ONSITE JAVA API

Support

The ScientiaMobile Support Forum 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 © 2018 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 Java API: User Guide

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.**

The Java OnSite API release zip file contains a wurfl.xml (or wurfl.zip if compressed), located under /core.zip/release.zip. The wurfl.xml is a database, or Device Definition repository (DDR), that holds data about all the devices known to WURFL and it's fundamental to every WURFL based application.

If you are using Maven to handle external dependencies, you can import WURFL API by following this tutorial.

If you do not use Maven or any other dependency management tool, all the dependencies you need are packed into the onsite zip release file under core.zip/release.zip, where you will find a file wurfl-version_number.jar and a directory lib, which contains all the dependencies needed by the WURFL Java API to compile and run. Copy the dependencies in your project and compile and run your java programs using the -cp option. If you need more information about Java compilation from the command line, you can read this article by Oracle

> ***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.*

The WURFLEngine is a high-level interface introduced to further abstract and simplify management of the WURFL API.

Initializing the WURFLEngine is enough to start using the API and is the entry point to all WURFL functionalities. The default implementation of WURFLEngine is GeneralWURFLEngine.

> ***Please note*** *that since version 1.8.0.0 the package names have changed to com.scientiamobile : update your configuration accordingly. If by any chance you need to update with the old package name, a legacy release package will be present. All new modules, like WurflUpdater, will only be available with the new package names.*

```
import com.scientiamobile.wurfl.core.Device;
import com.scientiamobile.wurfl.core.GeneralWURFLEngine;

public class Wurfl {

    public static void main(String[] args) {
        // wurfl.xml path can be either absolute or relative to the application classpath root.
        GeneralWURFLEngine wurfl = new GeneralWURFLEngine("wurfl.xml");
        // load method is available on API version 1.8.1.0 and above
        wurfl.load();
```

```
        String user_agent = "Mozilla/5.0 (Linux; Android 4.2.1; N9600 Build/JOP40D) AppleWebKit/537.36 (KHTML, like Gecko) Chr
ome/42.0.2311.111 Mobile Safari/537.36";

        Device device = wurfl.getDeviceForRequest(user_agent);
        System.out.println("Is Tablet: " + device.getCapability('is_tablet'));
        System.out.println("Can Assign Phone Number: " + device.getCapability('can_assign_phone_number'));
    }
}
```

If you are configuring a web application using a web.xml file, you can:

```
<listener>
  <listener-class>
     com.scientiamobile.wurfl.core.web.WURFLServletContextListener
  </listener-class>
</listener>

<context-param>
  <param-name>wurfl</param-name>
  <param-value>/WEB-INF/wurfl.zip</param-value>
</context-param>

<context-param>
  <param-name>wurflPatch</param-name>
  <param-value>/WEB-INF/patch_1.xml,/WEB-INF/patch_2.xml</param-value>
</context-param>
```

WURFLEngine implementation is loaded lazily (ie: when getDeviceById or getDeviceForRequest are called for the first time), but starting from version 1.8.1.0 you can load it programmatically by invoking method load.

```
WURFLEngine wurfl = new GeneralWURFLEngine("wurfl.xml");
wurfl.load();
```

Capability Filtering

In order to reduce memory usage and increase performance, you can specify a subset of the 500+ WURFL capabilities to load into memory. The list of the selected capabilities can be set by calling the setCapabilityFilter method on a WURFLEngine instance. The list can be passed as either a string array (String[]) or as a generic collection (Collection<String>):

```
String[] capabilities = {
    "device_os",
    "brand_name",
    "model_name",
    "release_date",
    "has_qwerty_keyboard"
};
```

If you are configuring using a web.xml file, you can apply a filter by:

```
<context-param>
    <param-name>capability-filter</param-name>
    <param-value>
        device_os
        brand_name
        model_name
        release_date
        has_qwerty_keyboard
    </param-value>
</context-param>
```

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 the getVirtualCapability() method from a Device object:

```
device.getVirtualCapability("is_mobile");
```

Variable Name	Type	Description
is_app	enumerable	Tells you if the Requesting HTTP Client is an App or not. The control capability is called controlcap_is_app (virtual_capability group) and can have values default, force_true and force_false
is_smartphone	enumerable	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.
is_robot	enumerable	This is a virtual capability that tells you if the HTTP Client is a Bot (robot, crawler or other programmable agent that stalks the web). Control capability is is_robot (virtual_capability group) and can have values default, force_true and force_false.

Variable Name	Type	Description
is_mobile	enumerable	This is just an ALIAS for is_wireless_device. There's no control capability associated to this virtual capability.
is_full_desktop	enumerable	This is just an ALIAS for ux_full_desktop. There's no control capability associated to this virtual capability.
is_windows_phone	enumerable	Check if device runs any version of Windows Phone OS. This virtual capability relies on the device_os (product_info group) capability.
is_ios	enumerable	Check if device runs any version of iOS. This virtual capability relies on the device_os (product_info group) capability.
is_android	enumerable	Check if device runs any version of Android OS. This virtual capability relies on the device_os (product_info group) capability.
is_touchscreen	enumerable	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.
is_largescreen	enumerable	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.

Variable Name	Type	Description
is_wml_preferred	enumerable	True if the device is better served with WML. Capability relies on preferred_markup (markup group).
is_xhtmlmp_preferred	enumerable	True if the device is better served with XHTML MP (Mobile Profile). Capability relies on preferred_markup (markup group).
is_html_preferred	enumerable	True if the device is better served with HTML. Capability relies on preferred_markup (markup group).
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).
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).
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).
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).
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

Variable Name	Type	Description
complete_device_name	string	Concatenates brand name, model name and marketing name (where available) of a device into a single string.
is_phone	enumerable	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.
is_app_webview	enumerable	This virtual capability returns true if a HTTP request is from an app based webview.
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.
advertised_app_name	string	This virtual capability will return the name of the application that generated the User-Agent or the HTTP request.

Configuring with the Spring Framework

The WURFL API (starting with 1.4) is completely decoupled from the Spring framework. This allows non-Spring users to import and use WURFL without having to include the Spring library and everything involved with it.

Using Spring, does not mean that the web.xml file is no longer involved. You must tell your application that all the features of the API are now supported through Spring.

In practice, this means that web.xml will contain the following lines:

```
<context-param>
  <param-name>contextConfigLocation</param-name>
  <param-value>/WEB-INF/wurfl-ctx.xml</param-value>
</context-param>
  :
```

```
<listener>
  <listener-class>
    org.springframework.web.context.ContextLoaderListener
  </listener-class>
</listener>
```

In the wurfl-helloworld-spring-{version}.war web application, you can find an easy way to configure the WURFL API using Spring. The wurfl.ctx context file will contain:

```
<bean id="WurflHolder" class="com.scientiamobile.wurfl.core.GeneralWURFLEngine">
  <constructor-arg index="0" value="classpath:/wurfl.zip" />
  <!-- <constructor-arg index="1" value="<< patch here >>"/> -->
  <!-- <constructor-arg index="2" value="<< more patches here >>"/> -->
</bean>

<!-- DeviceCacheProvider -->
<bean id="deviceCacheProvider" class="com.scientiamobile.wurfl.core.cache.LRUMapCacheProvider" />
<!-- <bean id="deviceCacheProvider" class="com.scientiamobile.wurfl.core.cache.DoubleLRUMapCacheProvider" /> -->
<!-- <bean id="deviceCacheProvider" class="com.scientiamobile.wurfl.core.cache.HashMapCacheProvider" /> -->
<!-- <bean id="deviceCacheProvider" class="com.scientiamobile.wurfl.core.cache.NullCacheProvider" /> -->
<!-- <bean id="deviceCacheProvider" class="com.scientiamobile.wurfl.core.cache.EhCacheProvider" /> -->
```

If you would like to set a capability filter, you will need to add the following properties to your wurfl-ctx.xml:

```
<property name="capabilityFilter">
    <set>
    device_os
    brand_name
    model_name
    release_date
    has_qwerty_keyboard
    </set>
</property>
```

Finally, you can simply access the WURFLEngine instance with the code here (taken from a common Servlet method):

```
protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
  WebApplicationContext wac = WebApplicationContextUtils.getWebApplicationContext(getServletContext());
  WURFLEngine engine = (WURFLEngine)wac.getBean(WURFLEngine.class.getName());
  [...]
}
```

Configuring WURFL updater

API version 1.8.0.0 introduces WURFL updater; a new set of classes which allow a client using WURFL to automatically update. In order to use the WurflUpdater, you must have your personal WURFL Snapshot url in the following format: https://data.scientiamobile.com/xxxxx/wurfl.zip. Your personal URL to wurfl.zip file can be found into the My Account > View Account link for WURFL OnSite for Java in Scientiamobile website private area. Also, do note that rootPath passed to WURFLEngine constructor must be writable from the process/task that is executing the Java API since WurflUpdater will update rootPath file.

The suggested setting for WURFL updater is "periodic" mode; i.e. WurflUpdater will periodically check to see if a new version of the wurfl.zip has been released, and if so, download it and reload the engine with the new version; all while the standard WurflEngine is running and serving requests.

Running "periodic" updates.

```
String rootPath = "wurfl.zip";
WURFLEngine engine = new GeneralWURFLEngine(rootPath);
```

```
// remember to modify the url below with your personal WURFL Snapshot url
WURFLUpdater updater = new WURFLUpdater(engine, "https://data.scientiamobile.com/xxxxx/wurfl.zip");
updater.setFrequency(Frequency.DAILY);
updater.performperiodicUpdate();
```

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, WurflUpdater can also run in "on demand" mode, i.e. check for a new version of the wurfl.zip once and then stop.

Running "on demand" update.

```
String rootPath = "wurfl.zip";
WURFLEngine engine = new GeneralWURFLEngine(rootPath);
// remember to substitute url below with your personal WURFL Snapshot url
WURFLUpdater updater = new WURFLUpdater(engine, "https://data.scientiamobile.com/xxxxx/wurfl.zip");
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.

Proxy configuration for WURFL Updater

Sometimes you may need to configure a proxy to run a WURFLUpdater instance. In this case, the initialization becomes:

```
String rootPath = "wurfl.zip";
WURFLEngine engine = new GeneralWURFLEngine(rootPath);
// proxy settings
String proxyHost = "proxy.example.com" // replace it with your proxy host
int proxyPort = 80; // replace with your proxy port
ProxySettings proxy = new ProxySettings(proxyHost, proxyPort, Proxy.Type.HTTP);
// ProxySettings proxy = new ProxySettings(proxyHost, proxyPort, Proxy.Type.SOCKS); // SOCKS proxy type is also supported.

// remember to modify the url below with your personal WURFL Snapshot url
WURFLUpdater updater = new WURFLUpdater(engine, "https://data.scientiamobile.com/xxxxx/wurfl.zip", proxy);
```

Java application samples

The Java OnSite API release zip contains two web application samples that can be helpful to understand how you can setup your own WURFL based web application:

- wurfl-helloworld-servlet.zip
- wurfl-helloworld-spring.zip

For the application to work, a wurfl.xml (.zip) file must be added in the application classpath before compiling it.

> ***Note:*** *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**

> ***IMPORTANT - Decommissioning of engine target options***

Prior to version 1.9 of the API, users could choose between set_engine_target_high_accuracy and set_engine_target_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.

© 2018 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.