"The first step in a great mobile experience"

INFUZE VARNISH MODULE USER GUIDE

Support

The ScientiaMobile Enterprise Support Portal 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 Varnish: User

Guide

This document is aimed at developers and system administrators who intend to install and configure the WURFL InFuze Module for Varnish-Cache on Unix, Linux, and other Unix-based systems such as FreeBSD. In the rest of the documentation, we will refer to the module as the WURFL VMOD (WURFL Varnish 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 foundhere. Release notes for each API can be found here.

WURFL Varnish Module availability

The WURFL Varnish module is included in the release package for multiple major versions of Varnish Cache. To see which versions are currently supported, please refer to the release package contents.

Installing WURFL Varnish Module

You may already have an instance of Varnish Cache running on your system. In this case, we recommend that you stop any running Varnish Cache instance and check if you have the correct version of Varnish Cache installed.

> ***Warning:*** *Some older Linux distributions, such as Fedora 15, may include outdated Varnish libraries that are incompatible with the WURFL VMOD. Please ensure you are running a supported version of Varnish Cache before installing this module.*

Installing the WURFL Varnish Module on Ubuntu

You can get the Varnish Cache software from packagecloud.io, chosing the desired repo and then following the installation instructions related to deb packages.

Once you have obtained the WURFL Varnish module deb package from ScientiaMobile, you can install it with:

```
sudo dpkg -r varnish-mod-wurfl
sudo dpkg -i varnish-mod-wurfl-x.y.z.0.varnish-5.2.1.x86_64.deb
```

WURFL VMOD packages are versioned to match the Varnish Cache versions they support. For example, the package shown above is intended for Varnish Cache 5.2.1. Be sure to install the WURFL VMOD package that corresponds to your specific Varnish Cache version.

> *WURFL Varnish Module Installation Folder*
> *Please note that the WURFL Varnish module deb package will install the module.so file in /usr/lib/varnish/vmods/ and will try to create symbolic links in/usr/lib/x86_64-linux-gnu/varnish/vmods and/or /usr/local/lib/varnish/vmods.*
> *If your Varnish installation uses a different directory for VMODs, you may encounter the following error on startup:*
> *..../vmods/libvmod_wurfl.so: open shared object file: No such file or directory.*
> *To resolve this, please manually create a symbolic link to /usr/lib/varnish/vmods/libvmod_wurfl.so in your Varnish VMODs folder.*

Install WURFL Varnish Module on RedHat/Fedora/CentOS

You can get the Varnish Cache software from packagecloud.io, chosing the desired repo and then following the installation instructions related to rpm packages.

Once you have downloaded the WURFL Varnish module rpm package from ScientiaMobile, you can install it with:

```
sudo rpm -e varnish-mod-wurfl
sudo rpm -i varnish-mod-wurfl-x.y.z.0.varnish-5.2.1.x86_64.rpm
```

WURFL VMOD packages are versioned to match the Varnish Cache versions they support. For example, the package shown above is intended for Varnish Cache 5.2.1. Be sure to install the WURFL VMOD package that corresponds to your specific Varnish Cache version.

> *WURFL Varnish Module Installation Folder*
> *Please note that the WURFL Varnish module deb package will install the module .so file in /usr/lib/varnish/vmods/ and will try to create symbolic links in /usr/lib64/varnish/vmods and/or /usr/local/lib/varnish/vmods.*
> *If your Varnish installation uses a different directory for VMODs, you may encounter the following error on startup:*
> *..../vmods/libvmod_wurfl.so: open shared object file: No such file or directory.*
> *To resolve this, please manually create a symbolic link to /usr/lib/varnish/vmods/libvmod_wurfl.so in your Varnish VMODs folder.*

The installation process is now complete. Make sure to check the Configuration Guide and WURFL Varnish Module Examples sections to verify that everything was installed correctly.

> *Varnish Tuning*
> *Starting libwurfl v1.8.3.1, the Varnish run time parameter thread_pool_stack needs to be tuned to meet WURFL requirements.*
> *Its value should be changed to at least 96k by executing the varnishd daemon with the following command: varnishd -p thread_pool_stack=96k.*
> *For Varnish versions prior to 4.0, please use varnishd -p thread_pool_stack=163840 instead.*

WURFL Data Snapshot

To perform lookups, you will need a copy of your WURFL data snapshot (also referred to as the wurfl.xml). While there is one included in the release package, it is intended to be a sample and will not contain all of your licensed capabilities. Your licensed WURFL data snapshot can be accessed by following these directions.

Configuration Guide

Varnish utilizes Varnish Configuration Language (VCL), a domain-specific language that can be used to define HTTP-request handling and media caching policies for the Varnish-Cache HTTP accelerator. For more information on VCL, please check the Varnish 5 VCL, Varnish 4.1 VCL, Varnish 4 VCL or the Varnish 3 VCL online documentation as well as other examples of VCL Usage.

Shown below is an example of a varnish.sample.vcl configuration file for WURFL setup in Varnish 5. Please refer to the directives guide which explains each element in detail (Table 1), their parameters, constraints, and default recommended settings. In order to test the correct installation of the WURFL Varnish module, you can use the VCL script located in /usr/share/wurfl/varnish.sample.vcl:

```
vcl 4.0;

import wurfl;
import std;

backend default {
    .host = "127.0.0.1";
    .port = "8080";
}

sub vcl_init {
    ### WURFL root definition. User MUST specify this path in order to make WURFL engine correctly start.
    wurfl.set_root("/usr/share/wurfl/wurfl.zip");

    ### WURFL Updater allows for seamless updates of the WURFL engine with new data downloaded from Scienti
amobile.
    ### Updater configuration must be done after wurfl.set_root
    ### WURFL file should be either .zip or .xml.gz and match wurfl.set_root file type
    ### Put your personal updater URL taken from the Scientiamobile customer Vault.
    ### Valid values for the updater checking frequency (how often the updater checks for any new WURFL data fi
le
    ### to be downloaded and used by the engine) are DAILY,WEEKLY
    ### Updater log file (wurfl-updater.log) may be found in "wurfl.set_root" folder. The folder and wurfl.zip file (w
hich must already be present for the updater to work, should be writable by Varnish
    #wurfl.updater("https://data.scientiamobile.com/xxxxx/wurfl.zip","DAILY");
    #wurfl.updater("https://data.scientiamobile.com/xxxxx/wurfl.zip","WEEKLY");

    ### WURFL patches definition (as much as needed, patches will be applied in the same order as specified).
    #wurfl.add_patch("/path/to/patch1.xml");
    #wurfl.add_patch("/path/to/patch2.xml");

    ### WURFL cache: one of the following
    wurfl.set_cache_provider_lru(100000);
    #wurfl.set_cache_provider_none();

    wurfl.load();

    ### WURFL Updater startup. Must be done after wurfl.load
    ### With wurfl.updater_start the updater will execute an asynchronous check DAILY or WEEKLY (depending on
 wurfl.updater parameters)
    ### With wurfl.updater_runonce the updater will run only once executing a synchronous check
    # wurfl.updater_start();
    # wurfl.updater_runonce();

    if (wurfl.error()) {
            std.syslog(3, wurfl.error());
            return (fail);
    }
}

sub vcl_miss {
    ### Print requested static capabilities
    std.syslog(0, wurfl.get_capability("is_console"));

    ### Print useful WURFL setup information
    std.syslog(0, wurfl.get_api_version());
    std.syslog(0, wurfl.get_wurfl_info());
    std.syslog(0, wurfl.get_last_load_time());

    ### Print some virtual capabilities values
    std.syslog(0, wurfl.get_virtual_capability("advertised_browser"));
    std.syslog(0, wurfl.get_virtual_capability("advertised_browser_version"));
    std.syslog(0, wurfl.get_virtual_capability("advertised_device_os"));
    std.syslog(0, wurfl.get_virtual_capability("advertised_device_os_version"));
    std.syslog(0, wurfl.get_virtual_capability("is_largescreen"));
    std.syslog(0, wurfl.get_virtual_capability("is_app"));

    return(fetch);
}

sub vcl_pipe {
    return(pipe);
}

sub vcl_pass {
    return(pass);
}
```

Available Varnish Module functions list

Syntax	Description	Availability
set_root(string)	Defines the location (path) of the WURFL data file.	1.4
updater(string, string)	Allows seamless update of WURFL engine with new data downloaded from Scientiamobile. A call to set_root must precede it. 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 set_root file type and updater data url file type match so you may need to change the set_root file type accordingly. • 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 set_root folder and file must be writable by varnish. The wurfl-updater.log file in set_root folder will contains details on Updater activity.	1.8.3
add_patch(string)	Adds one or more custom patch files to the WURFL repository.	1.4

set_engine_target_default()
or
set_engine_target_fast_desktop_browser_match()
or
set_engine_target_high_performance()
or
set_engine_target_high_accuracy()
	These configuration options are deprecated and will be removed in a future release.	1.4

Syntax	Description	Availability
set_cache_provider_lru() or set_cache_provider_none(int)	In order to increase performance while processing real HTTP traffic, we suggest setting up an LRU cache. The LRU caching strategy will speed up lookup operations on processed User Agents by keeping them in an LRU map. By default the cache will be set to 30000 entries which accounts for 7 to 10 MB of additional memory usage. Specific concerns regarding memory usage apart, users are advised to size their cache generously (100,000 or more) to increase performance. For more information, please see LRU Cache Mechanism.	1.4
add_requested_capability(string)	Defines one or more WURFL Static Capabilities to be loaded into memory run-time.	1.4
load()	Loads WURFL engine and makes it available to Varnish.	1.4
updater_start()	Starts the WURFL Updater and executes an asynchronous check DAILY or WEEKLY (depending on updater parameters). A load call must precede it in order to correctly start the Updater.	1.8.3
updater_runonce()	Runs the WURFL Updater only once, executing a synchronous check. A load call must precede it in order to correctly start the Updater.	1.8.3
error()	Returns the last WURFL call error message or empty if no errors were found.	1.4
const char * get_capability(string)	Returns the WURFL static capability value for the detected device as a zero terminated ASCII string. If the static capability is missing, this function returns 0.	1.4

Syntax	Description	Availability
int get_capability_as_int(string)	Returns the WURFL static capability value for the detected device as an integer value if the required static capability exists. If the static capability is missing, or the static capability's value is not a valid integer, this function returns 0.	1.4.4
bool get_capability_as_bool(string)	Returns the WURFL static capability value for the detected device as a boolean value if the required static capability exists. If the static capability is missing, or the static capability's value is not a valid boolean, this function returns 0.	1.4.4
const char * get_virtual_capability(string)	Returns the WURFL virtual capability value for the detected device as a zero terminated ASCII string. If the virtual capability is missing, this function returns 0.	1.5.0
bool get_virtual_capability_as_bool(string)	Returns the WURFL virtual capability value for the detected device as a boolean value if the required virtual capability exists. If the virtual capability is missing, or the virtual capability's value is not a valid boolean, this function returns 0.	1.5.0
int get_virtual_capability_as_int(string)	Returns the WURFL virtual capability value for the detected device as an integer value if the required virtual capability exists. If the virtual capability is missing, or the virtual capability's value is not a valid integer, this function returns 0.	1.5.0
const char * get_original_useragent()	Returns the original useragent coming with this particular web request.	1.5.1
const char * get_normalized_useragent()	Returns the normalized useragent.	1.5.1.3
const char * get_api_version()	Returns the currently used Libwurfl API version.	1.5.1

Syntax	Description	Availability
const char * get_engine_target()	**This function is deprecated and will be removed in a future release.**	1.5.1
const char * get_wurfl_info()	Returns a string containing informations on the parsed WURFL data file and its full path.	1.5.1
const char * get_last_load_time()	Returns the UNIX timestamp of the last time WURFL has been loaded successfully.	1.5.1

set_useragent_priority_override_sideloaded_browser_useragent()
or
set_useragent_priority_use_plain_useragent()
	These configuration options are deprecated and will be removed in a future release.	1.5.2

get_useragent_priority_as_string()	**This function is deprecated and will be removed in a future release.**	1.5.2

Environment Info and Virtual Capabilities in WURFL Varnish Module

Since virtual capabilities are automatically calculated by WURFL, there are no explicit commands to request them.

There are some functions used to retrieve the virtual capabilities values and also some useful WURFL environment configuration information.

Please take a look at the VCL example and the functions table specified above.

WURFL Varnish Module Examples

If you try the following examples, you should restart Varnish each time a change is made.

Example 1:

In this example, we declare the vcl_recv subroutine, which is called when the complete request has been received and parsed. Its purpose is to decide whether an HTTP request should be served. Thus, certain conditions should be satisfied prior to serving the request itself. For example, the req object represents a single request and its parameters can be accessed via "dotted notation".

In our case, we test if the URL of the request is equal to / (root). Moreover we check if the browser runs on a device which has a resolution height equal to 960 pixels. If both of these conditions are true then we can use the directive set to assign a specific home page URL (/wide.html) to the request parameter URL.

The wurfl object is accessible after all the required variables have been initialized (see the varnish.sample.vcl script described before for more info). We can query WURFL for a static capability value simply by calling the function get_capability(cap_name).

The subroutine vcl_error is called every time we hit an error. At this point, the request has been cached by Varnish and as a result we have access to the obj object's variables in order to check if some error has occured.

For example, we can read the status variable, which contains the HTTP response status code returned by the server, and act accordingly by showing a message to the user which describes the problem or restarts the transaction to connect to the server.

Please note that, in the subroutine vcl_error, we return the error object to the client, which in turn can take some further actions.

```
##################################################################
########## Wurfl Static Capability switch example 1 ##############
###################################################################
sub vcl_recv {
    if ( req.url  == "/" && wurfl.get_capability("resolution_height") == "800") {
        set req.url = "/wide.html";
    }
}

sub vcl_error {
    if (obj.status == 750) {
        set obj.http.Location = "/wideversion/";
        set obj.status = 302;
           return(deliver);
    }
}
```

Example 2:

This example is similar to the previous one and shows how to throw an error explicitly to the client when both conditions are satisfied. Note that in this case, the request will be discarded by Varnish.

```
###################################################################
########### Wurfl Static Capability switch example 2 #############
###################################################################
sub vcl_recv {
    if ( req.url  == "/" && wurfl.get_capability("resolution_height")  == "800") {
        error 750 "Moved Temporarily";
    }
}

sub vcl_error {
    if (obj.status == 750) {
        set obj.http.Location = "/wideversion/";
        set obj.status = 302;
           return(deliver);
    }
}
```

Example 3:

In this example we call the hash_data() function, which will hash the data passed as parameter. Using the + operator, we can concatenate strings and then pass the result to the hash function. In this case, we consider concatenating the requested URL with the Device ID matched by WURFL.

```
###################################################################
############# Caching only URL + Device ID example 3 ############
#### Content is cached only if the URL+Device_ID is matching #####
###################################################################
sub vcl_hash {
    hash_data(req.url+req.http.host+wurfl.get_device_id());
    return (hash);
}
```

You should restart Varnish every time you make a change to the configuration in order to force a reload of the VCL configuration:

```
pkill varnishd
varnishd -f /usr/share/wurfl/varnish.sample.vcl
```

> ***Note:*** *In order to use service varnish start you should set properly all the required parameters in /etc/sysconfig/varnish.*

> ***Note:*** *Due to a Varnish limitation, the only working log file is the syslog. If you want to check the error messages and other issues> that may have occurred, you should look into /var/log/messages.*