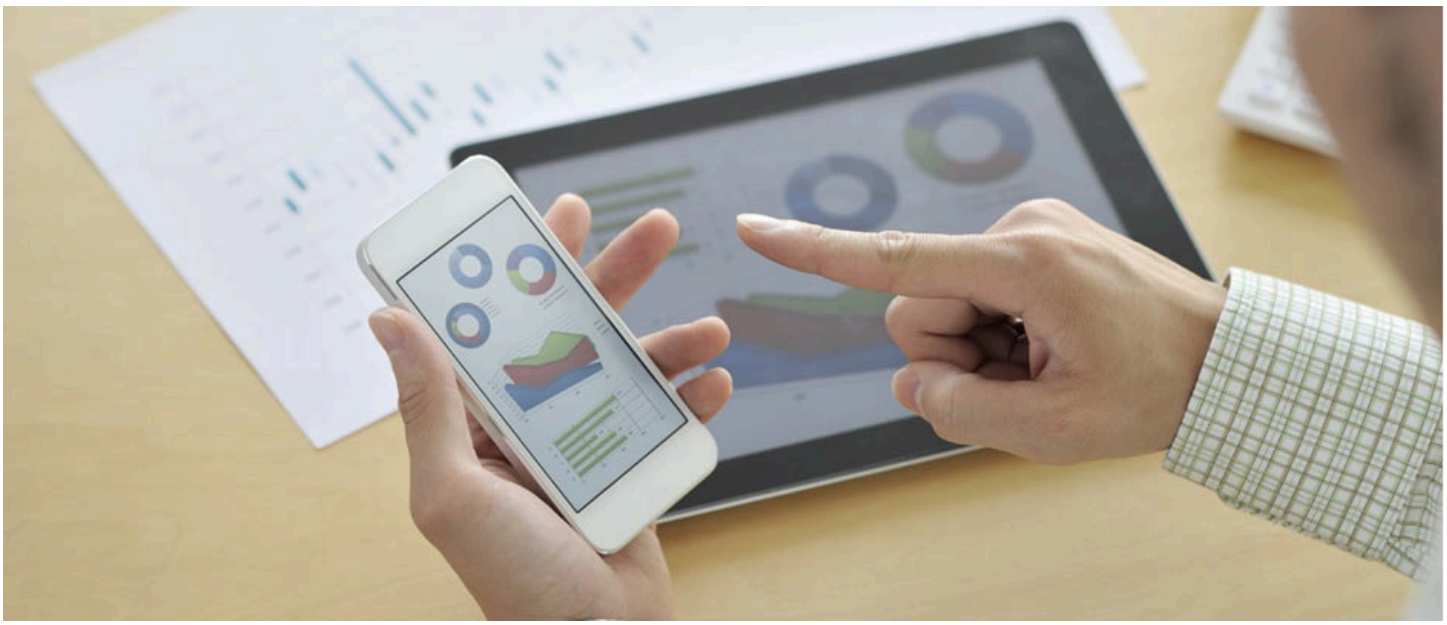
scientiamobile

"The first step in a great mobile experience"



WURFL INFUZE MODULE FOR GOLANG : 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 InFuze Module for golang : User Guide

WURFL InFuze for golang is a GO language module wrapping the WURFL C API and encapsulating it in two golang types to provide a fast and intuitive interface. It is compatible with Linux and macOS platforms. The minimum supported Go version is declared in the module's `go.mod` file.

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**.

Installation on Linux and macOS

InFuze for golang is available as a public Go module hosted at **github.com/WURFL/golang-wurfl**. To add it to your project, run:

```
go get -u github.com/WURFL/golang-wurfl
```

Then import it in your code as shown in the **Usage** section below.

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**. It's strongly recommended, as first step, to download the latest version of your WURFL data snapshot as shown in the example below.

Usage

Here is an example to get started:

```go
package main

import (
    "fmt"
    "os"

    wurfl "github.com/WURFL/golang-wurfl"
)



func main() {

    var wengine *wurfl.Wurfl
    var device *wurfl.Device

    // Download WURFL data file
    // substitute https://data.scientiamobile.com/xxxxxxx/wurfl.zip with your data URL from your Snapshot generator page in the Customer Vault
    // the second parameter is the destination folder where the zip file will be extracted. It must be writable by the process.
    if err := wurfl.Download("https://data.scientiamobile.com/xxxxxxx/wurfl.zip", "."); err != nil {
        fmt.Printf("Error downloading WURFL file: %v\n", err)
        os.Exit(1)
    }

    wengine, err := wurfl.Create("./wurfl.zip", nil, nil, -1, wurfl.WurflCacheProviderLru, "100000")

    if err != nil {
        fmt.Printf("Error creating WURFL engine: %v", err)
        os.Exit(1)
    }

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

    device, err = wengine.LookupUserAgent(ua)
    if err != nil {
        fmt.Printf("Error in lookup: %v\n", err)
        os.Exit(1)
    }

    deviceid, err := device.GetDeviceID()
    if err != nil {
        fmt.Printf("Error getting device ID: %v\n", err)
    }
    fmt.Println(deviceid)
```

```
    deviceOs, err := device.GetStaticCap("device_os")
    if err != nil {
        fmt.Printf("Error getting static capability: %v\n", err)
    }
    fmt.Println(deviceOs)

    isAndroid, err := device.GetVirtualCap("is_android")
    if err != nil {
        fmt.Printf("Error getting virtual capability: %v\n", err)
    }
    fmt.Println(isAndroid)

    device.Destroy()
    wengine.Destroy()
}
```

Create the WURFL Engine once, then lookup UserAgent and get the static and virtual capabilities needed in your implementation (**NOTE:** virtual capabilities are calculated at runtime).

If you already have a HTTP request, you can directly pass it to the WURFL API by using `wengine.LookupRequest()`. Alternately you can test passing a HTTP request to the WURFL API by creating a mock request, stuffing it with the desired test headers and passing them to the WURFL API as mentioned in the previous sentence. This is particularly useful for detecting requests with User-Agent Client Hints. In either case you can retrieve the resolved device as shown below. The static and virtual capabilities can then be requested as shown in the first snippet.

```
    req, _ := http.NewRequest("GET", "http://example.com", nil)
    UserAgent := "Mozilla/5.0 (Linux; Android 10; K) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/10.0.0.0 Mobile Safari/537.36"
    req.Header.Add("User-Agent", UserAgent)
    req.Header.Add("Sec-CH-UA", "\" Not A;Brand\";v=\"99\", \"Chromium\";v=\"110\",
\"Google Chrome\";v=\"110\"")
    req.Header.Add("Sec-CH-UA-Full-Version", "110.0.4430.91")
    req.Header.Add("Sec-CH-UA-Platform", "Android")
    req.Header.Add("Sec-CH-UA-Platform-Version", "11")
    req.Header.Add("Sec-CH-UA-Model", "SM-M315F")

    device, err = wengine.LookupRequest(req)
    if err != nil {
        fmt.Printf("Error in lookup: %v\n", err)
        os.Exit(1)
    }
```

Additional documentation on User-Agent Client Hints and how to collect them is available here.

WURFL Updater

If you want to keep your `wurfl.zip` up to date with the ScientiaMobile data release schedule, then you might want to use the Updater features.

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):

```
uerr := wengine.SetUpdaterDataURL(Url)
if uerr != nil {
    fmt.Printf("SetUpdaterDataUrl returned : %s\n", uerr.Error())
}
```

Specify the periodicity you would like for update checks:

```
_ = wengine.SetUpdaterDataFrequency(wurfl.WurflUpdaterFrequencyDaily)
```

Do note that a wurfl.zip file must already be present in a writable path in order for the updater to check the file and determine whether or not it needs to update the file. You can either manually download this file by going to your WURFL Snapshot URL or by following these directions.

Start the updater:

```
uerr = wengine.UpdaterStart()
if uerr != nil {
  fmt.Printf("UpdaterStart returned : %s\n", uerr.Error())
}
```

Updater will run a daily 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.

Populating IAB's OpenRTB V2 (ORTB2) Device Type

Starting with WURFL API 1.13.6.0, `ORTB2Getdevicetype()` will return a value corresponding to the device's OpenRTB V2 devicetype. More information on this topic is available in this blog post.

> ***Note:*** *The following static capabilities: is_ott, is_console, physical_form_factor and the following virtual capabilities: form_factor are required to be present in the loaded wurfl.xml file for this method.*

Checking User-Agent frozenness and HTTP headers quality

Starting from version 1.12.5.0 the `Wurfl` struct provides two new methods: `IsUserAgentFrozen()`, `GetHeaderQuality()`.

`IsUserAgentFrozen(ua string)` returns a boolean value which, if true, means that the input User-Agent string won't be updated by the sender browser.

`GetHeaderQuality()` returns an enumeration value that describes the HTTP headers quality. It has three possible values:

- **HeaderQualityFull**: all the headers needed for a successful WURFL detection are present. Eg. a header with all UA-CH header fields present
- **HeaderQualityBasic**: only some of the headers needed for a successful WURFL detection are present.
- **HeaderQualityNone**: no UA-CH headers are present

API Reference

wurfl

`import wurfl "github.com/WURFL/golang-wurfl"`

- Overview
- Index

Overview

Package wurfl - WURFL InFuze Golang module

Package wurfl - WURFL InFuze Golang module wurfl is a golang package wrapping the WURFL C API and encapsulating it in 2 golang types to provide a fast and intuitive interface. It is released for linux/macos platforms.

Index

- Constants
- Variables
- func APIVersion() string
- func Download(url string, folder string) error
- type Device

    - func (d *Device) Destroy()
    - func (d *Device) GetCapabilities(caps []string) map[string]string
    - func (d *Device) GetCapability(cap string) string
    - func (d *Device) GetCapabilityAsInt(cap string) (int, error)
    - func (d *Device) GetDeviceID() (string, error)
    - func (d *Device) GetMatchType() int
    - func (d *Device) GetNormalizedUserAgent() (string, error)
    - func (d *Device) GetOriginalUserAgent() (string, error)
    - func (d *Device) GetParentID() string
    - func (d *Device) GetRootID() string
    - func (d *Device) GetStaticCap(cap string) (string, error)
    - func (d *Device) GetStaticCaps(caps []string) (map[string]string, error)
    - func (d *Device) GetUserAgent() (string, error)
    - func (d *Device) GetVirtualCap(vcap string) (string, error)
    - func (d *Device) GetVirtualCapabilities(caps []string) map[string]string
    - func (d *Device) GetVirtualCapability(vcap string) string
    - func (d *Device) GetVirtualCapabilityAsInt(vcap string) (int, error)
    - func (d *Device) GetVirtualCaps(caps []string) (map[string]string, error)
    - func (d *Device) IsRoot() bool
    - func (d *Device) ORTB2GetDevicetype() (int, error)
- type DeviceHandler
- type HeaderQuality
    - func (hq HeaderQuality) String() string
- type Updater
- type Wurfl
    - func Create(Wurflxml string, Patches []string, CapFilter []string, EngineTarget int, CacheProvider int, CacheExtraConfig string) (*Wurfl, error)
    - func (w *Wurfl) Destroy()
    - func (w *Wurfl) GetAPIVersion() string
    - func (w *Wurfl) GetAllCaps() []string
    - func (w *Wurfl) GetAllDeviceIds() []string
    - func (w *Wurfl) GetAllVCaps() []string
    - func (w *Wurfl) GetAttr(attr int) (int, error)
    - func (w *Wurfl) GetEngineTarget() string
    - func (w *Wurfl) GetHeaderQuality(r *http.Request) (HeaderQuality, error)
    - func (w *Wurfl) GetInfo() string
    - func (w *Wurfl) GetLastLoadTime() string
    - func (w *Wurfl) GetLastUpdated() string
    - func (w *Wurfl) GetUpdaterUserAgent() string
    - func (w *Wurfl) GetUserAgentPriority() string
    - func (w *Wurfl) GoStringToCStringUsingMap(capname string) *C.char
    - func (w *Wurfl) HasCapability(cap string) bool
    - func (w *Wurfl) HasVirtualCapability(vcap string) bool
    - func (w *Wurfl) IsUserAgentFrozen(ua string) bool

- func (w *Wurfl) LookupDeviceID(DeviceID string) (*Device, error)
- func (w *Wurfl) LookupDeviceIDWithImportantHeaderMap(DeviceID string, IHMap map[string]string) (*Device, error)
- func (w *Wurfl) LookupDeviceIDWithRequest(DeviceID string, r *http.Request) (*Device, error)
- func (w *Wurfl) LookupRequest(r *http.Request) (*Device, error)
- func (w *Wurfl) LookupUserAgent(ua string) (*Device, error)
- func (w *Wurfl) LookupWithImportantHeaderMap(IHMap map[string]string) (*Device, error)
- func (w *Wurfl) SetAttr(attr int, value int) error
- func (w *Wurfl) SetLogPath(LogFile string) error
- func (w *Wurfl) SetUpdaterDataFrequency(Frequency int) error
- func (w *Wurfl) SetUpdaterDataURL(DataURL string) error
- func (w *Wurfl) SetUpdaterDataURLTimeout(ConnectionTimeout int, DataTransferTimeout int) error
- func (w *Wurfl) SetUpdaterLogPath(LogFile string) error
- func (w *Wurfl) SetUpdaterUserAgent(userAgent string) error
- func (w *Wurfl) SetUserAgentPriority(prio int)
- func (w *Wurfl) UpdaterRunonce() error
- func (w *Wurfl) UpdaterStart() error
- func (w *Wurfl) UpdaterStop() error

- type WurflHandler

Package files

[interface.go] [wurfl.go]

Constants

```
const (
    WurflEngineTargetHighAccuray             = C.WURFL_ENGINE_TARGET_HIGH_ACCURACY
    WurflEngineTargetHighPerformance         = C.WURFL_ENGINE_TARGET_HIGH_PERFORMANCE
    WurflEngineTargetDefault                 = C.WURFL_ENGINE_TARGET_DEFAULT
    WurflEngineTargetFastDesktopBrowserMatch =
C.WURFL_ENGINE_TARGET_FAST_DESKTOP_BROWSER_MATCH
)
```

Engine Target possible values DEPRECATED as of 1.9.5.0

```
const (
    WurflUserAgentPriorityOverrideSideloadedBrowserUserAgent =
C.WURFL_USERAGENT_PRIORITY_OVERRIDE_SIDELOADED_BROWSER_USERAGENT
    WurflUserAgentPriorityUsePlainUserAgent                  =
C.WURFL_USERAGENT_PRIORITY_USE_PLAIN_USERAGENT
)
```

UserAgent priority possible values DEPRECATED as of 1.9.5.0

```
const (
    // WurflAttrExtraHeadersExperimental is deprecated since 1.12.5.0 and should not
be used.
    WurflAttrExtraHeadersExperimental = C.WURFL_ATTR_EXTRA_HEADERS_EXPERIMENTAL
    // WurflAttrCapabilityFallbackCache attribute controls the capability fallback
cache (needs libwurfl version 1.12.9.3 or above)
    WurflAttrCapabilityFallbackCache = C.WURFL_ATTR_CAPABILITY_FALLBACK_CACHE
)
```

```
const (
    // WurflAttrCapabilityFallbackCacheDefault is the default setting for capability
fallback cache
    WurflAttrCapabilityFallbackCacheDefault =
C.WURFL_ATTR_CAPABILITY_FALLBACK_CACHE_DEFAULT
    // WurflAttrCapabilityFallbackCacheDisabled disables the capability fallback
cache
    WurflAttrCapabilityFallbackCacheDisabled =
C.WURFL_ATTR_CAPABILITY_FALLBACK_CACHE_DISABLED
    // WurflAttrCapabilityFallbackCacheLimited sets the capability fallback cache to
limited mode
    WurflAttrCapabilityFallbackCacheLimited =
C.WURFL_ATTR_CAPABILITY_FALLBACK_CACHE_LIMITED
)
```

To control capability fallback cache, needs libwurfl version 1.12.9.3 or above

```
const (
    WurflCacheProviderDefault = -1
    WurflCacheProviderNone    = C.WURFL_CACHE_PROVIDER_NONE
    WurflCacheProviderLru     = C.WURFL_CACHE_PROVIDER_LRU
    // Deprecated: use WurflCacheProviderLru instead
    WurflCacheProviderDoubleLru = C.WURFL_CACHE_PROVIDER_DOUBLE_LRU
)
```

Cache Provider possible values

```
const (
    WurflMatchTypeExact           = C.WURFL_MATCH_TYPE_EXACT
    WurflMatchTypeConclusive      = C.WURFL_MATCH_TYPE_CONCLUSIVE
    WurflMatchTypeRecovery        = C.WURFL_MATCH_TYPE_RECOVERY
    WurflMatchTypeCatchall        = C.WURFL_MATCH_TYPE_CATCHALL
    WurflMatchTypeHighPerformance = C.WURFL_MATCH_TYPE_HIGHPERFORMANCE
    WurflMatchTypeNone            = C.WURFL_MATCH_TYPE_NONE
    WurflMatchTypeCached          = C.WURFL_MATCH_TYPE_CACHED
)
```

Match type

```
const (
    WurflEnumStaticCapabilities    = C.WURFL_ENUM_STATIC_CAPABILITIES
    WurflEnumVirtualCapabilities   = C.WURFL_ENUM_VIRTUAL_CAPABILITIES
    WurflEnumMandatoryCapabilities = C.WURFL_ENUM_MANDATORY_CAPABILITIES
    WurflEnumWurflID               = C.WURFL_ENUM_WURFLID
)
```

Wurfl enumerator type

```
const (
    WurflUpdaterFrequencyDaily  = C.WURFL_UPDATER_FREQ_DAILY
    WurflUpdaterFrequencyWeekly = C.WURFL_UPDATER_FREQ_WEEKLY
)
```

Wurfl updater frequency

Variables

```
var Version = "1.26"
```

Version is the current version of this package.

func APIVersion

```
func APIVersion() string
```

APIVersion returns version of internal InFuze API without an initialized engine

func Download

```
func Download(url string, folder string) error
```

Download downloads the WURFL data file from the specified URL and saves it to the specified folder. If the download is successful, it returns nil. Otherwise, it returns an error.

type Device

```
type Device struct {
    Device C.wurfl_device_handle
    Wurfl  C.wurfl_handle
    // contains filtered or unexported fields
}
```

Device represent internal matched device handle

func (*Device) Destroy

```
func (d *Device) Destroy()
```

Destroy device handle, should be called when when device attributes are not needed anymore

func (*Device) GetCapabilities

```
func (d *Device) GetCapabilities(caps []string) map[string]string
```

GetCapabilities Get a list of Static Capabilities Deprecated: GetCapabilities is deprecated. Use GetStaticCaps instead.

func (*Device) GetCapability

```
func (d *Device) GetCapability(cap string) string
```

GetCapability Get a single Capability Deprecated: GetCapability is deprecated. Use GetStaticCap instead.

func (*Device) GetCapabilityAsInt

```
func (d *Device) GetCapabilityAsInt(cap string) (int, error)
```

GetCapabilityAsInt gets a single capability value that has a int type It returns an error if the requested capability is not a numeric one (ie: brand_name)

func (*Device) GetDeviceID

```
func (d *Device) GetDeviceID() (string, error)
```

GetDeviceID Get wurfl_id string from device handle

func (*Device) GetMatchType

```
func (d *Device) GetMatchType() int
```

GetMatchType Get type of Match occurred in lookup

func (*Device) GetNormalizedUserAgent

```
func (d *Device) GetNormalizedUserAgent() (string, error)
```

GetNormalizedUserAgent Get the Normalized (processed by wurfl api) userAgent (Only for internal use/tooling)

func (*Device) GetOriginalUserAgent

```
func (d *Device) GetOriginalUserAgent() (string, error)
```

GetOriginalUserAgent Get the original userAgent of matched device (the one passed to lookup)

func (*Device) GetParentID

```
func (d *Device) GetParentID() string
```

GetParentID - Retrieve the parent device id of this device.

func (*Device) GetRootID

```
func (d *Device) GetRootID() string
```

GetRootID - Retrieve the root device id of this device.

func (*Device) GetStaticCap

```
func (d *Device) GetStaticCap(cap string) (string, error)
```

GetStaticCap Get a single cap using new C.wurfl_device_get_static_cap() that returns error

func (*Device) GetStaticCaps

```
func (d *Device) GetStaticCaps(caps []string) (map[string]string, error)
```

GetStaticCaps Get a list of Static Capabilities

func (*Device) GetUserAgent

```
func (d *Device) GetUserAgent() (string, error)
```

GetUserAgent Get default UserAgent of matched device (might be different from UA passed to lookup)

func (*Device) GetVirtualCap

```
func (d *Device) GetVirtualCap(vcap string) (string, error)
```

GetVirtualCap Get Virtual Cap with new C.wurfl_device_get_virtual_cap() that manages errors

func (*Device) GetVirtualCapabilities

```
func (d *Device) GetVirtualCapabilities(caps []string) map[string]string
```

GetVirtualCapabilities Get a list of Virtual Capabilities Deprecated: GetVirtualCapabilities is deprecated. Use GetVirtualCaps instead.

func (*Device) GetVirtualCapability

```
func (d *Device) GetVirtualCapability(vcap string) string
```

GetVirtualCapability Get Virtual Capability Deprecated: GetVirtualCapability is deprecated. Use GetVirtualCap instead.

func (*Device) GetVirtualCapabilityAsInt

```
func (d *Device) GetVirtualCapabilityAsInt(vcap string) (int, error)
```

GetVirtualCapabilityAsInt gets a single virtual capability value that has a int type It returns an error if the requested virtual capability is not a numeric one (ie: brand_name)

func (*Device) GetVirtualCaps

```
func (d *Device) GetVirtualCaps(caps []string) (map[string]string, error)
```

GetVirtualCaps Get a list of Virtual Capabilities

func (*Device) IsRoot

```
func (d *Device) IsRoot() bool
```

IsRoot - true if device is device root

func (*Device) ORTB2GetDevicetype

```
func (d *Device) ORTB2GetDevicetype() (int, error)
```

ORTB2GetDevicetype - Returns the OpenRTB V2 (ORTB2) devicetype value based on WURFL capabilities.

type DeviceHandler

```
type DeviceHandler interface {
    GetMatchType() int
    GetVirtualCapabilities(caps []string) map[string]string
    GetVirtualCaps(caps []string) (map[string]string, error)
    GetVirtualCapability(vcap string) string
    GetVirtualCap(vcap string) (string, error)
    GetVirtualCapabilityAsInt(vcsp string) (int, error)
    GetCapabilities(caps []string) map[string]string
    GetStaticCaps(caps []string) (map[string]string, error)
    GetCapability(cap string) string
    GetStaticCap(cap string) (string, error)
    GetCapabilityAsInt(cap string) (int, error)
    IsRoot() bool
    GetRootID() string
    GetParentID() string
    GetDeviceID() (string, error)
    GetNormalizedUserAgent() (string, error)
    GetOriginalUserAgent() (string, error)
    GetUserAgent() (string, error)
    Destroy()
}
```

DeviceHandler defines API methods for the Wurfl Device handle

type HeaderQuality

```
type HeaderQuality int
```

HeaderQuality represents the header quality value

```
const (
    // HeaderQualityNone no User Agent Client Hints are present.
    HeaderQualityNone HeaderQuality = C.WURFL_ENUM_UACH_NONE
    // HeaderQualityBasic only some of the headers needed for a successful WURFL
detection are present.
    HeaderQualityBasic HeaderQuality = C.WURFL_ENUM_UACH_BASIC
    // HeaderQualityFull all the headers needed for a successful WURFL detection are
present.
    HeaderQualityFull HeaderQuality = C.WURFL_ENUM_UACH_FULL
)
```

func (HeaderQuality) String

```
func (hq HeaderQuality) String() string
```

type Updater

```
type Updater interface {
    SetUpdaterDataURL(DataURL string) error
    SetUpdaterDataFrequency(Frequency int) error
    SetUpdaterDataURLTimeout(ConnectionTimeout int, DataTransferTimeout int) error
    SetUpdaterLogPath(LogFile string) error
    UpdaterRunonce() error
    UpdaterStart() error
    UpdaterStop() error
    SetUpdaterUserAgent(userAgent string) error
    GetUpdaterUserAgent() string
}
```

Updater defines API methods for the Updater

type Wurfl

```
type Wurfl struct {
    Wurfl               C.wurfl_handle
    ImportantHeaderNames []string
    // contains filtered or unexported fields
}
```

Wurfl represents internal wurfl infuze handle

func Create

```
func Create(Wurflxml string, Patches []string, CapFilter []string, EngineTarget int,
CacheProvider int, CacheExtraConfig string) (*Wurfl, error)
```

Create the wurfl engine.Parameters : Wurflxml : path to the wurfl.xml/zip file Patches : slice of paths of patches files to load CapFilter : list of capabilities used; allow to init engine without loading all 500+ caps DEPRECATED: EngineTarget : As of 1.9.5.0 has no effect anymore CacheProvider : WurflCacheProviderLru or WurflCacheProviderDoubleLru CacheExtraConfig : size of single or double lru caches in the form "100000" or "100000,30000"

func (*Wurfl) Destroy

```
func (w *Wurfl) Destroy()
```

Destroy the wurfl engine

func (*Wurfl) GetAPIVersion

```
func (w *Wurfl) GetAPIVersion() string
```

GetAPIVersion returns version of internal InFuze API

func (*Wurfl) GetAllCaps

```
func (w *Wurfl) GetAllCaps() []string
```

GetAllCaps return all capabilities names

func (*Wurfl) GetAllDeviceIds

```
func (w *Wurfl) GetAllDeviceIds() []string
```

GetAllDeviceIds returns a slice containing all WURFL device IDs

func (*Wurfl) GetAllVCaps

```
func (w *Wurfl) GetAllVCaps() []string
```

GetAllVCaps return all virtual capabilities names

func (*Wurfl) GetAttr

```
func (w *Wurfl) GetAttr(attr int) (int, error)
```

GetAttr : get we engine attributes

func (*Wurfl) GetEngineTarget

```
func (w *Wurfl) GetEngineTarget() string
```

GetEngineTarget - Returns a string representing the currently set WURFL Engine Target. Possible values are "HIGH_ACCURACY", "HIGH_PERFORMANCE" or "INVALID". DEPRECATED: will always return default value

func (*Wurfl) GetHeaderQuality

```
func (w *Wurfl) GetHeaderQuality(r *http.Request) (HeaderQuality, error)
```

GetHeaderQuality return the header quality value of HTTP request headers

func (*Wurfl) GetInfo

```
func (w *Wurfl) GetInfo() string
```

GetInfo - get wurfl.xml info

func (*Wurfl) GetLastLoadTime

```
func (w *Wurfl) GetLastLoadTime() string
```

GetLastLoadTime - get last wurfl.xml load time

func (*Wurfl) GetLastUpdated

```
func (w *Wurfl) GetLastUpdated() string
```

GetLastUpdated - get last wurfl.xml update time

func (*Wurfl) GetUpdaterUserAgent

```
func (w *Wurfl) GetUpdaterUserAgent() string
```

GetUpdaterUserAgent - gets the User Agent for the updater

func (*Wurfl) GetUserAgentPriority

```
func (w *Wurfl) GetUserAgentPriority() string
```

GetUserAgentPriority - Tells if WURFL is using the plain user agent or the sideloaded browser user agent for device detection DEPRECATED: will always return default value

func (*Wurfl) GoStringToCStringUsingMap

```
func (w *Wurfl) GoStringToCStringUsingMap(capname string) *C.char
```

GoStringToCStringUsingMap returns a C string pointer for the given capability name using a cached map.

func (*Wurfl) HasCapability

```
func (w *Wurfl) HasCapability(cap string) bool
```

HasCapability - HasVirtualCapability return true is the vcap exists

func (*Wurfl) HasVirtualCapability

```
func (w *Wurfl) HasVirtualCapability(vcap string) bool
```

HasVirtualCapability - HasVirtualCapability return true if the vcap exists

func (*Wurfl) IsUserAgentFrozen

```
func (w *Wurfl) IsUserAgentFrozen(ua string) bool
```

IsUserAgentFrozen : returns true if an useragent is frozen

func (*Wurfl) LookupDeviceID

```
func (w *Wurfl) LookupDeviceID(DeviceID string) (*Device, error)
```

LookupDeviceID : lookup by wurfl_ID and return Device handle

func (*Wurfl) LookupDeviceIDWithImportantHeaderMap

```
func (w *Wurfl) LookupDeviceIDWithImportantHeaderMap(DeviceID string, IHMap
map[string]string) (*Device, error)
```

LookupDeviceIDWithImportantHeaderMap : Lookup deviceID using header values found in IHMap. IHMap must be filled with Wurfl.ImportantHeaderNames values

func (*Wurfl) LookupDeviceIDWithRequest

```
func (w *Wurfl) LookupDeviceIDWithRequest(DeviceID string, r *http.Request) (*Device,
error)
```

LookupDeviceIDWithRequest : lookup by wurfl_ID and request headers and return Device handle

func (*Wurfl) LookupRequest

```
func (w *Wurfl) LookupRequest(r *http.Request) (*Device, error)
```

LookupRequest : Lookup Request and return Device handle

func (*Wurfl) LookupUserAgent

```
func (w *Wurfl) LookupUserAgent(ua string) (*Device, error)
```

LookupUserAgent : lookup up useragent and return Device handle

func (*Wurfl) LookupWithImportantHeaderMap

```
func (w *Wurfl) LookupWithImportantHeaderMap(IHMap map[string]string) (*Device,
error)
```

LookupWithImportantHeaderMap : Lookup using header values found in IHMap. IHMap must be filled with Wurfl.ImportantHeaderNames values

func (*Wurfl) SetAttr

```
func (w *Wurfl) SetAttr(attr int, value int) error
```

SetAttr : we engine attributes

func (*Wurfl) SetLogPath

```
func (w *Wurfl) SetLogPath(LogFile string) error
```

SetLogPath - set path of main libwurfl log file (updater has a separate log file)

func (*Wurfl) SetUpdaterDataFrequency

```
func (w *Wurfl) SetUpdaterDataFrequency(Frequency int) error
```

SetUpdaterDataFrequency - set interval of update checks

func (*Wurfl) SetUpdaterDataURL

```
func (w *Wurfl) SetUpdaterDataURL(DataURL string) error
```

SetUpdaterDataURL - set your HTTPS updater URL, as shown on your **Snapshot generator** page in the Customer Vault

func (*Wurfl) SetUpdaterDataURLTimeout

```
func (w *Wurfl) SetUpdaterDataURLTimeout(ConnectionTimeout int, DataTransferTimeout
int) error
```

SetUpdaterDataURLTimeout - set connection and data transfer timeouts (in millisecs) for updater http call. 0 for no timeout, -1 for defaults

func (*Wurfl) SetUpdaterLogPath

```
func (w *Wurfl) SetUpdaterLogPath(LogFile string) error
```

SetUpdaterLogPath - set path of updater log file

func (*Wurfl) SetUpdaterUserAgent

```
func (w *Wurfl) SetUpdaterUserAgent(userAgent string) error
```

SetUpdaterUserAgent - set User Agent for the updater http call

func (*Wurfl) SetUserAgentPriority

```
func (w *Wurfl) SetUserAgentPriority(prio int)
```

SetUserAgentPriority - Sets which UA wurfl is using (plain or sideloaded) DEPRECATED. Since 1.9.5.0 has no effect anymore

func (*Wurfl) UpdaterRunonce

```
func (w *Wurfl) UpdaterRunonce() error
```

UpdaterRunonce - start updater process once and wait for termination

func (*Wurfl) UpdaterStart

```
func (w *Wurfl) UpdaterStart() error
```

UpdaterStart - start the updater thread

func (*Wurfl) UpdaterStop

```
func (w *Wurfl) UpdaterStop() error
```

UpdaterStop - stop the updater thread

type WurflHandler

```
type WurflHandler interface {
    GetAPIVersion() string
    Download(url string, folder string) error
    SetLogPath(LogFile string) error
    IsUserAgentFrozen(ua string) bool
    LookupDeviceIDWithImportantHeaderMap(DeviceID string, IHMap map[string]string)
(DeviceHandler, error)
    LookupWithImportantHeaderMap(IHMap map[string]string) (DeviceHandler, error)
    LookupDeviceIDWithRequest(DeviceID string, r *http.Request) (DeviceHandler,
error)
    LookupRequest(r *http.Request) (DeviceHandler, error)
    LookupUserAgent(ua string) (DeviceHandler, error)
    GetAllDeviceIds() []string
    LookupDeviceID(DeviceID string) (DeviceHandler, error)
    Destroy()
    GetAllVCaps() []string
    GetAllCaps() []string
    GetInfo() string
    GetHeaderQuality(r *http.Request) (HeaderQuality, error)
    GetLastLoadTime() string
    HasCapability(cap string) bool
    HasVirtualCapability(vcap string) bool
    SetAttr(attr int, value int) error
    GetAttr(attr int) (int, error)
    GetLastUpdated() string
}
```

WurflHandler defines API methods for the Wurfl Infuze handle

© 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.