

Compositor

API Reference

© 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 | 10-07-2018 | Coen Custers | Initial version. |
| 0.2 | 24-09-2018 | Coen Custers | Update after refactoring. |
| 0.3 | 08-09-2018 | Coen Custers | Added Zorder and Below-methods + reordering methods |
| 0.4 | 09-09-2018 | Coen Custers | Reorder introduction and configuration |
|  |  |  |  |

Table of Contents

[1.1 Case sensitivity 3](#__RefHeading___Toc3451_323876262)

[1.2 Acronyms, Abbreviations and Terms 3](#__RefHeading___Toc3446_2680829492)

[1.3 Standards 4](#__RefHeading___Toc3448_2680829492)

[1.4 References 4](#__RefHeading___Toc3450_2680829492)

[1.5 Open Issues 4](#__RefHeading___Toc3452_2680829492)

[1.6 Limitations 5](#__RefHeading___Toc3454_2680829492)

[2. Plugin 6](#__RefHeading___Toc3453_323876262)

[2.1 Configuration of Controller 6](#__RefHeading___Toc3458_2680829492)

[2.2 Introduction of the Compositor Plugin 7](#__RefHeading___Toc3458_26808294921)

[2.3 Application Programming Interface (API) 7](#__RefHeading___Toc3460_2680829492)

[2.3.1 General information 7](#__RefHeading___Toc3462_2680829492)

[2.3.2 Get Clients in ZOrder 8](#__RefHeading___Toc3510_460643453)

[2.3.3 Get Client Geometry 8](#__RefHeading___Toc3512_460643453)

[2.3.4 Get Screen Resolution 8](#__RefHeading___Toc3514_460643453)

[2.3.5 Reset Screen Resolution 9](#__RefHeading___Toc3516_460643453)

[2.3.6 Switch Client to Top. 9](#__RefHeading___Toc3455_323876262)

[2.3.7 Place ClientA Below another ClientB. 10](#__RefHeading___Toc3518_460643453)

[2.3.8 Switch Input to <client> 10](#__RefHeading___Toc3457_323876262)

[2.3.9 Reset Opacity 11](#__RefHeading___Toc3461_323876262)

[2.3.10 Reset Visibility 11](#__RefHeading___Toc3463_323876262)

[2.3.11 Reset Geometry 11](#__RefHeading___Toc3465_323876262)

[2.3.12 Kill <client> 12](#__RefHeading___Toc3520_460643453)

## Case sensitivity

All identifiers, 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. A single plugin can be instantiated multiple times, but each 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 of the request from the Proxy and executes the call, on behalf 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  <https://www.iso.org/iso-8601-date-and-time-format.html> |
| [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](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:



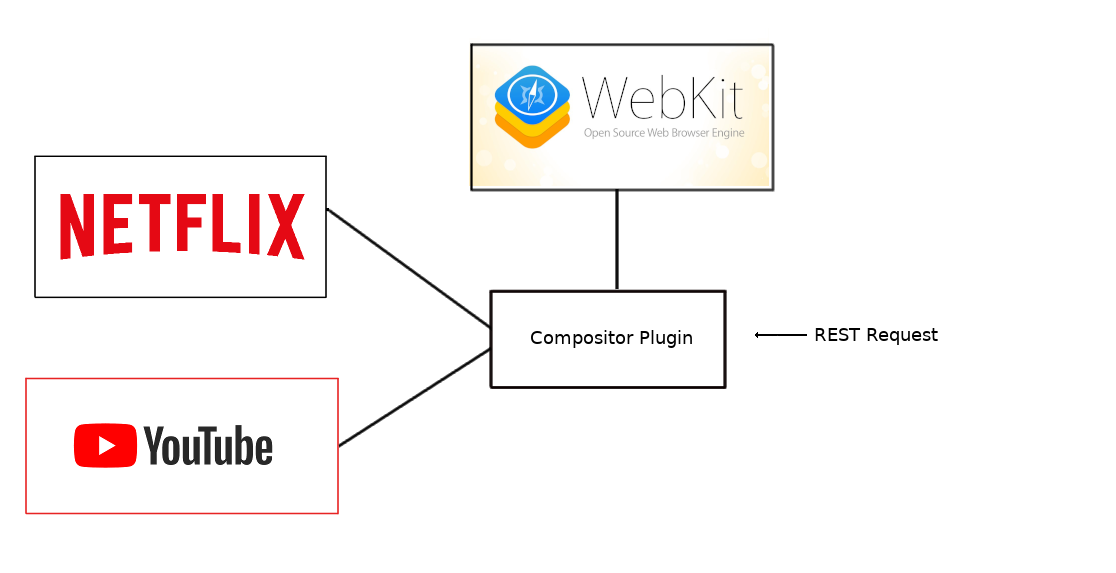
**Be aware of:** implementation choice is needed or side-effect needs to be handled.



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

# Plugin

## Introduction

In the illustration above one is able to see the overall working of the Compositor-plugin. One can distinguish the three clients: *WebKitBrowser*, *Netflix* and *Youtube* and the Compositor-plugin.

Whenever a REST- request is send to the plugin. The plugin will then handle the request for the compositor-server. So, basically the Compositor-plugin tasks can be subdivided into three main tasks:

1) It acts as REST-service that handles all requests it receives from its compositor-clients.

2) It contains a compositor-server that renders and handles the input (keyboard) events of all clients.

3) It will manage its clients and determines which client is active (on top of the screen).

As a compositor-server it will determine which client gets the focus (is on top) and becomes active. The Compositor-plugin is able to change the geometry, the input-source, the opacity, the visibility or to kill each client. Next to setting the overall screen resolution.

## Configuration of Controller

|  |  |
| --- | --- |
| callsign | [string] the instance name for the plugin e.g Compositor. |
| classname | [string] Compositor. |
| locator | [string] libWPECompositor.so |
| autostart | [bool] should the 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 controller:

|  |  |
| --- | --- |
| useragent | [string] the useragent that is used during communication with the web server. |
| url | [string] the URL that is loaded upon starting the browser. |
| injectedbundle | [string] the name of the so that is loaded to extend the HTML5 with customized javascript handlers. |
| cookiestorage | [string] postfixed path to the persistent location on where cookies will be stored. |

## Application Programming Interface (API)

### General information

Using one of both methods will retrieve for you the information of actual running clients.

|  |  |
| --- | --- |
| Request: | GET /Service/Compositor/ or *Service/*Compositor/Clients |
| Success: | HTTP/1.1 200 OK List information of all clients  { information of all clients } |

### Get Clients in ZOrder

Using this method will retrieve for you the list of all clients in z-order (name: *client-list* in this document).

Each client has an z-order-value that determines its position with respect to the screen. The ordering is that the top position is closest to the screen, the next zorder-value first behind the top, and so on.

|  |  |
| --- | --- |
| Request: | GET /Service/Compositor/ZOrder |
| Success: | HTTP/1.1 200 OK List clients from top to bottom  { client\_list } |

### Get Client Geometry

Using this method to get the geometry of the client.

|  |  |
| --- | --- |
| Request: | GET /Service/Compositor/*Geometry*/<client> |
| Success: | HTTP/1.1 200 OK Get screen resolution  { geometry of client } |
| Failure: | HTTP/1.1 400 BAD REQUEST Could not retrieve Geometry, could not find client  HTTP/1.1 400 BAD REQUEST Could not retrieve Geometry, client not specified |

### Get Screen Resolution

Using this method to get the resolution of the screen.

|  |  |
| --- | --- |
| Request: | GET /Service/Compositor/Resolution |
| Success: | HTTP/1.1 200 OK Get screen resolution  { resolution } |

### Reset Screen Resolution

Using this method to reset the screen resolution by passing index.

|  |  |
| --- | --- |
| **Resolution** | **[*index*]** |
| *480i* | *1* |
| *480p* | *2* |
| *720p* | *3* |
| *720p50Hz* | *4* |
| *108024Hz* | *5* |
| *1080i50Hz* | *6* |
| *1080p50Hz* | *7* |
| *1080p60Hz* | *8* |
| *2160p50Hz* | *9* |
| *2160p60Hz* | *10* |

|  |  |
| --- | --- |
| Request: | POST /Service/Compositor/*Resolution/[index*] |
| Success: | HTTP/1.1 200 OK Update screen resolution to [index] |
| Failure: | HTTP/1.1 400 BAD REQUEST invalid parameter for resolution: |

### Switch Client to Top.

Using this method to get a <client> to the top position (of the <client>-list).

Whenever a <client> is on top, it will shown on the screen and it will be in focus.

|  |  |
| --- | --- |
| Request: | POST /Service/Compositor/*<client>*/Top |
| Success: | HTTP/1.1 200 OK Set <client> to the top the client-list |
| Failure: | HTTP/1.1 400 BAD REQUEST Client name is not registered |

### Place ClientA Below another ClientB.

Using this method to place a <client-a> below another <client-b> (of the client-list).

Whenever one wants to reorder the client-list. One can slide a <client-a> just below a <client-b> in the client-list.

|  |  |
| --- | --- |
| Request: | POST /Service/Compositor/*<client-a>*/PutBelow/<client-b> |
| Success: | HTTP/1.1 200 OK Slide <client-a> just behind <client-b> |
| Failure: | HTTP/1.1 400 BAD REQUEST Client name is not registered  HTTP/1.1 400 BAD REQUEST Could not change z-order for Client. Client is not registered  HTTP/1.1 400 BAD REQUEST Could not change z-order for Client. Client relative to which operation should be executed is not registered  HTTP/1.1 400 BAD REQUEST Could not change z-order for Client.  Unspecified problem  HTTP/1.1 400 BAD REQUEST Could not change z-order for Client  <client-b>. Not specified relative to which client |

### Switch Input to <client>

Using this method to re-direct all the input-events to the <client>.

Each client has its own display, surface and keyboard (input-device). Analogous to setting the surface of a <client> to the top, one is able to direct the input-events to that specific <client>.

In the end the client is shown on the screen and all the input-events will act as if they are only send to the <client>.

|  |  |
| --- | --- |
| Request: | POST /Service/Compositor/*<client>*/Input |
| Success: | HTTP/1.1 200 OK Update the Input of the <client>HTTP/1.1 400 BAD REQUEST Client name is not registered |
| Failure: | HTTP/1.1 400 BAD REQUEST Client name is not registered |

### Reset Opacity

Using this method to reset the opacity-value of a <client>:

The opacity of a <client> surface can range from 0 till 255, that will represent an opacity of the surface from 0% till 100%.

Thus, for example, whenever someone wants an 25% opacity, one replaces [index] with the value 64.

|  |  |
| --- | --- |
| Request: | POST /Service/Compositor/*<client>*/Opacity/[index] |
| Success: | HTTP/1.1 200 OK Update to opacity to [value] |
| Failure: | HTTP/1.1 400 BAD REQUEST Client name is not registered |

### Reset Visibility

Using the method to make the surface of the <client> to hidden or visible:

|  |  |
| --- | --- |
| **Visibility** | **[value]** |
| hidden | Hide |
| visible | Show |

|  |  |
| --- | --- |
| Request: | POST /Service/Compositor/*<Client>*/Visible/[value] |
| Success: | HTTP/1.1 200 OK Update the visibility to [Hide] or [Show] |
| Failure: | HTTP/1.1 400 BAD REQUEST Client name is not registered |

### Reset Geometry

Using the method to update the geometry of the client surface:

The geometry is denoted as coordinates of a rectangle, where:

- [X] = x-coordinate of surface

- [Y] = y-coordinate of surface

- [W] = width of surface

- [H] = height of surface

|  |  |
| --- | --- |
| Request: | POST /Service/Compositor/*<Client>*/Geometry/[X]/[Y]/[W]/[H] |
| Success: | HTTP/1.1 200 OK Update the geometry([X],[Y],[W],[H]) |
| Failure: | HTTP/1.1 400 BAD REQUEST Could not set rectangle for Client <client>. Not all information provided |

### Kill <client>

Using this method to kill the <client>:

Whenever a <client> is killed, the execution of the <client> is stopped and all its resources will be released.

|  |  |
| --- | --- |
| Request: | POST /Service/Compositor/*<client>*/Kill |
| Success: | HTTP/1.1 200 OK Kill the <client> |
| Failure: | HTTP/1.1 400 BAD REQUEST Client name is not registered |