KDP &External Interface

# Available callbacks

1. To enable KDP to communicate with an external javascript interface, first pass a flashvar to the KDP:

externalInterfaceDisabled=false

1. jsCallbackReady – the first callback of the player. The javascript function is passed the id of the player embedded in the page.

function jsCallbackReady(objectId) {

window.kdp = document.getElementById(objectId);

}

1. To draw data from KDP components using the javascript interface, use the evaluate callback. The format of the callback is:

evaluate('{component.property}'); //The curly brackets are required.

For example, if you wish to extract the name of the entry from the KDP:

function getName() {

var entry\_name = kdp.evaluate('{mediaProxy.entry.name}');

alert('Entry name: '+entry\_name);

}

This callback is valid for any object that is externalized by the player. These include any component in the uiconf (buttons, labels etc) which are accessed by their id and the KDP changeable data, such as flashvars and media information (“VO”s).

1. It is also possible to set KDP values using the javascript API callback setKDPAttribute (for KDP versions after 3.2) or setAttribute (for KDP versions < 3.2).

The way to use this callback is:

setKDPAttribute(host,property,value)

For instance: if my KDP UI contains a Label:

<Label id="my\_label" width="18" height="18" text="Hello World!" />

And I use the callback:

setKDPAttribute('my\_label','text','Hello Kaltura!');

The label’s text will change to show “Hello Kaltura!”.

Note that this callback does not require the ‘host’ property to be passed with curly brackets (unlike the evaluate callback).

1. The KDP is based on the pureMVC framework. Therefore, any significant changes and events that occur in the KDP are passed as NOTIFICATIONS. The KDP externalizes these notifications for use by javascript.

In order to listen for a specific notification, use the following callback:

addJsListener(notificationName:String, listenerFunctionName:String)

Example: We want to listen for the player beginning to play:

function addPlayListener() {

kdp.addJsListener('playerPlayed', 'playListener');

}

function playListener() {

alert ('Player is playing!');

}

In some cases you’ll want to remove listeners that are no longer necessary.

Using the example above, if you wanted the alert to only happen once, you would use:

function playListener() {

alert ('Player is playing!');

removeJsListener('playerPlayed', 'playListener');

}

It is also possible to create custom notifications, if, for instance you wish to listen for a particular button being pressed. The way to do this is:

1. Extract the button’s id from the Uiconf xml. If the button’s xml tag is :

<Button id="my\_play\_btn" width="18" height="18" command="play" />

Then the button’s id = my\_play\_btn .

1. Add property kClick=”sendNotification(‘myNotification’)” via the flashvars tag that belongs to the Embed code of the player. Example:

<param name="flashvars" value="… &amp;my\_play\_btn.kClick=sendNotification('playPressed')" />

This makes sure that the button starts the KDP playing and also dispatches notification

‘playPressed’. The notification string is completely up to you.

1. In the script area of your web page add the functions:

function addPlayPressedListener() {

kdp.addJsListener('playPressed', 'playPressedListener');

}

function playPressedListener() {

alert ('Play button was pressed!');

kdp.removeJsListener('playPressed', 'playListener');

}

# Important notes

1. JS listeners are collected. If you add a JS listener for a certain notification, and then add another one for the same notification, the new listener will not override the old one. Both will be executed in the order in which they were added. Therefore, it is advised to strategize, and take care to remove unnecessary listeners to prevent unexpected occurrences in your application.
2. In every KDP released so far, the JS listeners are executed first; this means that if you added a JS listener for the “playerPlayed” notification (to be informed that the player actually started playing) and then used that listener to pause the player ([player\_object].sendNotification('doPause'), the JS listener will take precedence over the player’s actions. This is true for KDP versions up to 3.4.10.1. In the following KDP versions this behavior is changes, and the player ALWAYS takes precedence over the JS listeners.