scientiamobile

"The first step in a great mobile experience"

ONSITE PHP API

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

Copyright © 2025 ScientiaMobile, all rights reserved. WURFL Cloud, WURFL OnSite, WURFL and, InFuze WURFL InSight and respective logos are trademarks of ScientiaMobile. Apache is the trademark of the Apache Software Foundation. NGINX is the trademark of Nginx Software Inc. Varnish is the trademark of Varnish Software AB

WURFL OnSite PHP API: User Guide

> ***Note:*** *If you are moving from our legacy Database API to PHP API, please refer to our detailed migration guide to ensure a smooth transition. Also, there are breaking changes in API release v1.8. For documentation of releases before API v1.8 please refer to the README file included with that version. If you need aditional help for versions prior to 1.8 feel free to contact us at: support@scientiamobile.com.**

Installation

To enable WURFL on your application you must register for a free account on scientiamobile.com and **download the latest release from your File Manager.** Once you have downloaded the latest release, extract the files to be accessible from your PHP enabled webserver.

> ***Note:*** *The WURFL API is closely tied to thewurfl.xml file. New versions of the wurfl.xml are compatible with old versions of the API by nature, but the reverse is **not** true. Old versions of the wurfl.xml are **not** guaranteed to be compatible with new versions of the API.*

Requirements

- PHP >= 7.4
- extension: xml (XMLReader)
- Composer - The WURFL OnSite PHP API uses Composer which is a tool for dependency management in PHP. It allows you to declare the libraries your project depends on and it will manage (install/update) them for you. Download and instructions can be found here.

Install dependencies

Inside of your project folder, run the following command:

```
composer install --optimize-autoloader --no-dev
```

WURFL Data Snapshot

To perform lookups, you will need a copy of your WURFL data snapshot (also referred to as thewurfl.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.

Adds WURFL PHP API to an existing project using composer.json

If your project uses Composer, you can simplify maintenance by specifying the WURFL PHP API as an artifact repository:

- copy the WURFL PHP API ZIP package in a folder which will contain the archives
- add the following configuration to your composer.json:

```
"repositories": [
  {
     "type": "artifact",
     "url": "/path/to/directory/with/zips/"
  }
],
"require": {
  "scientiamobile/wurfl-api": "*"
}
```

- run composer install to install the dependencies

> ***Note:*** *To update the package, simply copy the new ZIP archive in the package folder and run composer update*

Configuring WURFL

Starting in version 1.8.0.0, the API uses a dependency container to configure and inject the required dependencies.

To configure WURFL you need to create a PHP configuration file returning a Container object.

As reference or starting point, please refer to theconfig/config.php.example file.

1. Create a new WURFL Dependency Container with optional settings

```
// Example: Create WURFL Dependency Container
$container = new \ScientiaMobile\WURFL\Container\Container([
  'wurfl_snapshot_url' => 'https://data.scientiamobile.com/xxxxx/wurfl.zip',
  'wurfl_capability_filter' => [
      'device_os',
      'device_os_version',
      // ...
  ],
  'wurfl_patches' => [
      '/full/path/to/patch-1.xml',
      '/full/path/to/patch-2.xml',
      // ...
  ],
  // ...
]);
```

   The Container takes an optional argurment as an associative array of settings with the following keys:

   - **wurfl_db**: a full path to the WURFL DB file (default to: resources/wurfl.xml)
   - **wurfl_snapshot_url**: The WURFL Snapshot URL from your customer vault on scientiamobile.com (click on "View Accout" next to your OnSite license). Ex. https://data.scientiamobile.com/xxxxx/wurfl.zip.
   - **wurfl_patches**: an array containing the WURFL patches' full path
   - **wurfl_capability_filter**: an array with the capabilities to load in order to reduce memory usage and increase performance
   - **wurfl_storage_path**: the absolute path to the storage dir. Default to "storage"
   - **wurfl_debug**: a boolean that enables the debug mode (default to: false)

   > ***Note:*** *if wurfl_snapshot_url is specified the wurfl_db setting will be ignored.*

2. Create a Storage adapter using either the StorageFactory, or by simply instantiating one of the Storage classes.

   The API supports the following storage mechanisms:

   - File (default)
   - SQLite through the PDO extension
   - Redis through the redis extension
   - MySQL through the PDO or the mysqli extension
   - MongoDB through the mongodb extension
   - Memory

```
// Example: Create a File storage adapter
use ScientiaMobile\WURFL\Storage\StorageFactory;
$storage = StorageFactory::createFileStorage('/path/to/storage/folder');
```

3. Create a Cache adapter using either the CacheFactory factory, or by simply instantiating one of the Cache classes. The API supports the following caching mechanisms:

    - File (default)
    - APCu (APC User Cache)
    - Redis
    - SQLite
    - Memcache
    - MySQL
    - Null (no caching)

    ```
    // Example: Create a File cache adapter
    use ScientiaMobile\WURFL\Cache\CacheFactory;
    use ScientiaMobile\WURFL\Cache\CacheAdapterInterface;

    // Time to live for cache item
    $ttl = CacheAdapterInterface::NEVER; //Default
    $cache = CacheFactory::createFileCache($ttl, '/path/to/storage/folder');
    ```

4. Add the storage and cache adapters to the container

    ```
    $container->setStorageAdapter($storage);
    $container->setCacheAdapter($cache);
    ```

> ***Note:*** *Since the default configuration uses the File adapter for both Storage and Caching, you'll need to make sure your webserver can write to the* ***storage*** *directories.*
>
> ***Note:*** *As reference on how configure and use the Evalution version please refer to the section: [WURFL PHP API Basic Usage for the Evaluation version]*

WURFL Updater

API version 1.8.0.0 introduces WURFL updater - a new command line utility which allows a client using WURFL to automatically update while the standard WURFLEngine is running and serving requests. In order to use the WURFL Updater, you must have your personal WURFL Snapshot URL (wurfl_snapshot_url) in the following format: https://data.scientiamobile.com/xxxxx/wurfl.zip where xxxxx is replaced with you personal access token. Also, do note that the wurfl_storage_path specified in the dependency container must be writable from the process/task and a wurfl.zip file must already be present in order for the Updater to run and determine whether or not the file needs to be updated.

The wurfl-updater utility is located under:

- /your/project/root/path/vendor/bin/ for project that uses the API as composer dependency
- /path/to/wurfl-api/ for project that uses the API as standalone library

The wurfl-updater utility can be used to runon demand or periodic updates.

Running "periodic" updates.

This is the suggested configuration since it allows periodic checks for an updated version of the wurfl.zip file, downloads it, and builds the repository with the updated version.

The quickest way to enable periodic updates is to use a time-based job scheduler, like a crontab:

```
# Crontab example
# .---------------- minute (0 - 59)
```

```
# | .------------- hour (0 - 23)
# | |   .--------- day of month (1 - 31)
# | |   | .------- month (1 - 12) OR jan,feb,mar,apr ...
# | |   | | .----- day of week (0 - 6) (Sunday=0 or 7)  OR sun,mon,tue,wed,thu,fri,sat
# | |   | | |
# * *   * * *   <user>  <command>

# Run the wurfl-updater script with daily frequency
30 3 * * *       www-data  /path/to/php -d memory_limit=1024M /path/to/wurfl-updater -c /path/to/config.php > /dev
/null
```

Running "on demand" update.

```
php -d memory_limit=1024M /path/to/wurfl-updater -c /path/to/config.php
```

On demand update runs only once per call.

WURFL PHP API Basic Usage

```
require_once './vendor/autoload.php';

// Require the dependency container object
$container = require_once '/path/to/config/config.php';
// Create a new instance of WURFLEngine
$wurfl_engine = new \ScientiaMobile\WURFL\WURFLEngine($container);
// Get the device for the current request
$device = $wurfl_engine->getDeviceForHttpRequest();
// Get the value for a static capability
$device->getCapability("is_wireless_device");
// Get the value for a virtual capability
$device->getVirtualCapability("is_smartphone");
```

> ***IMPORTANT:*** *Empty header values are treated as valid and those headers are not discarded. If you build your HTTP request programmatically from a data source such as logs, DB data, spreadsheet, etc., please make sure that you DO NOT add headers with empty strings as values (this may also be the result of "casting" a NULL / NONE / NaN to a string):*
>
> *Avoid:*
>
> ```
> {headerName}:{headerValue}
> ```
>
> *Use:*
>
> ```
> if  notNullOrEmpty(headerValue):
>    {headerName}:{headerValue}
> ```

WURFL PHP API Basic Usage for the Evaluation version

The Evaluation package comes with an XML file containing several of our popular WURFL Device Capabilities that can be loaded directly from the API without use of the WURFL Updater.

```
require_once './vendor/autoload.php';

// Require the dependency container object
$container = new \ScientiaMobile\WURFL\Container();
// Build the WURFL Repository using the WURFL evaluation XML
\ScientiaMobile\WURFL\Repositories\RepositoryManager::build($container);
// Create a new instance of WURFLEngine
$wurfl_engine = new \ScientiaMobile\WURFL\WURFLEngine($container);
// Get the device for the current request
$device = $wurfl_engine->getDeviceForHttpRequest();
// Get the value for a static capability
$device->getCapability("is_wireless_device");
// Get the value for a virtual capability
$device->getVirtualCapability("is_smartphone");
```

> ***Note:*** *Because this configuration of the WURFL repository is built directly from the WURFL API, the first request could take a few seconds to be served.*

Capability Filtering

To reduce memory usage and increase performance, specify a subset of the 500+ WURFL capabilities.

You can set the desired capabilities as settings in the Dependency Container:

```
// Example: Create WURFL Dependency Container
$container = new \ScientiaMobile\WURFL\Container\Container([
    'wurfl_capability_filter' => [
        'device_os',
        'device_os_version',
        ...
    ]
]);
```

If you make any changes, or add a capability filter, you will need to reload the WURFL file so that only the capabilities you've specified will be loaded. Do note that the auto-reload system will not detect the change in your configuration and will reload the WURFL data automatically. You will need to either touch the WURFL file, or delete your persistence folder.

> ***Note**: If you are upgrading from a previous version, do note that, starting in WURFL API version 1.8.0.0, you no longer need to specify WURFL mandatory capabilities.*

Virtual Capabilities

Virtual capabilities are an important feature of the WURFL API that obtain values related to the requesting agent out of the HTTP request as a whole (as opposed to limiting itself to capabilities that are found in WURFL).

In order to compute its final returned value, a virtual capability may look at regular (non-virtual) capabilities as well as parameters derived from the HTTP request at run-time. Virtual capabilities are useful to model aspects of the HTTP Client that are not easily captured through the finite number of profiles in WURFL.

To retrieve the value of a virtual capability, use thegetVirtualCapability() method from a Device object:

```
$is_smartphone = $device->getVirtualCapability('is_smartphone');
```

Variable Name	Type	Description
is_app	boolean	Tells you if the Requesting HTTP Client is an App or not.

Variable Name	Type	Description
is_smartphone	boolean	This is a virtual capability that will tell you if a device is a Smartphone for some arbitrary (and subject to change) definition of Smartphone by ScientiaMobile. The virtual capability returns true or false. Patch files can use the is_smartphone control capability to override the value returned by the virtual capability. Control capability is_smartphone can take value default, force_true and force_false. private
is_mobile	boolean	This is just an ALIAS for is_wireless_device. There's no control capability associated to this virtual capability. private
is_full_desktop	boolean	This is just an ALIAS for ux_full_desktop. There's no control capability associated to this virtual capability. private
is_windows_phone	boolean	Check if device runs any version of Windows Phone OS. This virtual capability relies on the device_os (product_info group) capability. private
is_ios	boolean	Check if device runs any version of iOS. This virtual capability relies on the device_os (product_info group) capability. private

Variable Name	Type	Description
is_android	boolean	Check if device runs any version of Android OS. This virtual capability relies on the device_os (product_info group) capability. private
is_touchscreen	boolean	This virtual capability tells you whether a device has a touch screen. There is no control capability. Mostly an alias for pointing_method == touchscreen (product_info group) capability. private
is_largescreen	boolean	True if the device has a horizontal and vertical screen resolution greater than 480 pixels. Relies on the resolution_width and resolution_height (display group) capabilities. private
is_wml_preferred	boolean	True if the device is better served with WML. Capability relies on preferred_markup (markup group). private
is_xhtmlmp_preferred	boolean	True if the device is better served with XHTML MP (Mobile Profile). Capability relies on preferred_markup (markup group). private
is_html_preferred	boolean	True if the device is better served with HTML. Capability relies on preferred_markup (markup group). private
advertised_device_os	string	This virtual capability will infer the name of the Device OS based on user-agent string analysis (and possibly the analysis of other HTTP headers and WURFL capabilities). private
advertised_device_os_version	string	This virtual capability will infer the version of the Device OS based on user-agent string analysis (and possibly the analysis of other HTTP headers and WURFL capabilities). private

Variable Name	Type	Description
advertised_browser	string	This virtual capability will infer the name of the browser based on user-agent string analysis (and possibly the analysis of other HTTP headers and WURFL capabilities). private
advertised_browser_version	string	This virtual capability will infer the version of the browser based on user-agent string analysis (and possibly the analysis of other HTTP headers and WURFL capabilities). private
form_factor	enumerable	This virtual capability will return one of the following values that identify a client's form factor: Desktop, Tablet, Smartphone, Feature Phone, Smart-TV, Robot, Other non-Mobile, Other Mobile private
complete_device_name	string	Concatenates brand name, model name and marketing name (where available) of a device into a single string. private
is_phone	boolean	This is a virtual capability that will tell you if a device is a mobile phone . The virtual capability returns true or false. Patch files can use the is_phone control capability to override the value returned by the virtual capability. Control capability is_phone can take value default, force_true and force_false. private
is_app_webview	boolean	This virtual capability returns true if a HTTP request is from an app based webview. private

Variable Name	Type	Description
device_name	string	Concatenates brand name and marketing name of a device into a single string. If marketing name is not available, model name is used instead. private
advertised_app_name	string	This virtual capability will return the name of the application that generated the User-Agent or the HTTP request. private
is_robot	boolean	

Checking User-Agent frozenness and HTTP headers quality

Starting from version 1.12.5.0 the HttpRequest class provides two new methods: isUaFrozen(), headerQuality().

isUaFrozen(string $user_agent) returns a boolean value which, if true, means that the input User-Agent string won't be updated by the sender browser.

headerQuality() returns an enumeration value that describes the HTTP headers quality. It has three possible values:

- **HEADER_QUALITY_FULL**: all the headers needed for a successful WURFL detection are present. Eg. a header with all UA-CH header fields present
- **HEADER_QUALITY_BASIC**: only some of the headers needed for a successful WURFL detection are present.
- **HEADER_QUALITY_NONE**: no UA-CH headers are present

Example:

```
$headers = [
  'User-Agent' => 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/96.0.0.0 Safa
ri/537.36',
  'Sec-Ch-Ua' => '" Not A;Brand";v="99", "Chromium";v="96", "Google Chrome";v="96"',
  'Sec-Ch-Ua-Platform' => 'Linux',
];
$request = new HttpRequest($headers);
$is_ua_frozen = $request->isUaFrozen(); // true
$header_quality = $request->headerQuality(); // basic
```

Enable Fast Desktop Browser Match

> *Deprecated since 1.9.5.0*

Use this option when you have significant amounts of desktop browser traffic compared to mobile device. Will return "generic_web_browser" wurfl_id for the majority of web browsers.

```
// Enable the fast desktop browser match.
$wurfl_engine->enableFastDesktopBrowserMatch();
```

Examples

There is a usage example included with the PHP API in the examples/demo folder.

The file examples/demo/index.php demonstrates how you can use the API to display the capabilities of a visiting device.

IMPORTANT - Decommissioning of Engine Target options

Prior to version 1.9 of the API, users could choose between High Accuracy and High Performance 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 deprecate this option to simplify configuration and go in the direction of uniform API behavior in different contexts. Customers who may find themselves in the unlikely situation of having to analyze significant amounts of legacy web traffic may still enable the old high-performance internal behavior by enabling the ENGINE_TARGET_FAST_DESKTOP_BROWSER_MATCH option in their engine target configuration. Please note that users with the old HIGH_PERFORMANCE or HIGH_ACCURACY target engine will not receive any error. The old behavior will not be triggered, though. The DEFAULT target (corresponding to the old HIGH_ACCURACY) will be used instead.

© 2025 ScientiaMobile Inc.

All Rights Reserved.

NOTICE: All information contained herein is, and remains the property of ScientiaMobile Incorporated and its suppliers, if any. The intellectual and technical concepts contained herein are proprietary to ScientiaMobile Incorporated and its suppliers and may be covered by U.S. and Foreign Patents, patents in process, and are protected by trade secret or copyright law. Dissemination of this information or reproduction of this material is strictly forbidden unless prior written permission is obtained from ScientiaMobile Incorporated.