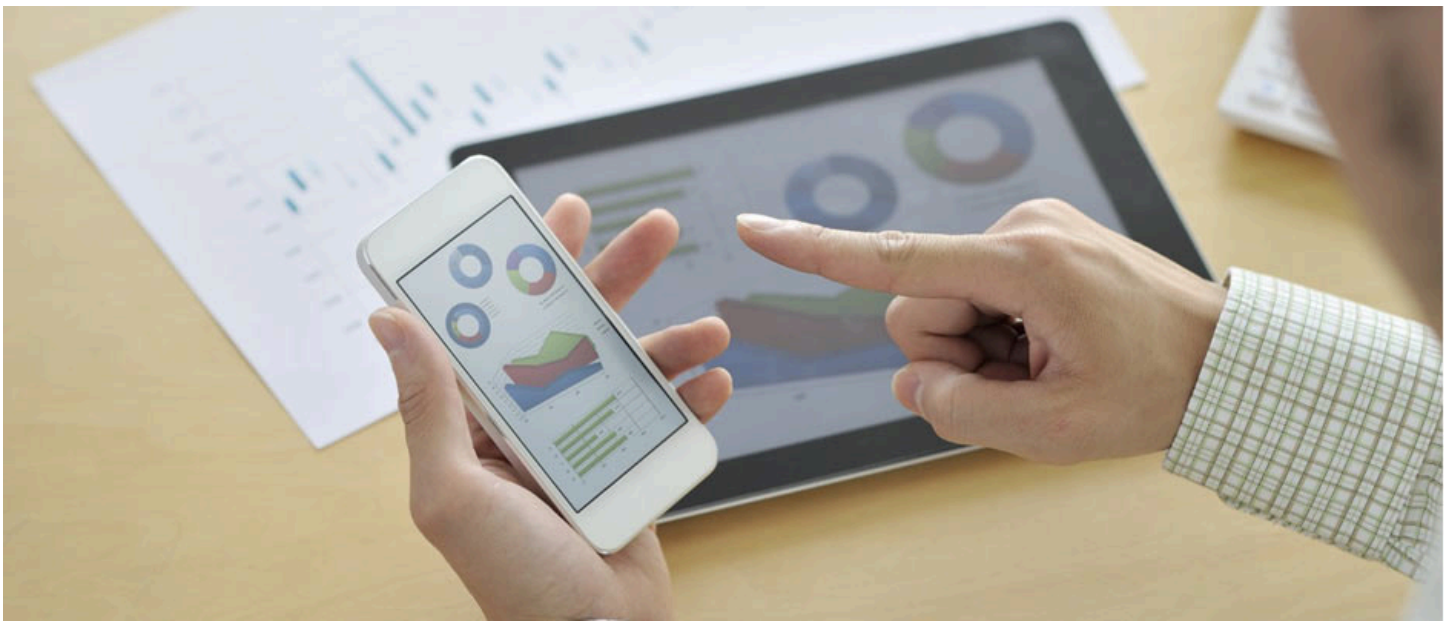
scientiamobile

"The first step in a great mobile experience"



WURFL ONSITE .NET API: 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

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 OnSite .NET API: User Guide

Installation

Start a Trial or Purchase a License

To access the installation package, please create a **free ScientiaMobile account** and **contact us** to initiate a trial or to purchase a license.

Where to Download the Installation Package

To download the installation package, **log in to your existing ScientiaMobile account**. On the **My Account** page of the Customer Vault, click **Manage** on your OnSite license, then click **Downloads**. Click **Download** next to a release of the WURFL API associated with your license to download it directly. To download it from the command line or another machine instead, open the ▫ menu next to the release and choose **Copy download link**, **Copy curl command**, or **Show link and command...**

> ***Note:*** *Download links are short-lived and only download the selected package. Copy a new link each time you need one, rather than saving it in scripts.*

For details on downloading WURFL data snapshots, please refer to **this documentation**.

The `Wurfl.dll` file must be added as a reference to any WURFL project.

> ***Note:*** *As of version 1.13.3.0 we support :*

1. .NET Framework 4.5.2, 4.6.2, 4.7.2, 4.8, 4.8.1
2. .NET Core 3.1
3. .NET 5.0, 6.0, 7.0, 8.0, 9.0

> ***Note:*** *The WURFL API is closely tied to the* `wurfl.zip` *file. New versions of the* `wurfl.zip` *are compatible with old versions of the API by nature, but the reverse is* ***not*** *true. Old versions of the* `wurfl.zip` *are* ***not*** *guaranteed to be compatible with new versions of the API.*

WURFL OnSite .NET API NuGet package

WURFL OnSite .NET API is available as a **NuGet package** too. To install, use the ScientiaMobile NuGet URL https://nuget.scientiamobile.com/repository/wurfl-onsite/. To browse from VisualStudio:

1. Go to **Tools -> NuGet Package Manager -> Package Manager Settings**
2. Then go to **Package Sources** and add a new Package Source
3. Give the new Package Source a **Name and Source URL** (https://nuget.scientiamobile.com/repository/wurfl-onsite/)
4. Next, go to **Tools -> NuGet Package Manager -> Package Manager Console** and run the following command: `Install-Package WURFLOnSite` (`Install-Package WURFLOnSite.NETCore` for .NET Core projects)
5. Once the command is executed, **enter your ScientiaMobile account credentials**

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**.

Getting Started

WURFL OnSite .NET API bases its operations on two main objects:

- a **WURFL Manager** object implementing the `IWURFLManager` interface.
- a **Device** object implementing the `IDevice` interface.

The **WURFL Manager** object should be instantiated only once in your application.

The **WURFL Manager** object offers several methods (among others) for you to gain access to the in-memory representation of the **Device Definition Repository (DDR)**.

```
public interface IWURFLManager
{
        .
        .
        IDevice GetDeviceForRequest(String userAgent);
        IDevice GetDeviceForRequest(HttpRequest request);
        IDevice GetDeviceById(String deviceId);
        .
        .
}
```

All of these methods return a **Device** object (implementing the `IDevice` interface) which represents the matched device model.

The **Device** object offers several methods (among others) for you to access the matched device data.

```
public interface IDevice
{
        .
        .
        String GetCapability(String name);
        String GetVirtualCapability(String name);
        IDictionary<String, String> GetCapabilities();
        IDictionary<String, String> GetVirtualCapabilities();
        String Id { get; }
        .
        .
}
```

In the next sections we'll see a sample **Console Application** using device detection and accessing WURFL static capabilities and virtual capabilities.

Console Application Usage

In your Console Application project, add the `Wurfl.dll` assembly as a reference.

> ***Note:*** *Beginning with version 1.8.4, the* `System.Web` *assembly must be referenced, even if you are building a* `Console Application`.

Add a new class named **WURFLSimpleTest** to your project with the following code.

```
using WURFL;
using WURFL.Config;

namespace YourNameSpace
{
    class WURFLSimpleTest
    {
        static void Main(string[] args)
        {
```

Before creating the WURFL manager instance, we download an up-to-date version of the WURFL file from the ScientiaMobile Snapshot generator into a writable directory of our choice.

```
            // Remember to modify the url below with your personal WURFL url or
you'll get a HTTP 402 error
            var wurflUrl = "https://data.scientiamobile.com/xxxxx/wurfl.zip";
            try
            {
                WURFLManager.WurflDownload(wurflUrl, ".");
            }
            catch(WURFLRuntimeException ex) {
                // handle the exception
            }
```

Create the `InMemoryConfigurer` object setting the WURFL data file path;

```
  try
            {
                InMemoryConfigurer configurer = new InMemoryConfigurer()
                    .MainFile(wurflFilePath);

                IWURFLManager manager = null;
```

Create the WURFL manager once, then lookup the UserAgent, and get the `Device-Id`, `Static Capabilities`, and `Virtual Capabilities` needed in your implementation (beware, Virtual Capabilities are calculated at runtime).

For further details on `Virtual Capabilities`, click **here**

```
                manager = WURFLManagerBuilder.Build(configurer);

                String ua = "Dalvik/1.6.0 (Linux; U; Android 4.3; SM-N900T
Build/JSS15J)";

                IDevice device = manager.GetDeviceForRequest(ua);

                Console.WriteLine("Device : {0}", device.Id);

                String capName = "brand_name";
                Console.WriteLine("Static Capability {0}: {1}", capName,
device.GetCapability(capName));

                String vcapName = "is_android";
                Console.WriteLine("Virtual Capability {0}: {1}", vcapName,
device.GetVirtualCapability(vcapName));
```

You can request a full list of Static and Virtual Capability name and values from the device instance.

```
                Console.WriteLine("--- Device Static Capabilities ---");
                foreach (KeyValuePair<string, string> dCap in
device.GetCapabilities())
                    Console.WriteLine("[{0}] = [{1}]", dCap.Key, dCap.Value);

                Console.WriteLine("--- Device Virtual Capabilities ---");
                foreach (KeyValuePair<string, string> vCap in
device.GetVirtualCapabilities())
                    Console.WriteLine("[{0}] = [{1}]", vCap.Key, vCap.Value);
                        }
```

WURFL will throw `Exceptions` in case of failure during the entire process

```
            catch (Exception e)
            {
                Console.WriteLine("WURFLSimpleTest throws this exception : {0} -
{1}", e.GetType(), e.Message);
            }
        }
    }
}
```

Passing the entire HTTP request for device detection

In addition to passing a User-Agent string to the WURFL API as shown in the example above, you can also pass a HTTP request for device detection. This is useful in cases where the HTTP request contains critical device information in places other than in the `User-Agent` header. This is most commonly seen in HTTP requests with **User-Agent Client Hints**.

You can either pass the HTTP request directly to `IDevice GetDeviceForRequest(HttpRequest request)` or instead pass a dictionary of the headers from the HTTP request:

```
            IDictionary<String, String> requestHeaders = new Dictionary<String,
String>
            {
                {"User-Agent", "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/100.0.0.0 Mobile Safari/537.36"},
                {"Sec-Ch-Ua", "\"Not_A Brand\";v=\"8\", \"Chromium\";v=\"120\",
\"Google Chrome\";v=\"120\""},
                {"Sec-Ch-Ua-Platform", "Android"},
                {"Sec-Ch-Ua-Platform-Version", "13.0.0"},
                {"Sec-Ch-Ua-Model", "Pixel 6"},
                {"Sec-Ch-Ua-Mobile", "?1"},
                {"Sec-Ch-Ua-Full-Version-List", "\"Not_A Brand\";v=\"8.0.0.0\",
\"Chromium\";v=\"120.0.6099.43\", \"Google Chrome\";v=\"120.0.6099.43\""},
                {"Sec-Ch-Ua-Arch", ""}
            };
            IDevice device = manager.GetDeviceForRequest(requestHeaders);
```

You can then request WURFL capabilities as shown in the **section above**.

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}
```

WURFL Cache

The WURFL manager has an `LRU` in-memory cache to preserve the result of previous detection.

If you want to enable the `LRU cache`, you can do it in `InMemoryConfigurer` object passing it the cache size:

```
configurer.SetCacheProvider(100000);
```

WURFL Updater

For API versions 1.8.1.1 and greater, you can keep your **wurfl.zip** file uptodate with Scientiamobile's data release schedule using the **WURFL Updater**.

To configure WURFL Updater, you will need your personal `WURFL Snapshot URL` (found on the **Snapshot generator** page of your license in the ScientiaMobile Customer Vault). You may configure the `frequency` for update checks.

Begin by adding the `Wurfl.Updater` namespace to your application.

```
using Wurfl.Updater;
```

Then, create a WURFLUpdater instance passing it the `manager` instance and your `updater url`

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

Note: the path of the **wurfl.zip** specified in the configurer at the moment of **WURFL Manager** creation must be writable from the process/task, and a **wurfl.zip** file must already be present in order for the Updater to determine whether or not it needs to update.
that is executing the .NET API, since `WURFLUpdater` will update the file denoted by its path.

There are two options in which you can invoke the updater.

- using the **PerformUpdate()** method which performs a single update check and then stop.

```
    updater.PerformUpdate();
```

- using the **PerformPeriodicUpdate()** method which performs update checks with a periodicity you can specify with the **SetFrequency(frequency)** method, chosing among `DAILY` `WEEKLY` (default is `DAILY`).

```
        updater.SetFrequency(Wurfl.Updater.Frequency.WEEKLY);
        updater.PerformPeriodicUpdate();
```

If you want to stop Periodic Updates, invoke the **StopPeriodicUpdate()** method

```
        updater.StopPeriodicUpdate();
```

> ***Note:*** *The **WURFL Updater** will check to see if a new version of the wurfl.zip has been released and, if so, download it and reload the **WURFL manager** with the new version; all while the **WURFL manager** still running*

and serving requests.

Dependencies for .NET Core projects

If your project uses .NET Core and you are manually referencing the `Wurfl.dll` assembly, you may need to add a PackageReference for `Microsoft.AspNetCore.Http.Abstractions`:

```
    <ItemGroup>
        <PackageReference Include="Microsoft.AspNetCore.Http.Abstractions"
Version="2.2.0" />
    </ItemGroup>
```

If you use **nuget** to integrate the WURFL API into your project, you can safely skip this section.

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 static capabilities that are found in WURFL).

Virtual Capabilities are calculated at runtime; in order to compute its final returned value, a virtual capability may look at static 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 get the value of a virtual capability:

```
var isSmartphone = device.GetVirtualCapability("is_smartphone");
```

The value associated with a virtual capability is always expressed as a string, even when it logically represents a number or a Boolean.

For further information on virtual capabilities, please refer to the **WURFL capabilities page**

© 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.