ustudio Javascript Player API

Introduction

uStudio's Javascript Player API allows an external web page to communicate with uStudio HTML5 players. All players, when embedded via standard `<iframe>`, communicate using `postMessage`. This is supported in all modern browsers (IE8+, Chrome, Safari 4+, and Firefox 3+). (To learn more, visit the Mozilla postMessage page.)

One of the benefits of uStudio's modular, extensible approach to players is that any Javascript developer can extend the core functionality of the player with custom Player Modules. These modules communicate via an Event Bus, in a simple publish / subscribe model. The Javascript `postMessage` interface allows direct access to this bus, which not only includes basic operations such as play, pause, time updates, etc., but also events from any custom extension module.

Due to the breadth of variation that can occur between customized players, this documentation only includes the most common events. A method for logging all events in a player is included at the end of the document.

Additionally, there is a helper Javascript wrapper that is referenced later in the document which simplifies the process of embedding and communicating with embedded players.

Event Basics

Unless otherwise specified, all messages sent between the player and the parent window will be JSON-serialized. This ensures that basic types (integers, lists, simple objects, etc) are preserved in all supported browser environments. An example event looks like:

```
{
    "event": "Player.seekto",
    "arguments": [{value: 15.6}],
    "id": "PLAYER_ID"
}
```

The event name is the primary action identifier, and may be namespaced according to the module that defined it. It may represent a request for an action (as shown above), or an indication that an action has occurred (for instance, `Player.seeking` and `Player.seeked`). The arguments list represents any variable information that may be needed by a listener. In the seek event shown above, the value of 15.6 seconds is provided. Finally, the `id` value allows a page to know which player is broadcasting the event, in order to track player progress, etc.

Setting Up

After a Player has finished loading all modules, it will initiate the `postMessage` handshake with the outer page. First, it will broadcast a `uStudio.eventsReady` event to the parent window. The parent window must send a `uStudio.pageReady` event before the player will continue broadcasting remaining events. The `pageReady` event must include one argument, which is a simple Javascript object containing the `url` of the outer page and the `id` the page wishes to use to track the player's future events.

Here is an example of a simple handshake (note this example uses the W3C event subscription, not Microsoft's proprietary `attachEvent`, and expects uStudio is the only service using postMessage on the page):

```
window.addEventListener('message', function(event) {
    var source = event.source;
    var data = JSON.parse(event.data);
    if (data.event === 'uStudio.eventsReady') {
        source.postMessage(JSON.stringify({
            'event': 'uStudio.pageReady',
            'arguments': [{
                'url': window.location.href,
                'id': data.id
            }]
        }), event.origin);
    } else {
        console.log(data);
    }
});
```

The most important attribute for tracking events is the `id` parameter. By default, the player will generate a unique, random id for every player embed (even for the same video), but this can be changed by providing a different id with the `pageReady` event. In the example above, we simply pass the same value we received from the `eventsReady` event. The other option for providing a custom id is to append a query parameter `?_playerid=CUSTOMID` to the embed URL. Note that this may leave the generation of the id up to a CMS, etc. as the HTML is being generated.

For browser security reasons, all `postMessage` broadcasts must include the origin of the target window. Using `event.origin` (as shown above) will ensure that you are sending the proper protocol, domain, and port without having to hard code the embed code URL in the outer page. This should be stored for later use (for each player), since sending an action to the player may not necessarily be in response to another broadcasted event.

After the `eventsReady` event is received by the player, it will begin broadcasting all subsequent events. It is therefore important that a page listens for `uStudio.eventsReady` (before the rest of the document loads) and immediately fires `uStudio.pageReady` in response to each player so that the maximum number of events are broadcast to the outer page.

All subsequent events will still pass through the `message` event callback. It is up to the end application to distribute events appropriately across individual players.

Javascript Wrapper Library

The above work can be reduced by using a helper Javascript library provided by uStudio. It is located at `http://scripts.ustudio.com/player.latest.js`, and can be used to wrap player embeds and simply subscribe and publish events. Simply copy and paste the following line to include the library into an HTML page (before you perform any actions with the library):

```
<script src="http://scripts.ustudio.com/player.latest.js"></script>
```

The following Javascript example shows the creation of a new embed, attaching an event listener, adding the player to the page, and then sending an event from a button click.

```
var player = uStudio.Player.create(
  "http://app.ustudio.com/embed/DESTINATION/VIDEO", 480, 270);

// Hooking up a play button already on the page.
document.getElementById("play-button").onclick = function() {
  player.send("Player.play");
};

// Adding a listener for the "playing" state, so we can make
// the page background dark.
player.listen("Player.playing", function() {
  document.body.className = "dark";
});

// Adding a listener for time updates, so we can display
// the current play time.
player.listen("Player.timeupdate", function(timeUpdateInfo) {
  console.log("Time: " + timeUpdateInfo.currentTime);
});

// Adding the player to the DOM via a placeholder.
player.replace(document.getElementById("new-player"));
```

By setting `uStudio.Player.LOGGING = true;`, all events from players embedded using the above approach will log events to the browser's console. This is a useful tool for understanding (and debugging) any configured or custom events for a given player.

Note that this helper script will not capture events for players which were added to the page as normal `<iframe>` – those will need to be tracked via other methods listed in this document.

Common Events

The following is an event reference guide for the most common events.

Event	**Arguments**	**Description**
Player.durationchange	{ duration: float (seconds) }	Indicates the video duration.
Player.ended	(none)	Indicates playback has stopped.
Player.pause	(none)	Pause video playback.
Player.paused	(none)	Indicates playback has paused.
Player.play	(none)	Start (or resume) vidoe plaback.
Player.playing	(none)	Indicates a player has started / resumed playback.
Player.seeked	{currentTime: float (seconds) }	Indicates a player has finished seeking.
Player.seeking	(none)	Indicates a player is currently seeking.
Player.seekto	{value: float (seconds) }	Instructs player to seek to specified seconds.
Player.timeupdate	{currentTime: float (seconds) }	Indicates current position (in seconds)
Playlist.selectVideo	{index: integer (video index) }	Swaps video playback to specified index.
Playlist.videoSelected	{video: { video metadata }}	Indicates a new video was selected.
Playlist.nextVideo	(none)	Selects next video for playback.
Playlist.previousVideo	(none)	Selects previous video for plaback.