Lib-Ray Draft Spec of Support Features

Developer guide for Supporting Lib-Ray - April 2011 DRAFT ("Version 0.2")

New In Version 0.2

After presenting my original prototype for the Lib-Ray concept ("Version 0.1") at the Texas Linux Fest, I received some feedback which changed the way I looked at the problem. Instead of thinking of the Lib-Ray disk as a videoplayer with menus tacked on via a browser, I've moved to thinking of the whole movie as a multimedia website contained within the HTML5 browser. This introduces a few limitations – the embedded video player lacks a few features compared to native viewers like VLC – but overall it's a big win because it eliminates the menu-integration problems that I had with version 0.1.

I also found that quite a bit of the needed functionality exists already in the version 6.0.472.63 of Chromium that is currently released, so that, with version 0.2, it is actually possible to load the prototype disk and watch it.

I still consider this a prototype, however (hence the "0.x" version), because although some important functionality works, there are are some fundamental problems as well. In particular, although some subtitle and audio track support is implemented on the disk, based on the WHATWG HTML5 draft[1], for example, these features are not supported by any known browser, including the Chromium browser I used for testing. This means not only that the support will not work until those features are implemented, but also that I may have made errors in interpreting or implementing the draft.

Support Levels

To make sure we're understanding each other, I've decided to identify two tiers of "support" for Lib-Ray.

"Zero"

"Level Zero" refers to the incidental support provided by platforms which already support viewing complex Theora video streams and have HTML5-compatible browsers that can work with the Lib-Ray HTML menus.

On such platforms, it is possible to watch Lib-Ray movies, but the user experience is compromised by a number of navigation and control problems that may require "work arounds" (such as manually controlling audio and subtitle tracks in the video player, while the "setup" menus for the disk do not work).

We do want to encourage reporting of such systems, but also distinguish it from "true" support for Lib-Ray, so we call this "Level Zero Support for Lib-Ray".

Right now, the best example of this is a GNU/Linux system running a recent version of the Chromium web browser.

[1] (as of April 2011 – see: *http://www.whatwg.org/specs/web-apps/current-work/multipage/index.html*)

"One"

"Level One" support on the other hand, means that some special accomodation has been made to play Lib-Ray disks similarly to DVD or Blu-Ray disks: the disk autoloads into the player, the menu is displayed, setup controls work, the video can be played, and after playback, control is returned to the menu. These should also handle remote controls signals in a sensible way. The rest of this document is dedicated to providing this "Level One Support for Lib-Ray".

1 Auto-loading Behavior

In order to auto-detect a Lib-Ray disk, the player should, on mounting a new medium, check for the 'meta.cnf' file in the top level directory of the disk. The mere existence of this file is a strong clue that this is a Lib-Ray disk. But this fact can be further verified by reading the file and looking for the line:

[Lib-Ray] which marks the mandatory "LibRay" section of this file.
The line:

LibRayVersion: X.X allows you to differentiate versions of the standard. This draft standard is marked as version "0.2".

1.1 Loading the Disk

Once the disk has been identified as Lib-Ray, it should be safe to treat it as one. To do this, a Lib-Ray-enabled browser should be loaded (preferably, but not essentially in fullscreen/kiosk mode), and the file 'index.html' in the top level of the disk should be opened. From here, the menu system will control the next actions.

With Chromium, this looks like this:

```
$ chromium-browser --kiosk /path/to/disk/index.html
```

1.2 Query Based Loading is Compliant

For security or other reasons, the user or distribution developer may not want disks to load automatically. In these cases, it's reasonable to simply offer the ability to open the Lib-Ray disk, or otherwise communicate this to the user, and the platform will still be compliant.

2 Correct "Play" and "Stop" Button Behavior in the Browser

In the Lib-Ray version 0.2 prototype, controls that should be mapped to remote control buttons use the HTML5 "accesskey" attribute to identify the correct keystroke sequence.

Compliant systems with remote controls should map these to the appropriate keys. This can be done with LIRC[2], for example.

In version 0.2, there are two keys to map: "Play" should map to "Alt+P" and "Stop" should map to "Alt+Q".

As in version 0.1, the Play button will also have the "tabindex" attribute set to "1" to make the "Play" button the default button when the menu is loaded.

You'll note that the "Stop" button is mapped to a browser control in v0.2, because we never actually leave the browser during playback – the video feature is technically played in a fullscreen window within the browser (which is different from the expectation with v0.1).

3 Required HTML5 Object Model for Full Lib-Ray Support

Instead of calling for a plugin to provide a custom object model as in v0.1, in v0.2 we merely require that the following elements of the WHATWG draft standard are fully implemented:

3.1 Accesskey Attributes ("8.4.2 The accesskey attribute")

In order to implement the special button behaviors (described in section 2), the browser must support the "accesskey" attribute for anchors. This is described in the WHATWG draft and is implemented in Chromium 6.0.472.63. No particular mapping to "actual key combination" is required, but any macros for the remote control system will have to be adjusted to match the browser's behavior. With Chromium, the key combinations appears to simply be "Alt" plus the indicated key (so "Play is "Alt+P" and "Stop" is "Alt+Q").

3.2 Multiple Audio Track Support ("4.8.10.10 Media resources with multiple media tracks")

The audioTracks features must be supported.

We provide alternate audio tracks (including commentaries) in either Ogg FLAC or Ogg Vorbis streams embedded within the Ogg Theora movie stream. These need to be selectable from Javascript in the setup menus:

media . audioTracks

Returns a MultipleTrackList object representing the audio tracks available in the media resource.

Note also: "4.8.10.10.1 TrackList objects"

The audio tracks are expected to provide this interface:

[2] See: `http://www.lirc.org/`

tracks . length

Returns the number of tracks in the list.

name = tracks . getName(index)

Returns the name of the given track, if known, or the empty string otherwise.

language = tracks . getLanguage(index)

Returns the language of the given track, if known, or the empty string otherwise.

enabled = audioTracks . isEnabled(index)

Returns true if the given track is active, and false otherwise.

audioTracks . enable(index)

Enables the given track.

audioTracks . disable(index)

Disables the given track.

3.3 Multiple Subtitle Track Support ("4.8.10.12 Timed text tracks")

Lib-Ray will provide subtitle tracks either "out-of-band" via external SRT-format subtitle files, or via "in-band" Ogg Kate subtitles. The browser should support both. Following the draft standard, all out-of-band tracks identified by "track" tags will be numbered first, followed by any tracks embedded in the video resource. Although the current prototype uses "out-of-band" tracks to avoid limitations on the interpretation of Ogg Streams (see the next section for issues regarding video playback support in the browser), future releases will probably emphasize "in-band" subtitle information with Ogg Kate as this is a more versatile standard, and also works when the video file is viewed outside of the HTML5 disk-menu context.

This requires that the following interfaces are supported ("4.8.10.12.5 Text track API"):

media . textTracks . length

Returns the number of text tracks associated with the media element (e.g. from track elements). This is the number of text tracks in the media element's list of text tracks.

media . textTracks[n]

Returns the TextTrack object representing the nth text track in the media element's list of text tracks.

track . track

Returns the TextTrack object representing the track element's text track.

textTrack . kind

Returns the text track kind string.

textTrack . label

Returns the text track label.

textTrack . language

Returns the text track language string.

textTrack . readyState

Returns the text track readiness state, represented by a number from the following list:

TextTrack . NONE (0) The text track not loaded state.

TextTrack . LOADING (1) The text track loading state.

TextTrack . LOADED (2) The text track loaded state.

TextTrack . ERROR (3) The text track failed to load state.

textTrack . mode

Returns the text track mode, represented by a number from the following list:

TextTrack . OFF (0) The text track disabled mode.

TextTrack . HIDDEN (1) The text track hidden mode.

TextTrack . SHOWING (2) The text track showing and showing by default modes.

Can be set, to change the mode.

In particular, Lib-Ray sets the current subtitle track by altering the "mode" element of the TextTrack objects using Javascript.

3.4 Positioning ("4.8.10.6 Offsets into the media resource")

For scene selection and resume functionality, the following interface is needed:

media . duration

Returns the length of the media resource, in seconds, assuming that the start of the media resource is at time zero.

Returns NaN if the duration isn't available.

Returns Infinity for unbounded streams.

media . currentTime [= value]

Returns the current playback position, in seconds.

Can be set, to seek to the given time.

Will throw an INVALID_STATE_ERR exception if there is no selected media resource or if there is a current media controller. Will throw an INDEX_SIZE_ERR exception if the given time is not within the ranges to which the user agent can seek.

media . initialTime

Returns the initial playback position, that is, time to which the media resource was automatically seeked when it was loaded. Returns zero if the initial playback position is still unknown.

Scene selection and resume functionality is implemented in Lib-Ray using assignments to the `media.currentTime` attribute.

Testing reveals that positioning playback offsets into the video resource already works with Chromium 6.0.472.63.

4 Web Storage ("Web Storage W3C Working Draft 08 February 2011")

The state of video settings as determined by the Lib-Ray setup menus is stored using "Session Storage", as described in the W3C working draft on "Web Storage"[3]. The following interface is used:

sessionStorage.setItem(key, value)

Both "key" and "value" will be strings (Lib-Ray converts other types to string and back).

[3] See: `http://www.w3.org/TR/webstorage/#the-sessionstorage-attribute`

value = sessionStorage.getItem(key)

The value is returned as a string, based on the key.

All menu state information, including the current audio and subtitle tracks to play and the playback position are relayed to the "feature.html" page in the Lib-Ray disk using SessionStorage. If this is not supported, then those features will not work.

This is already implemented in Chromium 6.0.472.63, but does not work correctly in tested Mozilla browsers.

5 Video Player and Codec Support

The embedded video player must be able to play the Lib-Ray video file. In version 0.2, I have made some compromises to allow this file to allow playback on existing Chromium browsers, but the limitations should be removed in order to enable full support (and this may be required in later versions):

Ogg

The container format for the video stream is Ogg[4], as defined by Xiph.org[5].

Theora

The only currently-support video codec for Lib-Ray v0.2 is Theora[6]. Later versions may add other acceptable free codecs as long as they can be embedded in an Ogg stream.

FLAC

The principle audio stream format is Ogg FLAC[7].

Vorbis

Support for Vorbis[8] audio streams should also be available.

Kate

Subtitles will be embedded in the video file as Ogg Kate[9] streams.

Support for these features is incomplete in Chromium 6.0.472.63 as tested. In particular, Ogg FLAC tracks are silently ignored (they should play normally).

[4] See: `http://xiph.org/ogg/`

[5] See: `http://xiph.org`

[6] See: `http://theora.org/`

[7] See: `http://flac.sourceforge.net/` for the base FLAC codec, and `http://wiki.xiph.org/FLAC` for information about streaming in an Ogg container (OggFLAC).

[8] See: `http://xiph.org/vorbis/`

[9] See: `http://wiki.xiph.org/Kate`

Remove or Raise the 20-Stream Limit?

There seems to be an arbitrary numerical limitation of 20 streams in a single multimedia file (I discovered this experimentally during testing – I'm not aware of any documentation of this limit or why it should exist), which is highly limiting, especially with respect to subtitle tracks. One of the design goals of Lib-Ray is to provide support for many subtitle and audio tracks so as to accomodate global ("no region code") releases without special per-country modifications.

For the prototype "Sintel" disk, for example, there were 40 usable contributed subtitle tracks. It's not hard to imagine this number being as high as 200 for some very popular disks, and because OggKate is very compact, this would still have little impact on the bitrate of the Ogg stream, so it should be possible to provide such numbers of streams.

As a work around, version 0.2 prototype Lib-Ray disks should have a Vorbis soundtrack loaded as the first audio track so that default playback will have normal sound.

Author

Terry Hancock <digitante@gmail.com>, Anansi Spaceworks - Lib-Ray Project
For more information, see http://lib-ray.org