scientiamobile

"The first step in a great mobile experience"

INFUZE RUBY 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 Ruby: User Guide

WURFL InFuze for Ruby is a module wrapping the WURFL C API and encapsulating it in an object oriented manner to provide a fast and intuitive interface. It is available for Linux and MacOS X platforms for Ruby versions 1.9 or higher.

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.

Installation on Linux/MacOS X

WURFL InFuze for Ruby is available as a gem and requires the WURFL InFuze C API to be installed. If you have not already installed libwurfl, instructions can be found here.You will needruby-dev and rubygems:

On debian based Linux:

```
$ sudo apt-get install ruby-dev
$ sudo apt-get install rubygems
```

On Redhat/CentOS:

```
$ sudo yum install ruby-devel
$ sudo yum install rubygems
```

Now install the gem:

```
$ sudo gem install --no-ri --no-rdoc wurfl-<version>.gem
```

Installation on Windows

The Ruby module has been tested with a RubyInstaller environment on Windows. On Windows, WURFL InFuze libs go to C:\Windows\System32.

Install the gem:

```
C:\> gem install --no-ri --no-rdoc wurfl-<version>.gem
```

> ***Note:*** *if you encounter any problem with RubyInstaller, please verify that you have correctly set up the environment variables.*

Usage

Here is an example to get started:

```
require 'wurfl_obj'

# create Wurfl object
MyWurfl = Wurfl.new("wurfl.zip", patches=[], engine_target=:WURFL_ENGINE_TARGET_DEFAULT,\
                cache_provider=:WURFL_CACHE_PROVIDER_LRU, cache_extra_config="10000")

# Define a test User Agent
UserAgent = "Mozilla/5.0 (Linux; Android 5.0; SAMSUNG SM-G925 Build/LRX21V) AppleWebKit/537.36\
          (KHTML, like Gecko) SamsungBrowser/4.0 Chrome/44.0.2403.133 Mobile Safari/537.36"

# Lookup a UserAgent string (renamed device method "parse_useragent" to "lookup_useragent" starting
# with v1.9.0.0)

begin
   dev = MyWurfl.lookup_useragent(UserAgent)
```

```
    rescue WurflError => err
        puts err.errorcode
        puts err.errormessage
end

# print deviceid
print("deviceid = " + dev.get_device_id() + "\n") # device ID

# print device capability
print("is_mobile = " + dev.get_capability("is_mobile") + "\n") # capability

# print virtual capability
print("is_ios = " + dev.get_capability("is_ios") + "\n") # virtual capability

# get a set of capabilities
print(dev.get_capabilities(["brand_name", "model_name", "is_ios"]))

# Get all capabilities
print(dev.list_device_capabilities())

# Release the device object to avoid memory leaks
dev.release()
```

WURFL Updater

If you want to keep your wurfl.zip up-to-date with the ScientiaMobile data release schedule, please consider applying WURFL Updater.

To start, after creating your WURFL Engine, set your personal WURFL Snapshot URL (in the form "https://data.scientiamobile.com/xxxxx/wurfl.zip", with "xxxxx" replaced with your personal access token, located in your license account page):

```
begin
  MyWurfl.set_updater_data_url("https://data.scientiamobile.com/<your access token>/wurfl.zip")
  rescue WurflError => err
    puts err.errorcode
    puts err.errormessage
end
```

Optionally, specify which periodicity (DAILY or WEEKLY, default is DAILY) you would like for update checks:

```
MyWurfl.set_updater_data_frequency(:WURFL_UPDATER_FREQ_DAILY)
```

Then start the updater:

```
begin
  MyWurfl.updater_start()
  rescue WurflError => err
    puts err.errorcode
    puts err.errormessage
end
```

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

Do note that the path should be writable, and a wurfl.zip file must already be present in order for the Updater to determine whether or not it has to pull an update.

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

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

Please note also:

- 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 his 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 updater. This means that, if you plan to use the updater, you **MUST** use a compressed (i.e, a ZIP or a XML.GZ) file as data file in the engine construction call.

Please note that set_updater_data_frequency() sets how often the updater **checks** for any 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