scientiamobile

"The first step in a great mobile experience"

DOCKER GETTING STARTED

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 Microservice for Docker

Using WURFL Microservice for Docker will allow you to run your own WURFL-based device detection service in your hosting infrastructure by deploying familiar Docker images as containers through a ScientiaMobile Docker repository. If you are not familiar with Docker and the possibilities of Docker containers, here is a good place to get started.

WURFL Microservice Client API (WM Client)

Given an HTTP Request and a device capability name, the WURFL Client API will return the corresponding property value.

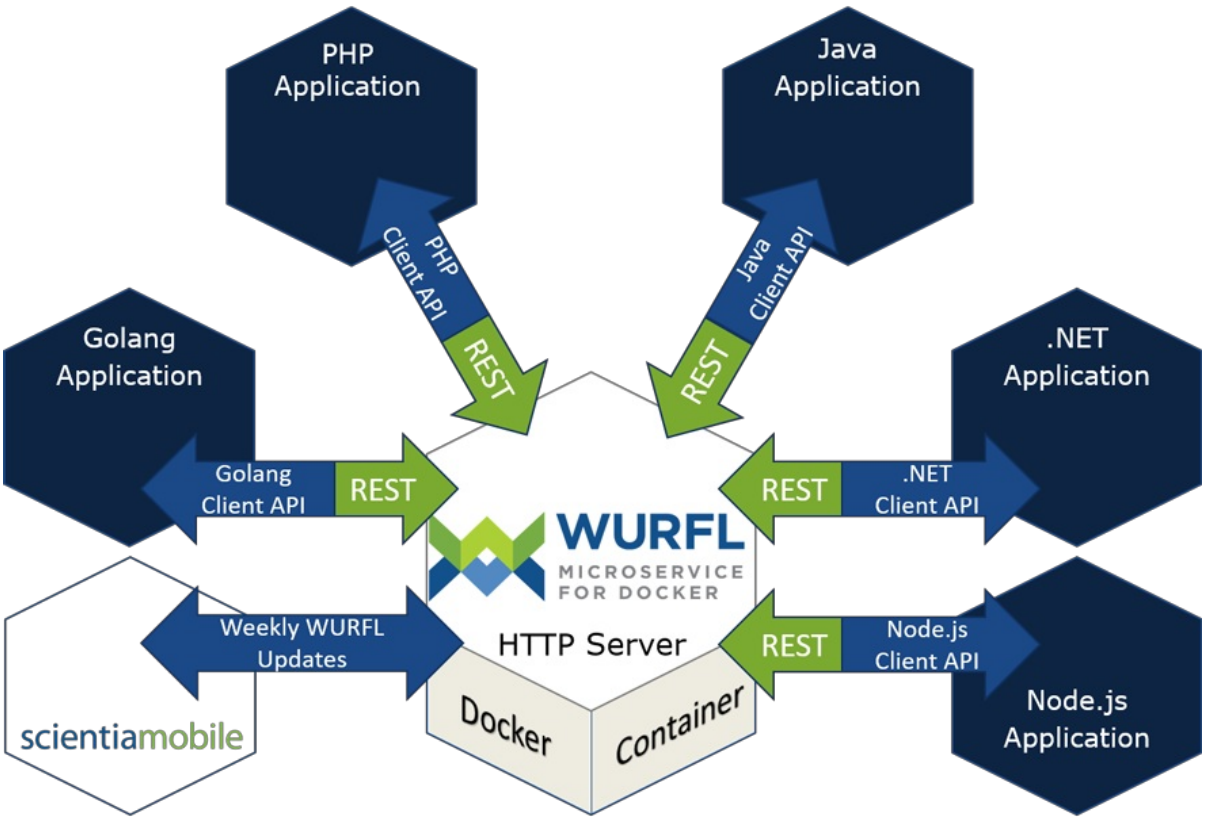
You can integrate the WURFL Microservice for Docker with major programming languages, including: You can integrate the WURFL Microservice Standard, Basic or Pro Edition with major programming languages, including:

- .NET (C#)
- Java
- PHP
- Node.js
- Golang.
- Python.
- Scala
- Rust

WURFL Microservice Architecture

WURFL Microservice Server (**WM Server**) is a service that exposes its functions to the WURFL Microservice Client API (**WM Client**). The Client API depends on the availability of the WM server to work.

The following diagram explains the overall architecture of WURFL Microservice as deployed through the ScientiaMobile Docker Private Registry. The Image will start an instance of the WURFL Microservice Server as a container. In addition to supporting the REST interface for API clients, the service will periodically query ScientiaMobile for updates to the WURFL Device Description Repository in order to include the profiles of newly released devices.



> ***Note 1:*** *While the WM Client feels like a "standalone API" for most practical purposes, under the hood it requires interaction with the WURFL Microservice HTTP server which introduces some latency (hugely mitigated by a built-in caching layer in the WM Client). For this reason, ScientiaMobile does not refer to the WURFL Microservice Client API (WM Client) as a "WURFL API" (that name is reserved for the [WURFL OnSite APIs]).*

> ***Note 2****: communication between the WM Client API and the WM Server happens through a REST API, but this is totally transparent to users of WURFL Microservice and ScientiaMobile acknowledges this aspect of the WURFL Microservice product for sake of transparency. Unless you have a very specific use-case(s), we strongly discourage you from utilizing the REST Interface directly. Not only does the WM Client provide a caching layer (which delivers much greater performance), but ScientiaMobile reserves the right to modify the internal REST API or even to switch to different non-REST protocols in future versions of the product without notice.*

Deploying the Docker Container and Setting up the WM Client APIs

1. (*Only for first-time WURFL Microservice users*) Obtain a license for the Product from ScientiaMobile.

This will let you use your **ScientiaMobile credentials** to login into the ScientiaMobile private registry and enable deployment of the product.

2. Login to the ScientiaMobile Docker Registry.

docker login docker.scientiamobile.com -u <username>

Details on the Docker login command can be found here.

Docker Images are currently available based on the set of Capabilities (Device Properties) and API Programming Language licensed from ScientiaMobile.

3. Use Docker to download, deploy, and start the WURFL Microservice HTTP server in an environment which makes sense for you (i.e. development, test, production, etc).

```
docker run --name wm-server --rm -p 8080:80 \
  -v /tmp/wm-server-logs:/var/log/wm-server \
  -e WM_CACHE_SIZE="200000" \
  -e WM_MAX_CORES=7 \
  -e WM_WURFL_UPDATER_DATA_URL=https://data.scientiamobile.com/xxxxx/wurfl.zip
  -e WM_ACCESS_LOG_TO_FILE=true \
  -e WM_ERROR_LOG_TO_FILE=true docker.scientiamobile.com/<license_id>/wurfl-microservice.server
```

- To reduce server load it is suggested to use a big server side cache:WM_CACHE_SIZE="200000" will do for a medium server, "300000" for a large one.
- Use WM_MAX_CORES to limit the number of threads used by the server (with no option server will use all necessary threads to keep serving requests).
- WM_WURFL_UPDATER_DATA_URL allow to download an up-to-date version of wurfl.zip before wurfl engine started, and enable automatic update of it (replace https://data.scientiamobile.com/xxxxx/wurfl.zip with your data URL from ScientiaMobile Vault)
- WM_ACCESS_LOG_TO_FILE=true will log HTTP access /var/log/wm-server/access.log
- WM_ERROR_LOG_TO_FILE=true will log errors in /var/log/wm-server/error.log

Important Note for Users of the Old DOUBLE LRU Cache Provider (pre 1.2.0.0): for backwards compatibility, older configurations are still supported and will not generate errors or warnings, but internally the new SINGLE LRU Cache is adopted for better performance.

4. Verify that the HTTP Server is running:

Copy the IP address of the web server, open a browser and visit the http://<instance-ip>/v2/status/json URL address. You should see a small JSON object that vouches for the health of your installation.

Alternatively, you can use curl from a shell terminal: assuming your IP is42.31.167.253, the following command will confirm that the Docker Image has been deployed successfully and that the WM Server is up and running.

```
$ curl  http://42.31.167.253/v2/status/json
{
   "lookup_request": 0,
   "lookup_useragent": 0,
   "lookup_device_id": 0,
   "make_model_requests": 0,
   "server_info_requests": 8,
   "v1_capabilities_requests": 0,
   "not_found_404": 0,
   "server_uptime": 23171
}
```

if your instance responds with something like this, everything looks good. Your WURFL Microservice server is up and running!

The WM Client API now has a WM server to talk to.

The WM Client API now has a WM server to talk to. More about Client APIs later.

5. (Only for first-time WURFL Microservice users) Install the WM Client API package and import the API in your project to start using it in your application.

WM Client APIs are available for the following supported languages (regardless of the capability set/tier): Go (golang), Java, .NET, Node.js and PHP.

WM Client APIs for the respective languages are provided on the ScientiaMobile public github repository:

- dotNET : https://github.com/WURFL/wurfl-microservice-client-dotnet
- Golang : https://github.com/WURFL/wurfl-microservice-client-golang
- Java : https://github.com/WURFL/wurfl-microservice-client-java
- Node.js : https://github.com/WURFL/wurfl-microservice-client-nodejs
- PHP : https://github.com/WURFL/wurfl-microservice-client-php
- Python : https://github.com/WURFL/wurfl-microservice-client-python
- Rust : https://github.com/WURFL/wurfl-microservice-client-rust

You can find the documentation for each WM Client API here:

1. Golang (GO Language)
2. Java
3. Microsoft .NET
4. PHP
5. Node.js
6. Python
7. Rust

Maven Central, Nuget and NPM also carry the WM Client API libraries for Java, .NET and Node.js respectively.