



ONSITE SCALA 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 © 2024 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 Scala API: User Guide

The Scala API is built as a wrapper around the WURFL Java API to expose methods to be available from within a Scala application. Below are the requirements that are needed in order to start including WURFL into your Scala project.

Requirements

- Java 8 JDK or above
- Scala library 2.12.x/2.13.x/3.x
- WURFL Java jar 1.11.0.0 or greater (it is highly recommended to use the same version of Scala and Java jar files)

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, you will simply need to add the WURFL Java JAR and the wurfl-scala.jar to your build path. Current distribution contains four JAR files, one for each supported Scala version. Pick the one that fits your needs.

Note: The WURFL API is closely tied to the wurfl.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.

Caching

It is possible to configure the API with a caching strategy in order to increase performance. Once you have instantiated your WURFL object, you must set the cache provider that you would like to use.

The available caching strategy is LRUMapCacheProvider.

LRUMapCacheProvider:

```
// Create a WURFL object
val wurflScala = new Wurfl(new GeneralWURFLEngine("classpath:/resources/wurfl.zip"))

// Set cache provider
wurflScala.setCacheProvider(new LRUMapCacheProvider)
```

Capability Filtering

Reduce memory usage and increase performance by specifying a subset of the 500+ WURFL capabilities.

You can set the desired capabilities using the setFilter() method:

```
// Create a WURFL object
val wurflScala = new Wurfl(new GeneralWURFLEngine("classpath:/resources/wurfl.zip"))

// Set Capability Filter
wurflScala.setFilter(
  "device_os",
  "brand_name",
  "model_name"
)
```

Note: beginning with API 1.8.0.0, it is no longer necessary to specify all mandatory capabilities in a filter.

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.

```
val wurflScala = new Wurfl(new GeneralWURFLEngine("classpath:/resources/wurfl.zip"))
var device = wurflScala.deviceForRequest("Mozilla/5.0 (Linux; U; Android 4.2.2; GT-I9505 Build/JDQ39) AppleWebKit/534.30 (KHTML, like Gecko) Version/4.0 Mobile Safari/534.30")
```

```
// Access a virtual capability
var smartphone = device.virtualCapability("is_smartphone")
```

Variable Name	Type	Description
is_app	boolean	Tells you if the Requesting HTTP Client is an App or not.
is_smartphone	boolean	<p>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.</p> <p>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.</p> <p>Control capability is_smartphone can take value default, force_true and force_false. private</p>
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

Variable Name	Type	Description
is_windows_phone	boolean	<p>Check if device runs any version of Windows Phone OS.</p> <p>This virtual capability relies on the device_os (product_info group) capability. private</p>
is_ios	boolean	<p>Check if device runs any version of iOS.</p> <p>This virtual capability relies on the device_os (product_info group) capability. private</p>
is_android	boolean	<p>Check if device runs any version of Android OS.</p> <p>This virtual capability relies on the device_os (product_info group) capability. private</p>
is_touchscreen	boolean	<p>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</p>
is_largescreen	boolean	<p>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</p>
is_wml_preferred	boolean	<p>True if the device is better served with WML. Capability relies on preferred_markup (markup group). private</p>
is_xhtmlmp_preferred	boolean	<p>True if the device is better served with XHTML MP (Mobile Profile). Capability relies on preferred_markup (markup group). private</p>

Variable Name	Type	Description
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
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

Variable Name	Type	Description
is_phone	boolean	<p>This is a virtual capability that will tell you if a device is a mobile phone .</p> <p>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.</p> <p>Control capability is_phone can take value default, force_true and force_false. private</p>
is_app_webview	boolean	<p>This virtual capability returns true if a HTTP request is from an app based webview. private</p>
device_name	string	<p>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</p>
advertised_app_name	string	<p>This virtual capability will return the name of the application that generated the User-Agent or the HTTP request. private</p>
is_robot	boolean	

Configuring WURFL updater

API version 1.8.2.0 introduces the WURFL updater; a new set of classes which allow a client using WURFL to automatically update the WURFL file. In order to use the WURFL Updater, you must have your personal 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 file path used to create the WURFL engine must be writable from the process/task that is executing the Scala API since Updater will update the file denoted by that path.

The suggested setting for the WURFL updater is "periodic" mode - in which theUpdater will periodically check to see if a new version of the wurfl.zip file has been released and, if so, download it and reload the engine with the new version. This can occur while the standard Wurfl engine is running and serving requests.

Running "periodic" updates.

```
val updater = Updater.apply(engine, "https://data.scientiamobile.com/xxxxx/wurfl.zip", patchPaths, proxySettings)
updater.performPeriodicUpdate(Frequency.DAILY)
```

proxySettings and patchPaths can be null

Being a periodic task, the updater will run perpetually until `updater.StopPeriodicUpdate()` is called.

Periodic update does not return a result. Failed/successful results must be checked in log files/console messages.

If needed, Updater can also run in "on demand" mode - in which it will check for a new version of the wurfl.zip once and then stop.

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

Running "on demand" update.

```
val updater = Updater.apply(engine, "https://data.scientiamobile.com/xxxxx/wurfl.zip", patchPaths, proxySettings)
UpdateResult result = updater.PerformUpdate()
```

On demand update runs only once per call and returns a result that can be used to programmatically check if update has been successful or, in case of failure, get an error message.

Sample

A sample getting started project can be found below:

```
package com.scientiamobile.wurfl.examples
import com.scientiamobile.wurfl.Updater
import com.scientiamobile.wurfl.Wurfl
import com.scientiamobile.wurfl.core.GeneralWURFLEngine
import com.scientiamobile.wurfl.core.cache.{DoubleLRUMapCacheProvider, HashMapCacheProvider, LRUMapCacheProvider}
import com.scientiamobile.wurfl.core.matchers.MatchType
import com.scientiamobile.wurfl.core.updater.Frequency
```

```
object Demo {
```

```
  def main(args: Array[String]) {
```

```
    print("Running scala version ")
    println(scala.util.Properties.scalaPropOrElse("version.number", "unknown scala version"))
```

```
    // Create WURFL passing a GeneralWURFLEngine object with a wurfl xml
    val engine = new GeneralWURFLEngine("wurfl.zip")
    val wurflWrapper = new Wurfl(engine)
```

```
    // Set cache provider
    wurflWrapper.setCacheProvider(new LRUMapCacheProvider)
```

```
    // Set Performance/Accuracy Mode
    wurflWrapper.setTargetAccuracy
```

```
    // Set Capability Filter
    wurflWrapper.setFilter(
      "can_assign_phone_number",
      "marketing_name",
      "brand_name",
      "model_name",
      "is_smarttv",
      "is_wireless_device",
      "device_os",
      "device_os_version",
      "is_tablet",
      "ux_full_desktop",
      "pointing_method",
      "preferred_markup",
      "resolution_height",
```

```

    "resolution_width",
    "xhtml_support_level",
    "mobile_browser_version")

val updater = Updater.apply(engine, "https://data.scientiamobile.com/xxxxx/wurfl.zip", null, null)
updater.performPeriodicUpdate(Frequency.MINUTES)

// User-Agent here
var userAgent = ""

// Defining headers
var headers = Map("Accept-Datetime"->"Thu, 31 May 2007 20:35:00 GMT")
headers += ("Content-Type"-> "application/x-www-form-urlencoded")

var device = wurflWrapper.deviceForHeaders(userAgent, headers)
println("Device id: " + device.id)
val matchType = device.matchType
if (matchType == MatchType.conclusive)
{
    println("Match Type is conclusive")
}

val wireless = device.capability("is_wireless_device")
println("Is wireless: " + wireless.get)

val formFactor = device.virtualCapability("form_factor")
println("Form factor: " + formFactor.get)

println("CAPABILITIES: ")
println("-----")
val capabilities = device.capabilities
// foreach can receive a tuple (in this case keyvalue one and two
capabilities.foreach {kv => println(kv._1 + " - " + kv._2)}
println("VIRTUAL CAPABILITIES: ")
device.virtualCapabilities.foreach {kv => println(kv._1 + " - " + kv._2)}
println("-----")
updater.stopPeriodicUpdate
}
}

```

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}
```

IMPORTANT - Decommissioning of MatchMode 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 "remove" this option to simplify configuration and go in the direction of uniform API behavior in different contexts.

Accessing ScientiaMobile's private Maven repository using SBT

SBT is the de facto standard for building and packaging Scala applications. Many Scala developers prefer it over the classic Maven client to access Maven repositories too.

You can access ScientiaMobile's private Maven repository using SBT 1.5.0 or above, using this configuration.

1. Create a file called `repositories` under `<user_home>/sbt` and put this content inside it:

```
[repositories]
local
maven-local
maven-central
my-ivy-proxy-releases: https://maven.scientiamobile.com/repository/wurfl-onsite/, [organization]/[module]/(scala_[scalaVersion])/(sbt_[sbtVersion])/(revision)/[type]s/[artifact](-[classifier]).[ext]
my-maven-proxy-releases: https://maven.scientiamobile.com/repository/wurfl-onsite/
```

This will configure a list of repositories: your local one, Maven Central and ScientiaMobile's private repo.

2. Then create a file called `credentials` under the same directory and copy this content inside it, replacing the asterisks (`***`) with your actual repository credentials:

```
realm=Sonatype Nexus Repository Manager
host=maven.scientiamobile.com
user=***
password=***
```

3. The last step is to add a reference to the credentials file to your build file with this line:

```
credentials += Credentials(Path.userHome / ".sbt" / "credentials")
```

You can now use the usual `sbt update`, `sbt compile` commands on your project and download the WURFL API dependencies.

© 2024 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.