ARD – Player

Technische Dokumentation

ID: APSP2001
Version: 1.3
Status: Final

Inhaltsverzeichnis

1. Änderungshistorie

Version	Datum	Autor	Erläuterung
0.1	20.02.2012	Kadir Uludag	Erstellung erster Version
0.2	25.02.2012	R. A.W. Neumann	Diverse Überarbeitungen
0.3	28.02.2012	Saban Ünlü	Diverse Überarbeitungen
0.4	02.03.2012	Kadir Uludag	Diverse Überarbeitungen, Kapitelreihenfolge geändert
0.5	02.03.2012	R. A.W. Neumann	Diverse Überarbeitungen
0.6	04.03.2012	Florian Diesner	Diverse Überarbeitungen, Formulierungen
0.7	05.03.2012	Kadir Uludag	Diverse Überarbeitungen, Definition der JSON-Dateien, Angabe der optionalen-,pflicht- & Standardparameter, PlayerModel-Beschreibung
0.8	27.04.2012	Kadir Uludag	Aktualisierung der PlayerConfiguration; Konfigurationseinstellungen des Flash-Players
0.8	27.04.2012	Florian Diesner	Erweiterung Conviva, Config
1.0	14.08.2012	Florian Diesner	Integrationsanleitung
1.1	05.10.2012	Florian Diesner	Erweiterung des MediaCollection Abschnitts um die Definition der möglichen CDN Parameter
1.2	14.11.2012	Florian Diesner	Spezielle Installationsanweisungen, setInitialPlayHead
1.3	21.01.2013	Florian Diesner	Entfernen von Playerinstanzen
1.4	22.01.2013	Florian Diesner	Interaktion mit Playern über Callbacks
1.5	08.03.2013	Florian Diesner	Erweiterung der Konfiguration, Mehrere Quell-URLs für HTML5
1.6	03.04.2013	Florian Diesner	Erweiterung um Skin-Konfiguration für Flash
1.7	16.04.2013	Florian Diesner	Erweiterung um Pixel- und Player-Event Anbindung
1.8	01.07.2013	Florian Diesner	BaseAssetUrl Parameter aufgenommen.
1.9	15.07.2013	Florian Diesner	Korrektur CSS Dateinamen
1.10	21.10.2013	Florian Diesner	Erweiterung der Player-Events, Optionale Parameter ergänzt
1.11	21.02.2014	Klaus Panster	Anpassung der Schnittstellen in der Player-Config und Handling für DefaultQuality

1.11	24.02.2014	Florian Diesner	Ersatzlose Entfernung des Windows-Media Formats
1.12	25.03.2014	Klaus Panster	Entfernung WindowsMedia, Overlay, Podcast und Download Spezifizierung setDVREnabled, Resp. Preview-Image
1.13	04.04.2014	Klaus Panster	Hinzufügen der Sola-Schnittstellenbeschreibung
1.14	16.05.2014	Klaus Panster	Erweiterung Hinweistext zu DVR-Timeshift Hinweis zur Nicht-Kompression der skin.css Hinweis zur Schließung der Sicherheitslücke in der Pixelung
1.2	29.07.2014	Klaus Panster	Umstellung auf jQuery 2.1.1 Kapselung JS Hinzufügen von Sprungmarken Erweitern der Screen-Reeder-Fähigkeit, Player-Div im VollbildmodusEntfernen der Skin.css (Flash-Skinning)
1.3	31.03.2016	Florian Diesner	Korrektur Schreibweise der Funktion setDVREnabled.

2. Einführung

Achtung:
Da im Player-Release 3.7 tiefgreifende Veränderungen (Kapselung JS-Namespace und Update jQuery) im Player vorgenommen wurden, wird dringend empfohlen, diese neu aufgesetzte Dokumentation zu lesen!!! Das Dokument bezieht sich nur noch auf Eigenschaften, die auch im Release 3.7 unterstützt werden. Ältere Versionen werden nicht berücksichtigt. Lesen Sie dazu bitte die entsprechenden Anleitungen der Vorgänger-Versionen.

Im vorliegenden Dokument werden die unterschiedlichen Vorgehensweisen zum Initialisieren des ARD-Players beschrieben. Der Player kann auf drei unterschiedlichen Wegen angepasst werden. So steht neben der programmatischen Methode, auch die Initialisierung über JSON-Objekte zu Verfügung. Zusätzlich können mit Hilfe von Query-Strings (GET-Parameter) bestimmte Zustände des Players aktiviert und deaktiviert werden. Darüber hinaus kann ebenfalls das Erscheinungsbild und die Initialisierung beeinflusst werden.

3. Installation

Im nachfolgenden Abschnitt werden die benötigten Schritte erläutert, um den Player in Ihre Webseiten zu integrieren. Bitte beachten Sie, dass es sich hierbei lediglich um einen Leitfaden handelt. Die endgültige Integration kann, abhängig des Ziel-Setups von dem Leitfaden abweichen. Dies betrifft insbesondere eingesetzte JS-Bibliotheken von Dritten, und CSS-Namensräume.

Exemplarische Installationen sind im Release zu finden:

- index.html
- index.min.html

3.1. CSS

Im Release-Ordner /css sind alle CSS-Dateien vorhanden, die für eine korrekte Darstellung des Players benötigt werden:

CSS-Dateien in folgender Reihenfolge einbinden:

1. player.css
    a. Basis-Klasse, immer einbinden
    b. link rel
2. *** Mandanten ***.css
    - Ergänzende und überschreibende Skin-Klassen, individuell einbinden
    - link rel

Um einen standardkonformen Modus zu erreichen, ist optimaler weise ein **html5-doctype** zu setzen. Bei Verwendung eines anderen doctype kann es insbesondere bei älteren Browsern zu Abweichungen in der Darstellung kommen, die dann individuell pro Integrationsinstanz zu korrigieren sind. Die Korrektur dieser Abweichungen kann in den jeweiligen Mediatheks-Klassen per css-Hack erfolgen oder als Basiskorrektur in die bereits vorhandenen Korrekturklassen übernommen werden.

3.2. JavaScript

Im Release-Ordner /js sind alle JavaScript-Dateien vorhanden, die für eine korrekte Funktionsweise des Players benötigt werden. Bei der Integration gibt es dabei für die Player-Core-Bibliotheken zwei Möglichkeiten:

- Komprimierte und minimierte Versionen
    - js/base.min.js
    - js/addons.min.js
- Quelloffene, lesbare Version
    - js/com/*

Zeitgleich darf jeweils nur eine der beiden Varianten importierte werden, ansonsten ist mit Fehlfunktionen zu rechnen.

Neben den Core-Klassen werden folgende, externe Bibliotheken benötigt.
Bitte beachten Sie dass jQuery 2.1.1 nicht mit älteren Player- und jQuery-Versionen kompatibel ist!!!

- js/libs/jquery-2.1.1.min.js
    - jQuery ab Version >= 2

- js/libs/ jquery.tools-1.2.7.min.js
  - jQuery Tools
- js/libs/ jquery-ui-1.10.4.custom.min.js
  - Angepasste jQuery UI
- js/libs/ swfobject.js
  - SWF Object
- js/libs/ mm.useractivity.js
  - Erfassung der Mausaktivität
- js/libs/ json2.js
  - JSON für ältere Browser
- js/libs/ jquery.cookie.js
  - jQuery Cookie
- js/libs/ a11y-slider.ext.js
  - Slider für Lautstärkeregler
- js/libs/ shortcut.js
  - Tastatureingaben

Die Einbindung der Importe sollte so erfolgen:

```
<script src="base/js/libs/jquery-2.1.1.min.js"></script>
<script src="base/js/libs/jquery.tools-1.2.7.min.js"></script>
<script src="base/js/libs/jquery-ui-1.10.4.min.js"></script>
<script src="base/js/libs/swfobject.js"></script>
<script src="base/js/libs/mm.useractivity.js"></script>
<script src="base/js/libs/json2.js"></script>
<script src="base/js/libs/jquery.cookie.js"></script>
 <script src="base/js/libs/shortcut.js"></script>
<script src="base/js/libs/bigscreen.min.js"></script>
```

Bei dieser Variante wird eine ggf. bereits vorangegangene jQuery-Version in eine Variable geschrieben und mit den aktuellen Versionen überschrieben. Nach der Einbindung aller Bibliotheken wird die gemerkte jQuery-Version der Seite zur weiteren Verwendung wieder „zurückgegeben“.

3.2.1. JS-Kapselung

Der Player verfügt über eine spezielle Kapselung via separaten Namespace.
Diese Kapselung gewährt eine Immunität gegenüber unsauberen Anweisungen auf der jeweils eingebundenen Seite und sieht strukturell so aus:

```
(function ($) {

	$(window).ready(function () {
		var p = new ardplayer.Player("player00", pc, mc);

	});
}(ardplayer.jq.noConflict()))
```

Nach außen bleiben die Klassen des Players mit der Anweisung **ardplayer** weiterhin ansprechbar.

3.3. Spezielle Installationsanweisungen

In einigen Sonderfällen kann es notwendig sein, einen Player mittels Javascript-Schnittstelle wieder vollständig zu entfernen. Hierfür sind folgende Methoden vorgesehen:

- Player-Instanz
  - *dispose()* – Entfernt alle zugehörigen Elemente, und gibt Sie zur Gabadge-Collection frei. Diese Funktion muss zwingend aufgerufen werden, bevor das zugehörige DOM-Element (DIV) entfernt wird.
- GlobalModel, ErrorController, ViewController, PlayerModel, Cookie und QueryParser:
  - resetSingleton() – Setzt die entsprechende Klasse in den Urzustand zurück, und gibt alle referenzierten Objekte frei.

Ein Beispiel zur konkreten Anwendung finden Sie in der im Release enthaltenen Datei „index_config.html“.

Die Methode „removeArdPlayer“ unterstützt die Löschung eines einzelnen, oder einer Vielzahl von Playern inkl. DOM gleichzeitig:

- removeArdPlayer() oder removeArdPlayer(„all“)
  - Löscht sämtliche Player-Instanzen, die auf der aktuellen Seite erzeugt wurden.
- removeArdPlayer(„player01“)
  - Löscht einen einzelnen Player, der im DOM die ID „player01“ besitzt.
- removeArdPlayer(„player01“, „player02“)
  - Löscht mehrere Instanzen gleichzeitig, welche anhand der übergebenen IDs identifiziert werden.

3.4. Weitere Assets

Neben den CSS und JS Elementen werden folgende Ordner für eine korrekte Installation benötigt:

- img/
  Bilder und Sprite-Sheets, die vom Player eingebunden werden
- plugins/
  Ergänzende Module (z.B. Flash-Plugin), die zur Wiedergabe von Medien benötigt werden.
- Addons/
  Erweiterungen, wie z.B. UntertitelAddon, die zusätzlich zum Kernplayer geladen werden können

4. Programmatisch

Aufruf des Players mit programmatisch erstellten MediaCollection- und PlayerConfiguration-Objekten.

4.1. Beschreibung

Um die Abwärtskompatibilität zur vorhergehenden Version des ARD-Players beizubehalten, können Player nach wie zuvor mit Hilfe von JavaScript erstellt werden. Die URL, und mit ihr verbundene mögliche Querys, werden nicht verarbeitet, wenn der Player über eine gültige PlayerConfiguration sowie eine MediaCollection verfügt.

4.2. Definition zu den grundsätzlich genutzten Javascript-Klassen

4.2.1. MediaCollection

Die MediaCollection-Klasse beinhaltet alle Attribute des wiederzugebenden Mediums. Vom logischen Aufbau her entspricht es den Anforderungen an den Player, **einen** Inhalt in **mehreren** Qualitätsstufen und für **mehrere** Plugins zu Verfügung zu stellen.

Die MediaCollection kann somit inhaltlich **ein** Video in geringer, mittlerer, hoher und sehr hoher Qualität beinhalten. Jede dieser Qualitätstufe kann dann wiederum für Flash und HTML5 ein Plugin zu Verfügung gestellt werden.

Zusätzlich beinhaltet die MediaCollection aber unter anderem auch den, Vorschaubilder, Audiohintergründe, Kapitelmarken oder den Untertitelpfad.

4.2.2. PlayerConfiguration

Die PlayerConfiguration-Klasse verwaltet alle Konfigurationsmöglichkeiten des Players. Das beinhaltet neben der Wiedergabegröße eines Videos, der Autoplay-Funktion, unter anderem auch das ein- und ausblenden der Bedienelemente, das automatische Speichern der Wiedergabeeinstellungen, usw.

4.2.3. Player

In der Player-Klasse laufen MediaCollection und PlayerConfiguration zusammen. Ein Playerobjekt hat die Aufgabe den Player auf einer HTML-Seite so zu integrieren, das die dem Benutzer angebotenen Medien, Client- (Plugins) und Serverseitig (hinterlegte Quellen) miteinander korrespondieren.

Außerdem wird hier das Objekt bereitgestellt, welches für die Darstellung der Bedienelemente, das Umschalten zwischen Qualitäten und Plugins verantwortlich ist.

4.2.4. PlayerModel

Das PlayerModel wird für jeden einzelnen Player erstellt. In dieser Klasse kann man die Werte für diverse Eigenschaften setzen und erhalten. Diese Eigenschaften sind:

- Metadaten → Metadaten für den laufenden Stream
- Time → Zeit des laufenden Streams
- Duration → Gesamtdauer des laufenden Streams

Des Weiteren bietet sie unterschiedliche Funktionen an.

Diese statischen Methoden sind:

- getInstanceByPlayerID (playerID)-Methode: Zugriff auf die anderen Playerinstanzen, die sich mit dem referenzierten Player auf der Bühne befinden. Als Parameter wird die PlayerID-Eigenschaft verlangt, auf die zugegriffen werden soll.
- getAllPlayer(exceptPlayer)–Methode: Gibt ein Array zurück, das alle Player-Objekte beinhaltet. Ausgenommen dem Player der in der exceptPlayer-Eigenschaft definiert wurde.
- getPlayingPlayer(exceptPlayer)–Methode: gibt alle Player, zusammengefasst in einem Array, zurück die aktuell einen Stream abspielen. Ausgenommen dem Player der in der exceptPlayer-Eigenschaft definiert wurde.

4.3. Klassenbeschreibung und Abhängigkeiten

Im Folgenden werden die einzelnen Klassen und deren Zusammenspiel für die wichtigsten Elemente beschrieben.
Die Funktionsweise der nicht beschriebenen Elemente, sind als Inline-Kommentare den entsprechenden Quell-Dateien zu entnehmen.

4.3.1. MediaCollection

Die MediaCollection-Klasse beinhaltet alle Informationen die ein abzuspielendes Medium betreffen. Ein MediaCollection-Objekt wird dem Player-Objekt übergeben.

Instanziierung

```
new MediaCollection(type, isLive, defaultQuality_arr);
```

Attribute

- type. Typ: String: Abzuspielendes Medium (video/audio)
- isLive. Typ: Boolean: Gibt an, ob es sich um einen Live-Stream handelt.
- defaultQuality. Typ: Number oder Array: Die Standard-Qualität für die in der MediaCollection hinterlegten Qualität des Streams. 0 - s, 1 - m, 2 - l, 3 – xl

  Achtung: Ab R.3.6 kann auch ein Array übergeben werden, in dem die

bevorzugt anzuwählenden Qualitätsstufen vom Player abzuarbeiten sind, falls die erste gewünschte Qualität nicht abspielbar ist.
Wird dennoch nur eine Qualitätsstufe angelegt, so wird bei erfolglosem Wiedergabeversuch die im Player hinterlegte Standard-Reihenfolge **[„auto“, 0, 1, 2,3]** verwendet.

Methoden

```
void setSortierung = function (value1, value2)
```

Ändert die Sortierreihenfolge der Plugin-Abfrage (Standard: flash(0), html(1))

Parameter

- value1. Typ: Number: 0 - Flash, 1 - HTML5
- value2. Typ: Number: 0 - Flash, 1 - HTML5
- value3. Typ: Number: 0 - Flash, 1 - HTML5

```
void setAudioImage(url/Objekt);
```

Verlangt eine URL zu einem Bild, welches innerhalb des Audio-Player dargestellt werden soll.

Achtung ab dem R.3.6 ist es möglich, innerhalb eines Objekts, je Repräsentationsgröße eine separate URL zu übergeben! Wird dennoch nur eine URL angegeben, wird nur diese Verwendet.

Parameter

- url. Typ: String. Pfad zu dem Bild. Oder:
- objekt Beispiel: {s:“pfad_zu_s.png“, m:“pfad_zu_m.png“}

```
void setPreviewImage(url/Objekt);
```

Verlangt eine URL zu einem Bild, welches im Player anzeigt wird, wenn er sich im Status „Stop“ befindet. Ausgenommen ist der Audio-Player.

Achtung ab dem R.3.6 ist es möglich, innerhalb eines Objekts, je Repräsentationsgröße eine separate URL zu übergeben! Wird dennoch nur eine URL angegeben, wird nur diese Verwendet.

Parameter

- url. Typ: String. Pfad zu dem Bild. Oder:
- objekt Beispiel: {s:“pfad_zu_s.png“, m:“pfad_zu_m.png“}

```
void setDVREnabled(bool);
```

Verlangt booleschen Wert zur De-/ Aktivierung der DVR-Timeshift-Funktion im Player. Ausgenommen ist der Audio-Player.

Achtung: Voraussetzung für die korrekte Aktivierung der Funktion ist die Unterstützung des eingebundenen Streams! Grundsätzlich darf die Funktion nur bei Livestreams aktiviert (true) sein!

Parameter

- bool. Typ: Boolean. Flag, ob DVR aktiviert wird.

```
void mc.addMedia(plugin);
```

Übermittelt dem Player, dass ein Clip für ein bestimmtes Plugin verfügbar ist. Das Ausführen dieser Funktion erzeugt ein neues Objekt der Media-Klasse.

Parameter

- plugin. Typ: Number. 0 - Flash, 1 - HTML5

```
void addMediaStream(plugin, quality, server, stream, cdn );
```

Fügt dem in der MediaCollection gehaltenen Media-Objekt eine Quelle für eine bestimmte Qualitätsstufe hinzu. So kann dem Player übermittelt werden, welche Qualitätsstufen für ein Plugin verfügbar sind. Zusätzlich wird als letzter Wert ein Content Delivery Network für diese Datei verlangt.

Parameter

- plugin. Typ: Number. 0 - Flash, 1 - HTML5
- quality. Typ: Number. („auto“, 0, 1, 2, 3)
- server. Typ: String. Pfad auf dem Server.
- stream. Typ: String / Array. Pfad zu der Datei auf dem Server oder Liste von Dateien zur mehrfachen Quelldefinition (HTML5-Player, Beispiel siehe unten).
- cdn Typ: String. Definiert den CDN Namen. Standardwert ist default. Die möglichen Werte können in der Konfiguration des Flash-Plugins hinterlegt werden.

Grundsätzlich wird die CDN-Angabe nur bei der Flash-Wiedergabekomponente benötigt. Wird ein anderes Plugin gewählt, wird dieser Parameter ignoriert und in der Mediacollection freigelassen. Folgende Parameter sind zum Zeitpunkt der Erstellung der Dokumentation vorhanden:

- „default“ – Verwendung bei SMIL, TV1, Level3, HTTP**, Audio OnDemand**
- „akamai“ – Verwendung bei Akamai-Streams (RTMP, HTTP, ...)
- „limelight“ – Verwendung bei Limelight-Streams (RTMP, HTTP, ...)
- „icecast“ – Verwendung bei mp3-Live-Streams (icecast/shoutcast)
- „conviva“ – Verwendung bei Conviva-Konfigurationen (Sonderfall)

Wichtig: Bei Audio-OnDemand-Streams muss immer „default“ verwendet werden, da sonst keine Gewähr auf das Auslesen der Metadaten besteht (z.B. keine Duration-Angabe).

Für den HTML5-Player besteht die Möglichkeit mehrerer Quell-Pfade je Qualität (via Array) zu definieren.

Der Aufbau der MediaCollection sieht für den HTML5-Teil wie folgt aus:

```
mc.addMediaStream(1, „auto“, "",
                [
                   "http://sample.url/stream_ios.m3u8",
                   "http://sample.url/stream.mp4"
                ],
mc.addMediaStream(1, 0, "",
                [
                   "http://sample.url/stream_ios.m3u8",
                   "http://sample.url/stream.mp4"
                ],

                "default");

mc.addMediaStream(1, 1, "",
                [
                   "http://sample.url/stream_ios.m3u8",
                   "http://sample.url/stream.mp4"
                ],
                "default");

mc.addMediaStream(1, 2, "",
                [
                   "http://sample.url/stream_ios.m3u8",
                   "http://sample.url/stream.mp4"
                ],
                "default");
```

Es ist somit möglich, unter **„auto“** adaptive Streaming-Sets, wie neu im AV-Standard
beschlossen, mit zu integrieren.
Des Weiteren ist es natürlich auch möglich, das jeweilige Array mit anderen Formaten (ogg, webm, etc.) zu erweitern. z.B. so (entspricht nicht AV-Standard):

```
mc.addMediaStream(1, 0, "",
                [
                   "http://sample.url/stream_ios.m3u8",
                   "http://sample.url/stream.mp4",
                   "http://sample.url/stream.ogg",
                   "http://sample.url/stream.webm"
                ],
                "");
```

Jedes Element des Arrays wird als src-tag in den audio-/video-tag

hintereinander angelegt. Lediglich die Streaming-Types sind im Player direkt als Referenz hinterlegt und werden als type entsprechen mit in das audio-/video-tag hinter das jeweilige src-tag gerendert (zur Performance-Steigerung).
Damit verhält sich der Player HTML5-konform, indem er die einzelnen Sourcen in einem audio-/video-tag hinterlegt und der jeweilige Browser sich den ersten Stream nimmt, den er abspielen kann.
Dem entsprechend sollte auch auf die oben angegebene Reihenfolge zur Stream-Einbindung geachtet werden. Eine 100%-Garantie kann es aber leider dafür nicht geben, da wir hier absolut auf die internen Vorgehensweisen des Browser angewiesen sind (sollte ja auch der Clou bei HTML5 sein).

4.3.2. Media

Die Media-Klasse hält alle Informationen über die Clip-Quellen eines Plugins. Das Media-Objekt selbst beinhaltet mehrere MediaStream-Objekte, eines für jede Qualität.

4.3.3. MediaStream

Im MediaStream werden alle Informationen zu einer einzelnen Qualitätsstufe eines Plugins hinterlegt.

4.3.4. Chapters

Achtung: ab dem **Release 3.7** ist es möglich, dem Player auf verschiedenen Wegen Kapitelmarken mit zu geben. Ausführliche Informationen dazu erhalten Sie im Dokument **ARD-Player-Dokumentation-Sprungmarken-XXX.pdf**

4.3.5. PlayerConfiguration

Um Unterschied zur MediaCollection-Klasse hält die PlayerConfiguration-Klasse alle Informationen die unabhängig vom Medium sind und nur die Einstellung des Players betreffen.

Instanziierung

```
new PlayerConfiguration();
```

Methoden

```
void setInitialPlayhead (position);
```

Setzt die Initiale Abspielposition in Sekunden.

Parameter

- `position`. Typ: Number. Abspielposition in Sekunden.

```
void setRepresentation(representationClass);
```

Stellt die Darstellung auf ein Qualitätsniveau ein.

Parameter

- `representationClass`. Typ: String. Optionale Klasse, die der Repräsentation angehangen wird.

```
void setAutoSave(bool);
```

Setzt die automatische Speicherung der aktuellen Qualitätsstufe im Cookie. Das Cookie wird bei jedem Qualitäts- und Pluginwechsel gesichert.

Parameter

- bool. Typ: Boolean.

```
void setAutoPlay(bool/string);
```

Setzt das Flag, ob der Inhalt automatisch abgespielt wird. Bei „true“ wird die Variable `_autoplay` auf den Stringwert: "AUTOPLAY" gesetzt. Bei einem „false“ wird die Variable `_autoplay` auf den Stringwert: "NOTHING"

Wert	Auswirkung
true	Die **1.** Instanz aller eingebundenen Player beginnt direkt mit der Wiedergabe
false	Keine automatische Wiedergabe
„FORCE“	Die **letzte** Instanz aller eingebundenen Player beginnt direkt mit der Wiedergabe

Parameter

- bool/string. Typ: Boolean/String.

```
void setSubtitleUrl (subtitleUrl, subtitleOffset)
```

Verlangt eine URL zu einem XML, welches in Form eines Untertitels innerhalb des Player anzeigt wird. Ausgenommen hiervon ist der Audio-Player.

Parameter

- subtitleUrl Typ: String. Relativer Pfad zu der Untertitel-XML-Datei

- subtitleOffset Typ: Number. Verschiebung (in Sekunden) Untertitel

angezeigt werden sollen.

```
void setNoSubtitelAtStart(bool);
```

Dieser Parameter ermöglicht die automatische Anzeige der Untertitel.

Achtung: für eine bessere Integration/Verständlichkeit lautet der JSON-Parameter: `_showSubtitelAtStart` und ist invertiert (true = false)

Parameter

- bool. Typ: Boolean.

```
void setRememberCurrentTime(bool);
```

Verhindert oder erlaubt, dass die aktuelle Spielzeit bei einem Qualitäts-wechsel gespeichert wird.

Parameter

- bool. Typ: Boolean.

```
void setShowOptions(bool);
```

Verhindert oder erlaubt die Darstellung des Options-Layers innerhalb des Players, mit dessen Hilfe die kompletten Optionen für den Player vom Endbenutzer eingestellt werden können.

Parameter

- bool. Typ: Boolean.

```
void setShowOptions_Plugins(bool);
```

Verhindert oder erlaubt die Darstellung der Plugins (Flash/HTML5) innerhalb des Options-Layers des Players, mit dessen Hilfe die kompletten Optionen für den Player vom Endbenutzer eingestellt werden können.

Parameter

- bool. Typ: Boolean.

```
void setShowOption_Quality(bool);
```

Verhindert oder erlaubt die Darstellung der Qualitätsstufen innerhalb des Options-Layers des Players, mit dessen Hilfe die kompletten Optionen für den Player vom Endbenutzer eingestellt werden können.

Parameter

- bool. Typ: Boolean.

```
void setStartStopTime (start, stop);
```

Die hier übergeben Werte werden mit dem Aufruf von `setStartStopTime` innerhalb der automatisch erstellten PlayerConfiguration-Instanz verarbeitet. Mit diesem Parameter wird die zeitliche Spanne (Start- & Endzeit für den Stream) definiert.

Parameter

- start. Typ: Number. Startzeit in Sekunden
- stop. Typ: Number. Endzeit in Sekunden (optionaler Parameter)

Wird nur ein Parameter übergeben, so wird im Player nur die Startzeit gesetzt.

```
void setShowColorSettings (bool);
```

Gibt an, ob die Farbeinstellungen **für Flash** angezeigt werden sollen. Für HTML5 steht diese Option nicht zur Verfügung.

Parameter

- bool. Typ: Boolean. Zeigt die Einstellungen, wenn auf true

```
void setForceControlBarVisible (bool);
```

Gibt an, ob die Controlbar dauerhaft eingeblendet werden soll.

Parameter

- bool. Typ: Boolean. True deaktiviert das Ausfaden.

```
void setSolaAnalyticsEnabled(bool);
```

Parameter

- bool. Typ: Boolean. True aktiviert das Sola-Plugin.

```
void setSolaAnalyticsConfig("url"/obj);
```

Parameter

- URL. Typ: String. Übergabe der Sola-Konfig per URL. Oder ab R.3.6.2
- OBJ. Typ: JSON-OBJ. Übergabe der Sola-Konfig als JSON-Object.

Achtung: Weitere Informationen zur Sola-Konfiguration finden Sie in der ARD-Player-Sola-Kurz-Referenz im Ordner docs!

```
void setChaptersEnabled(funcRef);
```

Parameter

- bool. Typ: Boolean. True deaktiviert die Kapitel- bzw. Sprungmarken.

```
void setOnStatusChangeFuncton (funcRef);
```

Definiert ein Callback, welches bei Statuswechsel aufgerufen wird. Die Signatur der Callbackfunktion ist *callback(status)*. Die Liste der möglichen Status umfasst

- init – Div-Gerüst des Players wurde konstruiert
- ready – Player-Instanz wurde vollständig initialisiert (bei Flash erst nach Interaktion oder bei Autoplay).
- play – Die Player-Instanz beginnt mit dem Abspielvorgang
- pause – Die Player-Instanz wurde pausiert
- stop – Die Player-Instanz wurde gestoppt, der Abspielkopf springt zum Anfang zurück

Parameter

- funcRef. Typ: Function. Referenz auf die Callback-Funktion

```
void setOnMediaFinishedFunction (funcRef);
```

Definiert ein Callback, welches beim Erreichen des Clipendes aufgerufen wird.

Siehe auch: 4.5 Player-Events

Parameter

- funcRef. Typ: Function. Referenz auf die Callback-Funktion

```
void setOnTimeUpdateFunction (funcRef);
```

Definiert ein Callback, welches bei Aktualisierung des Abspielkopfes aufgerufen wird. Die Signatur der Funktion ist *callback(number).*

Siehe auch: 4.5 Player-Events

Parameter

- funcRef. Typ: Function. Referenz auf die Callback-Funktion

```
void setOnQualityChangeFunction (funcRef);
```

Definiert ein Callback, welches bei Änderung der Qualitätsstufe aufgerufen wird. Die Signatur der Funktion ist *callback(qualityId = „xl“, „m“, ...).*

Parameter

- funcRef. Typ: Function. Referenz auf die Callback-Funktion

```
void setOnPluginChangeFunction (funcRef);
```

Definiert ein Callback, welches bei Änderung des Wiedergabeplugins aufgerufen wird. Die Signatur der Funktion ist *callback(PluginId),* wobei die PluginId eine Konstante der Klasse *GlobalModel* ist (GlobalModel.FLASH, GlobalModel.HTML).

Siehe auch: 4.5 Player-Events

Parameter

- funcRef. Typ: Function. Referenz auf die Callback-Funktion

```
void setOnErrorCallbackFunction (funcRef);
```

Definiert ein Callback, welches bei Auftreten eines Wiedergabefehlers aufgerufen wird. Die Signatur der Funktion ist *callback(Kritisch-bool, Fehlerbeschreibung-String)*.

Parameter

- funcRef. Typ: Function. Referenz auf die Callback-Funktion

Gibt an, in welchem HTML-Element (jQuery-Identifier) das modale Fenster dargestellt wird. Standardmäßig wird hier „body“ verwendet.

Parameter

- jQueryString. Typ: String. Verweis auf den Element-Namen, der als Target für das modale Fenster dienen soll.

4.4. PlayerPixelController (Verpixelungs-Events)

Um eine statistische Erfassung zu vereinfachen, bietet der Player die Möglichkeit über Events Benutzerinteraktionen zu erfassen. Dafür steht die Kasse „PlayerPixelController“ zur Verfügung. Die Auslösung der Events unterliegt speziellen Bedingungen, die im Dokument „ARD-Player-Dokumentation-Pixelung-XXX.pdf“ aufgeführt sind.

Achtung:

Ab der Version 3.6.2 ist aufgrund der Schließung einer Sicherheitslücke die Pixelung via **JSON** nicht mehr möglich!

4.5. Player-Events

Eine Alterative zu den PlayerPixelController Events ist es, direkt die Player-Events abzugreifen. Diese unterliegen keinen gefilterten Bedingungen (nur einmalige Ausführung o.Ä.) und müssen vom Entwickler ggf. aufbereitet werden.

Mögliche Aktionen sind:

- Player. EVENT_INIT
  - Initialisierungsphase des Players beginnt.
- Player. EVENT_READY
  - Die Initialisierung des Players ist abgeschlossen.
- Player. EVENT_ERROR
  - Bei der Wiedergabe ist ein Fehler aufgetreten.
- Player.EVENT_LOAD_STREAM
  - Ladevorgang des Streams beginnt
- Player.EVENT_PLAY_STREAM
  - Abspielvorgang des Streams beginnt
- Player.EVENT_END_STREAM
  - Streamende wurde erreicht
- Player.EVENT_PAUSE_STREAM

  - Stream wurde pausiert
- Player.EVENT_STOP_STREAM
  - Stream wurde angehalten
- Player.EVENT_UPDATE_STREAM_TIME
  - Abspielzeit aktualisiert. Im Event wird die aktuelle Zeit als Timecode übergeben
- Player.VIEW_RESTORE_DEFAULTS
  - Die View wird in den Ursprungszustand zurückgesetzt. Das passiert, wenn zwischen Repräsentationen oder Plugins gewechselt wird.
- Player.SETUP_VIEW
  - Die Initialisierungsphase der View beginnt
- Player.SETUP_VIEW_COMPLETE
  - Die Initialisierung der View ist abgeschlossen.

Anbindung

```
$(player).bind(
   Player. SETUP_VIEW,
   function (event) {
      console.log(„Event ausgelöst“);
   }
);
```

5. Aufruf des Players mit JSON-Dateien

Im Folgenden wird der Aufruf des Players mit vorhandener MediaCollection- und PlayerConfiguration-JSON-Dateien, in einer definierten URL beschrieben.

5.1. Beschreibung

Jeder Player der auf einer Seite angelegt ist, kann von außen mit MediaCollection- und PlayerConfiguration versorgt werden. Hierfür dienen JSON-Dateien als Informations-Container..

5.2. Aufbau der PlayerConfiguration-JSON-Datei

Die PlayerConfiguration-JSON-Datei funktioniert als ein einzelnes Objekt. In dessen weiterer Benutzung werden alle weiteren Objekte und Parameter in geschweifte Klammern geschrieben. Zusätzlich werden diese mit einem Komma voneinander getrennt. Der zu definierende Parameter wird in Anführungszeichen geschrieben. Im Anschluss daran folgt ein Doppelpunkt der den Wert des Parameters angibt.

Bsp:
{
„Parameter“:Parameterwert,
„Parameter“:Parameterwert
}

Innerhalb eines Parameters können weitere Parameter-Objekte verschachtelt werden. In der PlayerConfiguration-JSON handelt es sich um ein Objekt vom Typ: _representationArray. Dieses Objekt definiert die Repräsentations-Darstellung des Players. Hierfür wird die `setRepresentation` Methode der automatisch erstellten PlayerConfiguration verwendet.

Parameter

```
{"_representationArray":[
    {
        "_representationClass":"m"
    }
],
```

Im Anschluss daran, werden optionale Parameter definiert, die auch für die Darstellung, bzw. das Verhalten des Players dienen.

```
"_startTime":10,                    (Siehe: setStartStopTime)
"_endTime":200,                     (Siehe: setStartStopTime)
"_autoplay":"AUTOPLAY",             (Siehe: setAutoPlay)
```

```
"_autosave":false,                          (Siehe: setAutoSave)
"_showOptions":true,                        (Siehe: setShowOptions)
"_showOptions_Plugins":true,                (Siehe: setShowOptions_Plugins)
"_showOptions_Quality":true,                (Siehe: setShowOptions_Quality)
"_rememberCurrentTime":false,               (Siehe: setRememberCurrentTime)
"_showSubtitelAtStart":true                 (Siehe: setNoSubtitelAtStart)
„_chaptersEnabled“:true,                    (Siehe: setNoSubtitelAtStart)
```

5.3. Aufbau der MediaCollection-JSON-Datei

Auch die MediaCollection-JSON-Datei ist als einzelnes Objekt zu verwenden, in dem alle weiteren Objekte und Parameter in geschweifte Klammern geschrieben und mit einem Komma voneinander separiert werden. Der zu definierende Parameter wird in Anführungszeichen geschrieben. Im Anschluss daran folgt ein Doppelpunkt der den Wert des Parameters angibt.

Bsp:

```
{
   „Parameter“:Parameterwert,
   „Parameter“:Parameterwert
}
```

Innerhalb des MediaCollection-JSON-Objektes werden die Parameter definiert die alle Informationen zu einem abzuspielenden Medium beinhalten.

```
"_type":"video"                             (Siehe: MediaCollection)
"_isLive":false,                            (Siehe: MediaCollection)
"_dvrEnabled":false,                        (Siehe: MediaCollection)
"_defaultQuality":[„auto“, 0, 1, 2, 3],     (Siehe: MediaCollection)
"_audioImage":"{
        "s":"../base/img/posterframe-s.jpg",
        "m":"../mandanten/sport/img/posterframe-m.jpg",
        "l":"../mandanten/rbb/img/posterframe-l.jpg",
        "xl":"../base/img/posterframe-xl.jpg"
    },                                      (Siehe: setAudioImage)
"_previewImage":{
        "s":"../base/img/posterframe-s.jpg",
        "m":"../mandanten/sport/img/posterframe-m.jpg",
        "l":"../mandanten/rbb/img/posterframe-l.jpg",
        "xl":"../base/img/posterframe-xl.jpg"
    },                                      (Siehe: setPreviewImage)
"_subtitleUrl":"testURL.xml",               (Siehe: setSubtitleUrl )
"_subtitleOffset":0,                        (Siehe: subtitleOffset)
```

Im Anschluss wird ein mediaArray-Objekt erstellt. Dort wird zuerst mit der _plugin-Eigenschaft definiert auf welchen Playertyp die Quellen verweisen. Im Anschluss, wird mit dem mediaStreamArray- eine Eigenschaft erstellt. Als Objekt kann diese wiederum weitere Eigenschaften beinhalten. Sie definieren die Streamquellen für die unterschiedlichen Qualitätsstufen.

Bei der Analyse des mediaArray-Objekts werden die Methoden `mc.addMedia`(0) & `addMediaStream`(0, ...) der automatisch erstellten MediaCollection verwendet. Die Methode

addMediaStream wird so oft Aufgerufen, wie es valide Werte gibt. Zudem wird in der addMediaStream-Funktion auch der CDN-Parameter mit Inhalt gefüllt. Hier wird definiert welcher CDN-Anbieter für den Stream genutzt wird, damit das richtige Plugin innerhalb von Flash genutzt werden kann.

Achtung:

Bei der Erstellung des MediaArrays ist darauf zu achten, dass nur die Stream-Objekte angelegt werden, die auch tatsächlich übergeben werden sollen! Existiert für eine Qualität kein Stream, so entfällt das komplette Objekt {}.

```
        "_mediaArray":
        [
          {
            "_plugin":0,
            "_mediaStreamArray":
           [
               {
                  "_quality":0,
                  "_server":"rtmpt://ndr.fcod.llnwd.net/a3715/d1/",
                  "_stream":"mp4:flashmedia/streams/ndr/testStream/TV-13-3901.lo"
               },
               {
                 "_quality":1,
                 "_server":"rtmpt://ndr.fcod.llnwd.net/a3715/d1/",
                 "_stream":"mp4:flashmedia/streams/ndr/testStream/TV-13-3901.med"
               }
           ]
          },
            "_plugin":1,
            "_mediaStreamArray":
            [
              {
                  "_quality":0,
                  "_server":"",
                   "_stream":"http://ard.rnd.gl-systemhaus.de/ard_test.mp4"
              },
             {
                   "_quality":1,
                   "_server":"",
                   "_stream":"http://ard.rnd.gl-systemhaus.de/ard_test.mp4"
              }
            ]
        }
],
```

Nach dem mediaArray-Objekt werden dann weitere Parameter definiert.

Im Anschluss wird die Sortierreihenfolge mit der _sortierArray-Eigenschaft definiert. Sie gibt die Prioritätenreihenfolge an, die der Ersteller der MediaCollection.json-Datei verlangt. Die hier übergeben Werte werden mit dem Aufruf von `setSortierung` in der automatisch erstellten MediaCollection-Instanz verarbeitet. Man kann maximal 2 Werte definieren und mindestens kein Wert. Dann wird die Standardreihenfolge “0, 1“ genutzt.

Erwartete Werte sind:

0 - Flash,
1 - HTML5

```
"_sortierArray":[0,1]
```

5.4. Zusätzlich genutzte Eigenschaften

5.4.1. subtitleOffset

Mit dem Offset definiert man mit welcher Verschiebung (in Sekunden) Untertitel angezeigt werden sollen. Damit kann man dafür sorgen, dass Untertitel Vor- oder Nach dem Ton erscheinen. Abweichungen in der Event-Gesteuerten Darstellung lassen sich damit abgleichen.

5.5. Vorgehensweise

Damit JSON-Daten verarbeitet werden können müssen je Player-Instanz zwei Query-Parameter definiert sein. Diese Parameter müssen mit der ID des jeweiligen Players beginnen und mit „mc“ (für MediaCollection) und „pc“ (für PlayerConfiguration) enden.

Der Parameterwert wiederum ist eine URL zu einer JSON Datei. Um einen Player mit dieser Variante zu erstellen, muss sowohl die MediaCollection also auch die PlayerConfig definiert sein!

5.6. Finale URL

SeitenURL`/?player01pc=pc.json&player01mc=mc.json`

6. Query Definitionen

Mit dieser Methode kann der Player mit der Definition der MediaCollection- und PlayerConfiguration-Objekten in der URL aufgerufen werden.

Wurde eine Player-Instanz ohne PlayerConfiguration und MediaCollection instanziiert, versucht diese (ferner nicht durch JSON gesteuert) die Daten aus dem Query-String zu beziehen.

Wichtig! Bei Multiinstanzen lässt sich lediglich ein Player über die Querys steuern.

6.1. Beschreibung

Im Folgenden werden die in einer URL definierbaren Parameter erläutert.

6.1.1. prio

Die hier übergeben Werte werden mit dem Aufruf von `setSortierung` in der automatisch erstellten MediaCollection-Instanz verarbeitet. Dabei werden die Prioritäten für die Darstellungs-PlugIns definiert.

Parameter:	Pflicht
Standardwert:	0, 1
Erwartete Werte sind:	0 - Flash, 1 - HTML5

```
prio= 1,0
```

6.1.2. img

Die hier übergeben Werte werden mit dem Aufruf von `setPreviewImage` der automatisch erstellten MediaCollection-Instanz verarbeitet. Hiermit wird eine URL für das Vorschaubild angegeben, dass Bild wird vor der Erstinitialisierung und nach „Stop“ dargestellt.

Parameter:	Optional
Standardwert:	„“
Erwarteter Wert:	Pfad zu dem Bild auf dem Server, relativ zu der index.html

```
?img=/test/test.jpg
```

6.1.3. imgAud

Dieser Aufruf, `setAudioImage`, übergibt die Werte der automatisch erstellten MediaCollection-Instanz. Hiermit wird eine URL für das Coverbild eines Audio-Players definiert.

Parameter:	Optional
Standardwert:	„“
Erwarteter Wert:	Pfad zu dem Bild auf dem Server, relativ zu der index.html

```
?imgAud=test/test.jpg
```

6.1.4. suburl

Die hier übergeben Werte werden mit dem Aufruf von `setNoSubtitelAtStart` der automatisch erstellten MediaCollection-Instanz verarbeitet. Hiermit wird eine URL für die Untertitel-XML übergeben und die entsprechende Schaltfläche angezeigt.

Parameter:	Optional

Standardwert: „“

Erwarteter Wert: Pfad zu der XML-Datei auf dem Server, relativ zu der index.html

```
?suburl=test/subtitle/5282090.xml
```

6.1.5. opt

Durch den Aufruf von `setShowOptions` werden die Werte der automatisch erstellten PlayerConfiguration-Instanz verarbeitet. Mit dieser Variable werden die Optionsanzeigeobjekte aktiviert.

Parameter: Optional

Standardwert: 1

Erwartete Werte sind: 0 - ausblenden
1 - anzeigen

```
?opt=1
```

6.1.6. time:

Die hier übergeben Werte werden mit dem Aufruf von `setStartStopTime` der automatisch erstellten PlayerConfiguration-Instanz verarbeitet. Hiermit wird der Offset (Start- & Endzeit für den Stream) definierbar.

Parameter: Optional

Standardwert: 0,0

Erwartete Werte: Startzeit in Sekunden (ganzzahlig positiv)
Endzeit in Sekunden (ganzzahlig positiv)

```
?time=10,200
```

6.1.7. rem

Die hier übergeben Werte werden mit dem Aufruf von `setRememberCurrentTime` der automatisch erstellten PlayerConfiguration-Instanz verarbeitet. Hiermit wird eingestellt, ob bei einem plugIn Wechselt bei der Letzt verblieben Zeit weiter gemacht wird!

Parameter: Optional

Standardwert: 1

Erwartete Werte sind: 0 - ausblenden
1 - anzeigen

```
?rem= 1
```

6.1.8. auto

Die hier übergeben Werte werden mit dem Aufruf von `setAutoPlay` der automatisch erstellten PlayerConfiguration-Instanz verarbeitet. Hiermit wird

der AutoPlay-Modus festgelegt. Bei iOS Geräten wird diese Angabe nicht unterstützt.

Parameter: Optional

Standardwert: Mobilgeräte → 0, Desktop → 1

Erwartete Werte sind: 0 - ausblenden
1 - anzeigen

```
?auto=1
```

6.1.9. autosv

Die hier übergeben Werte werden mit dem Aufruf von `setAutoSave` der automatisch erstellten PlayerConfiguration-Instanz verarbeitet. Hiermit wird der AutoSave festgelegt – sollte das nicht aktiviert sein erscheint bei der PlugIn-Auswahl eine Save Schaltfläche. Ein Klick darauf speichert die Auswahl dann in einem Cookie. Bei aktivierter Einstellung wird jeder Wechsel direkt gespeichert und als neuer Standardwert verwendet.

Parameter: Optional

Standardwert: 1

Erwartete Werte sind: 0 - ausblenden
1 - anzeigen

```
?autosv=1
```

6.1.10. mct

Definiert in der automatisch generierten MediaCollection den gesetzten Typ

Parameter: Pflicht

Standardwert: „“

Erwartete Werte sind: audio
video

```
?mct=video
```

6.1.11. mcl

Bescheibt in der automatisch generierten MediaCollection den isLive-Wert.
1 == true == live

Parameter: Pflicht

Standardwert: „“

Erwartete Werte sind: 0 - kein LiveStream
1 - LiveStream

```
?mcl=1
```

6.1.12. mcd

Definiert in der automatisch generierten MediaCollection den Std.-Qualitäts Wert.

Parameter: Pflicht

Standardwert: „“

Erwartete Werte: ganzzahlige positive Werte
(In Abhängigkeit der Hinterlegten Qualitäten)

```
?mcd=3
```

7. Erweiterte Query Definitionen

Die Unterparameter werden innerhalb der URL immer mit einem Komma getrennt.

Achtung! Die Angabe der Qualitäten beginnt von der kleinsten Qualität zur höchsten Qualität.

7.1. reps

Definiert die Repräsentations-Darstellung des Players. Hierfür wird die `setRepresentation` Methode der automatisch erstellten PlayerConfiguration verwendet. Diese Methode wird so oft Aufgerufen, wie es valide Werte gibt.
Unterparameter sind:

7.2. s4flash:

In diesem Query werden die Quellen zu den Flash-Filmen definiert. Hierfür werden die Methoden `mc.addMedia`(0) & `addMediaStream`(0, ...) der automatisch erstellten MediaCollection verwendet. Die Methode addMediaStream wird so oft Aufgerufen, wie es valide Werte gibt. Zudem wird in der addMediaStream-Funktion auch der cdn-Parameter mit Inhalt befüllt. Hier wird definiert welcher CDN-Anbieter für den Stream genutzt wird, damit das richtige Plugin innerhalb von Flash genutzt werden kann.
Unterparameter sind:

7.2.1. Server:

URL des Servers, aus dem der Stream geladen werden soll

Parameter: Pflicht

Standardwert: „"

7.2.2. Stream:

Dateiname der vom Streaming-Server geladen werden soll

Parameter: Pflicht

Standardwert: „"

7.2.3. CDN:

Beschreibt die Angabe des genutzten Content Delivery Network(CDN) für den Stream.

Parameter: Pflicht

Standardwert: „"

```
&s4flash=rtmpt://ndr.fcod.llnwd.net/a3715/d1/,mp4:flashmedia
/streams/ndr/2011/0928/TV-20110928-2313-
3901.lo,akamai,rtmpt://ndr.fcod.llnwd.net/a3715/d1/,mp4:flas
hmedia/streams/ndr/2011/0928/TV-20110928-2313-3901.med,
akamai,rtmpt://ndr.fcod.llnwd.net/a3715/d1/,mp4:flashmedia/s
treams/ndr/2011/0928/TV-20110928-2313-3901.hi,
akamai,rtmpt://ndr.fcod.llnwd.net/a3715/d1/,mp4:flashmedia/s
treams/ndr/2011/0928/TV-20110928-2313-3901.xhi
```

7.3. s4html:

In diesem Query werden die Quellen zu den HTML5-Filmen definiert. Hierfür werden die Methoden `mc.addMedia`(1) & `addMediaStream`(1, ...) der automatisch erstellten MediaCollection verwendet. Die Methode addMediaStream wird so oft Aufgerufen, wie es valide Werte gibt.

Unterparameter sind:

7.3.1. Server:

Der Server muss für einen HTML Stream nicht definiert sein und ist ein leerer String.

Parameter: Pflicht

Standardwert: „"

7.3.2. Stream:

Beschreibt den Dateinamen der vom Streaming-Server geladen werden soll.

Parameter: Pflicht

Standardwert: „"

7.3.3. CDN:

Diese Eigenschaft muss angegeben werden, obgleich sie nur bei Flash-Content genutzt wird. Es wird bei der Auswertung der unterschiedlichen s4html und s4flash-Eigenschaften auf die selbe Auswertungsfunktion verwiesen.

Parameter: Pflicht

```
Standardwert:                    „"
      ?s4html=,media/ard_test.mp4,,,media/ard_test.mp4,,,
media/ard_test.mp4,,
```

8. Beispiel einer URL mit Query Definitionen

StartURL/?reps=111,144,false,true,512,288,false,true,960,540,true,true,1280,720,true,true&tbd=1&opt=1&tb=1&tbq=1&tbqn=1&zoom=1&set=1&time=10,200&rem=0&auto=1&autosv=1&mcl=0&mcd=2&mct=audio&prio=2,1&dwn=console.log&pod=console.log&img=img/tmp-512x288.jpg&imgAud=img/tmp-512x288.jpg&suburl=suburl.xml,123&s4flash=rtmpt://ndr.fcod.llnwd.net/a3715/d1/,mp4:flashmedia/streams/ndr/2011/0928/TV-20110928-2313-3901.lo,akamai,rtmpt://ndr.fcod.llnwd.net/a3715/d1/,mp4:flashmedia/streams/ndr/2011/0928/TV-20110928-2313-3901.med,akamai,rtmpt://ndr.fcod.llnwd.net/a3715/d1/,mp4:flashmedia/streams/ndr/2011/0928/TV-20110928-2313-3901.hi,akamai,rtmpt://ndr.fcod.llnwd.net/a3715/d1/,mp4:flashmedia/streams/ndr/2011/0928/TV-20110928-2313-3901.xhi,akamai,&s4html=,media/ard_test.mp4,,,media/ard_test.mp4,,,media/ard_test.mp4,,,media/ard_test.mp4,&s4wmp=,WMPvideovideo.lo.wmv,,,WMPvideovideo.med.wmv,,,WMPvideovideo.hi.wmv,,,WMPvideovideo.xhi.wmv,,

9. Konfigurationseinstellungen des Flash-Players

9.1. Config.xml

Ist zuständig um den Flash-Player mit den notwendigen Information zum Aufbau der CDN-spezifischen Konfigurationen zu versorgen.

Die Informationen beinhalten:

- Pfade zu den unterschiedlichen Plugins, die genutzt werden (<plugin/>). Optional kann der Platzhalter „#baseUrl#“ verwendet werden, der duch den konfigurierten Basispfad ersetzt wird (s.u.).
- Definition der unterschiedlichen CDNs, und Definition der zu verwendenden Plugins (<cdn/>):
- id-Eigenschaft: die im `addMediaStream` als cdn-Eigenschaft eingefügt wird
- ref: dort wird eine Referenz auf die oben definierten Plugins erstellt
- alias: Alias-Name der in der cdn-Eigenschaft in der `addMediaStream` Funktion genutzt werden kann
- zusätzliche Informationen können mittels des Metadata-Knotens angegeben werden. Diese werden zur Einbindung spezifischer OSMF Plugins benötigt, und können abhängig des Mediums gefiltert werden:
- includein
- „live“, „vod“ oder „resource“

9.1.1. Beispiel einer config.xml-Datei

```
<config>
    <plugins>
          <plugin
id="captioningPlugin"><![CDATA[#baseUrl#plugins/flash/osmf/SMPTETTPlugin.swf]]></plugin>
          <plugin
id="smilPlugin"><![CDATA[#baseUrl#plugins/flash/osmf/SMILPlugin.swf]]></plugin>
```

```
<plugin
id="convivaStarterPlugin"><![CDATA[#baseUrl#test/plugin/ConvivaStarterPl
ugin.swf]]></plugin>
</plugins>


<networks>
        <cdn id="default">
            <plugin ref="smilPlugin"/>
            <plugin ref="captioningPlugin"/>
        </cdn>
            <cdn id="conviva">
                  <plugin ref="convivaStarterPlugin" />

                  <metadata
key="com.netTrek.gundl.qos.ConvivaStarterPlugin"
                        includeIn="resource">
                        <metadata key="config-path"
value="#baseUrl#test/conviva-config.xml" />
                  </metadata>
            </cdn>
    </networks>
</config>
```

9.2. Conviva-config.xml

Der Verweis auf diese Datei wird als Metadata-Information in der in der config.xml definiert, und ist standardmäßig auf einem G&L Server hinterlegt.

In den ersten drei Knoten werden die Werte definiert für:

- customerId: ID des Kunden, der bei Conviva registriert ist
- serviceUrl: URL an die Conviva eigene Daten zurückliefert
- pluginUrl: URL von der kundenspezifische Plugin abgerufen wird

Die weiteren Knoten beschreiben unterschiedliche Konfigurationen, die anhand der hinterlegten ID später im Player erreichbar sind.

Beispiel:

<configuration id=“euro“ />

Aus dem Player über:

mc.addMediaStream(0, 0, „“, „ref:euro“, „conviva“);

erreichbar. Mittels des Knoten „failover“ kann zu einer weiteren Konfiguration verlinkt werden, zu der automatisch gewechselt wird.

Ist lediglich ein Host, aber keine Einzelstreams definiert, wird von einem HDS Stream ausgegangen.

9.2.1. Beispiel einer conviva-config.xml-Datei

```
<conviva>
      <customerId>kundenID</customerId>
      <serviceUrl>http://livepass.conviva.com/</serviceUrl>
      <pluginUrl>http://livepassdl.conviva.com/OSMF/ConvivaOSMFPlugin_OS
            MF1_6_FP10_1.swf?customerName=${customerId}
```

Beispiel einer conviva-config.xml-Datei

```
        </pluginUrl>
        <configuration id="euro" live="true">
              <!-- -->
              <candidate id="AK-RTMP">
                    <host>rtmp://cp163901.live.edgefcs.net/live</host>
                    <stream bitrate="264" width="256"
                          height="144">wdr_euro_264@15802</stream>
                    <stream bitrate="564" width="512"
                          height="288">wdr_euro_564@15802</stream>
                    <stream bitrate="764" width="640"
                          height="360">wdr_euro_764@15802</stream>
              </candidate>
              <tags>
                    <tag key="policy">RTMP</tag>
                    <tag key="context">sportschau.de</tag>
              </tags>
              <failover ref="euro-failover" />
        </configuration>
  </conviva>
```