

# Presentation API Implementation

Lukas Tetzlaff, Nico Tasche, Simone Egger  
Advisor: Louay Bassbouss

## I. ABSTRACT

The scope of this document is set to report details about the implementation of the World Wide Web Consortium's Presentation API. The specification was published by the Second Screen Presentation Working Group and submitted as a Candidate Recommendation in July 2016 with the current Editor's Draft version being dated to February, 16 2017.

Paraphrased, the API aims to provide a generalized way of accessing and connecting display resources using web technology thus providing means to present content from a given website to a display and gain restricted remote access to their browsing context by messaging. Effectively this system relies on two dedicated roles, the Controller and the Receiver, obtained by the respective User Agents of on the one hand the initiating web page and on the other hand the display, whereas these may also be identical thus allowing a 1-User-Agent situation. This document and the implementation assume the 2-User-Agent case since 1-UA is to be most prominently used in conjunction with a native way of transmitting the rendered presentation for example by encoding to png and sending it to an external display entity.

## II. ARCHITECTURE

To facilitate using the Presentation API in terms of hosting a presentation, connecting to a presentation and using the established connection the interface to the user or rather the application developer is kept simple and straight-forward as to be seen in the VI. Generally speaking there are three distinct entities involved in the aforementioned processes, the first being the Receiving User Agent. In case of a browser vendor's implementation of the API this is most likely to be integrated into the browser natively or as a plugin, whereas in this implementation it's a polyfill housed in a loaded html-document, hereafter referred to as the *\*Receiver\**.

The counterpart is the Controlling Browser Agent which acts upon input from the controlling browsing context. Due to the closely coupled relation in the internal procedures of the specification those two entities have been combined into the *\*Controller\** which can be any regular web page enriched by the same aforementioned PresentationAPI-polyfill and scripts containing the desired controller logic of the application developer.

As soon as the Controller knows about the possibility to present on a remote display (Presentation Availability) it may start a Presentation Connection and instruct the Receiver to create a receiving browsing context, here referred to as the *\*Presentation\**, which is the final third component. This is another document written by the application developer that needn't be much different from the regular controlling page since it can identify if it is loaded as a Presentation and act accordingly.

Since the specification relies heavily on individual vendor-specific mechanisms this implementation also provides a configuration interface for the User Agent level that requires a set of handlers II-B. To prove this concept two distinct approaches were realized, as seen in VI, one relying purely on ajax and long-polling thus offering maximum compatibility and the second one on WebSockets for a more standard bidirectional low-latency communication, abstracted by third-party library SocketIO which by default also includes a long-polling fallback.

As of this implementation the required logic was split up in separate files that need to be included as seen below.

### A. Scripts

In the following table C denotes the Controller, R the Receiver and RC the Receiving Context.

TABLE I: Scripts

Script	C	R	RC	Description
util.js	Y	Y	Y	General utilities to stay vanilla
presentation.js	Y	Y	Y	Polyfill of Presentation API
presentationUserAgent.js	Y	Y		Vendor-specific realization
implementation.*.js	Y	Y		Implementation-specifcs
receiver.js		Y		Hosting once loaded / Backdrop
receivingContext.js			Y	Communication with R-UA
*.js				Client scripts that use the API

Each entity in a Presentation scenario includes the API Polyfill, whereas only Controller and Receiver include the tasks the User Agent shall fulfill globally or in the background to provide relative safety to the receiving context thus preventing it to be conquered by malicious Controllers and displaying

content that's not intend to be presented (think of a game without a fixed set of commands in which a Controller was able to inject code to manipulate player's scores or similar situations).

### B. Configuration

Every User Agent has a set of handlers that are to be configured apart from the - generally speaking - browser vendor specific functionality. These can be assigned by instantiating an `ImplementationConfig`-object. Per default this happens in the `implementation.*.js`-file according to the table above where the asterisk indicates the specific implementation type. This config object is then reflected onto the `PresentationUserAgent`-object on its instantiation which makes subsequent calls to any of those handlers by the predefined algorithms use the configured handler functions. Swapping configurations later on is also supported by passing the global `PresentationUserAgent`-object to the configuration's `configure` method.

## III. ALTERNATIVE APPROACHES

Priorly considered approaches included stricter separation of the respective contexts and User Agents using a top-level context like a tab as the User Agent and several child contexts (iframes) as the browsing contexts. Due to the requirement that custom objects such as `PresentationRequest` or `PresentationConnection` need to be passed between context and User Agent the approach raised several problems, mainly related to serialization of these custom objects to then be transmitted via the `Window.postMessage`-interface and deserialized, keeping these multiple object instances synchronized, rerouting function calls and offering proper garbage collection.

These problems originate from the circumstance that W3C specifications are usually meant to be implemented natively by browser vendors thereby omitting the preceding complications or falling back to proper solutions for this issue that have already been implemented.

## IV. SHORTCOMINGS

Due to those obstacles this implementation resorted to the concept described in II according to which some security aspects of the specification are not met, for instance:

---

```

1
2 6.6.1 Creating a receiving browsing context
3
4 When the user agent is to create a receiving
  browsing context, it must run the
  following steps:
5 ...

```

---

[Read More](#)

In this function points 1 to 10 are not met since the receiving browsing context is not spawned as a new top-level context.

---

```

1
2 This specification adds a new token,
  allow-presentation, to the set of tokens
  allowed in the sandbox attribute of an
  iframe. It adds a corresponding new flag
  to the sandboxing flag set:
3
4 The sandboxed presentation browsing context
  flag
5 This flag disables the Presentation API.

```

---

This kind of functionality can not be reliably enforced using non-native code hosted in one context since the method prohibiting this can simply be overridden by applying common reflection commands or generic functions such as `Object.defineProperty`.

Another flaw of the current implementation is that only textual data transmission was tested and applied even though generally binary data should not pose a significant problem since e.g. WebSocket-communication can easily handle this kind of data.

Being constrained by the above aspects the implementation could also be extended by more generic and anonymous display detection as discussed in 7.1 Personally identifiable information, browser-instance-wide synchronization of existing presentation connections (like caching the `presentationId`) or recognized displays.






## V. TEST COMPLIANCE

To identify problems in the implementation the W3C Testharness was applied, yielding decent results (see ??).

The tests were run using the regular approach the W3C recommends (see Web Platform Tests) with the addition of priorly injecting the necessary scripts (see I) and deferring the load of inline scripts running the actual tests by a nodejs script (see Github).

Comparing this outcome with official implementation results the test compliance is on par with if not higher than CD53 from June 2016 W3C Test results with fails mostly eventuating from the above design decisions (see II and IV) such as the necessity to construct a `TrustedEvent` which is reserved for the browser.

## Progress

Done!					
	Passed	Failed	Timeouts	Errors	Not Run
	80	22	0	0	0
Display:					

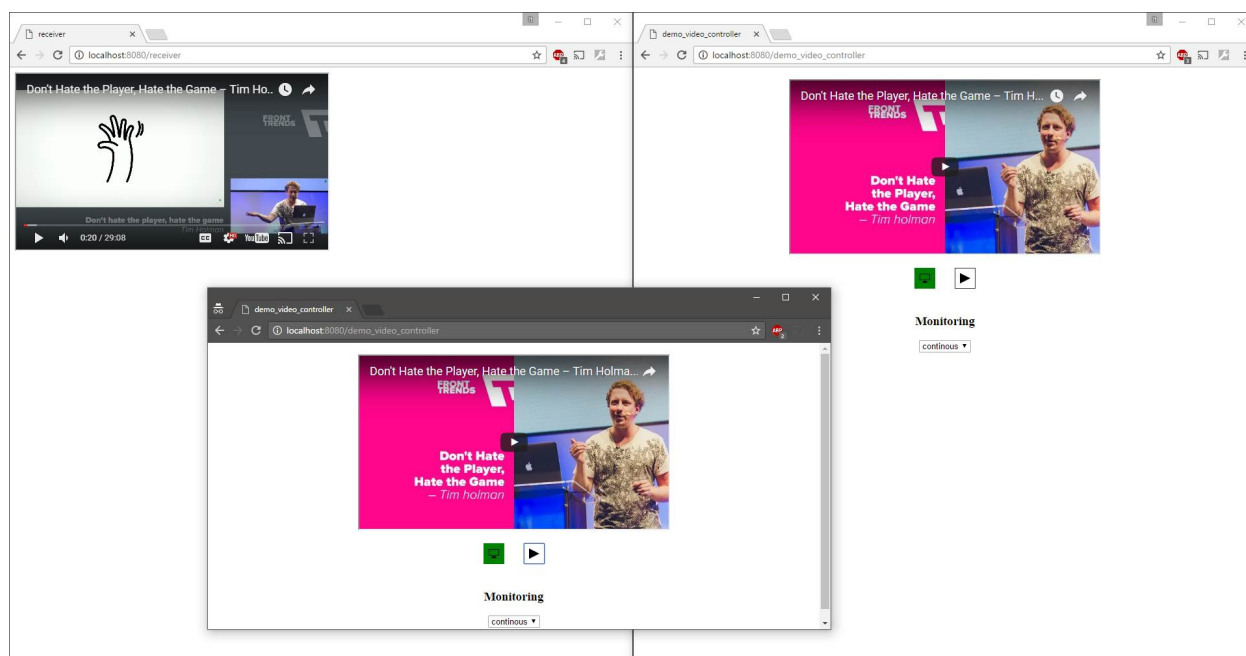
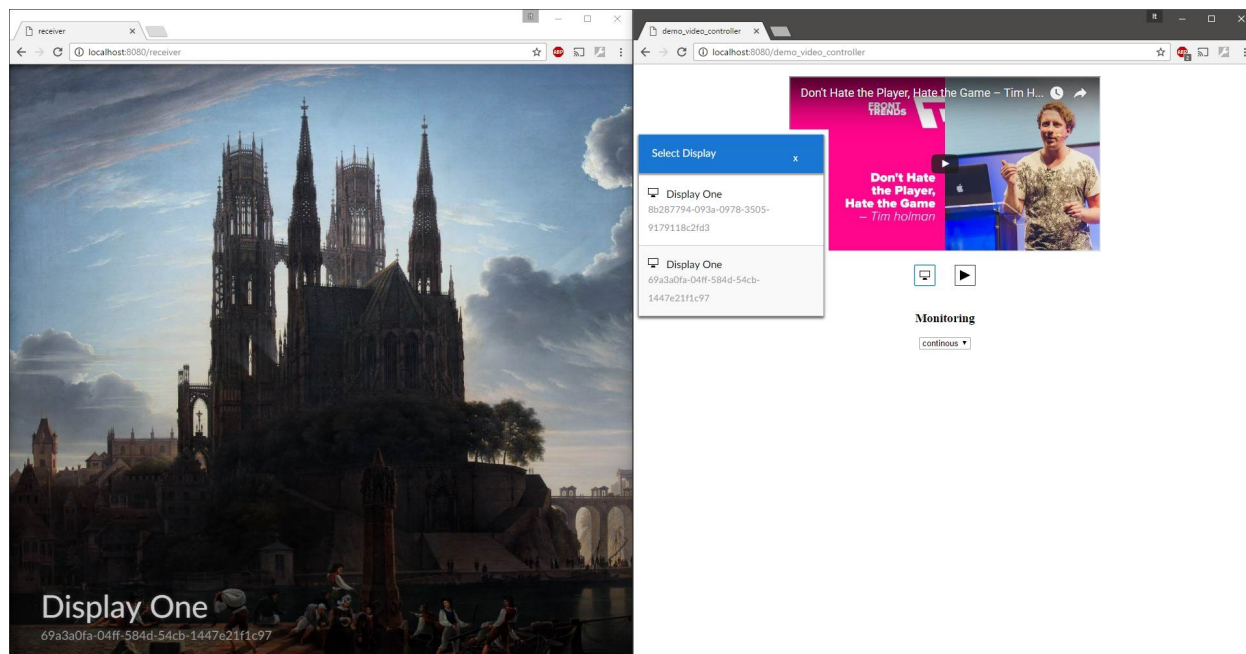
## VI. DEMO

Included in the presentation is a simple controller page incorporating an embedded youtube video whose url is - upon connecting to a presentation display hosted in a separate site - transmitted to the Receiver and then loaded there as a Receiving Browsing Context. Furthermore certain click events in the controller document invoke a function call in the Receiving Browsing Context via a predefined micro-protocol to play and pause the video. Similar to the Google Chromecast Device a backdrop image and the display identification are displayed when the Controller is idle. The II-B handlers for e.g. the connection handshake or message exchange are implemented with a Node.JS app incorporating a hybrid approach of Socket.io and a more compatible ajax fallback.

To start it just hit `npm start` in `%projectRoot%/server` and send your browser to `localhost:8080/receiver` and `localhost:8080/demo_video_controller`.

## VII. APPENDICES

The attached images show how the following usage looks like, after starting the display and thus the Receiver a backdrop image is visible. Loading the controlling application page instantiates the Controller. The user may now connect to a display of her choice after clicking the designated button. After the handshake is complete the url of the embedded video is exchanged and the user may toggle playback remotely. Similar to how Youtube on a Chromecast works other users may now enter the room (or the same user can reconnect if the presentationId is cached) and overwrite what is being shown on the display without revoking the right of the other participants to toggle video playback.



3/2/2017

Presentation API: Controlling UA: All Results

## Presentation API: Controlling UA: All Results

Test files: 24. Total subtests: 102

## Test Files

1. presentation-api-controlling-uaPresentationRequest\_error.html
2. presentation-api-controlling-uaPresentationRequest\_mixedcontent\_https.html
3. presentation-api-controlling-uaPresentationRequest\_mixedcontent\_multiple\_https.html
4. presentation-api-controlling-uaPresentationRequest\_sandboxing\_error.html
5. presentation-api-controlling-uaPresentationRequest\_sandboxing\_success.html
6. presentation-api-controlling-uaPresentationRequest\_success.html
7. presentation-api-controlling-uaGetAvailability.html
8. presentation-api-controlling-uaGetAvailability\_sandboxing\_success.html
9. presentation-api-controlling-uaIdharness.html
10. presentation-api-controlling-uaReconnectToPresentation\_notfound\_error.html
11. presentation-api-controlling-uaReconnectToPresentation\_sandboxing\_success.html
12. presentation-api-controlling-uaStartNewPresentation\_error.html
13. presentation-api-controlling-uaPresentationAvailability\_onchange-manual.html
14. presentation-api-controlling-uaPresentationConnection\_onclosed-manual.html
15. presentation-api-controlling-uaPresentationConnection\_onconnected-manual.html
16. presentation-api-controlling-uaPresentationConnection\_terminated-manual.html
17. presentation-api-controlling-uaPresentationRequest\_onconnectionavailable-manual.html
18. presentation-api-controlling-uaDefaultRequest\_success-manual.html
19. presentation-api-controlling-uaReconnectToPresentation\_success-manual.html
20. presentation-api-controlling-uaStartNewPresentation\_displaynotallowed-manual.html
21. presentation-api-controlling-uaStartNewPresentation\_displaynotfound-manual.html
22. presentation-api-controlling-uaStartNewPresentation\_sandboxing\_success-manual.html
23. presentation-api-controlling-uaStartNewPresentation\_success-manual.html
24. presentation-api-controlling-uaStartNewPresentation\_unsettledpromise-manual.html

Test	LT01
presentation-api-controlling-uaPresentationRequest_error.html	
Call PresentationRequest() constructor without presentation URL. TypeError Exception expected.	PASS
presentation-api-controlling-uaPresentationRequest_mixedcontent_https.html	
Creating a PresentationRequest with a priori unauthenticated URL in an HTTPS context throws a SecurityError exception.	FAIL
presentation-api-controlling-uaPresentationRequest_mixedcontent_multiple_https.html	
Creating a PresentationRequest with a set of URLs containing a priori unauthenticated URL in an HTTPS context throws a SecurityError exception.	FAIL
presentation-api-controlling-uaPresentationRequest_sandboxing_error.html	
Sandboxing: Creating a PresentationRequest from a nested context fails when allow-presentation is not set	FAIL
presentation-api-controlling-uaPresentationRequest_sandboxing_success.html	
Sandboxing: Creating a PresentationRequest from a nested context succeeds when allow-presentation is set	PASS
presentation-api-controlling-uaPresentationRequest_success.html	
Call PresentationRequest constructor with a valid relative presentation URL. No Exception expected.	PASS
presentation-api-controlling-uaGetAvailability.html	
Getting the presentation displays availability information.	PASS
presentation-api-controlling-uaGetAvailability_sandboxing_success.html	
Sandboxing: Retrieving display availability from a nested context succeeds when allow-presentation is set	PASS
presentation-api-controlling-uaIdharness.html	
Navigator interface: attribute presentation	PASS
Presentation interface: existence and properties of interface object	PASS
Presentation interface object length	PASS
Presentation interface object name	PASS
Presentation interface: existence and properties of interface prototype object	FAIL
Presentation interface: existence and properties of interface prototype object's "constructor" property	PASS
Presentation interface: attribute defaultRequest	PASS

file://X:\workspace\vb\web-platform-tests\presentation-api-controlling-ua\genial.html

1/4

3/2/2017

Presentation API: Controlling UA: All Results

Test	LT01
Presentation must be primary interface of navigator presentation	FAIL
Stringification of navigator presentation	FAIL
Presentation interface: navigator presentation must inherit property "defaultRequest" with the proper type (0)	FAIL
PresentationRequest interface: existence and properties of interface object	PASS
PresentationRequest interface object length	PASS
PresentationRequest interface object name	PASS
PresentationRequest interface: existence and properties of interface prototype object	FAIL
PresentationRequest interface: existence and properties of interface prototype object's "constructor" property	PASS
PresentationRequest interface: operation start()	PASS
PresentationRequest interface: operation reconnect(DOMString)	PASS
PresentationRequest interface: operation getAvailability()	PASS
PresentationRequest interface: attribute onconnectionavailable	PASS
PresentationRequest must be primary interface of navigator presentation.defaultRequest	FAIL
Stringification of navigator.presentation.defaultRequest	FAIL
PresentationRequest interface: navigator.presentation.defaultRequest must inherit property "start" with the proper type (0)	PASS
PresentationRequest interface: navigator.presentation.defaultRequest must inherit property "reconnect" with the proper type (1)	PASS
PresentationRequest interface: calling reconnect(DOMString) on navigator.presentation.defaultRequest with too few arguments must throw TypeError	PASS
PresentationRequest interface: navigator.presentation.defaultRequest must inherit property "getAvailability" with the proper type (2)	PASS
PresentationRequest interface: navigator.presentation.defaultRequest must inherit property "onconnectionavailable" with the proper type (3)	FAIL
PresentationRequest must be primary interface of presentation_request	FAIL
Stringification of presentation_request	FAIL
PresentationRequest interface: presentation_request must inherit property "start" with the proper type (0)	PASS
PresentationRequest interface: presentation_request must inherit property "reconnect" with the proper type (1)	PASS
PresentationRequest interface: calling reconnect(DOMString) on presentation_request with too few arguments must throw TypeError	PASS
PresentationRequest interface: presentation_request must inherit property "getAvailability" with the proper type (2)	PASS
PresentationRequest interface: presentation_request must inherit property "onconnectionavailable" with the proper type (3)	FAIL
PresentationRequest must be primary interface of presentation_request_uris	FAIL
Stringification of presentation_request_uris	FAIL
PresentationRequest interface: presentation_request_uris must inherit property "start" with the proper type (0)	PASS
PresentationRequest interface: presentation_request_uris must inherit property "reconnect" with the proper type (1)	PASS
PresentationRequest interface: calling reconnect(DOMString) on presentation_request_uris with too few arguments must throw TypeError	PASS
PresentationRequest interface: presentation_request_uris must inherit property "getAvailability" with the proper type (2)	PASS
PresentationRequest interface: presentation_request_uris must inherit property "onconnectionavailable" with the proper type (3)	FAIL
PresentationAvailability interface: existence and properties of interface object	PASS
PresentationAvailability interface object length	PASS
PresentationAvailability interface object name	PASS
PresentationAvailability interface: existence and properties of interface prototype object	FAIL
PresentationAvailability interface: existence and properties of interface prototype object's "constructor" property	PASS
PresentationAvailability interface: attribute value	PASS
PresentationAvailability interface: attribute onchange	PASS
PresentationConnectionAvailableEvent interface: existence and properties of interface object	PASS

file://X:\workspace\vb\web-platform-tests\presentation-api-controlling-ua\genial.html

2/4

3/2/2017

Presentation API: Controlling UA: All Results

Test	LT01
PresentationConnectionAvailableEvent interface object length	PASS
PresentationConnectionAvailableEvent interface object name	PASS
PresentationConnectionAvailableEvent interface: existence and properties of interface prototype object	FAIL
PresentationConnectionAvailableEvent interface: existence and properties of interface prototype object's "constructor" property	PASS
PresentationConnectionAvailableEvent interface: attribute connection	PASS
PresentationConnection interface: existence and properties of interface object	PASS
PresentationConnection interface object length	PASS
PresentationConnection interface object name	PASS
PresentationConnection interface: existence and properties of interface prototype object	FAIL
PresentationConnection interface: existence and properties of interface prototype object's "constructor" property	PASS
PresentationConnection interface: attribute id	PASS
PresentationConnection interface: attribute url	PASS
PresentationConnection interface: attribute state	PASS
PresentationConnection interface: operation close()	PASS
PresentationConnection interface: operation terminate()	PASS
PresentationConnection interface: attribute onconnect	PASS
PresentationConnection interface: attribute onclose	PASS
PresentationConnection interface: attribute onterminate	PASS
PresentationConnection interface: attribute binaryType	PASS
PresentationConnection interface: attribute onmessage	PASS
PresentationConnection interface: operation send(DOMString)	PASS
PresentationConnection interface: operation send(Blob)	PASS
PresentationConnection interface: operation send(ArrayBuffer)	PASS
PresentationConnection interface: operation send(ArrayBufferView)	PASS
PresentationConnectionCloseEvent interface: existence and properties of interface object	PASS
PresentationConnectionCloseEvent interface object length	PASS
PresentationConnectionCloseEvent interface object name	PASS
PresentationConnectionCloseEvent interface: existence and properties of interface prototype object	FAIL
PresentationConnectionCloseEvent interface: existence and properties of interface prototype object's "constructor" property	PASS
PresentationConnectionCloseEvent interface: attribute reason	PASS
PresentationConnectionCloseEvent interface: attribute message	PASS
presentation-api-controlling-uaReconnectToPresentation_notfound_error.html	
Calling "reconnect" with an unknown presentation ID fails with a NotFoundError exception	PASS
presentation-api-controlling-uaReconnectToPresentation_sandboxing_success.html	
Sandboxing: Reconnecting a presentation from a nested context succeeds when allow-presentation is set	PASS
presentation-api-controlling-uaStartNewPresentation_error.html	
The presentation could not start, because a user gesture is required.	PASS
presentation-api-controlling-uaPresentationAvailability_onchange-manual.html	
Monitoring the list of available presentation displays.	FAIL
presentation-api-controlling-uaPresentationConnection_onclosed-manual.html	
the onclose is fired and the connection state is closed.	PASS

file://X:\workspace\vb\web-platform-tests\presentation-api-controlling-ua\genial.html

3/4

3/2/2017

Presentation API: Controlling UA: All Results

Test	LT01
presentation-api-controlling-uaPresentationConnection_onconnected-manual.html	
the onconnect is fired and the connection state is connected	PASS
presentation-api-controlling-uaPresentationConnection_terminated-manual.html	
the onterminate is fired and the connection state is terminated	PASS
presentation-api-controlling-uaPresentationRequest_onconnectionavailable-manual.html	
The connectionavailable event was fired successfully.	PASS
presentation-api-controlling-uaDefaultRequest_success-manual.html	
[Optional] Starting a presentation from the browser using a default presentation request.	PASS
presentation-api-controlling-uaReconnectToPresentation_success-manual.html	
Reconnect to presentation success manual test	PASS
presentation-api-controlling-uaStartNewPresentation_displaynotallowed-manual.html	
Calling "start" when the user denied permission to use the display returns a Promise rejected with a NotFoundError exception.	PASS
presentation-api-controlling-uaStartNewPresentation_displaynotfound-manual.html	
Calling "start" when there is no available presentation display returns a Promise rejected with a NotFoundError exception.	PASS
presentation-api-controlling-uaStartNewPresentation_sandboxing_success-manual.html	
Sandboxing: starting a presentation from a nested context succeeds when allow-presentation is set	PASS
presentation-api-controlling-uaStartNewPresentation_success-manual.html	
Checking the chain of events when starting a new presentation	PASS
presentation-api-controlling-uaStartNewPresentation_unsettledpromise-manual.html	
Calling "start" when there is already an unsettled Promise returns a Promise rejected with an OperationError exception.	PASS

file://X:\workspace\vb\web-platform-tests\presentation-api-controlling-ua\genial.html

4/4