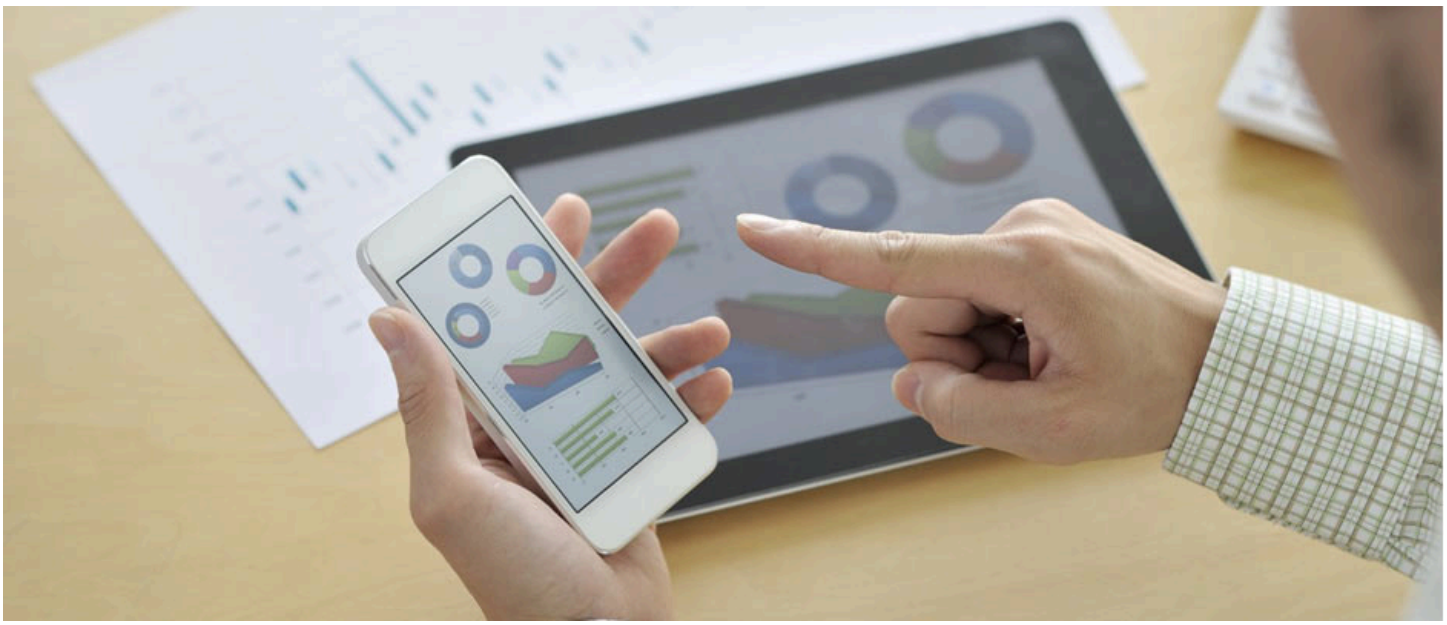
scientiamobile

"The first step in a great mobile experience"



WURFL INFUZE MODULE FOR PYTHON (PYWURFL)

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 © 2026 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 InFuze Module for Python (PyWURFL)

IMPORTANT NOTE: As of December 31st 2023, Python 2.7 will no longer be supported.

PyWURFL is a Python module wrapping the InFuze WURFL C API and encapsulating it in an object-oriented manner, to provide a fast, intuitive interface. Only Python3 builds are supported.

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 the **Downloads** section of your license in the **Customer Vault**.

If you have not already installed libwurfl, instructions can be found **here**. Changes in each release are listed in the **WURFL InFuze API Changelog**.

Compatibility

Python 2.7 is end-of-life and as of December 31st 2023, PyWURFL has dropped support for Python 2.7. This document assumes you are using Python 3.

Installation

PyWURFL is distributed as a Python wheel package, and should be installed using `pip`.

DO NOT use `pip install pywurfl` to install the WURFL InFuze for Python module.

Once you have installed `libwurfl`, download the **WURFL InFuze for Python** package from the **Customer Vault** (click **Manage** on your license, then **Downloads**). It will be a ZIP file like `infuze_python-1.12.0.0-linux-universal.zip`, follow these steps:

```
# Go to your Python project or any other directory if you're installing PyWURFL
globally
# Make a subdirectory for the PyWURFL contents
mkdir pywurfl
cd pywurfl

# Download the package here and unzip it
unzip infuze_python-*.zip

# You will see a tar file, which needs to be unpacked as well
tar xvf python-mod_wurfl-*.tar
cd dist
```

Now you will see the wheel (.whl) file `pywurfl-<pywurfl_version>-py3-none-any.whl`

Next, you must determine which Python environment you want to install the module into, for example:

```
# Global Python 3
$ python3 --version
Python 3.7.5

# Global Python 3.8
$ python3.8 --version
Python 3.8.0

# venv Python 3.8
$ ./my-project/bin/python --version
Python 3.8.0
```

> *Note: It's important that you install PyWURFL in the right environment for your project - you can install the module in any or all of them, but your project won't be able to use PyWURFL unless you have installed it that environment. If you're installing it globally, your "environment" is the global environment, and you can simply use* `python3`*, for example.*

Once you've determined which Python environment to use, you can install the module. In this example, we'll use the **Python virtual environment** in our test project `my-project`:

> *Attention:* ***DO NOT*** *use* `pip install pywurfl` *to install the WURFL InFuze for Python module.*

```
# Check the version of Python
$ ./my-project/bin/python --version
Python 3.8.0

# This is Python 3.8, so we install the `py3` package:
$ ./my-project/bin/python -m pip install ./pywurfl-3.9-py3-none-any.whl
```

> *If you get an error like* `No module named pip`*, you must first install* `pip`*. This may be available from your package manager (usually* `python3-pip`*), or you can* ***install it manually****.*

Python wheel packages have a specific naming convention, in the example above, we installed `pywurfl-3.9-py3-none-any.whl`.

To determine the right package for you, find the major version of your Python interpreter (with `python3 --version`, for example), and take the first digit, (ex: `Python 3.8.1` => `3`). This corresponds to the `python tag`, which starts with `py`, for example `Python 3.6.5` would be `py3`. As mentioned above, starting from version 3.0, pyWURFL packages are universal, the 'none' and 'any' tokens in the package name mean that it is not OS-specific and that it is suitable for all architectures.

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**.

Sample Usage

On Windows OS, if you have customized the libwurfl library installation path, you will have to set the environment variable `WURFL_SHARED_LIBRARY_PATH` with the full path of the libwurfl.dll file with the value you have chosen when installing libwurfl from the `.msi` installer.

Here is an example to get started using PyWURFL:

```
# Import WURFL InFuze (PyWURFL) module
from pywurfl.wurfl import Wurfl, wurfl_download, UPDATER_FREQUENCIES, CACHE_PROVIDERS

# before creating a WURFL engine, we download an updated version of the WURFL file in
the current directory
# destination directory must be writable
try:
    snapshot_url = "https://data.scientiamobile.com/xxxxx/wurfl.zip"

    # replace this URL with your customer specific one, to avoid error 402
    wurfl_download(snapshot_url, ".")
    print("WURFL file download succeeded")

except RuntimeError as e:
    print(f"An error occurred: {e}, using default file")

# Create a WURFL Engine with a 100,000 entries LRU cache
# and a WURFL updater
wurfl = Wurfl(root='wurfl.zip', cache_provider=CACHE_PROVIDERS['LRU'],
cache_extra_config="100000")
# start the updater to recevive the latest WURFL updates file
wurfl.set_updater_data_url(snapshot_url)
wurfl.set_updater_data_frequency(UPDATER_FREQUENCIES['DAILY'])
wurfl.updater_start()

# Lookup an HTTP request
user_agent = "Mozilla/5.0 (Linux; Android 15;  Pixel 9 Build/BP1A.250405.007; wv)
AppleWebKit/537.36 (KHTML, like Gecko) Version/4.0 Chrome/116.0.7103.61 Mobile
Safari/537.36 Line/15.5.4/IAB"
http_request = {
    "accept-encoding": "gzip, deflate, br",
    "accept-language": "en-US,en;q=0.9",
    "accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp",
    "User-Agent": user_agent,
    "Sec-CH-UA": "\"Chrome\";v=\"116\", \"HuaweiBrowser\";v=\"99\", \";Not A
Brand\";v=\"99\"",
    "Sec-CH-UA-Full-Version": "116.123.456",
    "Sec-CH-UA-Platform": "Android",
    "Sec-CH-UA-Platform-version": "15",
    "Sec-CH-UA-Model": "Pixel 9"
}

dev = wurfl.parse_headers(http_request)

# You can also lookup a device with just the user-agent string
# dev = wurfl.parse_useragent(user_agent)

# retrieve some properties and capabilities values
```

```
# WURFL device ID:
print("device id =", dev.id)

# Some static capabilities:
static_capabilities = ["model_name", "brand_name", "device_os"]

# Retrieve the value of a single static capability:
print("get_static_capability('model_name') =",
      dev.get_static_capability(static_capabilities[0]))

# Retrieve the value of many static capabilities at once:
print("get_static_capabilities(static_capabilities) =",
      dev.get_static_capabilities(static_capabilities))

# Some virtual capabilities:
virtual_capabilities = ["complete_device_name", "form_factor"]

# Retrieve the value of a single virtual capability:
print("get_virtual_capability('complete_device_name') =",
      dev.get_virtual_capability(virtual_capabilities[0]))

# Retrieve the value of many virtual capabilities at once:
print("get_virtual_capabilities(virtual_capabilities) =",
      dev.get_virtual_capabilities(virtual_capabilities))

# Make sure you release the device when you are finished
dev.release()

# example of getting all devices exposed by the currently running WURFL API
devices = wurfl.get_all_devices()
print("Printing all device ids:")
for d in devices:
    print(d.id)
    # it is extremely important to release each device when it's not needed anymore
    d.release()

# before program exit we must release the wurfl engine
if wurfl is not None:
    wurfl.release()
```

Here we create the `Wurfl` engine, then obtain the `device` object through lookup on a user agent string. Then we get device properties and static and virtual capability values as needed. Please note that virtual capabilities are calculated at runtime, so they might be significantly slower than static capabilities.

By running the above code, you should get an output like:

```
device id = google_pixel_9_ver1_suban150
get_static_capability('model_name') = Pixel 9
get_static_capabilities(static_capabilities) = {'model_name': 'Pixel 9',
'brand_name': 'Google', 'device_os': 'Android'}
get_virtual_capability('complete_device_name') = Google Pixel 9
get_virtual_capabilities(virtual_capabilities) = {'complete_device_name': 'Google
Pixel 9', 'form_factor': 'Smartphone'}
```

WURFL Updater

If you want to keep your `wurfl.zip` up-to-date with ScientiaMobile's data release schedule, please consider using the Updater features, available in WURFL InFuze for Python as follows:

After creating your WURFL engine, set your personal WURFL Snapshot URL (in the form "https://data.scientiamobile.com/xxxxx/wurfl.zip", available on the **Snapshot generator** page of your license in the Customer Vault):

```
wurfl = Wurfl('wurfl.zip')
try:
    wurfl.set_updater_data_url("https://data.scientiamobile.com/xxxxx/wurfl.zip")
except Exception as exception:
    print("Error while setting updater data URL: ")
    print(exception)
```

Note: you must use the same file type (`zip` or `gz`) in the updater URL that you use in the initial `Wurfl()` construction.

For long-running scripts, you can specify the frequency you want to check for updates: (DAILY or WEEKLY, default is DAILY):

```
wurfl.set_updater_data_frequency(wurfl.UPDATER_FREQUENCIES["DAILY"])
```

Note: the wurfl path must be writable, and a `wurfl.zip` file must already be present in order for the Updater to determine whether or not an update is required.

Then start the updater:

```
try:
    wurfl.updater_start()
except Exception as exception:
    print("Error while starting the updater: ")
    print(exception)
```

Updater will run a periodic check for the latest release of the `wurfl.zip` file, download it, and update the running engine to the latest version - all during normal application operations.

The internal updater also supports simple file logging, useful in debugging network problems and the like:

```
wurfl.set_updater_log_path("updater.log")
```

Please note that:

- The WURFL data file and the path where it resides, specified in the WURFL engine construction, **MUST** have write/rename access. The old data file will be replaced (i.e. a rename operation will be performed) with the updated version upon successful update operation completion, and the directory will be used for temp file creation, etc.
- ScientiaMobile does not distribute uncompressed XML data files via the updater. This means that, if you plan to use the updater, you **MUST** use the compressed (i.e. a ZIP or a XML.GZ) data file in the engine construction call.

`set_updater_data_frequency()` sets how often the updater **checks** for an updated data file, not how often the engine data file is actually updated.

The WURFL InFuze Updater functionality relies on availability and features of the well-known and widely available `curl` command-line utility. A check for `curl` availability is done in the `set_updater_data_url()` call.

© 2026 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. include('layouts.partials.license-footer')