater configuration must be done after wurfl_root.
    # -- WURFL file should be either .zip or .xml.gz and match wurfl_root file type.
    # -- Put your personal updater url taken from Scientiamobile customer Vault. If your license is expired, NGINX won't start with updater configured.
    # -- Valid values for the updater checking frequency (how often the updater checks for any new WURFL data file
    # -- to be downloaded and used by the engine) are DAILY,WEEKLY.
    # -- Updater log file (wurfl-updater.log) may be found in "wurfl_root" folder. The folder and wurfl.zip file should be writable
    # -- by NGINX process
    #wurfl_updater   https://data.scientiamobile.com/xxxxx/wurfl.zip DAILY;

    # -- WURFL patches definition (as much as needed, patches will be applied in the same order as specified in this conf file)
    #wurfl_patch     /path/to/patch1.xml;
    #wurfl_patch     /path/to/patch2.xml;
    #wurfl_patch     /path/to/patch3.xml;
```

```
# Increase the variable hash size
variables_hash_max_size 1024;
variables_hash_bucket_size 1024;

# -- WURFL UA priority: one of the following (default is wurfl_useragent_priority_override_sideloaded_browser_useragent)
wurfl_useragent_priority_override_sideloaded_browser_useragent;
#wurfl_useragent_priority_use_plain_useragent;

# -- WURFL cache: one of the following
#wurfl_cache_null;
wurfl_cache_double_lru 10000,3000;

# -- WURFL properties (formerly "WURFL default variables")
# -- Since WURFL API version 1.8.0.0, WURFL default variables except "wurfl_id" are no longer injected by default
# -- and have to be explicitly specified.
#wurfl_request_property  wurfl_root_id;
#wurfl_request_property  wurfl_isdevroot;
#wurfl_request_property  wurfl_useragent;
#wurfl_request_property  wurfl_engine_target;
#wurfl_request_property  wurfl_useragent_priority;
#wurfl_request_property  wurfl_info;
#wurfl_request_property  wurfl_api_version;
#wurfl_request_property  wurfl_last_load_time;
#wurfl_request_property  wurfl_normalized_useragent;

# -- WURFL user requested static capabilities (as an example, this is not a complete list)
#wurfl_request_capability is_console;
#wurfl_request_capability is_tablet;
#wurfl_request_capability is_wireless_device;

# -- WURFL user requested virtual capabilities (as an example, this is not a complete list).
# -- Since WURFL API version 1.7.1.0, virtual capabilities are no longer injected by default
# -- and have to be explicitly specified.
#wurfl_request_capability advertised_device_os;
#wurfl_request_capability is_android;

...

server {

    ...

    # -- WURFL injection rules
    #
    # -- The following rule lists define which urls hasn't/has to be injected with wurfl data.
    #
    # -- The urls will be processed in this manner:
    #
    # -- Check if a wurfl_do_not_process_url regex matches the url. If yes, the url
    # -- itself will not be injected and no further checks will be made
    # -- If no wurfl_do_not_process_url match, check if a wurfl_process_url regex
    # -- matches the url. If yes, the url itself will be injected
    #
    # -- The fallback behaviour in case the url doesn't match any
    # -- wurfl_do_not_process_url/wurfl_process_url rule is:
    # -- INJECTION if the list of wurfl_process_url is empty
    # -- NO INJECTION if the list of wurfl_process_url is not empty
    #
    #
    # ---- Black list: defines which urls hasn't to be injected with wurfl data
    # ---- syntax: wurfl_do_not_process_url <url regex> <rule name>
    #
    # wurfl_do_not_process_url .*\.(gif|jpeg|png|css) "Static contents";
    # wurfl_do_not_process_url .*\/img\/.* "All in img folder;
    #
    #
    # ---- White list: defines which urls has to be injected with wurfl data
    # ---- syntax: wurfl_process_url <url regex> <rule name>
    #
```

```
        # wurfl_process_url .*\.(php|php4|php5) "Php scripts";
        # wurfl_process_url .*\.(jsp|asp) "Jsp/Asp scripts";

        # -- Trace wurfl injections (default value = off)
        # -- The log format is:
        # -- WURFL: Server <server name> - Whitelist hit - resource: <url> - regex: <url regex> (rule name:<rule name>)
        # -- WURFL: Server <server name> - Blacklist hit - resource: <url> - regex: <url regex> (rule name:<rule name>)
        # -- <server name> is the value of server_name property, or, if empty, a number indicating
        # -- which server section the log is relative to
        #
        # wurfl_log_header_injection on;

        ...

        # Example of uses with PHP FastCGI (Note: PHP is not a requirement!)
        location ~ \.php$ {
            root           html;
            fastcgi_pass   127.0.0.1:9000;
            fastcgi_index  index.php;
            fastcgi_param  SCRIPT_FILENAME  /var/www$fastcgi_script_name;
            include        fastcgi_params;

            ############ WURFL data passed to fastcgi PHP scripts
            #
            ######## WURFL properties (formerly "WURFL default variables"): add here to make them available to fastcgi PHP scri
pts.
            ######## Property "wurfl_id" is injected by default (see http section) and could be specified here
            fastcgi_param  WURFL_ID                    $wurfl_id;
            #fastcgi_param  WURFL_ROOT_ID               $wurfl_root_id;
            #fastcgi_param  WURFL_ISDEVROOT             $wurfl_isdevroot;
            #fastcgi_param  WURFL_ORIGINAL_USERAGENT    $wurfl_useragent;
            #fastcgi_param  WURFL_ENGINE_TARGET         $wurfl_engine_target;
            #fastcgi_param  WURFL_USERAGENT_PRIORITY    $wurfl_useragent_priority;
            #fastcgi_param  WURFL_INFO                  $wurfl_info;
            #fastcgi_param  WURFL_API_VERSION           $wurfl_api_version;
            #fastcgi_param  WURFL_LAST_LOAD_TIME        $wurfl_last_load_time;
            #fastcgi_param  WURFL_NORMALIZED_USERAGENT  $wurfl_normalized_useragent;
            #
            #
            ######## WURFL capabilities: headers injected by into Request by wurfl engine.
            ######## Add here to make them available to fastcgi PHP scripts.
            #
            #### specify the static capabilities needed in webapp (as an example, this is not a complete list)
            #fastcgi_param  WURFL_IS_CONSOLE            $wurfl_cap_is_console;
            #fastcgi_param  WURFL_IS_TABLET             $wurfl_cap_is_tablet;
            #fastcgi_param  WURFL_IS_WIRELESS_DEVICE    $wurfl_cap_is_wireless_device;
            #
            #### specify the virtual capabilities needed in webapp (as an example, this is not a complete list)
            #fastcgi_param  WURFL_ADVERTISED_DEVICE_OS  $wurfl_cap_advertised_device_os;
            #fastcgi_param  WURFL_IS_ANDROID            $wurfl_cap_is_android;
        }
        ...
    }
}
```

WURFL NGINX Module command table

The following table shows directives which can be applied to configure and use our WURFL NGINX Module.

Section	Syntax	Description	Availability

Section	Syntax	Description	Availability
http	**wurfl_enable**	Enable/Disable the WURFL module. Since version 1.8.4.1, the WURFL module is disabled by default and you will need to issue the wurfl_enable on; to enable it. Otherwise all WURFL commands will be ignored and the WURFL module will not be loaded. Possible values: on/off. Default value: off.	1.9
http	**wurfl_root**	Defines the location (path) of the WURFL data file.	1.4

Section	Syntax	Description	Availability
	wurfl_updater	Allows seamless update of WURFL engine with new data downloaded from Scientiamobile. This directive must follow wurfl_root. It takes two parameters: • the data url (taken from your personal Scientiamobile Vault account, choosing between two data file types: .zip or .xml.gz) Take care that wurfl_root file type and wurfl_updater data url file types match so you may need to change the wurfl_root file type accordingly. Please note that in case your license is expired, NGINX won't start with updater configured. • the updater checking frequency (how often the updater checks for any new WURFL data file to be downloaded and used by the engine) which you can choose between DAILY and WEEKLY. In order to let the Updater perform its activities both the wurfl_root folder and file must be writable by NGINX. The wurfl-updater.log file in wurfl_root folder will contains details on Updater activity.	1.8.3
	wurfl_patch	Adds one or more custom patch files to the WURFL repository.	1.4

Section	Syntax	Description	Availability
	wurfl_target_default or **wurfl_target_fast_desktop_browser_match** or **wurfl_target_performance** DEPRECATED or **wurfl_target_accuracy** DEPRECATED	You can choose between wurfl_target_default suitable for generic traffic, and wurfl_target_fast_desktop_browser_match when you have significant amounts of desktop browser traffic compared to mobile device (this option will return generic_web_browser wurfl_id for the majority of web browsers). For wurfl_target_performance and wurfl_target_accuracy options, please read note about **Decommissioning of engine target**. Note that wurfl_target_performance and wurfl_target_accuracy will trigger the new wurfl_target_default behavior.	1.4

Section	Syntax	Description	Availability
	wurfl_cache_null or **wurfl_cache_lru** or **wurfl_cache_double_lru**	The caching strategies are also configurable. You can choose between NULL, LRU, or Double LRU cache mechanisms. The default is Double LRU, which is a two-cache strategy (one going from User-Agent to Device-Id, the other from Device-Id to Device). The default parameters are 30,000, 10,000 (maximum 30,000 elements for the User-Agent to device-id cache and maximum 10,000 elements for the device-id to device cache) and the values are in elements. The LRU cache comes with User-Agent to Device mapping only, and the NULL parameter will disable the cache mode. These parameters refer to the max capacity size of the cache itself in Kilobytes. For more information, please see LRU Cache Mechanism.	1.4

Section	Syntax	Description	Availability
	wurfl_request_capability	Enables injection of WURFL Static Capabilities/Virtual Capabilities in the HTTP request headers with your specified parameter(s). Syntax: **wurfl_request_capability <static capability/virtual capability name>** You can find the complete list of WURFL Static Capabilities and WURFL Virtual Capabilities here. If a Static Capability/Virtual Capability is specified in a wurfl_request_capability command, WURFL NGINX module set a NGINX variable whose name is **wurfl_cap_**	1.4

Section	Syntax	Description	Availability
	wurfl_useragent_priority_override_sideloaded_browser_useragent or **wurfl_useragent_priority_use_plain_useragent**	You can choose between **wurfl_useragent_priority_override_sideloaded_browser_useragent** (WURFL_USERAGENT_PRIORITY_OVERRIDE_SIDELOADED_BROWSER_USERAGENT) or **wurfl_useragent_priority_use_plain_useragent** (WURFL_USERAGENT_PRIORITY_USE_PLAIN_USERAGENT) user agent priorities. In High-Performance mode, desktop web browser detection is done programmatically without referencing the WURFL data. As a result, most desktop web browsers are returned as generic_web_browser WURFL ID for performance. The default user agent priority is wurfl_useragent_priority_override_sideloaded_browser_useragent.	1.5.2
	wurfl_request_property	Enables injection of WURFL Properties (see section WURFL Properties below) in the HTTP request headers. Syntax: **wurfl_request_property <variable name>** Not mandatory.	1.8

Section	Syntax	Description	Availability
server	**wurfl_do_not_process_url**	A regular expression defining which urls hasn't to be injected with wurfl data. Syntax: **wurfl_do_not_process_url <url regex> <rule name>** Not mandatory. You can specify more than one wurfl_do_not_process_url The urls will be processed in this manner: Check if a wurfl_do_not_process_url regex matches the url. • If yes, the url itself will not be injected and no further checks will be made. • If no, check if a wurfl_process_url regex matches the url. • • If yes, the url itself will be injected The fallback behaviour in case the url doesn't match any wurfl_do_not_process_url/wurfl_process_url rule is: • **INJECTION** if the list of wurfl_process_url is empty • **NO INJECTION** if the list of wurfl_process_url is not empty	1.8

Section	Syntax	Description	Availability
	wurfl_process_url	A regular expression defining which urls has to be injected with wurfl data. Syntax: **wurfl_process_url <url regex> <rule name>** Not mandatory. You can specify more than one wurfl_process_url. The urls will be processed in this manner: Check if a wurfl_do_not_process_url regex matches the url. • If yes, the url itself will not be injected and no further checks will be made. • If no, check if a wurfl_process_url regex matches the url. • • If yes, the url itself will be injected The fallback behaviour in case the url doesn't match any wurfl_do_not_process_url/wurfl_process_url rule is: • **INJECTION** if the list of wurfl_process_url is empty • **NO INJECTION** if the list of wurfl_process_url is not empty	1.8
	wurfl_log_header_injection	Enable/disable logging of occurred injections. You can choose between **on**/**off** Not mandatory. (default value is off)	1.8

WURFL Properties

The WURFL NGINX module sets some useful convenience NGINX variables to retrieve information regarding the currently active WURFL configuration.

These variables are automatically calculated and are injected in HTTP requests if specified in a wurfl_request_property

command.

Please note that the wurfl_id variable is injected by default so you don't have to sepcify it in a wurfl_request_property command.

WURFL Properties Table

Variable Name	Contents	Availability
wurfl_id	Contains the device ID of the matched device. It is injected by default so you don't have to specify it in a wurfl_request_property command	1.4
wurfl_root_id	Contains the device root ID of the matched device.	1.4
wurfl_isdevroot	Tells if the matched device is a root device. Possible values are "TRUE" or "FALSE"	1.4
wurfl_useragent	The original useragent coming with this particular web request	1.5.1
wurfl_api_version	Contains a string representing the currently used Libwurfl API version	1.5.1
wurfl_engine_target	Contains a string representing the currently set WURFL Engine Target. Possible values are "HIGH_ACCURACY", "HIGH_PERFORMANCE" or "INVALID"	1.5.1
wurfl_info	A string containing information on the parsed WURFL data file and its full path	1.5.1
wurfl_last_load_time	Contains the UNIX timestamp of the last time WURFL has been loaded successfully.	1.5.1
wurfl_normalized_useragent	The normalized useragent.	1.5.1.3
wurfl_useragent_priority	The user agent priority used by WURFL.	1.5.2

Running NGINX Web Server

Once you have configured NGINX and are ready to test the installion by launching the webserver, use the following

command:

```
sudo /usr/local/nginx/sbin/nginx
```

Tip: This guide only instructs you on how to start and stop NGINX manually. There are many different online guides that shows how to build init scripts for NGINX, here is an example site.

To see if NGINX is running, go to your server's IP address. You should see a 'welcome to NGINX' message, indicating that NGINX is installed. Alternatively, check to see if the NGINX process is running by executing this command:

```
sudo ps aux | grep nginx
```

To stop the NGINX service manually, you can take advantage of the pid file to identify process ID.

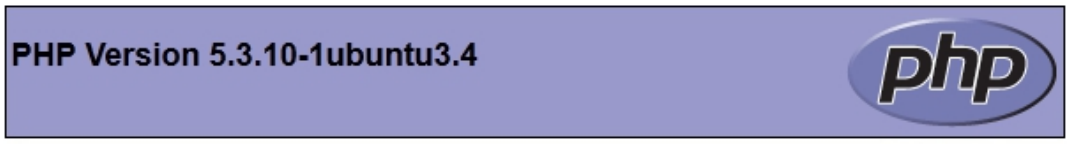
```
sudo kill `cat /usr/local/nginx/logs/nginx.pid`
```

Verify WURFL Installation

At the end of the installation procedure, your NGINX web server will be enhanced with the new WURFL API capabilities to support a variety of use-cases that rely on your NGINX instance becoming aware of device information. Among other things, NGINX will augment HTTP requests with new headers such as: X-Wurfl-Is-Tablet, X-Is-Full-Desktop and the relative values as defined in WURFL. This will enable services deployed downstream to take advantage of device detection in the simplest way possible (getHeader() and similar).

Note: Pre-selected capabilities and header names are shown for this example (the actual choice of header names is configurable).

The WURFL Module for NGINX enables organizations to leverage device and browser information within the very tools, frameworks and programming languages that they have previously elected to use. For example, assuming that a RIM BlackBerry 8830 device requests a PHP phpinfo() page (assuming that PHP is installed), the following information would be returned by the system thanks to the WURFL NGINX Module:



System	Linux ip-10-252-59-109 3.2.0-31-virtual #50-Ubuntu SMP Fri Sep 7 16:36:36 UTC 2012 x86_64

PHP Variables

Variable	Value
_SERVER["WURFL_ROOT_ID"]	blackberry8300_ver1
_SERVER["WURFL_ID"]	blackberry8300_ver1_vid102
_SERVER["WURFL_ISDEVROOT"]	FALSE
_SERVER["WURFL_CAP_BRAND_NAME"]	RIM
_SERVER["GATEWAY_INTERFACE"]	CGI/1.1
_SERVER["SERVER_SOFTWARE"]	nginx

PHP variables now include HTTP headers that contain device information. This information can be leveraged by downstream applications to tailor the content to the capabilities of the requesting device. For example, assuming we choose the X-Wurfl-* notation for the environment variables or HTTP headers, the following PHP code can be used to

determine if the requesting device is a tablet:

```
// PHP
if ($_SERVER['X-Wurfl-Is-Tablet'] == 'true') {
    //Do whatever makes sense for a tablet
}
```

Java, ASP.Net, Perl, Python and everything that can run in a CGI environment can take advantage of the functionality.

Running PHP Script on NGINX

In order to run the example script above, you will need to install PHP FastCGI. It is highly recommended you use PHP-FPM (FastCGI Process Manager) for its features and ease of installation. Use the following command to install the latest PHP-FPM package:

```
sudo apt-get install php5-fpm
```

Then run PHP5-FPM in the background, using the following init.d script to start the process:

```
sudo /etc/init.d/php-fpm start
```

At this point we are ready to create a simple test script. Test the capabilities filters with the following PHP script test.php to put in /usr/local/nginx/html:

```
//PHP - test.php
phpinfo(INFO_ENVIRONMENT);
```

The test.php script will return an extended result of what can be seen in Figure 1. You can use any kind of mobile device to load the site and see their unique device capabilities.

> ***IMPORTANT - Decommissioning of engine target options:*** *Prior to version 0.8 of the API, users could choose between wurfl_target_performance and wurfl_target_accuracy engine optimization options. These options had been introduced years ago to manage the behavior of certain web browsers and their tendency to present "always different" User-Agent strings that would baffle strategies to cache similar WURFL queries in memory.*
> *As the problem has been solved by browser vendors, the need to adopt this strategy has diminished and ultimately disappeared (i.e. there was no longer much to be gained with the high-performance mode in most circumstances) and ScientiaMobile elected to "remove" this option to simplify configuration and go in the direction of uniform API behavior in different contexts.*
>
> *When we wrote "remove" in the previous sentence, we were not being totally accurate. Customers who may find themselves in the unlikely situation of having to analyze significant amounts of legacy web traffic, may still enable the old wurfl_target_performance behavior by calling wurfl_target_fast_desktop_browser_match in their configuration.*
>
> *Please note that users with the old wurfl_target_performance target engine will not receive an error. The old behavior will not be triggered, though. The wurfl_target_default target (corresponding to the old wurfl_target_accuracy) will be used instead.*

License