HS-D/SS15

Rich Internet Applications

Camtasia für Lernvideos und Educasts

David Kay Posmyk & Alexander Dlugosch

27.09.2015

Inhalt

[Einleitung 2](#_Toc431080035)

[Camtasia Studio 3](#_Toc431080036)

[Smart Player 3](#_Toc431080037)

[Anwendungsfälle für Lernvideos und Educasts 4](#_Toc431080038)

[Erweiterung 5](#_Toc431080039)

[Erweiterbarkeit des Camtasia Smart Players 5](#_Toc431080040)

[Erweiterbarkeit von Camtasia Studio 5](#_Toc431080041)

[Zweckentfremdung 5](#_Toc431080042)

[Transformation des Projektformats 7](#_Toc431080043)

[Zielformat 7](#_Toc431080044)

[Konzeption 7](#_Toc431080045)

[Realisierung 7](#_Toc431080046)

[Verwendung 8](#_Toc431080047)

[Auswählen der Projektvorlage 8](#_Toc431080048)

[Produzieren des Videos 8](#_Toc431080049)

[Anreichern durch Metadaten 9](#_Toc431080050)

[Transformieren zu JSON 14](#_Toc431080051)

[Publizieren des Videos 14](#_Toc431080052)

[Probleme und Fehlerquellen 15](#_Toc431080053)

[Anhang 16](#_Toc431080054)

[Camtasia Projektformat 16](#_Toc431080055)

[Projektmetadaten 17](#_Toc431080056)

[Zeitleiste 18](#_Toc431080057)

[Medien 19](#_Toc431080058)

[Quiz-XML 22](#_Toc431080059)

[Datenbasis 24](#_Toc431080060)

[Anregungen zur Implementierung 28](#_Toc431080061)

# Einleitung

In dem Seminar *Rich Internet Applications* haben wir uns mit interaktiven / enriched Educasts und Lernvideos beschäftigt. Unser Schwerpunkt lag bei den Autorenumgebungen und der zugehörigen Player.

Wir haben uns einige Gedanken dazu gemacht, wie man vorhandene Autorentools und Player mit interaktiven Features anreichern kann. Da Camtasia Studio in unserer Hochschule eines der beliebtesten und meistgenutzten Werkzeuge zur Videoproduktion ist, entschieden wir uns im Laufe des Seminars dazu, uns konkret auf dieses zu konzentrieren.

Wir entwickelten außerdem ein Interesse an dem ePlayer[[1]](#footnote-2), da dieser (nach Angabe der Kommilitonen die sich ausgiebiger mit dem Thema Player und deren Features beschäftigt haben) der Einzige ist, der mit einer Reihe von Features aufwarten kann, die speziell für Lernvideos gedacht sind.

Das vorliegende Dokument repräsentiert einige unserer erarbeiteten Ergebnisse dieses Seminars und beschreibt unsere neuen Anforderungen, die wir an die Autorenumgebung Camtasia Studio gestellt haben. Wir erläutern, wie wir Camtasia Studio für unsere Bedürfnisse erweitert haben und wie die neuen Features nun in einem Camtasia Studio Projekt und dann letztendlich in einem Player, wie bspw. dem ePlayer, genutzt werden können.

Im Anhang definieren wir außerdem ein Dateiformat das, neben dem ePlayer, für weitere Player als eine gemeinsam genutzte Datenbasis verwendet werden kann, um unsere Features auch dort zu unterstützen.

Als weiteres Ergebnis unserer Arbeit liegt das Tool *Camtasia2Json* mitsamt Quellcode als weiteren Anhang zu diesem Dokument bei. Zur Nutzung der neuen Features ist dieses Tool unverzichtbar.

# Camtasia Studio

Camtasia Studio ist ein leistungsstarkes Programm zur Bildschirmaufnahme und Videobearbeitung. Es verfügt u.a. über folgende Features:

* Callouts: Texte, Bilder, Pfeile und Rahmen die "auf das Video" gelegt werden können.
* Hotspots: Interaktiv anklickbare Callouts (wenn über den *Smart Player* abgespielt).
* Zoomen: In bestimmte Teile des Videos hinein- und hinaus Zoomen.
* Untertitel (einsprachig)
* Marker: Bestimmte stellen im Video markieren um dorthin zu springen. Auch zum Erstellen von Kapiteln gedacht.
* Übergangseffekte und Animationen auf Videos und Callouts.
* Interaktive Quizze.

Weitere Hinweise zu Camtasia Studio und zur Nutzung der o.g. Funktionen finden sich in der Online-Hilfe[[2]](#footnote-3).

## Smart Player

In einer der Produktionsvarianten für die Videos bietet Camtasia Studio 8 an, dass Videomaterial zusammen mit einem speziellen HTML5 / Flash basierten Player zu exportieren. Der *Camtasia Smart Player* ermöglicht so das Abspielen eines Videos mit zusätzlichen interaktiven Funktionen für den Benutzer.

Folgende interaktive Funktionen werden dabei vom *Smart Player* unterstützt:

|  |  |
| --- | --- |
| * Darstellung von Kapiteln in einer Seitenleiste * Suchfunktion * Untertitel (nur eine Sprache) ein/aus * Wiedergabegeschwindigkeit regelbar * Vom Nutzer anklickbare Hotspots |  |

Die interaktiven Funktionen die der Player bietet sind also eher spärlich. Für hochwertige Educasts / Lernvideos sind weitere Funktionalitäten für den Benutzer wünschenswert. Auch das UI-Design des Players ist sehr schlicht gehalten und nicht sonderlich ansprechend.

# Anwendungsfälle für Lernvideos und Educasts

Für die Produktion von hochwertigen Lernvideos ziehen wir die folgenden Anwendungsfälle in Betracht:

1. Als Autor möchte ich ein Inhaltsverzeichnis in Baumform definieren können, um die Videos nach Kapiteln und Unterkapiteln inhaltlich zu strukturieren.
2. Als Autor möchte ich Dateien an das Video anhängen können, um den Benutzern Zusatzinformationen bereitzustellen.
3. Als Autor möchte ich Mehrsprachigkeit ermöglichen, um eine breitere Masse an Benutzern zu erreichen.
4. Als Autor möchte ich den Nutzer ermöglichen, die Inhalte in Sozialen Netzwerken zu teilen, um das Video populärer zu machen.
5. Als Autor möchte ich mein Video in verschiedenen Auflösungen und Formaten bereitstellen, so dass der Benutzer mein Video auf einem HD-Bildschirm sowie Smart Phone jeder Firma gut erkennen und auch mit einer tolerierbaren Datenrate betrachten kann.
6. Als Autor möchte ich für den Benutzer Notizen anlegen können, um bestimmte Abschnitte im Video zu annotieren und ggf. Literaturhinweise o.ä. zur Verfügung zu stellen.
7. Als Autor möchte ich Literaturhinweise geben können, um den Benutzern Zusatzinformationen bzw. Quellenangaben übersichtlich bereitzustellen.
8. Als Autor möchte ich Hinweistexte erstellen können, um etwaige Verlinkungen, Notizen, Kapitel kurz beschreiben zu können. Ich möchte dass der Benutzer vor dem anklicken erkennt, um welche Informationen es sich handelt.
9. Als Autor möchte ich für den Nutzer interaktive Inhalte über das Video legen können, um ….
   1. … direkt im Video einen anklickbaren Link erstellen zu können, um den Benutzer auf weitere Inhalte oder eine andere Position im Video zu verweisen.
   2. ... im Video einen iFrame einbinden zu können, um Webinhalte einzubinden, z.B. um Feedback über Web-Formulare einholen zu können.
   3. ... im Video kopierbare Elemente einbinden zu können, um den Nutzer die Möglichkeit zu geben Videoinhalte (z.B. Quellcode, Text, etc.) direkt per Klick zu kopieren.
10. Als Autor möchte ich mich selber während des Videos, nebenher zeigen (Autorenkamera), um einen persönlicheren Bezug zum Nutzer herzustellen. Ich möchte aber nicht, dass diese wichtige Inhalte meines Videos überdeckt oder den Benutzer ablenkt.
11. Als Autor möchte ich dem Benutzer anbieten das Inhaltsverzeichnis und auch mein gesamtes Video zu durchsuchen.
12. Als Autor möchte ich für den Nutzer ein- und ausblendbare Untertitel zur Verfügung stellen, um Barrierefreiheit zu ermöglichen und das Video durchsuchbarer zu machen.

# Erweiterung

Um die Anwendungsfälle abdecken zu können, ist offensichtlich eine Erweiterung des Camtasia Smart Players und auch von Camtasia Studio erforderlich.

## Erweiterbarkeit des Camtasia Smart Players

Eine Erweiterung des Players ist ausgeschlossen, da einerseits die JavaScript Datei minifiziert und chiffriert ist und zum anderen rechtlich geschützt ist. Außerdem verwendet der *Smart Player* ein XML-Dokument als Datenbasis, das für eine Zweckentfremdung nicht in Frage kommt, da zu viele wichtige Metadaten (wie z.B. Spurdaten) fehlen.

Daher muss auf einen anderen Player ausgewichen werden.

## Erweiterbarkeit von Camtasia Studio

Camtasia Studio ist nicht open-source und bietet grundsätzlich keine Schnittstelle für Plugins, oder ein Skriptinterface an. Durch die Zweckentfremdung der vorhandenen Features ist allerdings einige Zusatzfunktionalität denkbar. So könnte man z.B. durch eine spezielle Spur in der Zeitleiste dafür sorgen, dass ein dort vorhandenes Video als ein- und ausschaltbare Autorenkamera im Player dargestellt wird.

Wenn Zusatzfunktionalität über Zweckentfremdung ermöglicht werden soll, dann ist es natürlich auch erforderlich die Spurdaten, die Camtasia Studio abspeichert, durch den Player auszulesen. Hierfür sind grundsätzlich zwei Datenquellen in Erwägung zu ziehen:

* Die beim Produzieren mit dem Smart Player exportierte XML Datei.
* Die Camtasia Studio Projektdatei (Endung *camproj*) im XML Format.

Ersteres entfällt, da bereits auf den ersten Blick sichtbar ist, dass in diesem Format wesentliche Metadaten des Projekts (z.B. die in Camtasia angelegten Spuren) nicht enthalten sind. Daher ist nur letzteres in Betracht zu ziehen.

Eine umfangreiche Analyse zum Projektformat findet sich im Anhang unter Camtasia Projektformat.

## Zweckentfremdung

Aus den Anwendungsfällen ergibt sich, dass der Autor vor allem weitere Daten / Informationen für seine Benutzer hinterlegen möchte. Diese zusätzlichen Daten sind teilweise von der aktuellen Abspielposition d.h. von der Position auf der Zeitleiste abhängig. Es ist daher sinnvoll, diese Daten durch Callouts repräsentieren zu lassen, da diese auf den Spuren frei platziert werden, und beliebigen Textinhalt enthalten können.

Ein Callout kann frei formatierbaren Text enthalten, dieser eignet sich gut für Beschreibungstexte o.ä. ist allerdings ungeeignet um Metainformationen wie z.B. eine URL zu einer für den Benutzer herunterladbaren Datei zu hinterlegen. Daher ist eine Teilung von formatierbarem Inhalt und Metadaten sinnvoll, d.h. man erlaubt keinerlei Formatierungen für die Metadaten, wohl aber für sämtlichen andere Inhalte des Callouts. Zum Beispiel könnte bei einer Kapiteldefinition ein Callout angelegt werden, der einerseits einen frei formatierbaren Beschreibungstext enthält, und einen nicht formatierbaren Titel.

Um leicht zu unterscheiden, welche Callouts welche Arten von Daten repräsentieren, kann man außerdem spezielle Namen für Spuren definieren die sich durch ein Präfix wie bspw. *#!* klar von normalen Spuren abheben. Ein weiterer Vorteil solcher Spuren ist auch, dass man dadurch relativ einfach Mehrsprachigkeit über sämtliche neue Features ermöglichen kann, indem man mehrere Spuren derselben Art in unterschiedlichen Sprachen ermöglicht.

Muss man Daten in Listenform mit beliebig vielen Einträgen hinterlegen, z.B. um eine Liste von Literaturhinweisen an ein Kapitel anzuhängen, eignet es sich hierfür mehrere Callouts als ein Gruppenelement zusammen zu fassen, wobei dann jeder Callout einen Listeneintrag repräsentiert.

Aus den Anwendungsfällen ergeben sich folgende Features die via Zweckentfremdung abgebildet werden:

* Allgemeine Konfiguration und Metadaten zum Video.
  + Autorinformationen (können in Camtasia nicht **pro Projekt** konfiguriert und daher auch nicht über die Projektdatei ausgelesen werden).
  + Die URL für das Platzhalterbild vor dem Abspielen des Videos.
  + Einstellungen für das Social Sharing (z.B. welche Optionen der Player hier anbieten soll, welchem Benutzer auf Twitter gefolgt werden kann etc.)
* URLs zu Mediendateien (für mehrere Sprachen, Formate und Auflösungen).
* Zusätzliche Titel und Beschreibungstexte für andere Sprachen.
* Kapitel und optionale Kapitelkategorien.
  + Mit hinterlegbaren Literaturhinweisen (Verlinkungen).
  + Mit Tags zur Optimierung der Durchsuchbarkeit.
* Autorennotizen
  + An Kapitel gebunden (da Darstellung in der Kapitelstruktur ggf. Sinnvoll).
  + An bestimmter Position auf der Zeitleiste.
* Untertitel (in mehreren Sprachen).
* Hinterlegbare Hinweistexte bei allem was im Player durch den Benutzer angeklickt werden kann.
* Overlays / Hotspots ("auf dem Video" platzierbarer Inhalt, mit dem der Benutzer interagieren kann)
  + Mit freiem HTML Inhalt (auch iFrames).
  + Mit einigen Konfigurationsmöglichkeiten hinsichtlich der Interaktion (Kopierbarkeit, automatisches Ausblenden,
* Autorenkamera
  + Mit URLs zu Mediendateien (auch in mehreren Sprachen, Formaten und Auflösungen).

# Transformation des Projektformats

Da das Projektformat von Camtasia Studio sehr redundant ist und die Daten in einer schlecht verwertbaren, zu komplexen Struktur vorliegen, ist es sinnvoll das Format vorher in anderes, gut verwertbares Format umzuwandeln.

## Zielformat

Ideal ist hierfür das JSON (JavaScript Object Notation) Format geeignet, da es direkt für die Nutzung einer JavaScript basierten Webanwendung konzipiert wurde. Außerdem kann der ePlayer bereits einige unserer Anwendungsfälle abdecken, und nutzt selbst bereits das JSON Format.

Der ePlayer implementiert folgende Features, abgebildet durch sein JSON Format:

* Autorennotizen
* Kapitel (mit hinterlegbaren Literaturhinweisen / Dateiverlinkungen)
* Untertitel
* Mehrsprachigkeit bei allen Videoinhalten

Dieses Format eignet sich daher als grundlegende Datenbasis, muss aber erweitert werden, um auch die Daten für all unsere erdachten Features bereitstellen zu können. Das Format sollte dabei möglichst so erweitert werden, dass es nach wie vor ePlayer kompatibel bleibt, d.h. auch eine JSON Datei mit unseren ergänzten Inhalten sollte im ePlayer abspielbar bleiben.

In Zukunft kann dann entweder der ePlayer erweitert werden, oder ein gänzlich neuer Player entwickelt werden, der das Dateiformat vollständig unterstützen kann.

Eine detaillierte Definition der Datenbasis findet sich im Anhang unter Datenbasis.

## Konzeption

Da das Camtasia Projektformat in XML vorliegt, bietet sich sofort die Transformation mittels XSLT (Extensible Stylesheet Language Transformation) an. Mit dieser ist es relativ einfach möglich, ein komplexes XML-Dokument in das JSON Format zu transformieren. Außerdem ist eine nachträgliche Anpassung relativ leicht möglich, was den Wartungsaufwand verringert - denn es ist abzusehen, dass sich mit neuen Camtasia Studio Versionen auch das Projektformat ändern wird.

Da der formatierbare Text innerhalb von Callouts im Rich Text Format vorliegt, ist eine Konvertierung in HTML erforderlich, welche mit XSL nur sehr umständlich zu lösen wäre. Für eine Programmiersprache wie Java gibt es allerdings einige Bibliotheken (selbst die Standardbibliothek) die solche Funktionalitäten anbieten. Da XSL genau für solche Fälle das Aufrufen von Funktionen, die in einer anderen Programmiersprache definiert wurden, unterstütz, wäre eine Implementierung in Java und XSL daher angebracht.

## Realisierung

Es wurde eine Kommandozeilenanwendung *Camtasia2Json* mittels Java 8 und XSL realisiert. Sie ermöglicht einem Autor ein Video zu produzieren, das die von uns definierten Anwendungsfälle erfüllen kann, wenn es in einem entsprechenden Player abgespielt wird. Der ePlayer kann dabei einige der Anwendungsfälle abdecken.

Weitere Informationen zu der Technologie und den Bibliotheken, die bei der Realisierung verwendet worden sind, finden sich in der Datei *README.md* des *Camtasia2Json* Projekts.

# Verwendung

Im Folgenden wird schrittweise erläutert, wie ein Videoautor die durch uns definierten Anwendungsfälle mittels Camtasia Studio umsetzen kann. Ein grundlegendes Wissen über die Bedienung von Camtasia Studio 8 wird dabei vorausgesetzt. Der Workflow gestaltet sich folgendermaßen:

## Auswählen der Projektvorlage

Für sämtliche zweckentfremdete Funktionen sind spezielle Metadaten in Form von Spuren (folgend als Meta-Spuren bezeichnet), mit fest definierten Namenskonventionen erforderlich. Diese müssen jeweils bestimmte Callouts (folgend als Meta-Elemente bezeichnet) definieren. Damit Meta-Spuren und Meta-Elemente nicht von Hand angelegt werden müssen, stellen wir Projektvorlagen zur Verfügung die bei der Produktion eines neuen Videos verwendet werden sollten. Möchten Sie dagegen ein vorhandenes Camtasia Studio Projekt mit zweckentfremdeten Funktionen ausstatten, so empfehlen wir die gewünschten Meta-Spuren manuell anzulegen, und die Vorlagen für Meta-Elemente aus einem der Projektvorlagen durch Kopieren und Einfügen zu übernehmen.

Folgende Projektvorlagen stehen zur Verfügung:

* **Für die Nutzung im ePlayer optimiert (ein- und zweisprachig)**  
  Enthält nur die Metaspuren und –Elemente die vom ePlayer unterstützt werden.
* **Mit sämtlichen Features (ein- und zweisprachig)**  
  Enthält sämtliche Metaspuren und Elemente.

Die Vorlagen für zweisprachige Videos enthalten Spuren in der Standardsprache (Deutsch) sowie in einer Fremdsprache (Englisch).

## Produzieren des Videos

Fügen Sie nun in diese Projektvorlage sämtliche weitere erforderliche Spuren, Medien und Untertitel hinzu.

Wenn Untertitel nicht in das Video eingebrannt werden sollen, definieren Sie diese auf dem #!utitel Metatrack und schalten ihn auf Sichtbar.

Wenn auch Material für eine Autorenkamera vorliegt, dann erstellen Sie für diese eine zusätzliche normale Spur und fügen dort das entsprechende Videomedium ein. Verwenden Sie hierbei nicht die Spur #!autkam – diese dient nur der Konfiguration der Autorenkamera, sie ist nicht für das eigentliche Videomaterial geeignet.

Stellen Sie sicher, dass alle Metatracks (außer #!utitel) unsichtbar sind, so dass diese Sie bei der Bearbeitung des Videos nicht behindern, außerdem werden diese Tracks dann nicht mitgeschnitten oder produziert.

Wenn das Videomaterial und alle weiteren Medien in einem produzierbaren Zustand vorliegen (d.h. wenn Sie alle Nicht-Metaspuren fertig bearbeitet haben), dann deaktivieren Sie die Sichtbarkeit der #!utitel Spur und die der Spur für die Videodaten der Autorenkamera – falls Sie nicht wünschen, dass diese Features ins Video eingebrannten werden. Nun produzieren Sie das Video in allen benötigten Dateiformaten (mp4 und webm empfohlen) und legen die resultierenden Dateien im Ordner *publish* ab. Erstellen Sie dann ggf. ein Platzhalterbild und legen dieses auch in *publish* ab.

Wenn Sie eine Autorenkamera möchten, die nicht ins Video eingebrannt wird, dann muss diese in Form von zusätzlichen Videodateien vorliegen. Dies können Sie leicht dadurch erreichen, dass Sie die Sichtbarkeit aller Spuren, außer der auf der das Videomaterial ihrer Autorenkamera liegt, ausschalten. Stellen Sie dann sicher, dass die Autorenkamera den kompletten Bearbeitungsbereich einnimmt. Dann produzieren Sie das Video erneut in allen benötigten Dateiformaten und mit der Videoauflösung die sie für die Autorenkamera benötigen. Legen Sie die produzierten Dateien der Autorenkamera ebenfalls in *publish* ab.

Nachdem ihr Video fertig produziert ist, können Sie es nun mit interaktiven Features und Metadaten anreichern. Dazu schalten Sie für alle Metatracks den Sichtbarkeitsstatus um.

Zuerst konfigurieren Sie die Metaelemente vom Typ *Medium* auf der #!meta Spur. Diese müssen Ihre produzierten Videodateien per URL referenzieren, das werden die Dateien sein, die sich aktuell im Ordner *publish* befinden.

Wie Sie alle weiteren Metaelemente für Ihren Anwendungsfall nutzen können, lesen Sie im nächsten Abschnitt.

## Anreichern durch Metadaten

Für sämtliche Metaelemente gelten grundsätzlich folgende Einschränkungen:

* Visuelle Effekte werden nicht unterstützt (Ausnahme: Overlays unterstützen die Callout-Eigenschaft „Einblenden“ und „Ausblenden“.)
* Animationen werden nicht unterstützt und führen bei der Transformation ggf. zu Fehlern.
* Die Eigenschaft „Clip-Geschwindigkeit“ wird auf keinem Metaelement unterstützt.
* Ein Bild das auf die Zeitleiste eingefügt wird, kann nicht als Metaelement fungieren.

Es folgt eine kurze Erläuterung aller verfügbarer Metaspuren:

|  |  |  |
| --- | --- | --- |
| Spurname | Inhalt | Einschränkungen / Hinweise |
| #!konfig | Allgemeine Videokonfiguration bzw. Metadaten die nicht in den Projektinformationen hinterlegt werden können. | Muss genau einmal vorhanden sein. Keine Mehrsprachigkeit möglich. |
| #!meta | Definition der abzuspielenden Mediendateien und weiterer Metadaten die nicht in den Projektinformationen hinterlegt werden können. | Mehrsprachigkeit möglich. Muss in der Standardsprache vorhanden sein. |
| #!kategorien | Definitionen der Kapitel-Kategorien. | Mehrsprachigkeit möglich. Darf mehrmals vorkommen. Kann entfallen. |
| #!kapitel | Definitionen der Kapitel. | Mehrsprachigkeit möglich. Darf mehrmals vorkommen. Kann entfallen. |
| #!k-extras | Definitionen von Kapitelzusatzmaterial. | Mehrsprachigkeit möglich. Darf mehrmals vorkommen. Kann entfallen. |
| #!notizen | Definitionen von Autorennotizen. | Mehrsprachigkeit möglich. Darf mehrmals vorkommen. Kann entfallen. |
| #!utitel | Untertitel. | Mehrsprachigkeit möglich. Darf mehrmals vorkommen. Kann entfallen. |
| #!overlays | Overlays bzw. Hotspots. | Mehrsprachigkeit möglich. Darf mehrmals vorkommen. Kann entfallen. |
| #!autkam | Für die Konfiguration der Autorenkamera. | Kann nur einmal vorkommen. Keine Mehrsprachigkeit möglich. Kann nur ein Gruppenelement enthalten. |

Im Folgenden wird eine Konfigurationsübersicht zu den Meta-Elementen bereitgestellt.

Jeder Abschnitt ist wie folgt strukturiert:

* **Meta-Spur und -Element:** Das zu erstellende Element die Spur auf dem dieses zu erstellen ist.
* **Positionierung auf der Zeitleiste:** Die Besonderheiten, die bei der Positionierung des jeweiligen Elementes zu beachten sind.
* **Meta-Inhalt:** Dies sind Plichtangaben die im Inhaltsfeld des jeweiligen Elementes angegeben werden müssen und dessen Funktionalität beeinflussen.
* **ePlayer**: Ob und wie dieses Feature durch den aktuellen ePlayer unterstützt wird.

#### Konfiguration: Allgemein

Meta-Spur und -Element: *#!konfig*, Callout

ePlayer: Unterstützt.

Positionierung auf Zeitleiste: Beliebig.

Meta-Inhalt:

* KonfigTyp**:** [Allgemein] Die Art der Konfiguration. Muss immer diesen Wert haben.
* Autor**:** Name des Autors.
* Platzhalter URL**:** Die URL zu dem Bild das vor dem abspielen des Videos angezeigt wird. Kann leer gelassen werden.

#### Konfiguration: Social Share Funktionen

Meta-Spur und -Element: *#!konfig*, Callout

ePlayer: Nicht unterstützt.

Positionierung auf Zeitleiste: Beliebig.

Meta-Inhalt:

* KonfigTyp**:** [Social Sharing] Die Art der Konfiguration. Muss immer diesen Wert haben.
* ButtonsAktiv**:** [Ja|Nein] Aktivierung der Social Sharing Buttons.
* ShareURL**:** Die URL, die geteilt werden soll. Kann leer gelassen werden.

#### Konfiguration: Overlays

Meta-Spur und -Element: *#!konfig*, Callout

ePlayer: Nicht unterstützt.

Positionierung auf Zeitleiste: Beliebig.

Meta-Inhalt:

* KonfigTyp**:** [Overlays] Die Art der Konfiguration. Muss immer diesen Wert haben.
* Standardmäßig Aktiv?**:** [Ja|Nein] Ob Overlays standardmäßig eingeschaltet sind oder nicht.
* Ausschaltbar?**:** [Ja|Nein] Ob Overlays vom Benutzer ausgeschaltet werden können oder nicht.

#### Medien

Meta-Spur und -Element: *#!meta*, Callout

ePlayer: Unterstützt.

Positionierung auf Zeitleiste: Beliebig.

Meta-Inhalt:

* Metadaten Typ**:** [Medium] Die Art des Metadateneintrags. Muss immer diesen Wert haben.
* Mediendatei URL: [bspw. authcam.mp4] Die Quelle der Mediendatei.
* MIME Typ: [bspw. video/mp4] Der MIME Type der Mediendatei.

#### Zusätzliche Titel bzw. Beschreibungen

Normalerweise wird Titel und Beschreibungstext unter *Ansicht → Projekt → Projektinformationen* konfiguriert. Bei einem mehrsprachigen Video müssen allerdings zusätzliche Titel und Beschreibungstexte festgelegt werden.

Meta-Spur und -Element: *#!meta*, Callout

ePlayer: Unterstützt.

Positionierung auf Zeitleiste: Beliebig.

Meta-Inhalt:

* Metadaten Typ**:** [Titel bzw. Beschreibung] Die Art des Metadateneintrags. Muss immer diesen Wert haben.
* Titel bzw. Beschreibung: Der Titel bzw. Beschreibungstext.

#### Kapitel

**Meta-Spur und -Element**: *#!kapitel*, Callout

**ePlayer**: Unterstützt, aber nur mit Titel.

**Positionierung auf Zeitleiste:** Die Kapitelposition und -dauer wird anhand der Platzierung bestimmt.

Das Kapitel wird einer Kategorie zugeordnet, indem es sich Zeitlich im selben Zeitleistenbereich wie das Kategorielement befindet.

**Meta-Inhalt:**

* *Formatierbarer Text*: Beschreibungstext des Kapitels. HTML möglich. Kann leer gelassen werden.
* Titel: Titel des Kapitels.
* Tooltip: Hinweistext, kann leer gelassen werden.
* Tags: Schlüsselwörter wodurch dieses Kapitel beim Suchen schnell gefunden werden kann. Kann leer gelassen werden.

#### Kategorie

**Meta-Spur und -Element:** *#!kategorien*, Callout

**ePlayer**: Nicht unterstützt.

**Positionierung auf Zeitleiste:** Die Kategorieposition und -dauer wird anhand der Platzierung bestimmt.

**Meta-Inhalt:**

* *Formatierbarer Text*: Beschreibungstext der Kategorie. HTML möglich. Kann leer gelassen werden.
* Titel: Titel der Kategorie.
* Tooltip: Hinweistext, kann leer gelassen werden.

#### Autorennotiz

Meta-Spur und -Element: *#!notizen*, Callout

ePlayer: Unterstützt, aber nur Titel und *Formatierbarer Text*.

Positionierung auf Zeitleiste: Die Notizposition und -dauer wird anhand der Platzierung bestimmt.

Meta-Inhalt:

* *Formatierbarer Text*: Beschreibungstext der Notiz. HTML möglich.
* Titel: Titel der Notiz.
* Tooltip: Hinweistext, kann leer gelassen werden.
* Länge übernehmen?: [Ja|Nein] Anzeige über den entsprechenden Bereich der Zeitleiste, statt nur als Punkt.
* In Zeitleiste anzeigen?: [Ja|Nein] Anzeige in der Zeitleiste des Players.
* In Kapiteln anzeigen?: [Ja|Nein] Wenn diese Option aktiviert ist, dann ist die Position der Notiz abhängig von dem zuzuordnenden Kapitel und wird mit im Inhaltsverzeichnis abgebildet.
* Auf Bildschirm anzeigen?: [Ja|Nein] Ob für den Benutzer beim Erreichen der Zeitleistenposition ein Hinweis auf die Notiz eingeblendet wird.

#### Linkliste (zusätzlicher Kapitelinhalt)

Die Linkliste ist ein Anhang an ein Kapitel und somit von der Positionierung eines Kapitels abhängig. Eine Linkliste muss einem Kapitel zugeordnet werden können.

Die Liste wird als Gruppenelement auf der Zeitleiste repräsentiert. Sie besteht aus einem Element das dem Listenkopf entspricht und weiteren Elemente die je einen Link definieren.

Meta-Spur und -Element: *#!k-extras*, Gruppe (Name: #!linkliste, Inhalt: min. zwei Callouts)

ePlayer: Unterstützt, aber nicht Tooltip.

Positionierung auf Zeitleiste: Die Linkliste wird einem Kapitel zugeordnet, indem es sich Zeitlich im selben Zeitleistenbereich wie das Kapitelelement befindet.

Meta-Inhalt:

1. Listenkopf:

* *Formatierbarer Text*: Beschreibungstext der Liste. HTML möglich.
* Titel: Titel der Linkliste.
* Tooltip: Hinweistext, kann leer gelassen werden.

1. Link:

* Titel: Anzeigetitel des Links.
* URL: Zielseite des Links.
* Tooltip: Hinweistext, kann leer gelassen werden.

#### Autorenkamera

Meta-Spur und -Element: *#!autkam*, Gruppe (Name: beliebig, Inhalt min. zwei Callouts)

Besonderheit: Ein bis beliebig viele Callouts definieren die zu verwendenden Mediendateien und ein Callout definiert die Positionierung der Autorenkamera. Es darf nur eine Gruppe auf der Spur geben.

Positionierung auf Zeitleiste: Je nach Positionierung auf der Zeitleiste, wird die Autorenkamera eingeblendet.

Meta-Inhalt:

1. Positionierungs-Callout

* Positionsklasse: Positionierung des Overlays. Positionsklassen werden durch den jeweiligen Player definiert.
* Verschiebbar?: [Ja|Nein] Der Benutzer kann die Position der Autorenkamera verändern.
* Skalierbar?: [Ja|Nein] Der Benutzer kann die Größe der Autorenkamera verändern.
* Ausschaltbar?: [Ja|Nein] Der Benutzer kann die Autorenkamera deaktivieren.

1. Mediendatei-Callouts

* Mediendatei URL: [bspw. authcam.mp4] Die Quelle der Mediendatei.
* MIME Typ: [bspw. video/mp4] Der MIME Type der Mediendatei.
* Sprache: [bspw. de] Die Sprache der Mediendatei.

#### Overlay

Zusätzlicher Inhalt der über das Video gelegt wird. Die entsprechende Aktion wird in Camtasia über die Hotspot Eigenschaften konfiguriert. Das Overlay ist nur interaktiv, wenn es als Hotspot konfiguriert wird.

Meta-Spur und -Element: *#!overlays*, Callout

**ePlayer**: Nicht unterstützt.

Positionierung auf Zeitleiste: Je nach Positionierung auf der Zeitleiste, wird das Overlay eingeblendet.

Meta-Inhalt:

* *Formatierbarer Text*: Inhaltstext des Overlays. HTML möglich.
* Stilklasse: Verändert den Stil des Callouts. Stilklassen werden durch den jeweiligen Player definiert.
* Positionsklasse: Positionierung des Overlays. Positionsklassen werden durch den jeweiligen Player definiert.
* Auf Aktion warten?: [Ja|Nein] Hält das Video an und wartet auf Benutzerinteraktion.
* Bei Aktion schließen?: [Ja|Nein] Schließt das Overlay bei Benutzerinteraktion.
* Erzwinge Anzeige?: [Ja|Nein] Erzwingt die Anzeige des Overlays, egal ob der Benutzer die Anzeige von Overlays unterbindet.
* Kopierbarer Inhalt?: [Ja|Nein] Erlaubt dem Benutzer den Inhalt des Overlays in die Zwischenablage zu kopieren.
* Automatisch schließen?: [Ja|Nein] Das Overlay schließt sich automatisch, nach Ablauf der definierten Länge auf der Zeitleiste.
* Tooltip: Hinweistext, kann leer gelassen werden.

#### Untertitel

Meta-Spur und -Element: *#!utitel*

Meta-Element: Beschriftungen

Positionierung auf Zeitleiste: Je nach Positionierung auf der Zeitleiste, werden die jeweiligen Untertitel eingeblendet.

Meta-Inhalt: keine

#### How to: Erstellen eines Inhaltsverzeichnisses

**ePlayer**: Teilweise unterstützt, der ePlayer kann das Video nur eindimensional gliedern.

Das zweidimensionale Inhaltsverzeichnis wird durch eine Zusammensetzung von Kategorien und Kapiteln oder nur durch Kapitel (eindimensional) erzeugt. Hierbei gilt:

Kategorie (Ebene 1) und Kapitel (Ebene 2) oder nur Kapitel (Ebene 1).

#### How to: Umsetzung von Mehrsprachigkeit

**ePlayer**: Unterstützt.

Die Mehrsprachigkeit gilt für sämtliche Spuren, **außer** *#!konfig*, und *#!autkam*.   
Hierfür muss eine separate Spur erzeugt werden z.B. *#!kapitel[en]*.

Der Standardfall z.B. #!kapitel ist äquivalent zu #!kapitel[de]. Es sei denn innerhalb der Projekteinstellungen wurde die Sprache des Projekts geändert: Ansicht > Projekt > Projektinformationen > Sprache.

#### How to: Erstellung eines Literaturhinweises

**ePlayer**: Unterstützt, außer Option 3.

Option 1**:** Als Autorennotiz mit <a> Tag, siehe: Autorennotiz.

Option 2: Als Linkliste, die im Kapitel hinterlegt wird, siehe: Linkliste (zusätzlicher Kapitelinhalt).

Option 3: Im Beschreibungstext des Kapitels mit <a> Tag. Liegt im Gegensatz zu Opt.2 nicht in der Kapitelstruktur vor.

#### How to: iFrame einbinden

Um einen iFrame einzubinden, muss nur der entsprechende HTML Code als Inhalts- / Beschreibungstext in das Element eingefügt werden. Vor allem bei Autorenkameras sinnvoll.

## Transformieren zu JSON

Mit dem Tool *Camtasia2Json* können Sie nun ihr mit Metadaten angereichertes Camtasia Studio Projekt in eine JSON Datei konvertieren, die dann im ePlayer oder einen anderen Player, der dieses Format unterstützt, Verwendung finden kann. Beachten Sie dass das Tool *Camtasia2Json* das *Java Runtime Environment* der Version 8 benötigt.

Wenn Sie eine entsprechende Projektvorlage verwendet haben, dann ist die Ausführung von *Camtasia2Json* sehr einfach: machen Sie einfach einen Doppelklick auf die Convert to JSON.bat. Bei erfolgreicher Transformation findet ist das Ergebnis als Datei 1.json im Ordner *publish* zu finden.

Um das Tool manuell auszuführen, rufen Sie es von der Kommandozeile mit: java –jar CamtasiaToJson.jar /? auf. Sie erhalten dann eine Befehlsübersicht.

## Publizieren des Videos

Alle Dateien die sich nun im Ordner *publish* befinden sind für die Publikation vorgesehen und können auf einem oder mehreren entsprechenden Webserver/n zur Verfügung gestellt werden.

Beachten Sie, dass bei der Verwendung des ePlayers die Kategorisierung und Zusammenstellung der Playliste von globalen Konfigurationsdateien bestimmt wird, die nicht vom *Camtasia2Json* bereitgestellt werden. Wie diese letztendlich zu konfigurieren sind, entnehmen Sie bitte der Dokumentation des ePlayers.

## Probleme und Fehlerquellen

Beim Testen mit dem ePlayer fielen in der aktuellen Version (stand 24.09.15) folgende Fehler auf:

* Kapitel und Autorennotizen werden in der Kapitelstruktur nur Angezeigt, wenn sie in deutscher Sprache vorliegen (auch dann, wenn die Standardsprache des Videos Englisch ist).
* Untertitel müssen immer einen zeitlichen Abstand von ca. 500ms haben, sonst hält der ePlayer die Untertitel für Überlappungen und stellt nicht alle Untertitel dar. Dieser Abstand wird von *Camtasia2Json* automatisch eingefügt.
* Zu einem Kapitel muss immer zwingend mindestens ein Chart existieren, sonst erzeugt der ePlayer viele Warnungen auf der Konsole und stellt beim Ändern der Abspielposition auf der Zeitleiste einen leeren Thumbnail dar.

Camtasia Studio

* Die Projektdateien von Camtasia Studio sind nicht immer wohlgeformte XML-Dokumente, z.B. kann ein Kaufmann-Und (&), das in einem Freitextfeld in den Projektinformationen eingegeben wurde, verhindern dass *Camtasia2Json* die Projektdatei richtig einliest, da dieses nur wohlgeformtes XML unterstützt.

# Anhang

## Camtasia Projektformat

Es folgt eine grobe Analyse für das Projekt-Dateiformat von TechSmith Camtasia Studio in der Version 8.5.1.

Zu dem vorliegenden Format wird – auch auf Nachfrage seitens des TechSmith Kundenservices – keine Dokumentation zur Verfügung gestellt, weshalb für die vorliegenden Analyseergebnisse keine Gewährleistung für Vollständigkeit und Richtigkeit übernommen werden kann.

Es wurden nur jene Features des Dateiformats analysiert, die für eine Zweckentfremdung in Frage kommen. Dies beinhaltet allgemeine Metadaten zum Camtasia Studio Projekt und sämtliche Daten von Zeitleistenelementen, d.h. Spuren und Medien.

Um auf bestimmte Knoten im XML-Baum zu verweisen, wird XPath 2.0 verwendet.

**Allgemeines zum Format**

Die Datei liegt als wohlgeformtes XML 1.0 Dokument in der Kodierung UTF-8 vor. Es werden keine XML Erweiterungen oder Namensräume verwendet. Es liegt kein bekanntes DTD oder XML-Schema vor.

Für formatierbare Texte (wie z.B. in Callouts) wird das Rich Text Format[[3]](#footnote-4) verwendet.

### Projektmetadaten

<Project\_Data Version="8.00">

<Project\_Settings>

<ProjectWidth>640</ProjectWidth>

<ProjectHeight>360</ProjectHeight>

[…]

</Project\_Settings>

<ProjectID>36E9FE81-3E06-4668-ABB9-09848851D461</ProjectID>

<Project\_MetaData>

<Project\_MetaData\_Object>

<FieldArrayKey>8</FieldArrayKey>

<Value>Beispielprojekt</Value>

</Project\_MetaData\_Object>

[…]

</Project\_MetaData>

<CSMLData>

<GoProject id="1" version="3.0" >

<Project id="2" editRate="30/1" version="3.0">

[…]

<SourceBin id="3">

<Source id="4" src="C:\AnyFile.png" lastMod="20150724T140805" rect="(0,0,644,364)">640</GoProject>

</SourceBin>

[…]

</Project>

</GoProject>

</CSMLData>

[…]

</Project\_Data>

Der Wurzelknoten Project\_Data ist mit dem Attribut Version versehen, welches die Dateiversion des Camtasia Projekts wiederspiegelt.

Die ersten Kindknoten von Project\_Data repräsentieren einige relevante Daten zum Projekt:

|  |  |
| --- | --- |
| Knoten | Bedeutung |
| Project\_Settings/  ProjectWidth |  Project\_Settings/  ProjectHeight | Die Auflösung des Ausgabevideos. Kann 0 sein, wenn das Projekt noch keine Medien beinhaltet. Wird normalerweise automatisch bestimmt, kann aber auch in Camtasia Studio Konfiguriert werden. Entspricht nicht zwingend der Auflösung in dem das Video exportiert wird. |
| ProjectID | Eindeutige Projekt ID (GUID). |
| Project\_MetaData | Auflistung von beliebig vielen Project\_MetaData\_Object Knoten. |
| Project\_MetaData\_Object | Schlüssel- (FieldArrayKey) und Wertepaar (Value) für eine Metadateneigenschaft. Bekannte Schlüsselwerte:   * 8 – Titel * 9 – Thema * 10 – Kategorie * 11 – Schlüsselwörter (Zeilenumbrüche möglich) * 12 – Beschreibung (Zeilenumbrüche möglich) * 13 – Erstelldatum und Zeit (geprüftes Format, kein Freitext) * 14 – Format * 15 – Ressourcenkennung * 16 – Sprache (Freitext) * 17 – Relation * 18 – Quelle * 19 – Ressourcentyp * 20 – Abdeckung |
| CSMLData/GoProject/Project/  SourceBin | Dies sind Dateien die in Camtasia Studio im Clip-Auswahlfenster hinterlegt sind. Einige Medien verweisen auf diese Dateien anhand ihrer ID. |
| CSMLData/GoProject/Project/  SourceBin/Source | Eintrag für eine Mediendatei. Relevante Attribute:   * id – Medien auf der Zeitleiste verweisen teilweise auf diese ID. * src – Dateipfad der Quelldatei. Immer ein absoluter Pfad. |

### Zeitleiste

<Timeline id="11">

<Attributes>

<Attribute id="12" value="640" name="width" />

<Attribute id="13" value="360" name="height" />

<Attribute id="14" value="(0,0,0,255)" name="backgroundColor" />

</Attributes>

<GenericMixer id="15" name="Unified Mixer">

[…]

<Tracks>

[…]

</Tracks>

</GenericMixer>

<CaptionAttributes id="1888">

<Attribute id="1889" value="1" name="captionsEnabled" />

</CaptionAttributes>

<Parameters>

<Parameter id="1915" name="toc">

<Keyframes>

<Keyframe id="1916" time="0/1" value="Marker 1" leadInLength="0/1">

[…]

</Keyframe>

</Keyframes>

</Parameter>

<Parameter id="1917" name="quiz">

<Keyframes>

<Keyframe id="1918" time="286/1" value"[…]" leadInLength="0/1" />

</Keyframes>

</Parameter>

</Parameters>

</Timeline>

Es gibt immer nur eine Zeitleiste, wobei die ID variieren kann.   
Relevante Unterknoten von Project\_Data/CSMLData/GoProject/Project/Timeline:

|  |  |
| --- | --- |
| Knoten | Bedeutung |
| Attributes | Eigenschaften der Zeitleiste. |
| Attributes/Attribute | Schlüssel- (name) und Wertepaar (value) für eine Metadateneigenschaft. Bekannte Schlüsselwerte:   * width, height – Auflösung, wie in Project\_Settings. * backgroundColor – Farbe des Hintergrunds der Szene. Format: (R,G,B,A). Alpha kann in der Benutzeroberfläche nicht festgelegt werden. |
| CaptionAttributes | Metainformationen zu Untertiteln. |
| CaptionAttributes/Attribute | Schlüssel- (name) und Wertepaar (value) für eine Untertitel Metadateneigenschaft. Bekannte Schlüsselwerte:   * captionsEnabled – 1 (ja), 0 (nein). * captionsFontName – Schriftname. * captionsFontSize – Schriftgröße in Punkten. * captionsBackgroundColor – Hintergrundfarbe (R,G,B,A). * captionsForegroundColor – Textfarbe (R,G,B,A). * captionsAlignment – Ausrichtung. 0 (standard), 1 (links), 2 (mitte), 3 (rechts) * captionsOpacity – Durchsichtigkeit * captionsBackgroundEnabled – Hintergrundfarbe 1 (ja), 0 (nein) * captionsBackgroundOnlyAroundText – (0) Hintergrundfarbe nur Hinter eigentlichem Text, nicht im gesamten Untertitelbereich. * lang – Sprachidentifikator (z.B. „de“) * captionsDefaultFontSize – Standardschriftgröße in Pt. (scheint immer 0 zu sein) |
| Parameters/Parameter[name="toc"]/Keyframes | Liste von Zeitleistenmarkern. |
| Parameters/Parameter[name="toc"]/Keyframes/Keyframe | Repräsentiert einen zeitleisten Marker[[4]](#footnote-5). Relevante Attribute:   * time – Position auf der Zeitleiste. * value – Bezeichnung. |
| Parameters/Parameter[name="quiz"/Keyframes | Liste von Quizze. |
| Parameters/Parameter[name="quiz"]/Keyframes/Keyframe | Repräsentiert ein Quiz. Relevante Attribute:   * time – Position auf der Zeitleiste. * value – String-Repräsentation von XML-Daten zu einem Quiz. Siehe Abschnitt „Quiz-XML“ für Details. |

#### Spuren

Spuren kommen nicht nur als Unterknoten von Zeitleisten vor, sondern auch bei Group-Elementen (Gruppierungen, siehe *Medien*) und bei Medien des Typs *StitchedMedia*.  
GenericTrack Knoten haben normalerweise einen GenericMixer als Elternknoten. Dies gilt jedoch nicht, wenn sie in einem *StitchedMedia* Knoten vorkommen. Für Details zu StitchedMedia Knoten, siehe *Medien*.

#### Spuren in Zeitleisten

<Timeline>

<GenericMixer>

[…]

<Tracks>

<GenericTrack id="16">

[…]

</GenericTrack>

</Tracks>

<GenericMixer>

</Timeline>

GenericTrack Knoten

<GenericTrack id="16">

<Attributes>

<Attribute id="17" value="Track Name" name="ident" />

<Attribute id="18" value="0" name="audioMuted" />

<Attribute id="19" value="0" name="videoHidden" />

</Attributes>

[…]

<Medias>

[…]

</Medias>

</GenericTrack>

|  |  |
| --- | --- |
| Knoten | Bedeutung |
| GenericMixer/GenericTrack | Eine Spur. |
| GenericMixer/GenericTrack/Attributes | Eigenschaften der Spur. |
| GenericMixer/GenericTrack/Attributes/Attribute | Schlüssel- (name) und Wertepaar (value) für eine Eigenschaft. Bekannte Schlüsselwerte:   * ident – Name der Spur (Freitext). * audioMuted – 0 (nein), 1 (ja). * videoHidden – 0 (nein), 1 (ja). |
| GenericMixer/GenericTrack/Medias | Sämtliche Elemente auf der Spur. |

### Medien

Als Medien werden die Elemente auf den Spuren der Zeitleiste bezeichnet.  
Medien sind Unterknoten von GenericTrack/Medias, sie treten immer in der Reihenfolge auf, in der sie in der Zeitleiste platziert sind, also chronologisch.

|  |  |
| --- | --- |
| Knoten | Bedeutung |
| Callouts | Callout |
| Caption | Untertitel |
| Group | Gruppe |
| IMFile | Bilddatei |
| AMFile | Audiodatei |
| VMFile | Videodatei |
| ScreenIMFile | Unbekannt (Bildschirmaufnahmenfoto?) |
| ScreenVMFile | Bildschirmaufnahmedatei (trec oder camrec) |
| StitchedMedia | Im Mittelteil geschnittene Audio- und Videoelemente. (Nicht wenn Medium am Anfang oder am Ende geschnitten wurde!) |
| UnifiedMedia | Ein- und Ausblendeffekte von Medien und zusammengesetzte Medien wie Audio- und Videomedien. |

Folgende Vektoren können relevant sein:

* start – Position auf der Zeitleiste (Framenummer).
* duration – Dauer auf der Zeitleiste (in Frames).
* src – ID der verwendeten Quelldatei (nur für Medien die Dateinhalte repräsentieren).

|  |  |
| --- | --- |
| Knoten | Bedeutung |
| Transformer | Liste von Transformationen (beschrieben durch VectorParam Knoten) die auf dieses Element angewendet werden.  Für jedes Element sind immer VectorParam Knoten mit den folgenden Namen gegeben: translation, rotation, shear, scale, anchor. |
| Transformer/VectorParam |  Parameters/VectorParam | Vektor dessen Komponenten durch InterpolatingParam Kindknoten beschrieben werden.  Relevante Attribute:   * name – Name des Vektors. Gibt üblicherweise an, wie der Vektor anzuwenden ist.   Folgende Vektoren können relevant sein:   * translation – Translations-Transformation (3 Parameter: x, y, z?). * rotation – Rotations-Transformation (3 Parameter: x, y, z?). * scale – Skalierungs-Transformation (3 Parameter: x, y, z?). * anchor – Ankerpunkt für rotationen (3 Parameter: x, y, z?). * sourceCrop – Beschneidung des Elementinhalts (4 Parameter: links, rechts, oben, unten). * geometryCrop – Beschneidung des Elements (Parameter wie sourceCrop). |
| Parameters/  InterpolatingParam |  Parameters/VectorParam/  InterpolatingParam | Relevante Attribute:   * name – Name des zu setzenden Parameters. Falls InterpolatingParam Kindelement von VectorParam ist, ist dieses Attribut nicht vorhanden. * value – Wert des Parameters. Bei animierten Medien nicht vorhanden, da dort die Daten in einem Keyframes Knoten liegen.   Folgende (benannte) Parameter sind relevant:   * opacity – Transparenz. Value liegt zwischen 0 (unsichtbar) und 1 (keine Transp.). * volume – Die Lautstärke eines Mediums. Value liegt zwischen 0 und 1 |
| Parameters/  InterpolatingParam/  Keyframes | Liste von Keyframes zur Animationssteuerung. Nur bei animierten Medien vorhanden. |
| Parameters/  InterpolatingParam/  Keyframes/Keyframe | Ein Keyframe gibt einen Wert an den der *VectorParam* zu einem bestimmten Zeitpunkt annehmen muss. Keyframes sind auch dann bei **jedem** *VectorParam* gegeben, wenn tatsächlich nur **ein** *VectorParam* in der Liste der Transformer animiert wird – diese eigentlich unnützen Keyframes haben dann die value 0.  Relevante Attribute:   * time – Zeitpunkt relativ zur Position des Mediums auf der Zeitleiste in Frames. Eine negative nummer gibt an, dass der Keyframe den Standardwert (nicht animiert) des Mediums entspricht. * value – Wert zu dem interpoliert werden muss. * leadInLength – Länge der Animation. Tatsächlich beginnt das Animieren an der Position time – leadInLength und endet bei time. * interpolater – Die Funktion die zur Interpolation genutzt werden soll (easing-function). Aktuell scheinen nur leadin-linr (für lineare Interpolation) und leadin-eios (für exponentielle Interpolation) möglich zu sein. |
| GenericMixer | Nur für Medien vom Typ *Group*. Enthält *GenericTrack* Knoten mit weiteren Medien. |
| Parameters | Liste von Parametern die visuelle Attribute des Mediums beschreiben.  Für jedes Element sind immer VectorParam Knoten mit den folgenden Namen gegeben: sourceCrop, geometryCrop.  Zusätzlich gibt es auch immer einen InterpolatingParam Knoten mit dem namen: opacity. |
| Effects | Liste von visuellen Effekten die zu bestimmten Zeitpunkten (Keyframe gesteuert) angewandt werden. Effekte beziehen sich nicht auf die Ausrichtung des Mediums sondern auf die Veränderungen des Inhalts.  Wichtig ist hierbei, dass ein Text – wenn er auf ein Medium vom Typ IMFile gelegt wird – immer als ein GenericEffect Knoten repräsentiert wird. Der Text bei einem Callout wird im Vergleich dazu ganz anders gehandhabt. |
| Attributes | Metainformationen zum Medium. |
| Attributes/Attribute | Schlüssel- (name) und Wertepaar (value) für eine Metadateneigenschaft. Bekannte Schlüsselwerte:   * ident – Name / Bezeichner für das Medium (Freitext). * vectorNode – Repräsentiert Daten zur Darstellung eines Vektorobjekts. Wird für sämtliche visuelle Inhalte von Callouts genutzt, z.B. Speech Bubbles, Text, SVG Grafiken etc. |
| LeftEdgeMods |  RightEdgeMods | Gibt Effekte an, die auf der linken bzw. rechten Seite der Zeitleiste auf dem Element angewandt werden sollen. In Camtasia Studio werden diese in den Callouts Eigenschaften durch die Ein- bzw. Ausblenden Slider konfiguriert.  Relevante Attribute:   * length – Dauer des Effekts in Frames. * groupName – Art des Effekts. Aktuell scheint es nur OpacityFade zu geben. |
| ExtraData | Liste zusätzlicher Metadaten. |
| ExtraData/Entry | Ein Metadateneintrag.  Relevante Attribute:   * key – Name des Eintrags.   Folgende Einträge können relevant sein:   * HotspotInfo – Daten zu einem Hotspot. Eintrag ist auch dann vorhanden, wenn die „Hotspot erstellen“ in Camtasia Studio nicht angewählt ist - der Entry Knoten enthält dann allerdings keinen Kindknoten. |
| ExtraData/Entry/  HotspotInfo | Enthält sämtliche Hotspot-Daten.  Relevante Attribute:   * action – Art der Aktion die beim klicken auf den Hotspot ausgelöst werden soll.   + 0 steht für „Zum fortfahren Klicken“   + 1 für „Gehe zum Zeitpunkt“   + 2 für „URL öffnen“   + 3 für „Gehe zu Marker“ * pauseAtEnd – Ob am Ende des Callouts pausiert wird, Wert 0 oder 1. * gotoTime – Frame zu dem gesprungen werden soll. * url – Zu öffnende URL. * openUrlInNewWindow – Wert 0 oder 1. * markerTime – Frame des Markers zu dem gesprungen werden soll. |

#### Vector Node

|  |  |
| --- | --- |
| Knoten | Bedeutung |
| VectorNode | Daten zur Darstellung eines Vektorobjekts. Jedes Callout besitzt einen solchen Knoten. |
| VectorNode/DoubleParameters | Liste von numerischen Parametern. |
| VectorNode/DoubleParameters/InterpolatingParam | Relevante Attribute:   * name – Name des zu setzenden Parameters. * value – Wert des Parameters.   Folgende Parameter können relevant sein:   * Style – Index eines definierten Stils. * textDropShadow – Textschatten. Value ist 0 oder 1. * textVerticalAlign – Vertikale Textausrichtung. Value ist -1 für Oben, 0 für Mittig und 1 für Unten. |
| VectorNode/StringParameters | Liste von Zeichenfolgen-Parametern. |
| VectorNode/StringParameters/Parameter | Relevante Attribute:   * name – Name des zu setzenden Parameters.   Diese Parameter enthalten typischerweise Keyframes Knoten, die Werte der Parameter sind also animierbar. |
| VectorNode/StringParameters/Parameter/Keyframes/Keyframe | Ein Keyframe gibt einen Wert an den der Parameter zu einem bestimmten Zeitpunkt annehmen muss.  Relevante Attribute:   * time – Zeitpunkt relativ zur Position des Mediums auf der Zeitleiste in Frames. * value – Wert des Parameters.   Folgende Parameter sind relevant:   * text – Der Inhalt eines Mediums das Text enthalten kann im RTF-Format. Kommt bei Medien des typs *Callout* immer vor. |

### Quiz-XML

<Quiz id="1001" name="Das Programmiererquiz">

<Options feedback="1" graded="1" />

<Questions>

[…]

</Questions>

</Quiz>

|  |  |
| --- | --- |
| Knoten | Bedeutung |
| Quiz | Wurzelknoten. Attribut name gibt den Namen des Quiz an. |
| Quiz/Options | Einstellungen zum Quiz. Relevante Attribute:   * feedback – 1 (Antworten werden nach der Teilnahme angezeigt), 0 (keine antworten Anzeige) * graded – 1 (Quiz wird ausgewertet), 0 (keine Auswertung) |
| Quiz/Questions | Alle Fragen zum Quiz. |

#### Frageknoten

<MultipleChoice id="101" text="Welche Sprache ist nicht Objektorientiert?" scored="1">

<Answers>

<Answer id="1" text="C#" iscorrect="0" />

<Answer id="11" text="Java" iscorrect="0" />

<Answer id="12" text="C++" iscorrect="0" />

<Answer id="13" text="JavaScript" iscorrect="1" />

<Answer id="14" text="PHP5" iscorrect="0" />

</Answers>

</MultipleChoice>

<FillBlank id="104" text="Mit welcher Anweisung wird in Java typischerweise eine Verzweigung implementiert?" scored="1">

<Answers>

<Answer id="5" text="if" iscorrect="1" />

<Answer id="15" text="switch" iscorrect="1" />

<Answer id="17" text="?:" iscorrect="1" />

</Answers>

</FillBlank>

<Logical id="113" text="Wait Handles werden zur synchonisation von Threads eingesetzt." scored="0" istrue="1" />

<ShortAnswer id="106" text="Bitte geben Sie uns ein kurzes Feedback zum Quiz." scored="1" />

|  |  |
| --- | --- |
| Knoten | Bedeutung |
| MultipleChoice | Frage mit mehreren vorgegebenen Antwortmöglichkeiten. Relevante Attribute:   * text – Fragentext (Freitext). |
| FillBlank | Frage mit mehreren Antwortmöglichkeiten, die Antwort muss allerdings eingegeben werden. Sie ist dann richtig, wenn sie in der Liste der Antwortmöglichkeiten vorkommt. |
| MultipleChoice/Answers |  FillBlank/Answers | Alle Antworten zur Frage. |
| MultipleChoice/Answers/Answer |  FillBlank/Answers/Answer | Eine Antwortmöglichkeit. Relevante Attribute:   * text – Antworttext (Freitext). * iscorrect – 1 (dies ist die richtige Antwort), 0 (falsche Antwort). |
| ShortAnswer | Frage mit Freitextfeld als Antwort. |
| Logical | Frage mit Ja/Nein Auswahl als Antwort. Relevante Attribute:   * istrue – 1 (ja ist dir richtige Antwort), 0 (nein ist die richtige Antwort). |

## Datenbasis

Als Grundlage wird das Dateiformat des ePlayers in der Version 1.0 verwendet.

Es folgt eine ePlayer typische JSON Datei auf der linken Seite. Erweiterungen des Formats, zur Abdeckung unserer neuen Features, werden durch eine entsprechende Markierung gekennzeichnet. Der ePlayer ignoriert dabei jegliche nachträglich hinzugefügte JSON Attribute und bleibt somit kompatibel.

Auf der rechten Seite folgt eine Erläuterung der Werte die an den gegebenen Platzhaltern eingetragen werden sollen. Da keine Definition des ePlayer Formates vorliegt, wurden auch die durch den ePlayer definierten Attribute analysiert und miterläutert. Ein paar wenige Features des ePlayer-Formats werden von uns nicht verwendet oder anders umgesetzt, die Erläuterungen dazu sind entsprechend hervorgehoben.

|  |  |
| --- | --- |
| {  "id": 1,  "meta": {  "version": "1.0",  "cat": [Video Kategorie ID],  "generator": "[Generator Info]",  "dur": [Dauer],  "author": "[Autor]",  "date": "[Datum]",  "defaultLang": "[Standardsprache]",  "contact": "[Kontaktinformation]",  "socialShareUrl": "[Social Share Url]",  "enableSocialShareButtons": [SS Buttons?],  "tags: "[Tags]",  "poster": "[Poster Url]",  "titles": [  {  "lang": "[Sprache]",  "title": "[Text]"  }  ],  "descriptions": [  {  "lang": "[Sprache]",  "description": "[Text]"  }  ]  }  "authorNotes": [  {  "begin": [Sekunde des Beginns],  "dur": [Dauer der Notiz],  "title": "[Titel]",  "lang": "[Sprache]",  "content": "[Text]",  "tooltip": "[Tooltip Text]",  "chapter": [Kapitel Id],  "displayInTimeline": "[In Zeitleiste?]",  "displayInChapters": "[In Kapiteln?]",  "displayOnScreen": "[Auf Bildschirm?]"  },  ],  "captions": [  {  "begin": [Sekunde des Beginns],  "dur": [Dauer in Sekunden],  "lang": "[Sprache]",  "content": "[Text]"  }  ],  [In Zeitleiste?] Ob die Autorennotiz in der Zeitleiste sichtbar und anklickbar ist. Bool. Wert.  [In Kapiteln?] Ob die Autorennotiz als Teil der Kapitelstruktur angezeigt wird. Boolescher Wert.  [Auf Bildschirm?] Ob die Autorennotiz, sobald das Video die entsprechende Position erreicht hat, vom Player automatisch unübersehbar angezeigt wird (bspw. neben dem Player). Boolescher Wert.  [Tooltip Text] Hinweistext der beim hovern mit der Maus über das Element angezeigt wird  (title attribut in HTML).  [Kapitel Id] Id Referenz auf die Kategorie dem diese Notiz zugeordnet ist. Die Kapitel sind im chapters JSON Array definiert. | [Video Kategorie ID] ID Referenz auf die Kategorie des Videos. Die Kategorien werden durch globale Einstellungen des Players konfiguriert.  [Generator Info] Optional. Falls diese JSON Datei generiert wurde, stehen hier Versionsinformationen zum jeweiligen Tool.  [Dauer] Länge des gesamten Videos in Sekunden.  [Autor] Der Name des Videoautors. Besser: ID Referenz auf einen Autor der an einer globalen Stelle näher definiert wird.  [Datum] Datum des Videos im Format YYYY-MM-DD.  [Standardsprache] ID der Standardsprache.  [Kontaktinformation] Optional. Informationen wie mit dem Autor in Verbindung getreten werden kann. Vom ePlayer z.Z. ignoriert. Besser: Attribut streichen. Player sollte dies besser über die Autor ID ermitteln.  [Tags] Tags bzw. Schlüsselwörter die dabei helfen, das Video zu finden. Durch Leerzeichen getrennt.  [Kontakt Url] Url zum Kontaktieren des Autors. Kann auch eine mailto Adresse sein.  [Social Share Url] Basis-Url auf die sämtliche social share Buttons zum Teilen verwenden sollen.  [SS Buttons?] Social Share Buttons für dieses Video aktiviert? Boolescher Wert.  [Poster Url] Optional. Url zu dem Bild das vor beginn des Videos als Platzhalter angezeigt wird.  [Sprache] Sprachcode [[5]](#footnote-6)der Sprache. JSON-Objekte mit diesem Attribut können mehrmals, in verschiedenen Sprachen, vorkommen. Der Player wählt dann die entsprechende Sprache aus. Wird das Attribut nicht angegeben, geht der Player von de aus.  [Titel], [Text] Freier Text. Zeilenumbrüche in Form von \n werden entfernt ohne dass ein Zeilenumbruch eingefügt wird, dies gilt auch für andere Maskierungen mit \. Es werden jedoch beliebige HTML Tags unterstützt.  [Dauer der Notiz] Autorennotizen können zusätzlich auch mit einer Dauer angegeben werden. Hiermit kann der Autor auch bestimmte Abschnitte in seinem Video hervorheben, oder eine Notiz nur für eine bestimmte Dauer neben dem Video anzeigen lassen. |
| "chapters": [  {  "id": 1,  "lang": "[Sprache]",  "title": "[Titel]",  "begin": [Sekunde des Beginns],  "dur": [Dauer in Sekunden],  "videoType": "digital",  "description": "[Beschreibung]"  "category": "[Kategorie Id]",  "tooltip": "[Tooltip Text]",  "tags": "[Tags]",  "charts": [  {  "id": 1,  "begin": [Sekunde des Beginns],  "dur": [Dauer in Sekunden],  "title": "[Titel]",  "thumbnail": "[Thumbnail Url]"  },  ],  "additionals": [  {  "type": "[Zusatz-Typ]",  "title": "[Titel]",  "description": "[Text]",  "tooltip": "[Tooltip Text]",  "links": [  {  "title": "[Titel]",  "href": "[Link Url]",  "tooltip": "[Tooltip Text]"  }  ]  }  ]  }  ],  "categories": [  {  "id": 1,  "lang": "[Sprache]",  "title": "[Titel]",  "begin": [Sekunde des Beginns],  "dur": [Dauer in Sekunden],  "description": "[Beschreibung]",  "tooltip": "[Tooltip Text]"  }  ],  "authCam": {  "begin": [Sekunde des Beginns],  "dur": [Dauer in Sekunden],  "isMoveable": [Verschiebbar?],  "isResizeable": [Skalierbar?],  "isSwitchable": [Ausschaltbar?],  "position": "[Positionsklasse]",  "w": [Größe],  "h": [Größe],  "translateTransform": [x, y, z],  "rotateTransform": [x, y, z],  "shearTransform": [x, y, z],  "opacity": [Transparenz],  "fadeInDuration": [Dauer in Sek.],  "fadeOutDuration": [Dauer in Sek.],  "media": [  {  "src": "[Url abspielbarer Datei]",  "type": "[MIME Type]",  "lang": "[Sprache]"  }]  }, | [Kategorie Id] Id Referenz auf die Kategorie dem dieses Kapitel zugeordnet ist. Die Kategorien sind im categories JSON Array definiert.  [Thumbnail Url] Url zu einem Bild das, beim hovern mit der Maus über der Zeitleiste, als Vorschaubild angezeigt werden soll. Besser: charts komplett streichen. Thumbnails nicht als Metadaten und Bilddateien hinterlegen, sondern vom Player automatisch aus dem Video extrahieren lassen.  [Zusatz-Typ] Art des Kapitelzusatzes. Aktuell ist nur linklist bekannt welche eine Liste von verlinkten Literaturquellen zu einem Kapitel darstellt.  [Beschreibung] Beschreibungstext mit HTML unterstützung.  [Verschiebbar?] Ob das Autorenvideo vom Benutzer verschoben werden kann. Boolescher Wert.  [Skalierbar?] Ob das Autorenvideo vom Benutzer skaliert werden kann. Boolescher Wert.  [Ausschaltbar?] Ob das Autorenvideo vom Benutzer ein- und ausgeblendet werden kann.  [Positionsklasse] Name einer vom Player definierten Klasse die die Art der Positionierung von Authcams und Overlays vorgibt. Die Standardpositionierung sieht folgendes Verhalten vor:   * Der Ankerpunkt von Elementen ist ihre horizontale und vertikale Mitte. * Der Koordinatenursprung (0, 0) befindet sich in der Mitte des Videofensters.   Der Player kann Positionsklassen vorsehen die es beispielsweise erlauben ein Element anhand der linken oberen Ecke sowie derselben Ecke des Videofensters auszurichten. |

|  |  |
| --- | --- |
| "overlaySettings": {  "isEnabledByDefault": [Standardm. Aktiv?],  "isSwitchable": [Ausschaltbar?]  },  "overlays" [  {  "id": 1,  "lang": "[Sprache]",  "begin": [Sekunde des Beginns],  "dur": [Overlaydauer],  "content": "[Text]",  "isCopyableContent": [Kopierbar?],  "style": "[Stilklasse]",  "position": "[Positionsklasse]",  "action": "[Aktionstyp]",  "actionParams": {  "gotoPos": [Zielsekunde]  "href": "[Ziel Url]",  "inNewWindow": [Neues Fenster?],  "links": [  {  "title": "[Titel]",  "href": "[Link Url]",  "tooltip": "[Tooltip Text]"  }  ]  },  "waitForAction": [Auf Aktion warten?],  "closeOnAction": [Bei Aktion. schl.?],  "closeButton": [Schließen Button?],  "forceVisibility": [Erzwingte Anzeige?],  "w": [Größe],  "h": [Größe],  "translateTransform": [x, y, z],  "rotateTransform": [x, y, z],  "shearTransform": [x, y, z],  "opacity": [Transparenz],  "tooltip": "[Tooltip Text]",  "fadeInDuration": [Dauer in Sek.],  "fadeOutDuration": [Dauer in Sek.]  }  ],  "captionSettings": {  "fontName": "[Schriftart]",  "backgroundColor": [Farbe],  "foregroundColor": [Farbe],  "alignment": "[Ausrichtung]",  "opacity": [Transparenz],  "isBackgroundEnabled": [Hintergrundfarbe?],  "isBackgroundOnlyAroundText":  [Hintergrundfarbe nur um Text herum?]  },  "media": {  "digital": [  {  "src": "[Url abspielbarer Datei]",  "type": "[MIME Type]",  "lang": "[Sprache]"  }  ]  }  }  [Bei Aktion schl.?] Ob das Overlay beim Ausführen der Aktion (beim Anklicken) geschlossen wird oder geöffnet bleibt. Boolescher Wert.  [Schließen Button?] Ob der Player dem Benutzer erlaubt das Overlay zu schließen, ohne dessen Aktion auszuführen. Der Player kann hierbei bspw. beim Hovern über das Overlay einen zusätzlichen schließen Button an der oberen rechten Ecke darstellen. Boolescher Wert.  [MIME Type] Gibt das Format an in der eine Mediendatei vorliegt. JSON-Objekte mit diesem Attribut können mehrmals, mit unterschiedlichen MIME Typs, vorkommen. Der Player wählt ein entsp. unterstütztes Format zum abspielen aus. | [Standardm. Aktiv?] Ob sämtliche Overlays standardmäßig aktiviert sind oder nicht. Bool. Wert.  [Ausschaltbar?] Ob die Overlays vom Benutzer ein- und ausgeblendet werden können. Bool. Wert.  [Overlaydauer] Optional. Dauer in Sekunden. Wenn nicht gegeben, dann schließt sich das Overlay niemals automatisch.  [Kopierbar?] Ob der Inhalt kopierbar ist oder nicht (z.B. durch Selektion). Der Player kann hierfür aber auch z.B. einen zusätzlichen Kopierbutton im Overlay darstellen der dann den gesamten Inhalt kopiert. Boolescher Wert.  [Stilklasse] Der Name einer durch den Player vorgegebene Stilklasse (implementation z.B. durch CSS Klassen). Die Stilklasse bestimmt das Aussehen des Elements.  [Aktionstyp] Optional. Gibt den Namen einer durch den Player definierten Aktion an die durch klicken auf das Overlay ausgeführt wird.  Folgende Aktionen sind denkbar:   * none – Keine Aktion. * goto – Springt auf der Zeitleiste zu einer bestimmten Sekunde (Parameter gotoPos). * link – Hyperlink (Parameter href, type, inNewWindow). * continue – Führt keine tatsächliche Aktion aus, sondern setzt das Video nur beim anklicken fort. Funktioniert nur wenn [Auf Aktion warten?] true ist. * linklist – Beim Klick öffnet sich eine Liste von anklickbaren Links (Parameter links).   Die Parameter in den Klammern werden jeweils durch das actionParams JSON Objekt definiert.  [Auf Aktion Warten?] Ob der Player wartet, bis der Benutzer auf das Overlay geklickt hat bevor das Video fortgesetzt wird oder nicht. Boolescher Wert.  [Erzwingte Anzeige?] Ob das Overlay auch dann dargestellt wird, wenn Overlays vom Benutzer gänzlich ausgeschaltet worden sind.  Sinnvoll für z.B. weitere Social Sharing Optionen o.ä. am Ende des Videos. Boolescher Wert.  [Farbe] Als JSON Array in der Form [R, G, B, A].  [Transparenz] Transparenzwert von 0 (unsichtbar) bis 1 (sichtbar).  [Url abspielbarer Datei] Absolute oder relative URL zu einer abspielbaren Datei. |

### Anregungen zur Implementierung

Es folgen einige Anregungen zur Implementierung der neuen Features in einem Player.

Implementation einer Suchfunktion:

* Overlaytexte und Untertitel sollten in der aktuellen Sprache durchsucht werden können und gefundene Positionen auf der Zeitleiste markiert werden. Hierfür ist ggf. vorheriges Cachen der wichtigsten Worte in den Untertitel- bzw. Overlaydaten sinnvoll.
* Zudem können Kapitel hervorgehoben werden, wenn auf deren Tags Suchbegriffe zutreffen sollten.

Darstellung von [Text] Inhalten:

* Der Player kann das iframe Tag explizit unterstützen, d.h. es automatisch positionieren und stylen. Dies sollte insb. Innerhalb von Overlays gelten.
* Für code Tags kann der Player entweder ein Data-Attribut definieren mit dem die Sprache angegeben werden kann oder entsprechende CSS Klassen zur Formatierung erlauben.   
  Aufgrund dieser Angabe kann der Player dann den Text nach vorgegebenen Regeln automatisch formatieren, z.B. für das automatische Highlighting von Quellcode.

Ausrichtung von den Overlays / der Autorenkamera:

* Statt die Position und Ausrichtung anhand von Koordinaten zu beschreiben, werden diese mit Hilfe von Transformationen beschrieben - dies lässt sich in einem Player gut mit den CSS Attribut *transform* implementieren.
* Der Ankerpunkt jedes auszurichtenden Elements sollte sich in der Mitte des Elements befinden.
* Sämtliche Translationen beziehen sich auf die Mitte des Videos. Dies ermöglicht Elemente auch dann korrekte Auszurichten, wenn sich die Größe des Videofensters (z.B. durch den Vollbildmodus) ändert.

Daten die für einen Autor (wie er durch [Autor] referenziert wird) hinterlegt werden könnten:

* Name
* Titel (auch selbstdef. denkbar: z.B. „Angehender Master Medieninformatik“, „Der Bildtechnik-Coach“)
* Mini-Biographie
* E-Mail Adresse
* Twitter Konto (für Twitter „Folgen“ Button im Video)
* Facebook Profil Url (u.a. auch für Facebook „Folgen“ Button verwendbar)
* Fronter Profil Url
* Website Url
* Weitere Informationen Url

Die Autorendaten sollten in einer zusätzlichen, videounabhängigen Datenbank (z.B. eine JSON Datei) abgelegt sein.

1. eLecure Player, entwickelt von Marius Büscher im Rahmen eines Studentenprojekts. [↑](#footnote-ref-2)
2. <https://www.techsmith.de/tutorial-camtasia-8-dokumentation.html> [↑](#footnote-ref-3)
3. Rich Text Format. Ein von Microsoft eingführtes Dateiformat für formatierbare Texte. [↑](#footnote-ref-4)
4. Marker der direkt auf der Zeitleiste liegt und nicht an ein bestimmtes Element der Zeitleiste gebunden ist. [↑](#footnote-ref-5)
5. Wahrscheinlich RFC 1766 taugliches Format. z.B. „de“ oder „en-US“ [↑](#footnote-ref-6)