scientiamobile

"The first step in a great mobile experience"

INFUZE GOLANG 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

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 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 on Linux and macOS platforms for golang 1.7 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 and macOS

InFuze for golang is available as a binary package. To install it, untar the contents in your GOPATH folder. For example:

```
# cd $GOPATH
# tar xvzf golang-src-wurfl-1.12.10.0.tar.gz
```

Usage

Here is an example to get started:

```
package main

import (
    "fmt"
    "html"
    "net/http"
    "scientiamobile/wurfl"
)



func main() {

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

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

    if err != nil {
        fmt.Println(err)
    }

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

    device, err = wengine.LookupUserAgent(ua)

    deviceid, err := device.GetDeviceID()

    fmt.Println(deviceid)

    fmt.Println(device.GetCapability("device_os"))
    fmt.Println(device.GetVirtualCapability("is_android"))

    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 Mob
ile 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)
```

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 uptodate 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, with xxxxx replaced with your personal access token, located in your license account page):

```
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.

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.

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

Index

- Constants
- func APIVersion() string
- 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) GetDeviceID() (string, error)
    - func (d *Device) GetMatchType() int
    - func (d *Device) GetNormalizedUserAgent() (string, error)
    - func (d *Device) GetOriginalUserAgent() (string, error)
    - func (d *Device) GetRootID() string
    - func (d *Device) GetUserAgent() (string, error)
    - func (d *Device) GetVirtualCapabilities(caps []string) map[string]string
    - func (d *Device) GetVirtualCapability(vcap string) string
    - func (d *Device) IsRoot() bool
- 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) GetUserAgentPriority() string
    - 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) 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) SetUserAgentPriority(prio int)
- func (w *Wurfl) UpdaterRunonce() error
- func (w *Wurfl) UpdaterStart() error
- func (w *Wurfl) UpdaterStop() error

- type WurflHandler

Package files

interface.go](/src/scientiamobile/wurfl/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_SID
ELOADED_BROWSER_USERAGENT
    WurflUserAgentPriorityUsePlainUserAgent                  = C.WURFL_USERAGENT_PRIORITY_USE_PLAIN_USERAGEN
T
)
```

UserAgent priority possible values DEPRECATED as of 1.9.5.0

```
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

```
const (
    // Deprecated: since 1.12.5.0
    WurflAttrExtraHeadersExperimental = C.WURFL_ATTR_EXTRA_HEADERS_EXPERIMENTAL
)
```

func APIVersion

```
func APIVersion() string
```

APIVersion returns version of internal InFuze API without an initialized engine

type Device

```
type Device struct {
    Device C.wurfl_device_handle
    Wurfl  C.wurfl_handle
}
```

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

func (*Device) GetCapability

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

GetCapability Get a single Capability

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

func (*Device) GetOriginalUserAgent

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

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

func (*Device) GetRootID

func (d *Device) GetRootID() string

GetRootID - Retrieve the root device id of this device.

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) GetVirtualCapabilities

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

GetVirtualCapabilities Get a list of Virtual Capabilities

func (*Device) GetVirtualCapability

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

GetVirtualCapability Get Virtual Capability

func (*Device) IsRoot

func (d *Device) IsRoot() bool

IsRoot - true if device is device root

type DeviceHandler

```
type DeviceHandler interface {
    GetMatchType() int
    GetVirtualCapabilities(caps []string) map[string]string
    GetVirtualCapability(vcap string) string
    GetCapabilities(caps []string) map[string]string
    GetCapability(cap string) string
    IsRoot() bool
    GetRootID() 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
}
```

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, CacheExtraC
onfig 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)

LookupRequest : Lookup Request and return Device handle

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) 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) 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 is 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, erro
r)

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) 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 scientiamobile vault https updater url

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) 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

© 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.