

# **IP-5: Apple TV**

## **Untertitel**

### **Semesterarbeit / Diplomarbeit von:**

Blumer, Joel

Zirn, Andrea

**FHNW**

**Hochschule für Technik**

**Studiengang:**

iCompetence

### **Betreuende Dozenten:**

Simon Schubiger

Christoph Stamm

**Windisch, 22. Dezember 2016**

**Summary**  
Text

**Vorwort**  
Text

## Table of contents

<b>Table of contents</b>	<b>4</b>
<b>1 Introduction</b>	<b>6</b>
1.1 Motivation	6
1.2 Scope of work	6
1.3 Approach	6
<b>2 State of Technology</b>	<b>7</b>
2.1 Apple TV 4	7
2.2 Horizon App: Functionalities	7
2.3 Streaming	7
2.3.1 Components	7
2.3.1.1 Codecs	7
2.3.3.2 Container	8
2.3.3.3 DRM	9
2.3.3.4 Server/Client	9
2.3.3.5 HLS Authoring	11
2.4 Xcode	11
2.4.1 Swift	11
2.4.2 TVML	11
<b>3 Prototype</b>	<b>14</b>
3.1 Architecture	14
3.1.1 Streaming setup	14
3.1.1.1 HTTP Live Streaming	14
3.1.1.1.1 FFmpeg	14
3.1.1.1.2 Adaptive Streaming on Demand	14
3.1.1.1.3 DRM	21
3.1.1.1.3.1 Enabling ssl	22
3.1.1.1.3.2 Authentication with DRM	22
3.1.1.1.4 Live Streaming	23
3.1.1.1.5 Integration of the infrastructure in UPC	23
3.1.2 Application	23
3.1.2.1 folder structure:	23
3.1.2.2 files:	24
3.2 User interface and interaction Design	25
3.2.1 User interface Design principles	26
3.2.1.1 Color concepts	26
3.2.1.2 Layout concepts	26

3.2.2	Interaction Design principles	26
3.2.2.1	Navigation concept	26
3.2.3	Guidelines	26
3.2.3.1	tvOS Human Interface Guideines	27
3.2.3.2	UPC	33
3.2.4	Benchmarking	33
3.2.5	Wireframes and Click-prototype	33
3.3	Usability	33
<b>4</b>	<b>Results</b>	<b>35</b>
4.1	Findings	35
4.2	Outlook	35
<b>5</b>	<b>References</b>	<b>36</b>
L1.	Literatur	36
L2.	Internet	36
L3.	Interviews	36
<b>6</b>	<b>List of figures</b>	<b>37</b>
	<b>Acknowledgement</b>	<b>37</b>
<b>A.</b>	<b>Appendix</b>	<b>38</b>
A1.	Appendix 1	38
A1.1.	Appendix 2	38
A1.2.	Appendix 2	38
A2.	Declaration of honesty	38

## **1 Introduction**

Text

### **1.1 Motivation**

Text

### **1.2 Scope of work**

Text

### **1.3 Approach**

Text

## 2 State of Technology

INTRODUCTION OF THE WHOLE TOPIC. INCL. SWIFT, TVML, HTTP LIVE STREAMING AND APPLE TV 4

### 2.1 Apple TV 4<sup>1</sup>

INPUT: Hardware information

### 2.2 Horizon App: Functionalities

//TBD

### 2.3 Streaming

//Was gehört dazu. Überleitung und Einleitung auf alle folgenden Abschnitte  
//Im folgenden werden aufbau und funktion erklärt und spezifisch auf HLS eingehen

For streaming audio and video content to AppleTV only HTTP Live Streaming is supported<sup>2</sup>. HTTP Live Streaming lets you send audio and video content over HTTP. You need an ordinary web server for playback on iOS-based devices. Live broadcasting and video on demand streaming are both supported. In HTTP Live Streaming the content will be delivered in several small files, called segments. These segments typically have a duration of around 10 seconds.

#### 2.3.1 Components

Text

##### 2.3.1.1 Codecs<sup>3</sup>

A codec encodes data streams or signals. The purpose of a codec consists of storing and transferring data. It is possible to encrypt the data. The decoder is capable to reverse the encoding for playback or further editing. Codecs are mainly used in video editing, video conferencing and of course, streaming media. We will

---

<sup>1</sup> [https://developer.apple.com/library/content/documentation/General/Conceptual/AppleTV\\_PG/index.html](https://developer.apple.com/library/content/documentation/General/Conceptual/AppleTV_PG/index.html) (02.01.2017)

<sup>2</sup> [https://developer.apple.com/library/content/documentation/General/Conceptual/AppleTV\\_PG/](https://developer.apple.com/library/content/documentation/General/Conceptual/AppleTV_PG/) (29.12.2016)

<sup>3</sup> <https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html> (29.12.2016)

take a further look at the supported video and audio codecs for HTTP Live Streaming.

HTTP Live Streaming only supports the H.264 video codec with the following profile levels:

- H.264 Main profile 3.1: Apple TV 2 and later
- H.264 Main Profile 4.0: Apple TV 3 and later
- H.264 High Profile 4.0: Apple TV 3 and later

Since we are developing for Apple TV 4, the High Profile will be fine for us.

Regarding the supported audio codecs, there are only two available:

- HE-AAC or AAC-LC, stereo
- MP3 (MPEG-1 Audio Layer 3), stereo

MP3 is a very well known codec, but in contrast with the AAC-Codec, MP3 needs more bits transferring the audio content than the AAC-Codec. Despite MP3 needs more bits for transferring, the quality is better with AAC. In consideration of these facts, the “GoTo” codec for audio on apple TV is AAC.

Which profiles are supported for which devices? Apple Doku

### 2.3.3.2 Container

A container is a metafile format. It consists of specifications of how different elements of data and metadata exists in a single computer file.

Container are used for multimedia purposes. A container can support multiple audio and video codecs, so it's important to not only know the container format of a file, but rather knowing the exact audio and video codec of the file.

HTTP Live Streaming supports the following container formats:

//TBD  
Webm  
MKV  
OGG  
(NUT?)



### 2.3.3.3 DRM<sup>4</sup>

Digital content is protected by copyright laws, however, it is very difficult and time consuming to catch law-breakers. DRM follows a different strategy. The Approach is to make it impossible to steal content in the first place by encrypting it and require authorization for accessing the files. Therefore DRM describes the systematic approach of encrypting and protecting digital data from unauthorized access. It also restricts consumers from copying and redistributing content they already have purchased.

### 2.3.3.4 Server/Client<sup>5</sup>

The following scheme describes the typical composition of a HTTP Streaming environment.

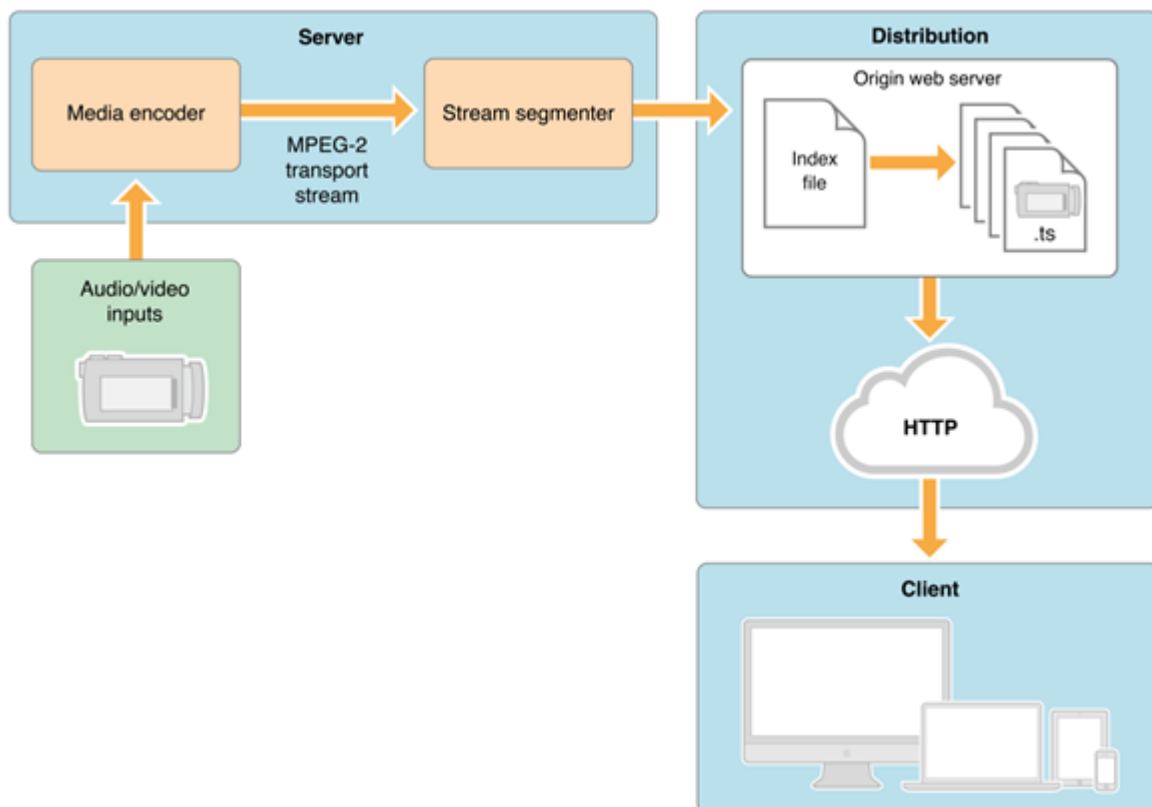


Figure1: Streaming scheme

<https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/HTTPStreamingArchitecture/HTTPStreamingArchitecture.html>

<sup>4</sup> <http://searchcio.techtarget.com/definition/digital-rights-management> (30.12.2016)

<sup>5</sup> <https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/HTTPStreamingArchitecture/HTTPStreamingArchitecture.html> (29.12.2016)

HTTP Live Streaming is divided in three parts, which consist of the server component, the distribution component and the client itself, which in most cases is an iOS-Device or computer.

1. The server component:

The server component has the main task to take input streams of media and encode them digitally, encapsulate them in a proper way for delivery and media distribution. For HTTP Live Streaming, the data needs to be transferred as a MPEG-2 transport stream, which means, the input has to be segmented in .ts files.

2. The distribution component:

The distribution component consists of a standard web server. The web server is responsible for accepting client requests and delivering the media and associated resources to the client. The MIME types of the web server need to be configured properly, which means the following MIME types have to be set:

File extension	MIME type
.m3u8	application/x-mpegURL
.ts	video/MP2T

Table1: MIME types

The web server needs to know the MIME type of the file so the content type of the response can be set properly. It also needs to provide the m3u8 playlist or index file, which refers to every segment and the according location and order. So the segments can be set back together with a corresponding Application.

3. The client software

The client software is responsible for determining the appropriate media to request, downloading them, and setting the segments together so the client can present the media in a single continuous stream to the user.

### 2.3.3.5 HLS Authoring<sup>6</sup>

TBD

## 2.4 Xcode<sup>7</sup>

Xcode is an IDE provided by Apple to create apps for iPhone, iPad, Mac, Apple Watch, and Apple TV. Xcode 8 is the newest version that among other things includes a runtime issues alert, a Memory Debugger and a Interface Builder.

### 2.4.1 Swift<sup>8</sup>

Swift is a programming language for macOS, iOS, watchOS and tvOS. The newest version Swift 3 is completely open source with source code, a bug tracker, mailing lists and regular development builds. More information about Swift can be found on: <https://swift.org>.

### 2.4.2 TVML

Apple's Television Markup Language (TVML) can be used to create individual pages inside of a client-server app.<sup>9</sup> The application stored on the Apple TV is the client that stores the URL that points to the initial TVMLKit JS file which is stored locally or on a server.

A TVML app is built up of three different parts.

- 1) **Xcode project:** creates the application that links to TVMLKit and provides the TVMLKIT Javascript (TVMLKit JS) environment.
- 2) **TVMLKit JS file:** controls the applications flow and business logic.
- 3) **TVML template:** displays information on the screen.

TVML is an easy way to create your own apps for AppleTV using the three frameworks TVMLKit JS, TVML and TVMLKit. Each framework interacts with each other.

---

<sup>6</sup>[https://developer.apple.com/library/content/documentation/General/Reference/HLSAuthoringSpec/index.html#apple\\_ref/doc/uid/TP40016596-CH4-SW1](https://developer.apple.com/library/content/documentation/General/Reference/HLSAuthoringSpec/index.html#apple_ref/doc/uid/TP40016596-CH4-SW1) (03.01.2017)

<sup>7</sup><https://developer.apple.com/xcode/> (31.12.2016)

<sup>8</sup><https://developer.apple.com/swift/> (31.12.2016)

<sup>9</sup>[https://developer.apple.com/library/content/documentation/LanguagesUtilities/Conceptual/ATV\\_Template\\_Guide/index.html#apple\\_ref/doc/uid/TP40015064](https://developer.apple.com/library/content/documentation/LanguagesUtilities/Conceptual/ATV_Template_Guide/index.html#apple_ref/doc/uid/TP40015064) (31.12.2016)

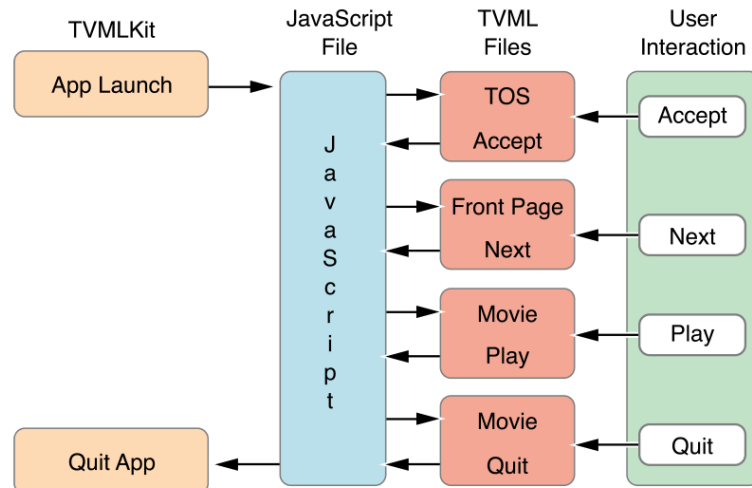


Figure x: Client-server application flow<sup>10</sup>

## TVMLKit JS

TVMLKit JS is a JavaScript API framework that is designed especially to work with Apple TV and TVML. TVMLKit JS combines perfectly the basic functionalities from JavaScript with the specialized API's for Apple TV. With TVMLKit JS you are not only able to display different TVML templates, but also control the flow of your application, load customized content and play media for example. More information can be found on: <https://developer.apple.com/reference/tvmljs>.

## TVML

TVML is an XML based language that is used to display information on the screen. Each TVML app is created using a set of predefined templates. Every template has a specific design and purpose that follows the Apple style guidelines. Those templates can be adapted to your needs using different styles and attributes. Any TVML file contains exactly one single template. More information can be found on: [https://developer.apple.com/library/content/documentation/LanguagesUtilities/Conceptual/ATV\\_Template\\_Guide/index.html](https://developer.apple.com/library/content/documentation/LanguagesUtilities/Conceptual/ATV_Template_Guide/index.html)

## TVMLKit

<sup>10</sup>[https://developer.apple.com/library/content/documentation/General/Conceptual/AppleTV\\_PG/YourFirstAppleTVApp.html#apple\\_ref/doc/uid/TP40015241-CH3-SW1](https://developer.apple.com/library/content/documentation/General/Conceptual/AppleTV_PG/YourFirstAppleTVApp.html#apple_ref/doc/uid/TP40015241-CH3-SW1) (31.12.2016)

This framework provides access to the TVMLKit JS environment and TVML. It also gives you the possibility to combine TVML and UIKit to create individual layouts. With this combination TVMLKit supports creating own views, elements and controllers and us them to customize every application to present your content as good as possible. More information can be found on:

<https://developer.apple.com/reference/tvmlkit>.<sup>11</sup>

---

<sup>11</sup> <https://developer.apple.com/library/content/documentation/TVMLKitJS/Conceptual/TVMLProgrammingGuide> (31.12.2016)

### 3 **Prototype**

Our application is divided in two major parts. On one hand, we have the whole streaming setup which is crucial for the video content played in the application. On the other hand, we have the tvOS application itself developed in Xcode with TVML.

#### 3.1 **Architecture**

TBD: Illustration of main architecture

##### 3.1.1 **Streaming setup**

//Describe the Infrastructure(Apache-VM, FFMPEG, the FFMPEG Inputs) +  
Diagramm  
//We use virtualbox to simulate the server component.  
//tbd

##### 3.1.1.1 **HTTP Live Streaming**

TBD

##### 3.1.1.1.1 **FFmpeg**<sup>12</sup>

FFmpeg is a cross platform multimedia framework, which allows you to decode, encode, transcode, mux, demux, stream, filter and play any codec and container formats which have ever been created.

##### 3.1.1.1.2 **Adaptive Streaming on Demand**

Adaptive Streaming describes the ability of the client to switch to the optimal resolution and codec to guarantee a buffer-free and continuous stream for the user. This is device based and follows a few conventions which are, amongst other things, based on Apple's definitions, recommendations and restrictions on HTTP Live Streaming on iOS-based devices<sup>13</sup>. The following Image describes the typical workflow of adaptive streaming. The difference between live streaming is simply that the whole workflow has to be continuously executed in real time which is going to claim a lot of CPU power. We will go quickly through each step:

---

<sup>12</sup> <https://www.ffmpeg.org/about.html> (29.12.2016)

<sup>13</sup> <https://developer.apple.com/library/content/documentation/NetworkingInternet/Conceptual/StreamingMediaGuide/UsingHTTPLiveStreaming/UsingHTTPLiveStreaming.html> (30.12.2016)

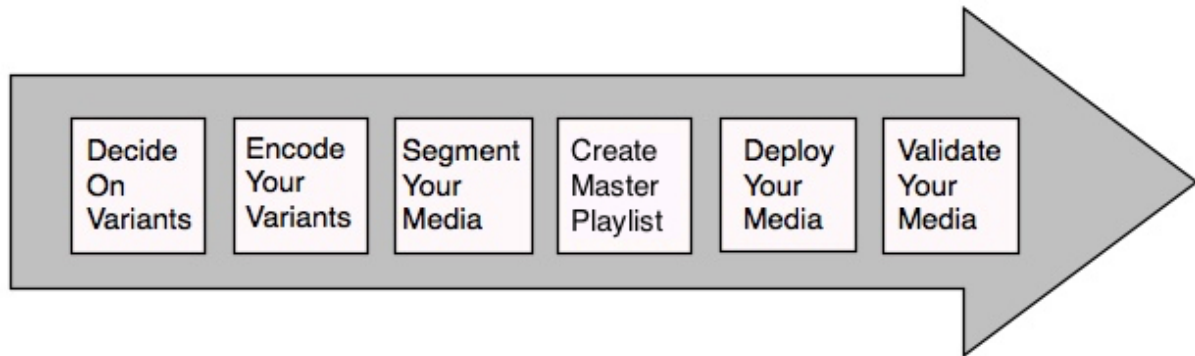


Figure 3: Adaptive Streaming Workflow  
[https://developer.apple.com/library/content/technotes/tn2224/\\_index.html](https://developer.apple.com/library/content/technotes/tn2224/_index.html)

### Decide on Variants:

The following table describes which dimensions and quality levels we used for the Horizon Apple TV Project. We ignored the cellular connection type since Apple-TV devices have to be connected over WiFi. Cellular connections will also make the stream quality quite low and unenjoyable. It is also recommended to create a distance of adjacent variants with the factor 1,5 or 2 regarding the Total Bit Rate<sup>14</sup>. Otherwise there will be too many and unnecessary variants which we simply will not need and would only use unnecessarily CPU power.

Connection	Dimensions	Frame Rate	Total Bit Rate	Total Video Bit Rate	Total Audio Bit Rate	Keyframes	Restrict Profile to:	Segment Size
WiFi	640 x 360	29.97	1240 kbps	1200 kbps	40 kbps	90	Main profile, 4.0	9
WiFi	960 x 540	29.97	1840 kbps	1800 kbps	40 kbps	90	Main profile, 4.0	9
WiFi	1280 x 720	29.97	2540 kbps	2500 kbps	40 kbps	90	Main profile, 4.0	9
WiFi	1280 x 720	29.97	4540 kbps	4500 kbps	40 kbps	90	Main profile, 4.0	9
WiFi	1920 x 1080	29.97	12000 kbps	11000 kbps	1000 kbps	90	High profile, 4.0	9

<sup>14</sup> [https://developer.apple.com/library/content/technotes/tn2224/\\_index.html#/apple\\_ref/doc/uid/DTS40009745-CH1-BITRATERECOMMENDATIONS](https://developer.apple.com/library/content/technotes/tn2224/_index.html#/apple_ref/doc/uid/DTS40009745-CH1-BITRATERECOMMENDATIONS)

WiFi	1920 x 1080	29.97	25000 kbps	24000 kbps	1000 kbps	90	High profile, 4.0	9
WiFi	1920 x 1080	29.97	40000 kbps	39000 kbps	1000 kbps	90	High profile, 4.0	9

Table 2: Variants table

Since we decided which variants we are going to use for our Apple-TV project, we need to encode the different variants. Therefore we need either Apples own media stream segmenter and developer tools, which is not the best choice since it only works on Apple devices, or the open source cross platform project FFmpeg.

### Encode your variants

We used FFmpeg to convert a video into the different variants. The following input converts the sample movie Input.mov into the first variant in table 2.

```
ffmpeg -i Source/Input.mov -force_key_frames "expr:gte(t,n_forced*3)" -c:v libx264 -vprofile main -vlevel 4.0 -s 640x360 -b:v 1200k -strict -2 -c:a aac -ar 44100 -ac 2 -b:a 40k Destination/640x360_1240.mov
```

We force the key frames for each variant to the corresponding frame rate multiplied with 3. Furthermore, we need to convert the restricted profile to the according value for each variant. In the first case, we need to convert the video to the mainProfile 4.0, which can be set with the options -vprofile and -vlevel. The next step is to write down the desired resolution, which is in this case 640x360. After that, we need to specify the video bitrate and codec. The audio codec of our choice is AAC since it has the lower bit rate compared with MP3 and it still has the better quality.

We need to repeat this step with each variant we specified in Table 2.

**//Explain -c:a etc**

### Segment your media

After we encoded each variant correctly, we need to segment the media. In order to do that, we can either use Apple's own Mediafilesegmenter or FFMPEG. For the sake of completeness we are going to take a look at both programs. We start with Apple's Mediafilesegmenter:

```
"mediafilesegmenter -start-segments-with-iframe -l -f Destination/640x360_1240/ -t 9 Source/640x360_1240.mov"
```

The Mediafilesegmenter will split the input video into segments with 9 seconds duration. The segments, if not specified, will be named fileSequence followed by the



number of the segment and the .ts file ending. Additionally the segmenter creates a m3u8 playlist, that is named “prog\_index.m3u8 per default, which refers to the different segments in the correct order. The duration of each segment is set in the “#EXTINF:” tag. Below we present you a snippet of the prog\_index.m3u8 playlist we created before:

```
#EXTM3U
#EXT-X-TARGETDURATION:9
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-INDEPENDENT-SEGMENTS
#EXTINF:9.00000,
fileSequence0.ts
#EXTINF:9.00000,
fileSequence1.ts
...
#EXTINF:9.00000,
fileSequence65.ts
#EXTINF:2.45833,
fileSequence66.ts
#EXT-X-ENDLIST
```

The mediafilesegmenter also creates an I-Frame only playlist defaultly named iframe\_index.m3u8 playlist. I-Frames are encoded video frames whose encoding does not depend on any other frame. An I-Frame only playlist is almost identical to a regular playlist. The difference is the #EXTINF tag refers to the span of the I-Frame. Which means it refers to the time between the presentation of the I-Frame in the media segment and the presentation time of the next I-Frame in the playlist. Below you will find the snippet of the iframe\_index.m3u8 playlist file we created with the same input from the mediafilesegmenter before:

```
#EXTM3U
#EXT-X-TARGETDURATION:9
#EXT-X-VERSION:4
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-I-FRAMES-ONLY
#EXTINF:3.00000,
#EXT-X-BYTERANGE:1128@376
fileSequence0.ts
#EXTINF:3.00000,
#EXT-X-BYTERANGE:46060@560428
fileSequence0.ts
#EXTINF:3.00000,
#EXT-X-BYTERANGE:82720@1103372
fileSequence0.ts
#EXTINF:2.87500,
#EXT-X-BYTERANGE:94000@564
```

```
fileSequence1.ts
...
#EXT-X-BYTERANGE:14476@564
fileSequence65.ts
#EXTINF:3.00000,
#EXT-X-BYTERANGE:13348@79148
fileSequence65.ts
#EXTINF:3.00000,
#EXT-X-BYTERANGE:17860@212440
fileSequence65.ts
#EXTINF:2.37500,
#EXT-X-BYTERANGE:18612@564
fileSequence66.ts
#EXT-X-ENDLIST
```

iOS 5 supports Fast Forward and Reverse Playback. The good thing is, that we do not need to encode or segment the content differently, we just need to specify where the I-Frames are and simply just refer to the iframe playlist to enable this feature.

//needs reference

Last but not least, the file segmenter creates a .plist file which stores the properties of the segmented media, including codec information, frame and bit rates.

//Insert FFMPEG solution with hls muxer

```
ffmpeg -i Source/Input.mov -hls_playlist_type vod Destination/out.m3u8
//segment length
```

## Create Master Playlist

After we encoded and segmented each variant, we need to create a master playlist. For this purpose we use Apple's Variantplaylistcreator. It is important that we refer to the prog\_index.m3u8 file first, followed by the associated .plist file. With the following input we create the master playlist file named all.m3u8 which includes all our previously encoded variants.

```
variantplaylistcreator -o all.m3u8 640x360_1240/prog_index.m3u8
640x360_1240/640x360_1240.plist 960x540_1840/prog_index.m3u8
960x540_1840/960x540_1840.plist 1280x720_2540/prog_index.m3u8
1280x720_2540/1280x720_2540.plist 1280x720_4540/prog_index.m3u8
1280x720_4540/1280x720_4540.plist 1920x1080_12000/prog_index.m3u8
1920x1080_12000/1920x1080_12000.plist 1920x1080_25000/prog_index.m3u8
1920x1080_25000/1920x1080_25000.plist 1920x1080_40000/prog_index.m3u8
1920x1080_40000/1920x1080_40000.plist
```

After we executed the command, we get a master playlist file, which helps the client switch to the optimal resolution based on the average-bandwidth it can use.

```
#EXTM3U
#EXT-X-STREAM-INF:AVERAGE-BANDWIDTH=1319324,BANDWIDTH=3808636,CODECS="mp4a.40.2,
avc1.4d4028",RESOLUTION=640x360,FRAME-RATE=24.000,CLOSED-CAPTIONS=NONE
```

```
640x360_1240/prog_index.m3u8
#EXT-X-STREAM-INF:AVERAGE-BANDWIDTH=1882542,BANDWIDTH=3047104,CODECS="mp4a.40.2,
avc1.4d4028",RESOLUTION=960x540,FRAME-RATE=24.000,CLOSED-CAPTIONS=NONE
960x540_1840/prog_index.m3u8
#EXT-X-STREAM-INF:AVERAGE-BANDWIDTH=2590844,BANDWIDTH=4554948,CODECS="mp4a.40.2,
avc1.4d4028",RESOLUTION=1280x720,FRAME-RATE=24.000,CLOSED-CAPTIONS=NONE
1280x720_2540/prog_index.m3u8
#EXT-X-STREAM-INF:AVERAGE-BANDWIDTH=4634323,BANDWIDTH=8353885,CODECS="mp4a.40.2,
avc1.4d4028",RESOLUTION=1280x720,FRAME-RATE=24.000,CLOSED-CAPTIONS=NONE
1280x720_4540/prog_index.m3u8
#EXT-X-STREAM-INF:AVERAGE-BANDWIDTH=11580520,BANDWIDTH=22715247,CODECS="mp4a.40.2,
avc1.640028",RESOLUTION=1920x1080,FRAME-RATE=24.000,CLOSED-CAPTIONS=NONE
1920x1080_12000/prog_index.m3u8
#EXT-X-STREAM-INF:AVERAGE-BANDWIDTH=25261530,BANDWIDTH=56162870,CODECS="mp4a.40.2,
avc1.640028",RESOLUTION=1920x1080,FRAME-RATE=24.000,CLOSED-CAPTIONS=NONE
1920x1080_25000/prog_index.m3u8
#EXT-X-STREAM-INF:AVERAGE-BANDWIDTH=30273689,BANDWIDTH=74271865,CODECS="mp4a.40.2,
avc1.640028",RESOLUTION=1920x1080,FRAME-RATE=24.000,CLOSED-CAPTIONS=NONE
1920x1080_40000/prog_index.m3u8
```

Unfortunately FFmpeg does not support such a master playlist function like the variantplaylistcreator does, we would need to create the playlist file manually. This means, we need the correct hex codec reference for the audio and video codec as well as the correct bandwidth, average bandwidth, resolution and the location of each m3u8 playlist file.

## Deploy your Media

Now that we have prepared the media for distribution, we need to host it on a web server. In our approach, we set up an Ubuntu(version 16.04.1) virtual machine and installed FFmpeg and Apache2 on it. Additionally we set the port forwarding regarding HTTP and HTTPS requests up as followed:

Name	Protokoll	Host-Port	Guest-Port
HTTP	TCP	8080	80
SSL	TCP	7070	443

Table x: Port Forwarding

We did this in order to interact with the virtual machine from our host machine and test the streaming and web hosting functionalities. Furthermore, we mapped a local folder on our hosting computer in the virtual machine in order to not have to transfer big chunks of video data for testing purpose. In order to do that, we used Virtualbox and installed the guest additions<sup>15</sup>. After that, we can specify in the Shared-Folder tab in the config, which folder on the host machine we want to enable in the virtual

<sup>15</sup> <https://www.virtualbox.org/manual/ch04.html> (03.01.2017)

machine. We can simply prepare the media on the hosting machine, in this case our mac computer, and start the virtual machine and automatically host the folder as a web folder. But we need to specify the new DocumentRoot folder in the /etc/apache2/sites-enabled/100-streaming.conf file first. The first row of the 100-streaming.conf file consist of the statement <VirtualHost \*:80>.

In Linux external mapped directories will be mapped in the /media directory with the prefix "sf\_". In our case we named the folder on the hosting machine Destination\_File\_Segmenter. Therefore the path in the virtual machine will be /media/sf\_Destination\_File\_Segmenter. So now we need to edit the DocumentRoot tag as followed: "DocumentRoot /media/sf\_Destination\_File\_Segmenter/". After we have done that, we need to set some important properties and configurations regarding the Directory we just specified as the DocumentRoot. In the same <VirtualHost \*:80> tag we need to add the following directory entry.

```
<Directory /media/sf_Destination_File_Segmenter/>
    ExpiresActive On
    ExpiresByType application/x-mpegURL "now plus 5 seconds"
    <FilesMatch "\.m3u8$">
        Header append Cache-Control "public"
    </FilesMatch>
    Options Indexes FollowSymLinks
    AllowOverride None
    Require all granted
</Directory>
```

First of all, we changed the cache-control header of m3u8 playlist files to 5 seconds. Which is roughly half the duration of a segment. After 5 seconds, the client needs to request the playlist file. The cache control mechanism guarantees that the client will always get the newest version of the playlist and not an obsolete copy.

## Validate your Media

//tbd

### 3.1.1.1.3 DRM

First of all, to implement DRM on our setup, we need to install OpenSSL on the Linux machine. With a simple apt-get install openssl this step is done. Furthermore, we need to create a self signed certificate. In order to do that, we open the linux terminal and type in the following command:

```
sudo openssl req -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/ssl/private/server.key -out /etc/ssl/certs/server.crt
```

Now we need to encrypt our segments so they can only be playable if the client has access to the associated key file. HTTP Live Streaming supports the Advanced Encryption Standard (AES) with a 128-bit key to encrypt videos. We added a new subfolder in our hosted web folder called secure with another subfolder called protected.

We use the Mediafilesegmenter to encrypt our video with the following command:

```
mediafilesegmenter -I -t 9 -f /secure/ -k /secure/protected/ -encrypt-key-url https://localhost:7070/secure/protected source/Input.mov
```

The -k option allows us to specify where we want to store the encryption key. The decryption key for decrypting the segments is specified with the -encrypt-key-url parameter. The default name of the key is crypt.key. The segments now will not be playable without the decryption key and therefore the segments will be useless if an unauthorized client finds out the location of the segments. Let us have a look at the generated m3u8 playlist file:

```
#EXTM3U
#EXT-X-TARGETDURATION:9
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-KEY:METHOD=AES-
128,URI="https://localhost:7070/secure/protected/crypt0.key",IV=0x9c09ce638
30d3a59675971c863f2c640
#EXTINF:9.00000,
fileSequence0.ts
#EXTINF:9.00000,
fileSequence1.ts
```

The encryption method equals AES-128. In encrypted playlists, this will always be the encryption method since it is the only supported encryption method for HTTP Live Streaming. The location of the Key and the initialisation vector are publicly viewable. Which means we need to ensure that only authorized access can read the crypt.key. Since AES-128 uses the Advanced Encryption Standard in cipher block chaining mode we do not need to worry about hiding the initialisation vector IV.

//more details

The Mediafilesegmenter is used for VOD's. If we want to encrypt a live stream, we need to use the mediastreamsegmenter with the same -k and -encrypt-key-url parameters as we did before.

#### **3.1.1.1.3.1 Enabling ssl**

After that, we need to config the /etc/apache2/sites-enabled/100-streaming.conf file again. We copy the whole <VirtualHost \*:80> content with all its tags. Then we paste it and make sure, it does not interfere with the original <VirtualHost \*:80> tag. We change the Port in the copy from 80 to 443. We also add the following three lines to the entry:

```
SSLEngine on
SSLCertificateFile      /etc/ssl/certs/server.crt
SSLCertificateKeyFile   /etc/ssl/private/server.key
```

We need to protect the /secure/protected location with the following statement in the VirtualHost \*:80 entry:

```
<Location /secure/protected>
SSLRequireSSL
</Location>
```

#### **3.1.1.1.3.2 Authentication with DRM**

//tbd

#### **3.1.1.1.4 Live Streaming**

Live Streaming consists of the same steps as adaptive streaming. The only difference is that the whole Workflow in Figure 2 has to be executed in real time.

#### **3.1.1.1.5 Integration of the infrastructure in UPC**

//tbd

### **3.1.2 Application**

Swift VS TVML with (dis-) + advantages

The architecture of our application follows the structure of Apple's TVML. For that matter our application consists of the following structure and files.

### 3.1.2.1 folder structure:

folder	description	file type(s)
IP5_Horizon/		-
IP5_Horizon/3rd party		Javascript
IP5_Horizon/content/data		JSON
IP5_Horizon/content/data/videos		JSON
IP5_Horizon/content/images		-
IP5_Horizon/content/images/Logos		png, .jpg
IP5_Horizon/content/images/Movies		png, .jpg
IP5_Horizon/content/images/Series		png, .jpg
IP5_Horizon/content/videos		.mp4
IP5_Horizon/layouts		HTML
IP5_Horizon/utilities		Javascript

### 3.1.2.2 files:

folder	description	file type
IP5_Horizon/AppDelegate.swift		Swift
IP5_Horizon/Info.plist		-
IP5_Horizon/main.js		Javascript
IP5_Horizon/RessourceLoader.swift		Swift
IP5_Horizon/3rd party/mustace.min.js		Javascript
IP5_Horizon/content/data/videoDatabase.json		JSON

IP5_Horizon/layouts/_searchResult .tvml		HTML
IP5_Horizon/layouts/expandedDetailText.tvml		HTML
IP5_Horizon/layouts/homeScreen.tvml		HTML
IP5_Horizon/layouts/LiveTV.tvml		HTML
IP5_Horizon/layouts/myPrime.tvml		HTML
IP5_Horizon/layouts/photos.tvml		HTML
IP5_Horizon/layouts/rootMenu.tvml		HTML
IP5_Horizon/layouts/Search.tvml		HTML
IP5_Horizon/layouts/Settings.tvml		HTML
IP5_Horizon/layouts/tvGuide.tvml		HTML
IP5_Horizon/layouts/video.tvml		HTML
IP5_Horizon/layouts/videoLogoOverlay.tvml		HTML
IP5_Horizon/layouts/videoRating.tvml		HTML
IP5_Horizon/utilities/DataController.js		Javascript
IP5_Horizon/layouts/EventHandler.js		Javascript
IP5_Horizon/layouts/Presenter.js		Javascript
IP5_Horizon/layouts/ResourceLoader.js		Javascript
IP5_Horizon/layouts/SearchHandler.js		Javascript



## 3.2 User interface and interaction Design

Interfaces are the window to view the capabilities of the software which makes it important to create a well-designed interface. The appearance and layout of a screen and the software's navigation can affect a user in many ways. If the software's design is inefficient, misleading or even confusing, people will have difficulties to interact with the product. It not only can lead to aggravation, increased stress, but also to frustration so it may chase the user away from the system permanently.<sup>16</sup>

The ideal user interface design process contains the following 7 steps:<sup>17</sup>

1. Provide a multidisciplinary design team
2. Solicit early and ongoing user involvement
3. Gain a complete understanding of users and their task
4. Create the appropriate design
5. Perform rapid prototyping and testing
6. Modify and iterate the design as much as necessary
7. Integrate the design of all the system components

TBD: our exact approach

### 3.2.1 User interface Design principles

Principles<sup>18</sup> TBD

#### 3.2.1.1 Color concepts

TBD<sup>19</sup>

#### 3.2.1.2 Layout concepts

TBD

### 3.2.2 Interaction Design principles<sup>20</sup>

The following principles from KISS (keep it simple and stupid) can help creating the user navigation ideal:<sup>21</sup>

1. direct access to objects

---

<sup>16</sup> (Galitz 2007, 5)

<sup>17</sup> (Galitz 2007, 60)

<sup>18</sup> (Galitz 2007, 127)

<sup>19</sup> (Galitz 2007, 691)

<sup>20</sup> (Galitz 2007, 71)

<sup>21</sup> adxd Slides

2. contextual tools
3. remain in place
4. provide attractive content
5. use transitions
6. objects should react immediately

more TBD

### **3.2.2.1 Navigation concept**

TBD

### **3.2.3 Guidelines**

The documentation of a guideline, a developed shared language, helps promoting consistency among multiple designers in aspects as usages, appearance and action sequences. A guideline includes best practices, examples derived from practical experience or studies.<sup>22</sup> As we are developing an application for tvOS we only go into the guidelines made from Apple regarding the development of an tvOS application.

#### **3.2.3.1 tvOS Human Interface Guideines**

##### Design Principles<sup>23</sup>

Since Apple TV is a unique platform it also has its own design requirements. When using Apple TV with the Siri Remote it is important to provide an interaction with gestures that are fluid and familiar to the user. The users should instinctively know where they are and what to do to achieve the goal. For that reason the text should be legible at a distance and a cluttered interface with unnecessary visual content shall be avoided.

Apple TV uses a focus model for navigation. An element is in focus when the user highlights the item, but has not clicked it. It must be clear for the user if something is in focus at a distance.

Clicking an item with the remote takes the user deeper into an app's hierarchy, pressing Menu always takes the user back. While navigating through content the user should feel a sense of immersion.

Sound is used throughout the system to enhance the visual experience.

---

<sup>22</sup> (Shneiderman 2010, Page)

<sup>23</sup> <https://developer.apple.com/tvos/human-interface-guidelines/overview/> (31.12.2016)

tvOS provides development framework for creating new application on Apple TV. A lot interface elements and frameworks are identical to those from the iOS app design. Nevertheless, it is crucial to create an application that is conform to a television screen and a remote.

### Remote<sup>24</sup>

The primary tool to navigate on the Apple TV is this remote. It is vital to implement an expected button behaviour which means that pushing the touch surface brings a user deeper into an app's hierarchy and pressing the Menu returns to the previous screen.

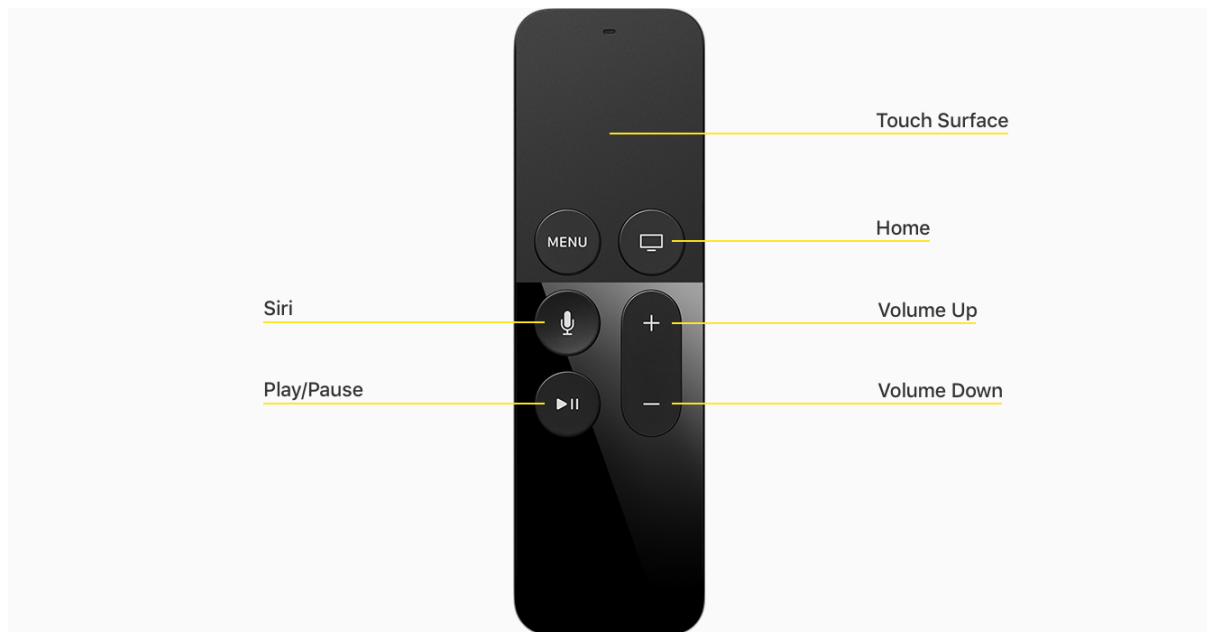
The touch surface of the remote detects three different single-finger gestures.

1. Swipe. The user can scroll through the content effortlessly with swiping up, down, left or right. The speed of the movement depends on the strength of the swipe.
2. Click. Clicking is the primary way of triggering actions, it can activate a control or select an item for example.
3. Tap. With this functionality you can navigate in different directions on a standard interface by tapping various regions on the screen. You can also use it to display hidden content.

---

<sup>24</sup> <https://developer.apple.com/tvos/human-interface-guidelines/remote-and-controllers> (02.01.2017)

While developing you should also keep in mind to avoid using standard gestures to execute nonstandard actions. Using standard gestures can help every user you get



along with your application faster and more efficiently.

Figure x: [https://developer.apple.com/tvos/human-interface-guidelines/images/remote-and-interaction-remote\\_2x.png](https://developer.apple.com/tvos/human-interface-guidelines/images/remote-and-interaction-remote_2x.png)

Another important possibility to navigate in the application in addition to the gestures are the buttons. It is crucial that the behaviour of the buttons is consistent and predictable in the context of the app.

Button	expected behaviour in apps
Home	returning to the Apple TV Home screen
Touch Surface <ul style="list-style-type: none"> <li>- tap/swipe</li> <li>- click</li> </ul>	navigating and changing focus activating a control or item and navigation deeper into the app
Menu	returning the user to previous screen or the Apple TV Home screen
Play/Pause	playing, pausing and resuming media playback


## User Interaction<sup>25</sup>

Navigation in the application should feel natural and support the structure of your application. The user normally navigates by moving through a collection of content visualized with individual screens. Generally, it makes it easier for user to move flawless in your application when using standard interface elements such as table or collection views and tab bars (especially when the application uses UIKit).


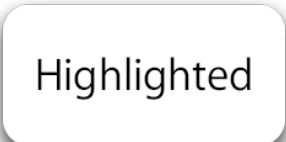

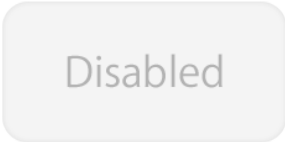
To provide a simple access to the content the information structure should be organized in a way that requires the fewest screens and gestures to move through it. Another point to bear in mind is that swiping to the side is easier and more familiar to the user than swiping up and down.

Apple TV works not only with a focus model as mentioned above, but also uses mostly indirect manipulation. This means that gestures move focus between objects and the system scrolls the interface to keep those focused elements visible. It is crucial that gestures and focus moves into the same direction. When tapping or swiping to the right focus should move right and the content may move left. In contrast, full-screen elements should use direct manipulation instead, which means that gestures move objects, rather than focus.

For an even more intuitive navigation there are five different focusable item states that appear visually distinct.

state	meaning
 Unfocused	The user is not interacting with this item at the moment, but could bring it into focus at any time.

<sup>25</sup> <https://developer.apple.com/tvos/human-interface-guidelines/user-interaction> (02.01.2017)

 <p>Focused</p>	<p>The user has currently set the focus on this element. While on focus the item stands out from the other content in the screen.</p>
 <p>Highlighted</p>	<p>This state is activated when the user has clicked on the item.</p>
 <p>Selected</p>	<p>This state displays the item when it was selected or activated in some way. (this state can appear whether the element is focused or unfocused)</p>
 <p>Disabled</p>	<p>There is no possibility to bring the element into focus or click it and so appears inactive.</p>

You should also consider the following aspects when developing an application:

- You should design the first launch experience for the user fun appealing. If you offer a tutorial make it quick, enjoyable and interactive and give the user the possibility to replay it in case they miss something the first time.
- Sometimes your application needs more time to load the content and if so, it is crucial to show an activity spinner that denotes something is happening. A loading screen is only recommended if the content needs more than 2 seconds to load.
- If authentication is a necessity in your application make sure that the authentication process is quick and easy. Keep in mind that most people interact with the remote and not a keyboard and therefore avoid extensive input from the user

The Apple TV video player lets users watch media and uses the remote with the following intuitive gestures to move through the media:<sup>26</sup>

<sup>26</sup> <https://developer.apple.com/tvos/human-interface-guidelines/visual-design> (02.01.2017)

- Gliding a thumb across the touch surface enables quick scanning quickly through a video
- Clicking on the left or right sides of the touch surface lets a user skip forward and backward in a video. Fast-forward or backward is possible by clicking and holding on the left and right sides of the touch surface
- Skipping forward or backward through chapters you have to click the touch surface to reveal the video timeline and then click again on the left and right sides of the touch surface
- Swiping downward displays additional information

### Visual Design

Apps on Apple TV show the same exact interface on every display and don't adapt their layout on the size of the screen. For this reason develop the app's user interface design for a screen resolution of 1920 x 1080 pixels.

Additionally, a spanning area of 60 pixels from the top and bottom and 90 pixels from the sides should be considered because it is difficult for the user to see content close to the edges.

When it comes to typography, Apple TV's system font is San Francisco with two variants: Text or Display. Apple TV automatically applies the most appropriate style of the font bases on the point size. San Francisco Text is better for text 39 points or smaller and San Francisco Display for text 40 points or larger. There is also the possibility to integrate a custom font in your application, but Apple advises every developer to stick to the default font unless your application has a compelling need for this custom font.

Colours are important to add your application an unique look. Therefore, choose a limited colour palette that coordinates with your app logo. Avoid colour combinations that are difficult to distinguish for colour-blind people, for example red and green. Keep in mind that every colour has its own connotations depending on the people's cultural background. You can find more information about this theme in chapter 3.2.1.1 Colour concepts.

On Apple TV there is the default light appearance and the dark appearance that can be activated for a deeper cinematic experience. Make sure that your app looks good in both appearances.

## Icons and Images<sup>27</sup>

All images on Apple TV are @1x resolution.

Every application must supply a small and a large icon and they must be submitted in two specific sizes.

- small icon: 400px x 240px
- large icon: 1280px x 768px

For creating an app icon layered images are required, which means that they must consist two to five layer to create a sense of depth. Layered images are, in conjunction with the parallax effect, at the essence of the Apple TV user experience. If your icon includes a logo, separate the logo from the background and if your icon includes text as well, bring the text to the front. Only with this separation the parallax effect occurs properly. Don't place your content too close to the edges because when the icon is in focus the edges of some layers may be cropped or are difficult to see. Every app must also supply a static shelf image with the size 2320px x 720px that is displayed when your app is in the top row of the Home screen and is in focus. You can also use dynamic top shelf imagery if you want to provide quick access to main content in your app.

Every app on Apple TV must also supply a launch image with a resolution of 1920px x 1080px.

The information above is only a short excerpt of the extensive tvOS Human Interface Guidelines. We consciously focused on topics that are relevant for our proof of concept.

### **3.2.3.2 UPC**

TBD: CI of upc

### **3.2.4 Benchmarking**

TBD.

### **3.2.5 Wireframes and Click-prototype**

TBD. With workflow diagramm

---

<sup>27</sup> <https://developer.apple.com/tvos/human-interface-guidelines/icons-and-images> (02.01.2017)



### **3.3 Usability**

Usability is a quality attribute that evaluates how easy an user interface is to use. The usability assessment should be integrated in the early stages of the whole product development process.<sup>28</sup>

The overall goal of usability testing is to improve a product or software by gathering data from which to identify usability deficiencies and expose design issues existing in products to they can be minimized and thus eliminating frustration for users.

The goal is to create a product that is useful to and valued by the target audience, is easy to use, helps people be effective at what they want to do and is satisfying to use.<sup>29</sup>

While testing we focused on the three critical measures of usability: effectiveness, efficiency and satisfaction. Satisfaction, tough, is in our point of view the most important measurement criterion because it not only relies on the other two measures, but it is the crucial factor when it comes to desirability. If a user is totally satisfied with the product the desire of reusing it increases highly.<sup>30</sup>

We decided to perform two different usability test on our product. The first one should focus on the layout of the interface and the information visualization. Furthermore, the second test analyses the navigation with the Apple TV remote.

#### **Usability test 1**

The main goal of this test was to find out if our developed design of the interface is efficient and effective. For this test we used a click-prototype based on the high fidelity wireframes. This prototype comprises the most important screens and functionalities which are being implemented in the application later. The test has been executed with one test person that correlated to our persona and contained four defined tasks and some open questions. The results of this test showed us that the test person was able to perform the task efficiently and successfully. Moreover, the design and the information visualization is intelligible and beautifully designed. You can find more detailed results and findings in the file Usabilitytest\_1 found in the appendix.

---

<sup>28</sup> (Galitz 2007, 64)

<sup>29</sup> (Rubin, Chisnell 2008, 22)

<sup>30</sup> (Barnum 2011, 11)

## **Usability test 2**

The main content of the second usability test was to test the navigation of the application with the Apple TV remote.

----- INSERT TEXT -----

You can find more detailed results and findings in the file Usabilitytest\_2 found in the appendix.

## **4 Results**

Text

### **4.1 Findings**

Notes:

- *live streaming application for tvOS is realizable within a moderate investment of time, effort and knowledge*
- *live streaming, especially adaptive live streaming needs huge CPU resources in order to generate an ideal workflow*
- *TVML is the ideal solution to create an streaming application which meets all necessary requirements and guidelines*
- *some gui elements, for example the tv guide, need to be developed manually because there is no template that can be used to create it perfectly*
- *good usability guaranteed with the implementation of TVML*

### **4.2 Outlook**

Notes:

- *integration of the UPC infrastructure into the app (TV channels, MyPrime, User information, ect.)*
- *implementing EPG (imdb, rotten tomato)*
- *implement new features to make the application unique for users (functionality to find users preferences in order to give individual recommendations)*
- *create and personalize a dynamic top shelf*
- *In order to implement adaptive live streaming you need a high quality streaming infrastructure with high CPU power*
- *implementing AVPlayer for even more video functionalities like queueing videos*

### **4.3 Reflection**

Text

## 5 References

### L1. Literatur

Buch:

Muster: Autor (Jahr): Titel, Auflage. Ort: Verlag.

- Galitz, Wilbert O. (2007): The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, Third Edition. Indiana: Wiley Publishing.  
Kurz: (Galitz 2007, Page)
- Shneiderman, Ben (2010): Designing the user interface: Strategies for Effective Human-Computer Interaction, fifth Edition. USA: Addison Wesley.  
Kurz: (Shneiderman 2010, Page)
- Barnum, Carol M. (2011): Usability Testing Essentials. Burlington: Elsevier Inc.  
Kurz: (Barnum 2011, Page)
- Rubin, Jeff and Chisnell, Dana (2008): Handbook of Usability Testing, Second Edition. Indiana: Wiley Publishing, Inc.  
Kurz(Rubin, Chisnell 2008, Page)

### L2. Internet

Neben Autorin, Autor, Titel und Veröffentlichungsdatum sind zwei weitere Punkte wichtig:  
*Bleuel, Jens (2000): Zitation von Internet-Quellen. <http://www.bleuel.com/ip-zit.pdf>  
(abgerufen am 28.12.2011).*

für Kurzbeleg:

*Link*

### L3. Interviews

## **6 List of figures**

## **Acknowledgement**

## **A. Appendix**

### **A1. Appendix 1**

Text

#### **A1.1. Appendix 2**

Text

#### **A1.2. Appendix 2**

Text

### **A2. Declaration of honesty**

Text