

WebKitBrowser

© 2017 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 | 30-10-2017 | O.Deveci | Initial version |
| 0.2 | 11-01-2017 | P.Wielders | API Press and Release added. |
|  |  |  |  |
|  |  |  |  |

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 4

1.6 Open Issues 5

1.7 Limitations 5

2. RemoteControl Plugin 6

2.1 Configuration 6

2.2 Key Mapping File 6

2.3 Application Programming Interface (API) 7

2.3.1 General information 7

2.3.2 Device actions 7

2.3.3 Key actions 8

2.3.4 Key mapping actions 8

2.4 Events 9

2.5 JSON definitions 9

2.5.1 General information 9

2.5.2 Codes for key actions 9

# Introduction

## Scope

This document describes the Plugin RemoteControl API interface. This plugin can be configured to be loaded and executed in the WPEFramework and offers user input functionality on the platform. 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 |

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. |

## 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.

# RemoteControl Plugin

## Configuration

|  |  |
| --- | --- |
| callsign | [string] the instance name for the plugin. Default: RemoteControl. |
| classname | [string] RemoteControl. |
| locator | [string] libWPEFrameworkRemoteControl.so |
| autostart | [bool] should the remote controller plugin be instantiated at the moment the WPEFramework is starts up. |
| configuration | [JSON] JSON object specifying the exact configuration for this plugin. See the next paragraph for details. |

Configuration of the RemoteControl:

|  |  |
| --- | --- |
| mapfile | [string] the key mapping file is loaded upon starting the remote controller plugin. |
| repeatstart | [integer] the timeout between key press and release in milliseconds. |
| repeatinterval | [integer] the interval to generate repeat event. |
| passon | [bool] pass through even the key is not in the loaded key mapping. |
| specific | [JSON] JSON array of object defining custom device settings. |

Specific settings:

|  |  |
| --- | --- |
| device | [string] the name of specific device will be instantiated when WPEFramework is starts up. |
| mapfile | [string] the key mapping file of custom device is loaded upon instantiation of specific device. |
| passon | [bool] pass through even the key is not in the loaded key mapping. |
| codemask | [string] the mask of raw codes: some devices generate different codes based on this mask to differentiate order of key presses. It is a hexadecimal string e.g: “0XFFFEFFFF”, 16th bit is flip-flop. This setting is optional. |

## Key Mapping File

The key map file consists of JSON objects of code pairs. API methods in 2.3 manipulates the mirror of these codes in the memory and make them persistent when Save action is triggered.

It is possible to have multiple key map files. The default one is *keymap.json* which is loaded once we activated Remote Control plugin. We can also load customer specific key mappings *(<device>.json*) which might be installed during build or generated on-the-fly.

|  |  |
| --- | --- |
| code | [string] the hexadecimal string of device raw key code e.g: “0x1” |
| key | [integer] the key code of keyboard keys e.g: 103 -> KEY\_UP |
| modifiers | [string] the array of string representation of keyboard modifiers: “shift” (or ”leftshift”), “rightshift”, “alt” (or “leftalt”), “rightalt”, “ctrl” (or “leftctrl”), “rightctrl”. e.g: [“ctrl”, “alt”] |
| \_comment | [string] the comment in the file. It is optional |
| char | [string] the text of key button. It is optional to make readability simpler. |

## Application Programming Interface (API)

### General information

Using this method, the names of all available devices, the metadata of specific device can be retrieved from the plugin.

|  |  |
| --- | --- |
| Request: | GET /Service/RemoteControl |
| Success: | HTTP/1.1 200 List of loaded remote devices  { device\_list } |
| Failure | HTTP/1.1 404 Unknown request path specified[[1]](#footnote-1) |

### Device actions

Using these methods, the metadata of specific device can be retrieved from the plugin or the pairing mode of device (if applicable) can be activated.

|  |  |
| --- | --- |
| Request: | GET /Service/RemoteControl/<device\_name>[[2]](#footnote-2) |
| Success: | HTTP/1.1 200 Specific device is loaded  { device\_info }  HTTP/1.1 204 Default device is loaded |
| Failure | HTTP/1.1 502 Error during loading of device. ErrorCode:<err> |

|  |  |
| --- | --- |
| Request: | PUT /Service/RemoteControl/<device\_name>/PairingMode |
| Success: | HTTP/1.1 200 Pairing mode active:<device\_name> |
| Failure | HTTP/1.1 501 Failed to activate pairing:<device\_name> |

### Key actions

Using these methods, the key related queries (get code details, ingest code) can be triggered.

|  |  |
| --- | --- |
| Request: | GET /Service/RemoteControl/<device\_name>?Code=<key\_code> |
| Success: | HTTP/1.1 200 Get key info of <device\_name>  { code\_object } |
| Failure | HTTP/1.1 404 Key does not exist in <device\_name>  HTTP/1.1 400 No key code in request |

The next method simulates a key press and release in one atomic action. A key press can be issued with a single API call. However, this will never lead to “repeat” keys.

|  |  |
| --- | --- |
| Request: | PUT /Service/RemoteControl/<device\_name>/Send  { code\_object } |
| Success: | HTTP/1.1 202 Soft key is sent to <device\_name> |
| Failure | HTTP/1.1 404 Key does not exist in <device\_name>  HTTP/1.1 400 No key code in request |

The next two methods allow to specify how long the key is pressed. The key pressed and the key released, as triggered by the method above, is split into two dedicated methods here. The system is now capable of generating repeat key events for the duration between key pressed and key released.

|  |  |
| --- | --- |
| Request: | PUT /Service/RemoteControl/<device\_name>/Press  { code\_object } |
| Success: | HTTP/1.1 202 Key pressed event is sent to <device\_name> |
| Failure | HTTP/1.1 404 Key does not exist in <device\_name>  HTTP/1.1 400 No key code in request |

|  |  |
| --- | --- |
| Request: | PUT /Service/RemoteControl/<device\_name>/Release  { code\_object } |
| Success: | HTTP/1.1 202 Key release event is sent to <device\_name> |
| Failure | HTTP/1.1 404 Key does not exist in <device\_name>  HTTP/1.1 400 No key code in request |

### Key mapping actions

Using this method, the key mapping related queries (add, delete, modify, save, load) can be triggered.

|  |  |
| --- | --- |
| Request: | PUT /Service/RemoteControl/<device\_name>  { code\_object } |
| Success: | HTTP/1.1 201 Code is added |
| Failure | HTTP/1.1 400 Bad json data format  HTTP/1.1 403 Code already exists |

|  |  |
| --- | --- |
| Request: | POST /Service/RemoteControl/<device\_name>  { code\_object } |
| Success: | HTTP/1.1 200 Code is modified |
| Failure | HTTP/1.1 400 Bad json data format  HTTP/1.1 403 Code does not exist |

|  |  |
| --- | --- |
| Request: | DELETE /Service/RemoteControl/<device\_name>  { code\_object } |
| Success: | HTTP/1.1 200 Code is deleted |
| Failure | HTTP/1.1 400 Bad json data format  HTTP/1.1 403 Key does not exist in <device\_name> |

|  |  |
| --- | --- |
| Request: | PUT /Service/RemoteControl/<device\_name>/Save |
| Success: | HTTP/1.1 200 OK |
| Failure | HTTP/1.1 500 File is not created |

|  |  |
| --- | --- |
| Request: | PUT /Service/RemoteControl/<device\_name>/Load |
| Success: | HTTP/1.1 200 OK |
| Failure | HTTP/1.1 404 File does not exist |

## Events

Events are not applicable for RemoteControl plugin.

## JSON definitions

### General information

device\_list

|  |  |
| --- | --- |
| devices | [JSON] JSON array which consists of the names of registered devices. |

device\_info

|  |  |
| --- | --- |
| - | [string] the metadata of specific device e.g: {“DeviceA”:”It is based on protocol A”} |

### Codes for key actions

code\_object

|  |  |
| --- | --- |
| code | [string] the hexadecimal string of device raw key code e.g: “0x1” |
| key | [integer] the key code of keyboard keys e.g: 103 -> KEY\_UP |
| modifiers | [string] the array of string representation of keyboard modifiers: “shift” (or ”leftshift”), “rightshift”, “alt” (or “leftalt”), “rightalt”, “ctrl” (or “leftctrl”), “rightctrl”. e.g: [“ctrl”, “alt”] |

1. HTTP 404 error code will be received for all requests of malformed or incorrect URLs. [↑](#footnote-ref-1)
2. <device\_name> points the specific device on which the actions in APIs are performed. [↑](#footnote-ref-2)