

WifiControl

© 2018 All rights reserved by Metrological

This document contains information which is proprietary and confidential to Metrological. It is provided with the expressed understanding that the recipient will not divulge its content to other parties or otherwise misappropriate the information contained herein. This information is furnished for guidance; specifications and availability of goods mentioned in it are subject to change without notice. No part of this publication may be reproduced, stored in a database, retrieval system or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording or otherwise, without the written prior permission of Metrological.

History

|  |  |  |  |
| --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Description** |
| 0.1 | 24-04-2018 | P. Wielders | Initial version |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

1. Introduction 4

1.1 Scope 4

1.2 Case sensitivity 4

1.3 Acronyms, Abbreviations and Terms 4

1.4 Standards 4

1.5 References 5

1.6 Open Issues 5

1.7 Limitations 5

2. Rationale behind tracing 6

2.1 Definition, logging versus tracing 6

2.2 Tracing in practice 6

2.3 Configuration 6

2.4 Flow of trace information. 6

3. TraceControl Plugin 8

3.1 Configuration 8

3.2 Application Programming Interface (API) 8

3.2.1 General information 8

3.2.2 State changes 8

3.3 Events 8

3.4 JSON definitions 9

3.4.1 General information (tracing\_info) 9

# Introduction

## Scope

This document describes the Plugin WifiControl API interface. This plugin can be configured to control the wireless network interface on the target device. The plugin is depending on the WPASupplicant application on the target device. This plugin is also depending on the NetworkControl plugin for IP configurations. NetworkControl and WifiControl are part of the WPEFramework infrastructure, for details on the WPEFramework API, refer to: [WPEF]

## Case sensitivity

All identifiers on the interface described here are case-sensitive. E.g. an id known in the plugin as 'C0FFEE' is not the same as 'c0ffee'.

All keywords, entities, properties, relations and actions should be treated as case-sensitive.

## Acronyms, Abbreviations and Terms

The next list provides an overview of acronyms and abbreviations used in this document and their definitions.

|  |  |
| --- | --- |
| **Acronym** | **Definitions** |
| API | Application Programming Interface |
| JSON | JavaScript Object Notation |
| UTC | Coordinated Universal Time |

Below terms are listed with their definitions, as used in this document.

|  |  |
| --- | --- |
| **Term** | **Definitions** |
| Callsign | The callsign is the name given to an instance of a plugin. One plugin can be instantiated multiple times, but each instance the instance name, callsign, must be unique. |
| Proxy | An object in one process space representing the “real” object in another process space. The Proxy takes care of marshalling the parameters. |
| Stub | An object in the process space that contains the actual object. The stub takes care of un-marshalling the request from the Proxy and executes the call, on behave of the Proxy object, on the real object |

## Standards

Date time formats between the systems shall be in UTC time and W3C (ISO 8601 profile) formatting [ISO 8601], e.g.: 2004-11-05T13:15:30Z. This way time discontinuities can be avoided due to daylight savings. Note that all interfacing systems must decode/encode the date time to the correct local time.

Languages used in the WPEFramework will be conform [ISO 639-1] using two letter language codes. If WPEFramework encounters a language code it does not recognize, it will use ‘xx’ instead. For a list of available two letter ISO language codes, please visit:  
<http://www.loc.gov/standards/iso639-2/php/code_list.php>

## References

This section lists the references made in this document:

|  |  |
| --- | --- |
| [WPEF] | WPEFramework API Reference  <https://github.com/WebPlatformForEmbedded/WPEFramework> |
| [HTTP] | Hypertext Transfer Protocol  <http://www.w3.org/Protocols> |
| [ISO 8601] | Date and time format  http://www.iso.org/iso/date\_and\_time\_format |
| [ISO-3166] | Country code specification  <http://www.iso.org/iso/country_codes.htm> |
| [ISO-639-1] | Language code specification (Alpha-2 code)  <http://www.loc.gov/standards/iso639-2/php/code_list.php> |
| [JSON] | JavaScript Object Notation  <http://www.json.org> |
| [URLENC] | URL Encoding  <http://www.w3schools.com/tags/ref_urlencode.asp> |

## Open Issues

This is a list of open issues that needs to be resolved:

* This document is still a work in progress.

## Limitations

The information described in this document is preliminary and subject to change in the future.

Legend:

****

**Be aware of:** implementation choice is needed or side-effect needs to be handled.



**Implementation advice:** Guide line for implementation mostly related to performance.

# Abstraction model

## Process/Component overview

WPEFramework

WPA Supplicant

Wifi

Control

HTML / JavaScript

[REST]

[Domain]

This document describes the RESTFull interface, indicated by [REST] in the picture above. The Wifi Control plugin described in the document, is the implementation that translates/converts the RESTFull calls towards the WPASupplicant.

The setup consists of 3 processes.

1. The WPASupplicant
2. WifiControl, together with the WPEFramework.
3. The HTML/Javascript running in a browser space.

Communication between WPASupplicant and WifiControl is realized by one domain socket connection. The communication from HTML /Javascript with the WifiControl can be multiple and is based on http restfull calls over TCP/IP.

## Model

**Network [BSSID]**

SSID

Signal

Frequency

Pairing

Key

Throughput

**Config [SSID]**

Open,WEP,WPA(2), Enterprise

Mode (AP/Link)

Depending on key/pair:

Username/Password/Hash

The plugin will scan for Network objects. Each network object represents a radio that was found during the scan. Networks are volatile and not cachec/stored on disk. A network is identified by the BSSID (Binairy SSID).

The user of the plugin can create Config objects. A config object is identified by the SSID. The config object contains all configuration information required to connect to a network or to setup a network. Config objects can be persisted on disk. These persisted config objects will be loaded during the startup of the plugin.

During startup the Network with the strongest Signal that has an associated Config [SSID] will be automatically connected.

It is also possible to start the Wifi as an AccessPoint. This way other devices can connect to this Wifi.

# WifiControl Plugin

## Configuration

|  |  |
| --- | --- |
| callsign | [string] the instance name for the plugin e.g. WLAN0. Default: WifiControl. |
| classname | [string] WifiControl. |
| locator | [string] libWifiControl.so |
| autostart | [bool] should the WifiControl plugin be instantiated at the moment the WPEFramework starts up. |
| configuration | [JSON] JSON object specifying the exact configuration for this plugin. See the next paragraph for details. |

Configuration of the WifiControl Plugin:

|  |  |
| --- | --- |
| connector | [string] the domain socket for WPASupplicant communication. Default: /var/run/wpa\_supplicant. |
| interface | [string] interface to control, using this plugin. Default: wlan0 |
| application | [string] full path to the wpa\_supplicant application. Default: /usr/sbin/wpa\_supplicant. |

## Application Programming Interface (API)

### General information

Using this method, actual trace status information can be retrieved from the WPEFramework.

|  |  |
| --- | --- |
| Request: | GET /Service/WifiControl |
| Success: | HTTP/1.1 200 OK  { status\_info } |

|  |  |
| --- | --- |
| Request: | GET /Service/WifiControl/Networks |
| Success: | HTTP/1.1 200 OK  { “networks”: [ network\_info ] } |

|  |  |
| --- | --- |
| Request: | GET /Service/WifiControl/Configs |
| Success: | HTTP/1.1 200 OK  { “configs”: [ config\_info ] } |

|  |  |
| --- | --- |
| Request: | GET /Service/WifiControl/Config/<ssid> |
| Success: | HTTP/1.1 200 OK  { config\_info } |
| Failure: | HTTP/1.1 204 NO\_CONTENT |

### Config operations

Using this method, a new config is created.

|  |  |
| --- | --- |
| Request: | PUT /Service/WifiControl/Config  { config\_info } |
| Success: | HTTP/1.1 200 OK |
| Failure: | HTTP/1.1 204 NO\_CONTENT |

Using this method, an existing config is updated.

|  |  |
| --- | --- |
| Request: | POST /Service/WifiControl/Config  { config\_info } |
| Success: | HTTP/1.1 200 OK |
| Failure: | HTTP/1.1 204 NO\_CONTENT |

Using this method, an existing config is deleted.

|  |  |
| --- | --- |
| Request: | DELETE /Service/WifiControl/Config/<SSID> |
| Success: | HTTP/1.1 200 OK |

### State operations

To connect (or start an AccessPoint) connect to an SSID using the following command.

|  |  |
| --- | --- |
| Request: | PUT /Service/WifiControl/Connect/<SSID> |
| Success: | HTTP/1.1 200 OK |
| Failure: | HTTP/1.1 204 NO\_CONTENT |

Using the next method the connected SSID is disconnected or an Access Pooint is taken offline..

|  |  |
| --- | --- |
| Request: | DELETE /Service/WifiControl/Connect/<SSID> |
| Success: | HTTP/1.1 200 OK |

## Events

Events are autonomous events, triggered by the internals of the plugin. These events will be broadcasted as JSON to all the connected web socket connections that where opened to this plugin.

## JSON definitions

### General information (\_info)

|  |  |
| --- | --- |
| event | [enum] Type of wifi event.  Values: “Connected”, “Disconnected”, “NetworkUpdate”. |
| networks | [uint32] the number of frames per second the browser is painting on the screen. |
| Hidden | [bool] true in case the browser is not visible (hidden) on screen. |
| suspended | [bool] true, in case the client is in a suspended mode. All system critical resources have been relinquished. |