
Table of Contents

1. Introduction

The Web & TV Testing Task Force is part of the Web & TV Interest Group. The Task Force is responsible for developing testing requirements specific to Web & TV applications. The focus is expected to be on HTML5 and associated specifications. The Task Force will work with the Web Testing Interest Group and other relevant W3C groups to ensure that Web & TV requirements are met by the W3C testing framework.

2. Background

Over the last couple of years, much work has been done to provide HTML5 with the features necessary to support commercial video. Adaptive bit-rate video formats, protected content and features required by commercial video regulation have all been developed to support common means of providing commercial video using standard interfaces. Being able to reliably test these new features is a necessary step towards ensuring high-quality implementations that run correctly and consistently on HTML5 browsers. The W3C has recognized the need for testing to encourage consistent and reliable implementation of HTML5-compliant browsers. The Web & TV Testing Task Force will work with other testing efforts in W3C to make sure the W3C tests are rich enough to support Web & TV use cases.

The task force set the following objectives:

- Collect Web & TV testing use cases (e.g. "Testing a browser embedded in a commercial TV" or "use of W3C tests by third-party certification organizations").

- List requirements for W3C test tool features to achieve the use cases (e.g. "ability to test embedded browsers").
- List prioritized requirements for W3C specification test coverage to achieve the use cases (e.g. HTML5, Media Source Extensions).
- Identify gaps in the current test tools.
- Identify gaps in the current test coverage.
- List features important to Web & TV members and classify them in terms of testing priority.
- Work with the Web Testing IG, Browser Testing and Tools WG, HTML Testing TF and other relevant W3C groups to communicate the requirements and develop a strategy to fill the identified gaps.
- Liaison with external organizations to inform them about the ongoing activities and gather input on the use cases and requirements.

3. Conformance

As well as sections marked as non-normative, all authoring guidelines, diagrams, examples, and notes in this specification are non-normative. Everything else in this specification is normative.

The key words must, must not, shall, should and should not in this specification are to be interpreted as described in RFC 2119 [*RFC2119*].

This specification only applies to one class of product: ***W3C Technical Reports*** . A number of specifications may be created to address the requirements enumerated in this document. In some cases the union of multiple parts of different specifications may be needed to address a single requirement. Nevertheless, this document speaks only of ***conforming specifications*** .

Conforming specifications are ones that address one or more requirements listed in this document. Conforming specifications should attempt to address should level requirements requirements unless there is a technically valid reason not to do so.

4. Terminology

Add terms and define them.

5. Requirements

The following testing requirements were selected by the task forced based on individual contributions, derived requirements from relevant use cases and surveys

submitted by individuals and other standards development organizations.

1. General Requirements
    1. Central Test Runner
        1. *One URL as central location for all tests*

There MUST be one URL as the destination for all W3C testing.

Having a single location reduces confusion and makes it more likely that the correct tests are performed as prescribed. It also allows for simple maintenance as changes can be made once and all users who go to the destination will have the same changes instantly.

        1. *One click to run all tests*
        2. *Clear results summarizing top pass/fail results*
        3. *Detailed pass/fail results for individual tests*
        4. *Test configuration options*
        5. *Certifiable test logs*
    2. Device Tests
        1. *Remote testing*
        2. *Differences for products vs. prototypes?*

2. Use Case Requirements
    1. Standardized APIs for test hooks
    2. An ecosystem (web sites, workshops, etc.) to get feedback from the community on bugs, priorities, features, etc.
    3. Provide one home for all W3C tests
    4. Performance measurement
        1. *Time to start a stream*
        2. *Average frame rate*
        3. *Maybe defined "acceptable" bound, but the measure is most important because different applications will require different performance.*
    5. Management of testing with various codecs and encoding schemes (e.g. adaptive bit rate)
    6. Management of testing with various externally provided CDM implementations
    7. Ensure testing can be performed with EME implementation in JavaScript
    8. Interface and testing to override cross-origin restrictions for specific devices and services authorized by the user
    9. Testing security of discovery process (e.g. user must specifically authorize access to each device)
    10. Synchronization of streams in MSE (e.g. works with ad insertion or concatenation of programs in continuous stream)

5.2.11. Seamless audio splicing in MSE (i.e. prevent audio artifacts and maintain synchronization between audio and other related signals)

11. **Network impairment/feedback mechanism to force MSE adaptive bit-rate implementations to change bit rate.**
12. **Test cases in support of EME**
    1. ***The same CDM across two different browsers yields the same result.***
    2. ***Two different CDMs can decrypt the same stream***

3. Company Requirements

Explain how the internal results were gathered and how they were used to generate test requirements.

4. SDO Requirements

Explain how aggregated SDO requirements were gathered and how they were used to generate test requirements.

5. Summary Feature Coverage Table

Put the cleaned up feature coverage table here.

6. Use Cases

Many of the requirements for testing Web & TV capabilities were derived from candidate use cases submitted to the Web & TV task force. The following use cases were accepted by the Task Force.

1. Category 1

1. Improve Web Platform Consistency – U1

Description:

- *Provide comprehensive test suites for HTML5 and related specifications to support fully portable web applications.*

Motivation:

- *We need to maintain a consistent web app platform across all browsers and devices.*
- *The cost of Web App support across browsers is still high. Each app company duplicates this cost, adding up to a huge expenditure.*
- *The number of browsers and devices continues to increase.*
- *Existing limited tests are focused on specification completion, not platform verification*

Dependencies:

- *New tests, tests frameworks and client test support will be needed.*

What needs to be standardized:

- *The user agent may need to standardize additional APIs for test hooks.*
- *W3C should provide websites (webplatform.org?) and/or workshops to get requirements, priorities and bugs from the app community and library authors.*

2. *Support External Testing/Certification Organizations – U2*

Build a single, consistent web site to run all W3C specification tests.

- ***Central**: Provide one home for all W3C tests.*
- ***Interactive or scripted**: Must be able to run tests either interactively or completely scripted.*
- ***Configurable**: Enable choice of which specs and/or individual tests to run.*
- ***One click/script**: Any mix of specification tests should all be runnable from one click or a single script.*
- ***Automated**: All tests must be as fully automated as possible.*
- ***Results**: Test runs must create a detailed pass/fail report.*

Motivation:

- *HTML5 being built into external specs: DLNA, OIPF, DTG, HbbTV, etc.*
- *Many of these organizations provide testing and/or certification. A consistent W3C test suite ensures all organizations measure W3C specs consistently.*
- *Current W3C tests difficult to use by external organizations due to inconsistencies of test frameworks, test locations, coverage and lack of maintenance.*

Dependencies:

- *Involves changes and additions to test suites and frameworks across W3C.*

What needs to be standardized:

- *Two external HTML5-related test suites meet the goals above and can be used as examples and/or to leverage technology*
  - *Khronos WebGL Conformance Test Suite* http://www.khronos.org/webgl/wiki/Testing/Conformance
  - *ECMA Test262* http://test262.ecmascript.org

3. *Browser Graphics Performance Testing – U3*

Description: Enable benchmarking of graphics relates specifications e.g. CSS Transition, Animation or Transform is important for devices with limited capability

(like TV-sets and STBs).

Support from the user agent to extract relevant metric may be needed.

Motivation: Functional testing of specifications like CSS Transform, Transition and Animation is not enough since the usability of such functionalities is heavily dependent on the browser capability of rendering the related graphic operations with acceptable performances. Being able to test such performances is then a key factor, especially when dealing with devices with limited hw capability.

Dependencies: As User Agent support may be required, discussion with relevant W3C WGs (e.g. WebPerf) may be required.

What needs to be standardized: The user agent may need to expose to Javascript metrics that can be used for benchmarking purposes.

- One example is the mozilla window.mozPaintCount https://developer.mozilla.org/en-US/docs/DOM/window.mozPaintCount
- Related discussion on a similar property for chromium is still open with no conclusion: http://code.google.com/p/chromium/issues/detail?id=65348 http://code.google.com/p/chromium/issues/detail?id=120796

4. Media Source Extension (MSE) Testing – U4

Description:

- *Provide comprehensive test suite for HTML5 Media Source Extensions (MSE) specification to support adaptive and live streaming to a variety of devices such as TVs / STBs, smart phones, tablets, and PCs, etc*

Specific Areas of Test:

- *Seamless splicing. In scenarios such as adaptive bitrate, branching and commercial insertion, multiple media streams are spliced into a single continuous program. Care must be taken when processing audio signals at each of the splice points: the audio content on each side of the splice might not have been authored in anticipation of a splice or, when audio is coded, the coded audio frame boundaries of each stream might not align with each other or the video frame boundaries of the companion video content. Common requirements: (a) the splice should be specified on a video frame boundary common to both streams, (b) the video should have identical characteristics in both streams, and (c) the audio should be coded using a frame-based audio codec and the coded audio frame boundaries should not be aligned with the video frame boundaries of the companion video program, (d) the results should apply whether audio and video content are transmitted in a single multiplex or as separate streams, and (e) the stream should be carried in a ISO BMFF container.*

- *When splicing of two unrelated streams: (i) no harmful audio artifacts should be generated, and (ii) audio and video should remain synchronized within +/- 40 ms following the splice. Additional requirements: (a) the coded audio frame boundaries of the two streams should not be aligned.*
- *When splicing two streams authored with anticipation of splicing at a specified video frame boundary: (a) no artifact should be generated and (b) audio and video synchronization should remain unchanged following the splice. Additional requirements: (a) the coded audio frame boundaries of the two streams should be aligned.*

Motivation:

- *HTML5 MSE extends HTMLMediaElement to allow JavaScript to generate media streams for playback, thus facilitates a variety of use cases like adaptive streaming, time shifting live streams, ad insertion and video editing, etc.*
- *MSE is specifically important for a variety of devices such as TVs / STBs, smart phones, tablets, and PCs etc.*
- *Supporting MSE Testing will accelerate the time-to-market of MSE-enabled device, offer the consumers with MSE-enabled video services and provide end users with better user experience.*

Dependencies:

- *User Agent support is required as the JavaScript library needs to be extended to support MSE.*
- *Support for ISO BMFF byte stream format*
- *Support at least one frame-based audio codec*

What needs to be standardized:

User Agent needs to be exposed with MSE-enabled JavaScript library.

5. Encrypted Media Extension (EME) Testing – U5

Description:

- *Provide comprehensive test suite for HTML5 Encrypted Media Extensions (EME) specification to enable playback of protected content, and related use cases ranging from simple clear key decryption to high value video*

Specific Areas of Test:

(1) *CDM portability*: 1 CDM type, 2 browsers, 1 stream

This would be a test of whether two browsers using the same type of CDM can

decode the same encrypted stream. For example, if BrowserA and BrowserB both include support for the XYZ CDM, the test would be whether BrowserA and BrowserB can both decode and display the same stream that is meant to be decoded by the XYZ CDM. This could be tested on both clear-key and non clear-key CDMs.

(2) "Common Encryption": 2 CDM types, 2 browsers, 1 stream

This would be a test of whether two browsers using different types of CDM can decode the same encrypted stream. For example, if BrowserA supports XYZ CDM and BrowserB supports UVW CDM and if both XYZ and UVW support the same common encryption format, the test would be whether BrowserA and BrowserB can both decode and display the same stream using two different CDMs. This must be tested on two non clear-key CDMs.

(3) "HTML/CSS transformations": 1 CDM, 1 browser, 1 stream

This would be a test of whether a CDM correctly transforms the video given a variety of HTML and CSS transformations, such as scaling, rotation and occlusion. This could be tested on both clear-key and non clear-key CDMs.

Motivation:

- *HTML5 EME extends HTMLMediaElement to allow JavaScript to select content protection mechanisms, control license/key exchange, and implement custom license management algorithms.*
- *It supports a wide range of use cases without requiring client-side modifications in each User Agent for each use case. This also enables content providers to develop a single and robust application solution for all devices (TVs / STBs, smart phones, tablets and PCs etc) supporting a range of content decryption and protection technologies.*
- *Supporting EME Testing will accelerate the time-to-market of EME-enabled device, offer the consumers with EME-enabled video services and provide end users with better user experience without dependency on Flash or SilverLight.*

Dependencies:

- *User Agent support is required as the JavaScript library needs to be extended to support EME.*

What needs to be standardized:

User Agent needs to be exposed with EME-enabled JavaScript library.

6. Network Service Discovery (NSD) Testing – U6

Description:

- *Provide comprehensive test suite for Network Service Discovery (NSD) specification to enable a web app to discover and subsequently communicate with HTTP-based services advertised via common discovery protocols within the current network*

Motivation:

- *NSD allows JavaScript to discover, communicate, select and manage other network services, specifically in a local network such as home environment.*
- *It supports a wide range of use cases in home network, for example, web-based TV remote control of multiple home appliances, offband messaging channel of communication, support of UPnP, JSON-RPC, Protocol Buffers or other messaging formats, and related privacy and authorization mechanisms.*
- *Supporting NSD Testing will accelelate the time-to-market of NSD-enabled device in home environment and other circumstances, and provide end users with better user experience in managing multiple services in a local network.*

Dependencies:

- *User Agent support is required as the JavaScript library needs to be extended to support NSD.*

What needs to be standardized:

User Agent needs to be exposed with NSD-enabled JavaScript library.

7. Browser Synchronization – U7

Description:

- Provide test requirements to ensure multiple browsers with possibly different CDMs, accessing different streams, can be synchronized
- In cases where browsers are accessing AV streams of the same content, one or more browser streams must be able to synchronously start or stop. This enables streams to synchronously 'switch' from one browser to another.
- Requirements may include HTML support of timing and synchronization of data streams
- For example, Browser A, running CDM-A, accessing video stream A, and Browser B, running CDM-B, accessing audio stream B. Lip sync test.
- Requirements may include HTML support of timing and synchronization of data streams, including STOP and RESUME functions.

Motivation:

- "In a multi-screen viewing environment, synchronization of media presentation on the browsers on each individual screen/device is necessary

to have a coherent orchestration of multi-device content presentation experience.

- *To enable content stream that initially runs on one browser to be re-directed to another browser.*
- *The streams may be a encrypted media stream, and each browser may be using a different* **What needs to be standardized or what requirements does this use case add:**
- *Granularity of synchronization, specified in time unit, or video frame, e.g.*
- *Behavior of browser to hand-off the stream to another browser.*

7. Next Steps

The next step is to submit this document to the relevant W3C Testing group(s) and make any necessary updates based on comments received from the various groups.

8. Acknowledgements

Thanks to all the individuals and organizations who contributed to this requirements document.