The Wowza™ Player JavaScript API is an extension of the Wowza Player embed code that allows you to interact with your player instance programmatically from inside your webpage or web application.

The Wowza Player JavaScript API works with Wowza Player as well as Wowza Player for Ultra Low Latency. To learn more about streaming to Wowza Player for Ultra Low Latency, see Configure Wowza Player for Wowza Streaming Cloud ultra low latency streams.

### WowzaPlayer object

**WowzaPlayer** is the object that Wowza Player places in the global DOM and provides access to the Wowza Player Javascript API. Use the **WowzaPlayer.create** static method (in your player embed code) to set up a new player or use the **WowzaPlayer.get** method to retrieve a player instance so you can make other Wowza Player API calls.

**Note:** You can use `$wp` as a shortcut for your **WowzaPlayer** global object. For example:

```javascript
var myPlayer = $wp.get('playerElement');
```

### Set up a WowzaPlayer instance

To make API calls, you must add a **WowzaPlayer** object instance to your webpage to interact with. Our [Getting started with Wowza Player tutorial](#) explains the steps for accessing and embedding Wowza Player in your webpage. Player embed code should resemble the sample code below.

```html
```
Note: Significant updates to Wowza Player may alter the API. To force your WowzaPlayer instance to use the latest stable version of a specific player release, replace latest in the player URL in your embed code to [version-number]-latest. For example:

For more information on previous updates to Wowza Player, see the Wowza Player release notes.

The WowzaPlayer.create static method in your player embed code creates an instance of WowzaPlayer and returns a reference to the new WowzaPlayer instance. If you call this static method after an embedded player has already been created, a new player is created to replace the previous player. This is useful for changing player configuration properties at runtime.

create (PlayerElement : string, ConfigJSON : {}) : WowzaPlayer

Where:
• **playerElement** is the player ID defined in the HTML element in your player embed code.

• **ConfigJSON** is the JSON literal containing the configuration properties for the **WowzaPlayer** instance. See [Wowza Player configuration properties](#) for available values.

The WowzaPlayer object is created and initiated asynchronously, so some playback method calls may not function until the player has entered the **READY_TO_PLAY** or **PLAYING** states. For more information about player states, see the list of **WowzaPlayer.State** values. To learn how to call an event listener when the **WowzaPlayer** instance is ready and playback method calls can be used, see the [onLoad event instance method](#).

Retrieve an existing WowzaPlayer instance

The **WowzaPlayer.get** static method returns the **WowzaPlayer** instance that was created with the specified **PlayerElement** player ID.

```typescript
def get ( PlayerElement : string ) : WowzaPlayer
```

If no **WowzaPlayer** instance matches **PlayerElement**, the method returns **null**.

Destroy a WowzaPlayer instance

The **destroy** method deallocates a **WowzaPlayer** instance.

```typescript
def destroy ( );
```

If you have the **WowzaPlayer** instance available, for example, **myPlayer**, you can do the following:

```typescript
myPlayer.destroy();
```

If you don’t have access to the **WowzaPlayer** instance, you can retrieve the player instance and then destroy it:

```typescript
thePlayer = WowzaPlayer.get('playerElement');
thePlayer.destroy();
```

**Wowza Player configuration properties**
Setter and getter instance methods

Control playback and get information about the player and stream playback with Wowza Player API setter and getter methods.

**Note:** You can chain Wowza Player instance methods that return a reference to the method’s own instance. For example:

<table>
<thead>
<tr>
<th>Control playback</th>
<th>Get playback status</th>
</tr>
</thead>
<tbody>
<tr>
<td>• WowzaPlayer.finish()</td>
<td>• WowzaPlayer.getCurrentState()</td>
</tr>
<tr>
<td>• WowzaPlayer.mute(isMuted)</td>
<td>• WowzaPlayer.getCurrentTime()</td>
</tr>
<tr>
<td>• WowzaPlayer.pause()</td>
<td>• WowzaPlayer.getDuration()</td>
</tr>
<tr>
<td>• WowzaPlayer.play()</td>
<td>• WowzaPlayer.getVolume()</td>
</tr>
<tr>
<td>• WowzaPlayer.seek(newTimeMS)</td>
<td>• WowzaPlayer.isLive()</td>
</tr>
<tr>
<td>• WowzaPlayer.setVolume(newVolume)</td>
<td>• WowzaPlayer.isMuted()</td>
</tr>
<tr>
<td></td>
<td>• WowzaPlayer.isPlaying()</td>
</tr>
</tbody>
</table>

Control playback

The following instance methods control playback in your player.

**finish() : WowzaPlayer**

Finishes stream playback and returns the current WowzaPlayer instance. For example:

```javascript
myPlayer.finish();
```

**Note:** See Stop or pause after a specified time with the Wowza Player JavaScript API for a code example using the `finish` method.

**mute (isMuted : boolean) : WowzaPlayer**

Set `isMuted` to `true` to mute playback volume or set to `false` to unmute and return the
playback volume to its previous level. Then, the method returns the current WowzaPlayer instance. For example:

```javascript
myPlayer.mute(true);
myPlayer.pause();
myPlayer.play();
myPlayer.seek(60000);
myPlayer.setVolume(75);
```

### pause () : WowzaPlayer

Pauses video playback (changes `WowzaPlayer.State` to `PAUSED`) and returns the current WowzaPlayer instance. For example:

```
Note: See Stop or pause after a specified time with the Wowza Player JavaScript API for a code example using the `pause` method.
```

### play () : WowzaPlayer

Starts video playback (changes `WowzaPlayer.State` to `PLAYING`) and returns the current WowzaPlayer instance. For example:

### seek (newTimeMS: number) : WowzaPlayer

Seeks to `newTimeMS` milliseconds into stream playback and returns the current WowzaPlayer instance. For example:

### setVolume (newVolume : number) : WowzaPlayer

Sets the current playback volume to `newVolume`. Volume ranges from 0-100. For example:

Get playback status

The following playback instance methods retrieve information from your player instance.

```javascript
getcurrentState() : number
```
Returns the current playback state. For example:

```
if (myPlayer.getCurrentState() == WowzaPlayer.State.PLAYBACK_COMPLETE) {
    console.log('Playback Complete');
}
console.log( 'Playhead time (ms): ' + myPlayer.getCurrentTime() );
```

For a list of all **WowzaPlayer.State** values and the states those numbers represent, see the table below.

<table>
<thead>
<tr>
<th>Return value</th>
<th>Player state</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0)</td>
<td>WowzaPlayer.State.UNKNOWN</td>
</tr>
<tr>
<td>(1)</td>
<td>WowzaPlayer.State.CREATED</td>
</tr>
<tr>
<td>(2)</td>
<td>WowzaPlayer.State.INITIALIZED</td>
</tr>
<tr>
<td>(3)</td>
<td>WowzaPlayer.State.PREPARING</td>
</tr>
<tr>
<td>(4)</td>
<td>WowzaPlayer.State.READY_TO_PLAY</td>
</tr>
<tr>
<td>(5)</td>
<td>WowzaPlayer.State.PLAYING</td>
</tr>
<tr>
<td>(6)</td>
<td>WowzaPlayer.State.PAUSED</td>
</tr>
<tr>
<td>(7)</td>
<td>WowzaPlayer.State.STOPPED</td>
</tr>
<tr>
<td>(8)</td>
<td>WowzaPlayer.State.PLAYBACK_COMPLETE</td>
</tr>
<tr>
<td>(9)</td>
<td>WowzaPlayer.State.SHUTDOWN</td>
</tr>
<tr>
<td>(10)</td>
<td>WowzaPlayer.State.APP_STOPPED</td>
</tr>
<tr>
<td>(11)</td>
<td>WowzaPlayer.State.ERROR</td>
</tr>
</tbody>
</table>

**getcurrentTime () : number**

Returns the current playhead time in milliseconds. For example:

```
getDuration () : number
```
Returns the duration of the current video asset in milliseconds. For example:

```
getDuration () : number
```

Returns the current playback volume. Volume ranges from 0-100. For example:

```
isVolume () : number
```

Returns `true` if the current video asset is a live stream or nDVR stream. Returns `false` if the current video asset is VOD. For example:

```
isLive () : boolean
```

Returns `true` if the current playback is muted. Returns `false` otherwise. For example:

```
isMuted () : boolean
```

Returns `true` if a stream is playing. Returns `false` if playback is paused or stopped. For example:

```
isPlaying () : boolean
```

**Event instance methods**

The **WowzaPlayer** class provides several methods for attaching and removing event listener functions. These event listener functions are called when certain events occur in the lifecycle of a **WowzaPlayer** instance. Event listeners may be wired and unwired at any time, independent of the state of the **WowzaPlayer** instance.

---

**Notes:** With the following exceptions, Wowza Player 1.0.08 or later is required to use the event instance methods on iOS devices:
The **onLoad** event method is supported on iOS in earlier versions of Wowza Player.

The **onBitrateChanged** event method isn't supported on iOS devices.

The **onPlay**, **onPause**, **onResume**, **onStop**, **onCompleted**, and **onSeek** event methods require Wowza Player 1.1 or later.

- **onBitrateChanged**, **removeOnBitrateChanged**
- **onChunklistURIChanged**, **removeOnChunklistURIChanged**
- **onCompleted**, **removeOnCompleted**
- **onError**, **removeOnError**
- **onLoad**, **removeOnLoad**
- **onMetadata**, **removeOnMetaData**
- **onPause**, **removeOnPause**
- **onPlay**, **removeOnPlay**
- **onPlaybackAttempt**, **removeOnPlaybackAttempt**
- **onPlaybackFailure**, **removeOnPlaybackFailure**
- **onPlaybackSuccess**, **removeOnPlaybackSuccess**
- **onPlayheadTime**, **removeOnPlayheadTime**
- **onReady**, **removeOnReady**
- **onResume**, **removeOnResume**
- **onSeek**, **removeOnSeek**
- **onStateChanged**, **removeOnStateChanged**
- **onStats**, **removeOnStats**
- **onStop**, **removeOnStop**
- **onVolume**, **removeOnVolume**

**onBitrateChanged**, **removeOnBitrateChanged**

```javascript
onBitrateChanged ( callback : function ( bitrateChangedEvent : {} ) ) : WowzaPlayer
```

**Note:** **onBitrateChanged** isn't supported on iOS devices.

**onBitrateChanged** attaches an event listener function that's called when the viewer switches between bitrate renditions in an adaptive bitrate stream. Then, the method returns the current **WowzaPlayer** instance.

**callback** is a function that receives a **bitrateChangedEvent** object literal with two properties:
Where oldBitrate is the previous bitrate and newBitrate is the new bitrate selected by the viewer or player. For example:

```javascript
bitrateChangedEvent : {
   oldBitrate : number,
   newBitrate : number
}
```

```javascript
bitrateChangedListener = function (bitrateChangedEvent) {
   console.log('Old bitrate: ' + bitrateChangedEvent.oldBitrate);
   console.log('New bitrate: ' + bitrateChangedEvent.newBitrate);
};
myPlayer.onBitrateChanged(bitrateChangedListener);
myPlayer.removeOnBitrateChanged(bitrateChangedListener);
```

removeOnBitrateChanged (callback : function (bitrateChangedEvent : {})) : WowzaPlayer

removeOnBitrateChanged removes the requested event listener function and returns the current WowzaPlayer instance. For example:

```javascript
onChunklistURIChanged, removeOnChunklistURIChanged
```

```javascript
onChunklistURIChanged (callback : function (chunklistURIChangedEvent : {})) : WowzaPlayer
```

onChunklistURIChanged attaches an event listener function that’s called when the player switches between redundant chunklists in a redundant stream. Then, the method returns the current WowzaPlayer instance.

callback is a function that receives a chunklistURIChangedEvent object literal with two properties:

Where prevChunklistURI is the URI of the previous chunklist and currentChunklistURI is the URI of the new chunklist selected by the player. For example:
removeOnChunklistURIChanged ( callback : function ( chunklistURIChangedEvent : {} ) ) : WowzaPlayer

**removeOnChunklistURIChanged** removes the requested event listener function and returns the current **WowzaPlayer** instance. For example:

```javascript
chunklistURIChangedListener = function ( chunklistURIChangedEvent ) {
  console.log('Previous chunklist: ' + chunklistURIChangedEvent.prevChunklistURI);
  console.log('Current chunklist: ' + chunklistURIChangedEvent.currentChunklistURI);
};
myPlayer.onChunklistURIChanged( chunklistURIChangedListener );
myPlayer.removeOnChunklistURIChanged( chunklistURIChangedListener );
```

onCompleted, removeOnCompleted

**onCompleted** attaches an event listener function that's called when playback completes and returns the current **WowzaPlayer** instance.

**callback** is a function that receives a **completedEvent** object literal with one property:

```javascript
completedEvent : { time : number }
```

Where **time** is the current playhead time in milliseconds. For example:

```javascript
completedListener = function ( completedEvent ) {
  console.log('Time: ' + completedEvent.time);
};
myPlayer.onCompleted( completedListener );
myPlayer.removeOnCompleted( completedListener );
```

removeOnCompleted ( callback : function () ) : WowzaPlayer

**removeOnCompleted** removes the requested event listener function and returns the current **WowzaPlayer** instance. For example:

```javascript
```

onError, removeOnError
onError (callback : function (errorEvent : {})) : WowzaPlayer

**onError** attaches an event listener function that’s called when an error occurs, and then returns the current **WowzaPlayer** instance.

**callback** is a function that receives an **errorEvent** object literal that has one property.

```javascript
    errorEvent : { message : string }
```

Where **message** explains the error that occurred. For example:

```javascript
    myPlayer.onError(errorListener);
```

removeOnError (callback : function (errorEvent : {})) : WowzaPlayer

**removeOnError** removes the requested event listener function and returns the current **WowzaPlayer** instance. For example:

```javascript
    myPlayer.removeOnError(errorListener);
```

onLoad, removeOnLoad

onLoad (callback : function () ) : WowzaPlayer

**onLoad** attaches an event listener that’s called when the **WowzaPlayer** instance is loaded. If the **WowzaPlayer** is already loaded, the **callback** function is called immediately. Then, it returns the current **WowzaPlayer** instance. For example:

```javascript
    onLoadListener = function () {
        console.log('WowzaPlayer loaded');
    }
    myPlayer.onLoad(onLoadListener);
```

removeOnLoad (callback : function () ) : WowzaPlayer

**removeOnLoad** removes the requested event listener function and returns the current **WowzaPlayer** instance. For example:

```javascript
    myPlayer.removeOnLoad(onLoadListener);
```
onMetadata, removeOnMetadata

onMetadata (callback: function (metaDataEvent: {})): WowzaPlayer

**onMetadata** attaches an event listener function that’s called when metadata is available. When the **onMetadata** event listener is attached, the method returns the current **WowzaPlayer** instance.

**Notes:**
- To make metadata available, you must insert AMF data or ID3 tags in Wowza Streaming Engine, use a source encoder to insert AMF data before ingesting into Wowza Streaming Cloud or Wowza Streaming Engine, or use automatically inserted ID3 PDT tags in Wowza Streaming Cloud. Based on the format of your stream and your desired workflow, see Developer’s guide to using timed metadata in Wowza workflows or Insert ID3 data events into MPEG-TS streams in Wowza Streaming Engine for more information.
- See Listen for metadata and trigger an action with the Wowza Player JavaScript API for a code example using the **onMetadata** method.

callback is a function that receives a single **metaDataEvent** object literal with several properties:

```javascript
metaDataEvent: {packetTime: number, time: number, type: string, [additional data varies by type] }
```

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>packetTime</td>
<td>The time, in milliseconds, as captured in the ingest stream.</td>
</tr>
<tr>
<td>time</td>
<td>The playback time, in milliseconds, at which the event occurred. For live streams, the time the client connected is 0.</td>
</tr>
<tr>
<td>type</td>
<td>A string representing the type of metadata.</td>
</tr>
<tr>
<td>[additional data varies]</td>
<td>Additional properties might be passed from Wowza Streaming Engine based on the type of</td>
</tr>
</tbody>
</table>
metadata and how you've configured Wowza Streaming Engine to insert the metadata (such as ID3 tags).

For example:

```javascript
removeOnMetadata (callback : function (metaDataEvent : {})) : WowzaPlayer

removeOnMetadata removes the requested event listener function and returns the current WowzaPlayer instance.
```

```javascript
onPause, removeOnPause

onPause (callback : function (pauseEvent : {})) : WowzaPlayer

onPause attaches an event listener function that's called when playback is paused and returns the current WowzaPlayer instance.

callback is a function that receives a pauseEvent object literal with one property:

```javascript
pauseEvent : {time : number}
```

Where time is the current playhead time in milliseconds. For example:

```javascript
removeOnPause (callback : function () ) : WowzaPlayer

removeOnPause removes the requested event listener function and returns the current WowzaPlayer instance. For example:
```

```javascript
metadataListener = function ( metaDataEvent ) {
    console.log('Packet Time: ' + metaDataEvent.packetTime);
    console.log('Time: ' + metaDataEvent.time);
    console.log('Type: ' + metaDataEvent.type);
};
myPlayer.onMetadata( metadataListener );
```

```javascript
pauseListener = function ( pauseEvent ) {
    console.log('Time: ' + pauseEvent.time);
};
myPlayer.onPause( pauseListener );
```
onPlay, removeOnPlay

onPlay (callback : function (playEvent : {})) : WowzaPlayer

onPlay attaches an event listener function that’s called when the player starts playing and returns the current WowzaPlayer instance.

callback is a function that receives a playEvent object literal with one property:

   playEvent : { time : number }

Where time is the current playhead time in milliseconds. For example:

```javascript
playListener = function (playEvent) {
    console.log('Time: ' + playEvent.time);
};
myPlayer.onPlay(playListener);
myPlayer.removeOnPlay(playListener);
```

removeOnPlay (callback : function () ) : WowzaPlayer

removeOnPlay removes the requested event listener function and returns the current WowzaPlayer instance. For example:

```javascript
myPlayer.removeOnPause(pauseListener);
```

**Note:** See [Show loading status during time to first frame with the Wowza Player JavaScript API](#) for a code example using the onPlay method.

onPlaybackAttempt, removeOnPlaybackAttempt

onPlaybackAttempt (callback : function (playbackEvent : {})) : WowzaPlayer

onPlaybackAttempt attaches an event listener function that’s called when the player attempts to play a source from a sourceURL and returns the current WowzaPlayer instance. During a playback attempt, Wowza Player attempts to connect to a sourceURL a maximum of three times. If more than one sourceURL is listed in a sources object, Wowza Player will fall back to the next sourceURL if playback for an initial source fails. For an example of fallback behavior logged to the console, see [Fall back to a second source with the Wowza Player](#)
JavaScript API. You can use the `onPlaybackAttempt` method with the `onPlaybackFailure` and `onPlaybackSuccess` methods to monitor whether a playback event succeeds.

`callback` is a function that receives a `playbackEvent` object literal with one property:

```
playbackEvent : { url : string }
```

Where `url` is the source URL that Wowza Player is attempting to play. For example:

```javascript
removeOnPlaybackAttempt( callback : function () ) : WowzaPlayer
```

`removeOnPlaybackAttempt` removes the requested event listener function and returns the current `WowzaPlayer` instance. For example:

```javascript
myPlayer.removeOnPlaybackAttempt (playbackAttemptListener);
```

`onPlaybackFailure, removeOnPlaybackFailure`

```javascript
onPlaybackFailure ( callback : function (playbackEvent : {}) ) : WowzaPlayer
```

`onPlaybackFailure` attaches an event listener function that's called when the player fails to play a source from a `sourceURL` and returns the current `WowzaPlayer` instance. If more than one `sourceURL` is listed in a `sources` object, Wowza Player will fall back to the next sourceURL after playback for an initial source fails. For an example of fallback behavior logged to the console, see Fall back to a second source with the Wowza Player JavaScript API.

`callback` is a function that receives a `playbackEvent` object literal with one property:

```
playbackEvent : { url : string }
```

Where `url` is the source URL that Wowza Player failed to play. For example:

```javascript
playbackFailureListener = function ( playbeackEvent ) {
    console.log('Failure: ' + playbackEvent.url);
};
myPlayer.onPlaybackFailure( playbackFailureListener );
```
removeOnPlaybackFailure( callback : function () ) : WowzaPlayer

**removeOnPlaybackFailure** removes the requested event listener function and returns the current **WowzaPlayer** instance. For example:

```javascript
myPlayer.removeOnPlaybackFailure (playbackFailureListener);
```

**onPlaybackSuccess, removeOnPlaybackSuccess**

**onPlaybackSuccess** ( callback : function (playbackEvent : {}) ) : WowzaPlayer

**onPlaybackSuccess** attaches an event listener function that's called when the player succeeds in connecting to a source from a **sourceURL** and returns the current **WowzaPlayer** instance. See Fall back to a second source with the Wowza Player JavaScript API for a code example with **onPlaybackSuccess**.

**callback** is a function that receives a **playbackEvent** object literal with one property:

```javascript
playbackEvent : { url : string }
```

Where **url** is the source URL that Wowza Player successfully connects to. For example:

```javascript
playbackSuccessListener = function (playbackEvent) {
  console.log('Success: ' + playbackEvent.url);
};
myPlayer.onPlaybackSuccess (playbackSuccessListener);
```

```javascript
myPlayer.removeOnPlaybackSuccess (playbackSuccessListener);
```

**removeOnPlaybackSuccess**

**removeOnPlaybackSuccess**( callback : function () ) : WowzaPlayer

**removeOnPlaybackSuccess** removes the requested event listener function and returns the current **WowzaPlayer** instance. For example:

```javascript
removeOnPlaybackSuccess( callback : function () ) : WowzaPlayer
```

**onPlayheadTime, removeOnPlayheadTime**

**onPlayheadTime** ( callback : function (playheadTimeEvent : {}) ) : WowzaPlayer

**onPlayheadTime** attaches an event listener function that's called at 500-millisecond intervals throughout playback. Then, the method returns the current **WowzaPlayer** instance.
callback is a function that receives a `playheadTimeEvent` object literal that has one property:

```javascript
playheadTimeEvent : { time : number }
```

The `playheadTimeEvent` object has a property, `time`, which is the current playhead time (in milliseconds). For example:

```javascript
myPlayer.onPlayheadTime( playheadTimeListener );
myPlayer.removeOnPlayheadTime( playheadTimeListener );
```

`removeOnPlayheadTime` removes the requested event listener function and returns the current `WowzaPlayer` instance. For example:

```javascript
myPlayer.removeOnReady (readyListener);
```

**Note:** See [Show loading status during time to first frame with the Wowza Player JavaScript API](#) for a code example using the `onPlayheadTime` method.

**onReady, removeOnReady**

```javascript
onReady ( callback : function () ) : WowzaPlayer
```

`onReady` attaches an event listener that's called when the `WowzaPlayer` instance enters the `READY_TO_PLAY` state. If the player has already been in the `READY_TO_PLAY` state, the `callback` function will fire immediately. Then, it returns the current `WowzaPlayer` instance.

```javascript
removeOnReady ( callback : function () ) : WowzaPlayer
```

`removeOnReady` removes the requested event listener function and returns the current `WowzaPlayer` instance. For example:

```javascript
onResume, removeOnResume
```
onResume (callback : function (resumeEvent : {})) : WowzaPlayer

**onResume** attaches an event listener function that’s called when the player resumes playing from a paused state and returns the current WowzaPlayer instance.

**callback** is a function that receives a **resumeEvent** object literal with one property:

```javascript
resumeEvent : { time : number }
```

Where **time** is the current playhead time in milliseconds. For example:

```javascript
removeOnResume (callback : function () ) : WowzaPlayer
```

**removeOnResume** removes the requested event listener function and returns the current WowzaPlayer instance. For example:

```javascript
onSeek, removeOnSeek
```

onSeek (callback : function (seekEvent : {})) : WowzaPlayer

**onSeek** attaches an event listener function that’s called when the player seeks and returns the current WowzaPlayer instance.

**callback** is a function that receives a **seekEvent** object literal with two properties:

```javascript
seekEvent : { oldTime : number, newTime : number }
```

Where **oldTime** is the previous playhead time in milliseconds. **newTime** is the new playhead time in milliseconds. For example:

```javascript
resumeListener = function (resumeEvent) {
    console.log('Time: ' + resumeEvent.time);
};
myPlayer.onResume( resumeListener );
```

```javascript
removeListener = function () {
    console.log('Remove Resumed :: ');
};
myPlayer.removeOnPause(removeListener);
```

```javascript
seekListener = function (seekEvent) {
    console.log('Old time: ' + seekEvent.oldTime);
    console.log('New time: ' + seekEvent.newTime);
};
myPlayer.onSeek(seekListener);
```
removeOnSeek (callback : function () ) : WowzaPlayer

removeOnSeek removes the requested event listener function and returns the current WowzaPlayer instance. For example:

```javascript
myPlayer.removeOnSeek (seekListener);
```

onStateChanged, removeOnStateChanged

onStateChanged (callback : function (stateChangedEvent : {})) : WowzaPlayer

onStateChanged attaches an event listener function that’s called when the WowzaPlayer changes state. Then, the method returns the current WowzaPlayer instance.

callback is a function that receives a stateChangedEvent object literal with two properties:

```javascript
stateChangedEvent : { previousState : number, currentState : number }
```

Where previousState is the previous player state, and currentState is the new player state. For example:

```javascript
onStateChangedListener = function (stateChangedEvent) {
if (stateChangedEvent.currentState == WowzaPlayer.State.PLAYING) {
  console.log('Playing');
} else if (stateChangedEvent.currentState == WowzaPlayer.State.PAUSED) {
  console.log('Paused');
}
}
myPlayer.onStateChanged(onStateChangedListener);
myPlayer.removeOnStateChanged(onStateChangedListener);
```

removeOnStateChanged (callback : function (stateChangedEvent : {})) : WowzaPlayer

removeOnStateChanged removes the requested event listener function and returns the current WowzaPlayer instance. For example:

```javascript
onStateChangedListener = function (stateChangedEvent) {
if (stateChangedEvent.currentState == WowzaPlayer.State.PLAYING) {
  console.log('Playing');
} else if (stateChangedEvent.currentState == WowzaPlayer.State.PAUSED) {
  console.log('Paused');
}
}
myPlayer.onStateChanged(onStateChangedListener);
myPlayer.removeOnStateChanged(onStateChangedListener);
```

onStats, removeOnStats

onStats (callback : function (statsEvent : {})) : WowzaPlayer

```javascript
onStatsListener = function (statsEvent) {
  console.log(statsEvent); // do something with the stats
}
myPlayer.onStats(onStatsListener);
myPlayer.removeOnStats(onStatsListener);
```
**onStats** attaches an event listener function that’s called when statistics become available. Then, the method returns the current **WowzaPlayer** instance.

**callback** is a function that receives a **statsEvent** object literal. **statsEvent** has various properties describing HLS stream playback stats:

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalBytes</td>
<td>The cumulative total number of downloaded bytes.</td>
</tr>
<tr>
<td>totalDroppedFrames</td>
<td>The cumulative total number of dropped frames.</td>
</tr>
<tr>
<td>throughput</td>
<td>Calculated as lastBytesDownloaded / (lastDownloadDuration + lastParseDuration) * 1000.</td>
</tr>
<tr>
<td>bandwidth</td>
<td>Calculated as lastBytesDownloaded / (lastDownloadDuration * 1000).</td>
</tr>
<tr>
<td>bitrate</td>
<td>The current stream bitrate.</td>
</tr>
<tr>
<td>lastBytesDownloaded</td>
<td>The number of bytes in the last chunk.</td>
</tr>
<tr>
<td>lastDownloadDuration</td>
<td>The time spent downloading the last chunk (in milliseconds).</td>
</tr>
<tr>
<td>lastParseDuration</td>
<td>The time spent parsing the last chunk (in milliseconds).</td>
</tr>
</tbody>
</table>

For example:

```javascript
myPlayer.onStats(statsListener);
```
removeOnStats (callback : function (statsEvent : {})) : WowzaPlayer

**removeOnStats** removes the requested event listener function and returns the current **WowzaPlayer** instance. For example:

```javascript
myPlayer.removeOnStats(statsListener);
```

```javascript
stopListener = function (stopEvent) {
    console.log('Time: ' + stopEvent.time);
};
myPlayer.onStop(stopListener);
myPlayer.removeOnStop(stopListener);
```

**onStop, removeOnStop**

onStop (callback : function (stopEvent : {})) : WowzaPlayer

**onStop** attaches an event listener function that’s called when the player stops playing, commonly after a user interacts with a live stream. Then, the method returns the current **WowzaPlayer** instance.

**callback** is a function that receives a **stopEvent** object literal with one property:

```javascript
stopEvent : { time : number }
```

Where **time** is the current playhead time in milliseconds. For example:

```javascript
myPlayer.onStop(stopListener);
```

```javascript
myPlayer.removeOnStop(stopListener);
```

removeOnStop (callback : function () ) : WowzaPlayer

**removeOnStop** removes the requested event listener function and returns the current **WowzaPlayer** instance. For example:

```javascript
removeOnStop(callback);
```

**onVolume, removeOnVolume**

onVolume (callback : function (volumeEvent : {})) : WowzaPlayer

**onVolume** attaches an event listener function that’s called when the volume changes. Then, the method returns the current **WowzaPlayer** instance.

**callback** is a function that receives a **volumeEvent** object literal with two properties:

```javascript
volumeEvent : { volume : number, muted : boolean }
```

Where **volume** is the current volume in percentage. For example:

```javascript
volumeListener = function (volumeEvent) {
    console.log('Volume: ' + volumeEvent.volume);
};
myPlayer.onVolume(volumeListener);
myPlayer.removeOnVolume(volumeListener);
```
volumeEvent : { volume : number, muted : boolean }

Where volume is the current volume (a value from 0-100). muted is true when playback is muted and is false otherwise. For example:

```javascript
removeOnVolume ( callback : function ( volumeEvent : {} ) ) : WowzaPlayer

removeOnVolume removes the requested event listener function and returns the current WowzaPlayer instance.
```