

Diplomarbeit

The StreamOnTheFly Video Extension Vodcasting and more...

Ausgeführt zum Zweck der Erlangung des akademischen Grades eines

Dipl.-Ing. (FH) Telekommunikation und Medien

am FH-Diplomstudiengang Telekommunikation und Medien St. Pölten

unter der Erstbetreuung von

Markus Seidl Bakk.

Zweitbegutachtung von

Dipl.-Ing. Grischa Schmiedl

ausgeführt von

Martin Schmidt

tm021103

0210038103

Ehrenwörtliche Erklärung

Ich versichere, dass

- ich diese Diplomarbeit selbständig verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt und mich auch sonst keiner unerlaubten Hilfe bedient habe.
- ich dieses Diplomarbeitsthema bisher weder im Inland noch im Ausland einem Begutachter/einer Begutachterin zur Beurteilung oder in irgendeiner Form als Prüfungsarbeit vorgelegt habe.

Diese Arbeit stimmt mit der von den Begutachtern beurteilten Arbeit überein.

St. Pölten, am 7. September 2006

.....

Zusammenfassung

Die vorliegende Diplomarbeit befasst sich mit der StreamOnTheFly Videoerweiterung, welche im ersten Halbjahr 2006 am Referat für Forschung und Entwicklung der Fachhochschule St. Pölten entwickelt wurde.

Bei StreamOnTheFly handelt es sich um ein webbasiertes Open Source Medienarchiv, welches speziell für die Bedürfnisse von freien Radios entworfen und entwickelt wurde und der Bereitstellung, der Archivierung sowie des Austausches von Sendungen dient. Es besteht aus mehreren Servern in Österreich und Ungarn, die Metadaten von vorhandenem Content unter Verwendung von XML-RPC austauschen.

Bis vor kurzem unterstützte die Software – da ja vorwiegend von Radiostationen verwendet – ausschließlich Audiomaterial. Die Videoerweiterung wurde nun durchgeführt, um StreamOnTheFly ebenso mit Videodateien kompatibel zu machen. Dies beinhaltete neben einer Anpassung der Metadaten die Umsetzung einer flexiblen Transcoding-Lösung, um eine Vielzahl mobiler Videoplayer zu unterstützen. Der Konvertierungsvorgang dient auch dazu, einen Preview-Clip im FLV Format zu erzeugen. Zusätzlich wurden Vodcasting-Funktionen implementiert, um Benutzern zu ermöglichen, Beiträge gemäß ausgewählten Kriterien automatisiert auf Ihre Computer geliefert zu bekommen.

Daher beschäftigt sich ein Teil dieser Diplomarbeit ausführlich mit Podcasting und Vodcasting; deren Geschichte und Funktionsweise werden genau erklärt. Ferner werden gängige mobile Videogeräte vorgestellt, ebenso wie die im Rahmen der StreamOnTheFly Videoerweiterung verwendeten Videocodecs.

Abschließend werden auch die Teilschritte der Implementierung, welche unter Anwendung des „Evolutionären Prototypings“ durchgeführt wurden, beschrieben.

Abstract

This diploma thesis deals with the StreamOnTheFly Video Extension which was carried out in the first half of the year 2006 at the R&D department of the University of Applied Sciences in St. Pölten.

StreamOnTheFly is a web-based Open Source media archive, which has been designed and developed especially to fit the requirements of free radio stations in terms of publishing, archiving and exchanging programmes. It consists of several servers in Austria and Hungary which sync metadata of existing content via XML-RPC.

Until recently, the software has only supported audio material as it has predominantly been used by radio stations. The Video Extension has been carried out to make StreamOnTheFly also compatible with video files. This project has implied the adaptation of metadata as well as a flexible transcoding solution to support a wide variety of mobile video devices. The conversion process also provides a clip in FLV format which is used for preview purposes. Additionally, vodcasting functionality has been implemented to offer users the automatic download of content which is selected by criteria according to user settings.

Therefore, a part of this diploma thesis deals extensively with podcasting and vodcasting; their history and functionality are explained in detail. Current mobile video devices are as well introduced as video codecs which are relevant to the StreamOnTheFly Video Extension.

Finally, the partial stages of the implementation process – which has been carried out according to “Evolutionary Prototyping” principles – are described.

Acknowledgements

In alphabetical order, I wish to thank the following people for their contributions to the successful completion of the StreamOnTheFly Video Extension and therefore for providing the prerequisite for this diploma thesis:

- *Dipl.-Ing. (FH) Matthias Husinsky*, scientific assistant at the R&D department of the University of Applied Sciences in St. Pölten, for supporting me in terms of research and specification. Especially mentioned should be his immersion in the field of current video codecs and his help concerning Sony PSP's incompatibility issues.
- *Dr. András Micsik* from the Hungarian Academy of Sciences in Budapest (SZTAKI) for sharing his immense StreamOnTheFly knowledge by answering my regularly mailed questions almost in real-time when I have got stuck in the deep of StreamOnTheFly's code once again.
- *Wolfgang Reutz*, scientific assistant at the Vorarlberg University of Applied Sciences for contributing his forward-looking thoughts to the specification of the Video Extension at a snowy kick-off meeting in Dornbirn and for telling me what a SVN "working copy" is ...
- *Jürgen Schmidt* from strg.at for once having introduced me to the very basics of Linux as well as helping me through some difficult stages of the development, especially at the end of March 2006.
- *Markus Seidl Bakk*, lecturer at the University of Applied Sciences in St. Pölten and custodian of both my practical work and this diploma thesis, for all support and for timely reminding me of the project schedule when we had to catch up ...

Table of Contents

1	Introduction	8
2	StreamOnTheFly	9
2.1	Open Source	9
2.1.1	The GNU General Public License (GPL)	11
2.2	StreamOnTheFly.....	12
2.2.1	History.....	12
2.2.2	Technical Overview	13
2.2.3	Functionality	15
2.2.4	Functionality on the station editor's side.....	17
2.2.5	Future Prospects.....	20
3	Podcasting and Vodcasting	21
3.1	Overview.....	21
3.2	History of Podcasting	21
3.2.1	RSS 0.92 and "Audioblogging"	22
3.2.2	Getting mobile	22
3.2.3	"Taking the world by storm"	24
3.3	History of Vodcasting	25
3.4	How it works.....	27
3.4.1	Creation and Publication	27
3.4.2	Subscription and Download	29
3.4.3	Schematic Overview – Podcasting and Vodcasting.....	32
3.4.4	A Vodcast Example Feed	33
3.5	Mobile Video Devices.....	34
3.5.1	iPod Video	35
3.5.2	Cell Phone	36
3.5.3	Personal Digital Assistant (PDA)	37
3.5.4	Sony PlayStation Portable	37

3.6	Relevant Video Formats	39
3.6.1	MPEG-1	39
3.6.2	MPEG-4	39
3.6.3	3GP	40
3.6.4	WMV	41
3.6.5	FLV	41
4	Development Process	42
4.1	Evolutionary Prototyping	42
4.2	Video Upload and File Check	43
4.3	Video Conversion	45
4.4	Video Metadata	51
4.5	Video Preview	54
4.6	New Advanced Search	57
4.7	Vodcasting	60
5	Final Conclusion	62
6	Appendix	64
6.1	Bibliography	64
6.1.1	Books	64
6.1.2	Conference Proceedings	64
6.1.3	University Material	65
6.1.4	Articles	66
6.1.5	Internet Ressources	67
6.2	List of Figures	73
6.3	Extracts from the Source Code	74
6.3.1	config.inc.php	74
6.3.2	functions.inc.php	77
6.3.3	sotf_Programme.class.php	77
6.3.4	sotf_VideoFile.class.php	79
6.3.5	sotf_VideoCheck.class.php	83
6.3.6	editFiles.php	86

1 Introduction

At the very beginning of this diploma thesis, one question arises: Why is the StreamOnTheFly Video Extension so important?

Since more and more users are connected to the Internet using broadband technology, the transfer of large amounts of data with unprecedented speed has become reality. Video consumption over the Web therefore has gained a new significance among certain – especially young – user groups. The amount of video blogs is continuously rising and so is the popularity of huge video platforms such as YouTube.¹

The technological development has also led to new possibilities in the field of "Video on Demand". This means to watch for example a certain TV programme without minding any schedules. It can simply be downloaded anytime and anywhere to a device of your choice. This could be a normal TV screen as well as a modern video enhanced phone or the iPod Video.

This finally leads to the vodcasting hype which began in 2005 and has taken "Video on Demand" to the next level of usability. It has never been easier: The user only has to plug a mobile device to his computer to automatically receive programmes fitting his interests.

StreamOnTheFly is a platform perfectly suited to provide a wide variety of such programmes, if it only would be able to handle video content.

If StreamOnTheFly would indeed include video functionality such as vodcasting, free TV stations and other independent media companies could be attracted by its features and gained as new partners in the network.

That is why the StreamOnTheFly Video Extension was overdue to be realized.

¹ [YOUTUBE]

2 StreamOnTheFly

This chapter gives an overview about the Open Source media archive StreamOnTheFly. It begins with defining the term "Open Source", followed by a short introduction to the GNU General Public License (GPL), which StreamOnTheFly underlies. Reaching the main aspect of this chapter, the historical background of StreamOnTheFly is as well mentioned as its original purpose and functionality.

2.1 Open Source

The term "Open Source" is commonly used for software which is free of license costs (formerly known as "Freeware"), but this is not always true. It is a registered trademark of the Open Source Initiative (OSI) and may only be used for software which underlies certain license models.² The concrete definition of "Open Source" according to the OSI consists of 10 paragraphs:

1. *Free Redistribution*

The license shall not restrict any party from selling or giving away the software as a component of an aggregate software distribution containing programs from several different sources. The license shall not require a royalty or other fee for such sale.

2. *Source Code*

The program must include source code, and must allow distribution in source code as well as compiled form. Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost preferably, downloading via the Internet without charge. The source code must be the preferred form in which a programmer would modify the program. Deliberately obfuscated source code is not allowed.

² cp. ZODL, p. 11

Intermediate forms such as the output of a preprocessor or translator are not allowed.

3. *Derived Works*

The license must allow modifications and derived works, and must allow them to be distributed under the same terms as the license of the original software.

4. *Integrity of The Author's Source Code*

The license may restrict source-code from being distributed in modified form only if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time. The license must explicitly permit distribution of software built from modified source code. The license may require derived works to carry a different name or version number from the original software.

5. *No Discrimination Against Persons or Groups*

The license must not discriminate against any person or group of persons.

6. *No Discrimination Against Fields of Endeavor*

The license must not restrict anyone from making use of the program in a specific field of endeavor. For example, it may not restrict the program from being used in a business, or from being used for genetic research.

7. *Distribution of License*

The rights attached to the program must apply to all to whom the program is redistributed without the need for execution of an additional license by those parties.

8. *License Must Not Be Specific to a Product*

The rights attached to the program must not depend on the program's being part of a particular software distribution. If the program is extracted from that distribution and used or distributed within the terms of the program's license, all parties to whom the program is redistributed

should have the same rights as those that are granted in conjunction with the original software distribution.

9. *License Must Not Restrict Other Software*

The license must not place restrictions on other software that is distributed along with the licensed software. For example, the license must not insist that all other programs distributed on the same medium must be open-source software.

10. *License Must Be Technology-Neutral*

No provision of the license may be predicated on any individual technology or style of interface. ³

2.1.1 The GNU General Public License (GPL)

The GNU GPL is the most common license model among the Open Source community.

The principles of this concept are stated as follows:

Free software is a matter of the users' freedom to run, copy, distribute, study, change and improve the software. More precisely, it refers to four kinds of freedom, for the users of the software:

- *The freedom to run the program, for any purpose (freedom 0).*
- *The freedom to study how the program works, and adapt it to your needs (freedom 1). Access to the source code is a precondition for this.*
- *The freedom to redistribute copies so you can help your neighbour (freedom 2).*
- *The freedom to improve the program, and release your improvements to the public, so that the whole community benefits (freedom 3). Access to the source code is a precondition for this.* ⁴

³ PERENS

2.2 StreamOnTheFly



Figure 1 - StreamOnTheFly's Home Page ⁵

StreamOnTheFly is an Open Source media archive whose primary aim is to offer free radio stations a platform for archiving, publishing and exchanging their programmes.

2.2.1 History

The development of the software began in 2002 as a collaboration of MTA SZTAKI (Computer and Automation Research Institute of the Hungarian Academy of Sciences in Budapest), "Public Voice Lab" (Vorarlberg), "Team Teichenberg" (Vienna) and – as the first radio station to use the software – Radio Orange (Vienna). From the beginning, the project has been funded by means of the European Union.

⁴ [GNU]

⁵ <http://medienarchiv.fh-stpoelten.ac.at>

In 2003, a few more partners joined the project: strg.at (Vienna) and the Universities of Applied Sciences in St. Pölten and Dornbirn, which all are part of the “Competence Network for Media Design”. The latter is a project within the “FH PLUS” programme which was initiated by the Austrian Research Promotion Agency (FFG) in 2002 and since then has carried out at Austria’s Universities of Applied Sciences.

2.2.2 Technical Overview

The StreamOnTheFly network consists of a couple of servers (so called “nodes”) which are all running the same software, but hosting different radio stations. Presently, following nodes are part of the system ⁶:

- *SZTAKI DSD, Hungary*
<http://radio.sztaki.hu/node/>
- *Radio Orange 94.0, Vienna*
<http://sendungsarchiv.o94.at/>
- *University of Applied Sciences St.Pölten*
<http://medienarchiv.fh-stpoelten.ac.at/>
- *University of Applied Sciences Vorarlberg*
<http://sotf.fhv.at/node/>

The variety of programmes originates from the fact that these servers are linked together. Via XML-RPC, metadata are exchanged across the network, so every server shows always up-to-date content of all nodes. The physically existing media files however always stay on the node they had been uploaded to.

In the following figure, a user is requesting a file from “Node 3” where he can see its metadata. In the background, the request is redirected to “Node 1”, where the physical file is stored in this example and then delivered to the user:

⁶ cp. [SOTF_ORG]

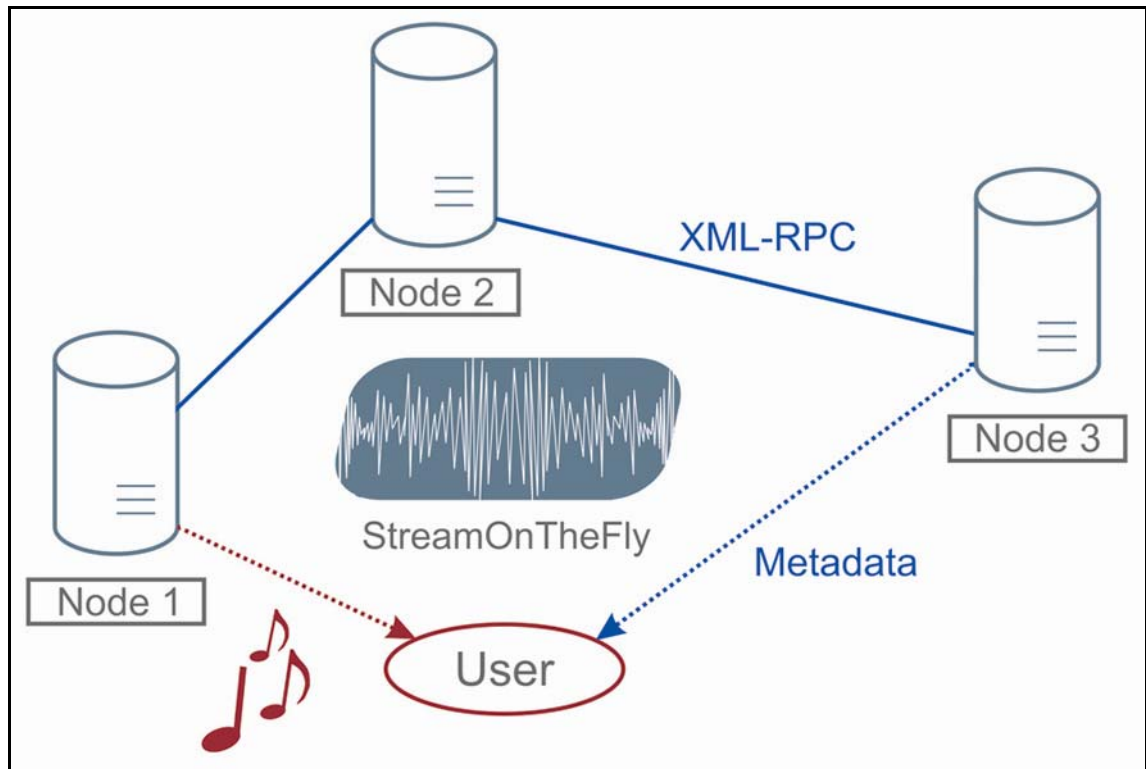


Figure 2 - Schema of the StreamOnTheFly Node Network

The synchronisation routine is part of the `cron.php` script which has to be run at regular intervals (therefore ideally as a cron job) defined by the node administrator.

StreamOnTheFly's system has been designed in compliance with a classical three tier architecture: the data layer is represented by a PostgreSQL database, the logic is written in PHP and the presentation tier is realized with flexible Smarty templates.

To facilitate the development process of the project, SVN has been chosen as versioning system. The constantly updated code is accessible on the project page at BerliOS ⁷.

⁷ [BERLIOS]

2.2.3 Functionality

As StreamOnTheFly's source code has continuously been enhanced for the past four years, it contains plenty of useful features which will be described in the following paragraphs.

When a user accesses the node's home page (see Figure 1), recently added programmes are shown in a list. For the likely case a desired programme is not part of that listing, a quick search is available on the right side of the screen. As an alternative, the user might also select an interesting radio station from a drop-down field to view all programmes which have been published by the chosen one. (A list of participating stations which also includes an overview of their series is also shown under "Stations" in the top navigation.)

If specific content is requested, performing an advanced search will be the right choice. This feature is always available in the navigation menu. In its original version, the advanced search interface looked like this:

Advanced search

Your query is:

Broadcast date	before	2006	08	07	+ -
AND					
Abstract	does not contain	election			+ -
AND					
Topic	contains	politics			+ -
AND					
Language	is	English			+ -

1. Add term to the query

☒ AND ☐ OR

Abstract
Broadcast date
Entry date
Expiry date

Add term Create new query

2. Run query

Sort results by Entry date ☒ Desc

Second sort by Station name ☐ Desc

Run query

Figure 3 - The Original "Advanced Search" Screen

Although it is possible to perform complex searches with this kind of form, users not being familiar with SQL have reported some difficulties. Therefore, a more usable solution was integrated in the context of the StreamOnTheFly video extension. (This new interface will be explained later in Chapter 4.6.)

On the advanced search result page, as well as nearby station or series lists, there is a "Podcast" link placed. Following it, the user reaches the accordant RSS feed of the displayed page. By adding its URL to a podcast client, he or she will automatically receive the new programmes of a station, new episodes of a series or up-to-date search results. (What podcasting exactly is and how it works will be the subject of Chapter 3.)

Another way to find a programme of your interest is to browse the StreamOnTheFly topic tree (Figure 4). It is an outright listing of various subjects a programme might deal with. In the original StreamOnTheFly version for audio programmes, this topic tree was very extensive – containing for example 68 kinds of sports alone. So, it had also to be simplified as a part of the video extension. Concretely, it was cut down from 260 to 126 entries.

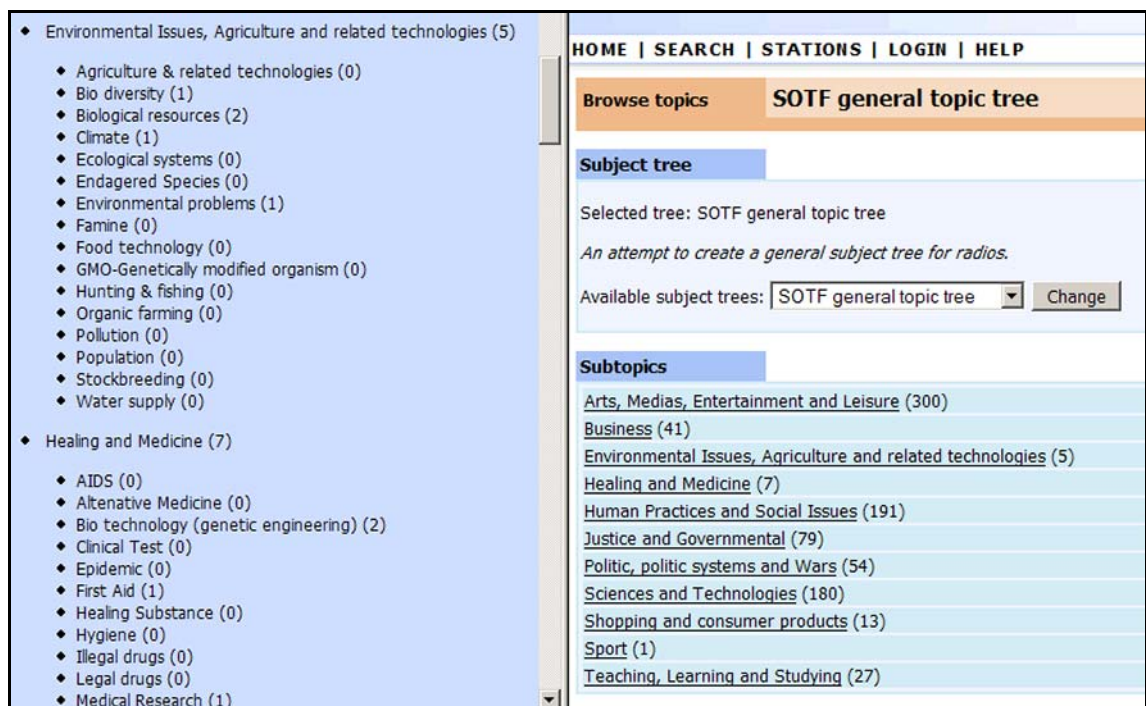


Figure 4 - Extract of the "Topic Tree" Screen

Once a single programme has however been found and is requested to be displayed, the programme detail page appears. It contains all available information about the programme which has been entered by a radio station's editor.

This leads to the next chapter, where the workflow for uploading programmes will be explained.

2.2.4 Functionality on the station editor's side

The process of adding programmes to the StreamOnTheFly network has to be carried out by a station's editor owning the necessary permissions. It mainly consists of these 4 steps:

- Uploading
- Converting
- Entering Metadata
- Publishing

The entry point to perform all these tasks is the "Editor's Console". It is a screen showing a list of recently added files along with their publication status, statistics and other pieces of relevant information.

The screenshot shows the 'Editor's Console' interface. At the top is a navigation bar with links: HOME | SEARCH | STATIONS | TOPICS | MY PLAYLIST | EDITORS' CONSOLE | ADMIN | PREFERENCES (PTMSCHMIDT) | LOGOUT | HELP. Below this is a section titled 'Create new programme' with instructions and links for FTP access and file management. A form allows selecting a file (bravia.mpg) and a series (sunny videos (881)) to add as a new programme. The main section, 'My programmes', displays a table of existing programmes with filters and sorting options. The table has columns for Title / abstract, Station / Series, Dates, Statistics, Flags, and Actions.

Title / abstract	Station / Series	Dates	Statistics	Flags	Actions
Animation Gläser CA-Projekt	sunny videos (881)	Entered: 2006-05-07 Produced: 2006-05-07 Expires:	visits: 49 listens: 0 downloads: 168	Published <input checked="" type="checkbox"/> None <input type="checkbox"/>	Edit files Edit metadata Delete Export in format: <input type="text"/>
TITLE 2 ABSTRACT 2	sunny videos (881) Test Series (881)	Entered: 2006-05-07 Produced: 2006-05-07 Expires:	visits: 34 listens: 2 downloads: 6	Published <input checked="" type="checkbox"/> None <input type="checkbox"/>	Edit files Edit metadata Delete Export in format: <input type="text"/>
boarding by andreas sabitzer	sunny videos (881)	Entered: 2006-05-09 Produced: 2006-05-09 Expires:	visits: 113 listens: 0 downloads: 400	Published <input checked="" type="checkbox"/> None <input type="checkbox"/>	Edit files Edit metadata Delete Export in format: <input type="text"/>
Dancing Computeranimations-Projekt	sunny videos (881)	Entered: 2006-05-09 Produced: 2006-05-09 Expires:	visits: 22 listens: 0 downloads: 70 Rating: 5 (by 1)	Published <input checked="" type="checkbox"/> None <input type="checkbox"/>	Edit files Edit metadata Delete Export in format: <input type="text"/>

Figure 5 - "Editor's Console" Screenshot

The upload of files can be done via HTTP, choosing "Manage your files" on this screen, or via FTP for which there is also a link placed on this page. Latter is the recommended method as HTTP uploads are likely to be aborted

when it comes to large video files. The FTP authentication is mapped against the PostgreSQL using mod_sql for ProFTPD ⁸, so a user can login to the server by providing his node access data.

Once the upload is done, the file will be listed in the selection field under “Create new programme”. In the background, all files in the user’s personal directory on the node server are checked with getID3 ⁹, a free PHP library for extracting useful information from MP3 and other media files. After they have been declared valid, they are offered for selection under “Create new programme”. In a second drop-down field the stations to which the logged in editor is allowed to add programmes are listed. This choice done as well, a click on “Add as new programme” opens the second screen of the process: the conversion page (editFiles.php).

This page will be very important in the context of the video extension. That’s how it looks like on a pure audio node:

Add new programme - step 1 Go to step 2 of 2

Files and links associated with programme

Programme audio

Filename	Format	Size	Last modified	Length	Stream access	Download access	
missing	24kbps_1chn_22050Hz.mp3						Convert from others
missing	128kbps_2chn_44100Hz.mp3						Convert from others
missing	64kbps_2chn_22050Hz.ogg						Convert from others
all_that_192kbps_2chn_44100Hz.mp3	192kbps_2chn_44100Hz.mp3	5745395	2006-08-09 01:54:12+02	239 s	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Delete

Add other formats of audio content Convert all required formats

Associated files

No files

Add files

Links

No links

Add new link

Figure 6 - editFiles.php on an Audio Node

⁸ [MOD_SQL]

⁹ [GETID3]

The main part of the screen belongs to the conversion facility which is handled with lame, a MP3 conversion tool which is freely available under the GNU license as well as with oggenc, which – as its name says – is responsible for ogg encoding. The formats listed here can be modified by the node administrator in StreamOnTheFly's config file in terms of bitrate, sample rate and numbers of channels.

This way, a small amount of available memory space might be compensated by for example choosing a lower bitrate. However, in the case of audio, this subject is not as severe as it will be when large video files are stored in the file system.

Most often, an editor will use the function "Convert all required formats", but he might also pick some formats out of the list to start the conversion separately. In any case, a popup window opens which dynamically shows the progress of the conversion. When it is done, an JavaScript alert window gives information about possible errors or conversion success.

It is also offered on the conversion screen to add additional files somehow related with the programme and to place a list of interesting web links.

The page will be left over the button "Go to step 2 of 2" which leads to editMeta.php, the page where the metadata manipulation is handled

The forms and buttons placed on this screen allow the editor to:

- add a programme to an existing series
- define a title, the language, a short abstract, some keywords and the production date (required fields)
- enter some additional metadata information as for example an alternative title or a programme genre
- associate topics from the StreamOnTheFly Topic Tree with the programme
- add persons who have contributed to the creation of the programme and to define their roles

- disclose who owns the rights of the programme (might be combined with a start and stop time, in the case it contains e.g. a music track)
- grant node users the permission to modify or delete the programme (whereas permissions from the respective station or series are inherited)
- upload an icon which will be displayed on the programme detail page and on the node's home page under "New Programmes".

As soon as everything has been completed according to the editor's desiderata, he has the possibility either to immediately publish the programme for every node user to see or to just save the entered data. If he does not publish his programme, the respective checkbox in the "Editor's Console" will remain unchecked. The publish process can then be initiated later by simply checking this field.

2.2.5 Future Prospects

Currently, there is work in progress at the University of Applied Sciences St. Pölten as well as in Budapest at MTA SZTAKI to redesign the node's user interface according to WAI (Web Accessibility Initiative) standards. A tableless renewal of the layout is overdue and so is a flexible appearance of the content using CSS.

Furthermore, it is planned to join in the "Web 2.0" hype by integrating a few AJAX elements on pages where it makes sense. In this context, a real-time search field could as well be implemented as a check whether a chosen username already exists during the registration process.

The most comprehensive plan however is to open the StreamOnTheFly nodes to a new target group: private podcasters. This would require a more usable adaptation of the whole user interface, especially of the "Editor's Console" to make uploading, file conversion and editing metadata more comprehensible. Apart from that, a considerate marketing concept would have to be applied.

3 Podcasting and Vodcasting

This chapter will explain these two terms, outline their history and show functional principles. Additionally, various types of mobile video devices will be introduced.

3.1 Overview

*Podcasting is the process of capturing an audio event, song, speech, or mix of sounds and then posting that digital sound object to a Web site or "blog" in a data structure called an RSS 2.0 envelope (or "feed").*¹⁰

The word podcasting itself derives from Apple's popular MP3 Player "iPod". This fact often leads to the misconception that this technology might only be used with that special device. In fact, this is not true at all. Almost every MP3 player is capable to handle podcasts. Often, they are even consumed on a normal PC.

The latter is especially often the case when it comes to video podcasting. The short word for this enhanced form of podcasting, "vodcasting", includes the syllable "vod-" which stands for "Video On Demand". That indicates well the nature of podcasting: Once audio or video material is created and published, the user is free to consume the content whenever he likes. (A more detailed description of the pod-/vodcasting workflow will follow in Chapter 3.4 - "How it works".)

3.2 History of Podcasting

Unless podcasting is regarded as a brand new thing, its beginnings date back to the year 2000.

¹⁰ MENG, p. 1

3.2.1 RSS 0.92 and "Audioblogging"

The concept of podcasting as the delivery of digital audio data via RSS enclosures was first mentioned by Tristan Louis, who mapped out a draft on October 13, 2000:

I'd like to suggest a few optional additions to the specification. Here are some ideas I'd like to throw around for discussion:

[...]

<sound></sound>: Points to either a VXML source file (which can be read by a VXML browser) or a sound file. For example, it could serve up a radio feed related to this story.

<video></video>: Same as above with video or SMIL file. ¹¹

Dave Winer then implemented Louis' suggestions as the known "enclosure"-tag in RSS 0.92. The first time it was used was on January 11, 2001. That day, Winer placed a song by "Grateful Dead" in his weblog this way. ¹²

The technology was also integrated in Winer's weblogging software "Radio Userland". By doing this, "Audioblogging" has become reality.

Cool to hear my own audio-blog post on the webtalk guys radio show where Mitch Ratcliffe guest hosted. I've included the full show as an enclosure in my RSS feed, if you are using Radio UserLand, you will automagically [sic] receive the mp3 file for immediate playback. ¹³

3.2.2 Getting mobile

But the developers have still not been totally content. Kevin Marks, having been part of the same developer group as Winer, meant on October 1, 2003:

¹¹ LOUIS

¹² WINER

¹³ CURRY 2002

Adam Curry and I had a chat about trying something more like blogging using the RSS 'enclosures'. I have the beginnings of a tool to automatically move audio posts into iTunes (and hence iPods) as I just can't listen to speech radio at the computer - I need to do it while driving. ¹⁴

Therefore, an Apple script was written which Adam Curry offered for download in his weblog on October 12, 2003 – almost exactly three years after Tristan Louis' draft. The script was called "RSS2iPod" and represented the missing link between "Radio Userland" and MP3 players. An important step to real podcasting has been taken:

Mac users can use this folder applescript to automagically [sic] update your iPod with any new mp3's that are downloaded to your Radio UserLand enclosures folder from enclosure aware RSS feeds. ¹⁵

In September 2004, the first client with a graphical user interface which automatically downloaded audio data enclosed in RSS feeds was released: "iPodderX", developed by August Trometer and Ray Slakinski from "Thunderstone Media". (Currently, the application is being rewritten under the name "Transistr". ¹⁶)

Finally it was on February 12, 2004 when the term "podcasting" appeared first on the Web. Ben Hammersley mentioned it in an entry of "The Guardian Technology Blog":

With the benefit of hindsight, it all seems quite obvious. MP3 players, like Apple's iPod, in many pockets, audio production software cheap or free, and weblogging an established part of the internet; all the ingredients are there for a new boom in amateur radio. But what to call it? Audioblogging? Podcasting? GuerillaMedia? ¹⁷

¹⁴ MARKS

¹⁵ CURRY 2003

¹⁶ cp. [THUNDERSTONEMEDIA]

¹⁷ HAMMERSLEY

In September 2004, Dannie J. Gregoire also used the term “podcaster” in a post to the “ipodder-dev” newsgroup on “how to handle getting past episodes”.

I guess one could argue that this is simply an rss/server side issue, and that the “podcaster” (yes, I like making up new words) should be responsible enough to offer a page of seperate feeds of old sodes by month/year/season/etc. ¹⁸

As Winer and Curry has taken up the expression “podcasting” as well, nothing could stop the widespreading of the term.

3.2.3 “Taking the world by storm”

The popularity of this new way of getting audio out of the Web extremely increased in winter 2004. Rex Hammock wrote in his blog on February 20, 2005:

About 20 weeks ago, on September 28, 2004, I heard the term “podcast” for the very first time when Doc Searls explained it on his “IT Garage” blog. He said then that the word podcast returned 24 results on Google. After reading the long, long NY Times piece yesterday on podcasting, I wondered how many results the word would return. Yesterday, it was 687,000, today it is 1,700,000. ¹⁹

In the meantime, the number of results has risen to 344,000,000.

In 2005, podcasts sprung up like mushrooms all over the Internet. In February, the first official podcast production company “Pwop Productions” was founded. In March, John Edwards was the first politician to launch his own podcast. In May, the first book on podcasting was written by Todd

¹⁸ GREGOIRE

¹⁹ HAMMOCK

Cochrane: "Podcasting – The Do It Yourself Guide". And in June, podcasting functions were added to Apple's iTunes (Version 4.9). ²⁰

Podcasting also has found its way to the Guinness Book of Records. The American comedian Ricky Gervais has got the entry for the "Most Successful Podcast" in the 2007 edition. His feed, which he launched in February 2006, reached 261.670 in its first month.

3.3 History of Vodcasting

Of course, the history of video podcasting is much shorter. It began on November 14, 2004 when Steve Garfield first used the expression "video podcast":

This is an experiment at doing a VideoPodcast. You can listen to it ~~on your iPod~~ in iTunes [sic], but also watch it on your computer. I'm preparing for the day when you can watch it on your iPod! ²¹

On March 15, 2005, the first episode of the Canadian video blog series "Tiki Bar TV" was published on iTunes. ²² On October, it regained popularity as it was used as a sample vodcast when Apple presented its "iPod Video":

When Apple Chief Executive Steve Jobs flashed a clip of the video blog Tiki Bar TV during the October launch of Apple's new video iPod, he turned the rollickingly sophomoric show into an instant online hit. Filmed in a 1950s-style bachelor pad, the bimonthly program is a farcical series of ad-libbed skits built around cocktails with names like the Volcano and the Red Oktober. Tiki Bar TV was launched on a lark about 10 months ago, and it attracts about 200,000 viewers, an audience the size of some established cable shows. ²³

²⁰ cp. WIKIPEDIA

²¹ GARFIELD

²² cp. [PODCAST_NET]

²³ GREEN

The first vodcast portal, vodcast.nl, was launched on July 3, 2005 in the Netherlands by Stef van der Ziel.²⁴

Also in July, the first vodcast directory, vodcasts.tv, went online.

As already mentioned in the last citation, Apple's iPod Video was introduced on October 12, 2005. This of course led to another vodcast boom: Video podcasts received over iTunes could now easily be transferred to an appropriate gadget to view them on the way.

The first official German institution which was to offer video podcasts was ARD. Since November 2005, the newscast "Tagesschau" is available in that format on a daily base.²⁵

On January 9, 2006, Google Video was launched:

*We have officially launched the Google Video store, the world's first open video marketplace. The store features thousands of titles representing a wide variety of entertainment and educational content, and more content is being added each day. With a quick keyword search, you can access a trove of high quality media for sale in addition to the free content already part of Google Video.*²⁶

The video podcasting hype finally also reached Germany's chancellor Angela Merkel. She has weekly been publishing a video feed since June 8, 2006 on currently important subjects such as integration of immigrants or the new health reform.²⁷

And still, there lies an enormous potential of growth in video podcasting. It is to expect that the day cheaper mobile video devices conquer the market

²⁴ cp. [VODCAST_NL]

²⁵ cp. [TAGESSCHAU]

²⁶ EVA

²⁷ [BUNDESKANZLERIN]

vodcasting will also be attractive to less wealthy target groups as a source of information and entertainment.

3.4 How it works

Podcasting is not complicated. It mainly consists of these four steps:

- I. Create or capture and edit the content.*
- II. Publish content to a web site or blog.*
- III. Subscribe to the content using an "RSS News Reader".*
- IV. Download the content [...].²⁸*

Creation and publication have to be done by the author of a podcast while subscription and download happen on the user's side.

As for vodcasting, the workflow is mainly the same. The most significant difference is, that the required RSS <enclosure> tag does not contain a file of type "audio/mpeg" but for example a "video/mov" file.

3.4.1 Creation and Publication

First of all, the content for the podcast has to be produced. Therefore, a podcast's author will need some capture devices and editing software. Either this person is member of a company which professionally deals with media production – like a radio station – or a private podcaster.

In the first case, the required devices and software will probably be already available as a part of the studio environment.

A private podcaster will first have to buy an audio or video recording device, depending on which kind of podcast he or she wants to realize. However, they do not have to be the most expensive products of their kind. For

²⁸ MENG, p. 3

publishing captured material over the Internet, even cheap solutions (available from € 200,--) might be sufficient.

Neither, much money has to be spent on professional production software such as Adobe products. There are many open source solutions available – for example Audacity²⁹ could be used for audio editing while Jahshaka³⁰ is a free but powerful video editor.

Once the media has been produced, it has to be compressed and encoded to an appropriate format so it can be easily transferred over the Internet and subsequently used as a podcast. Technically, there are no restrictions, but it has become common use that audio podcasts contain MP3 files ranging from 56 to 160 kBit/s at either 22 kHz 16-bit for pure voice or 44.1 kHz 16-bit resolution for music programmes.³¹ When it comes to video podcasts, it will have to be decided which players should be supported. A comparison of video devices and codecs which come into question will be the subject of Chapters 3.5 and 3.6, but in general, the following can be stated:

*You should choose 320x240 or a simple multiple (160x213 or 120x160). You should consider a nominal bitrate (for the higher resolutions) from 200 kbps-300 kbps, a maximum of 400 kbps (for variable bit rates), and 10-20 frames per second with an audio sample rate of 22,050.*³²

After completed encoding, the next step is to publish the podcast. Therefore, the media file has to be embedded into a RSS 2.0 feed which is readable by news readers. (RSS 2.0 is - to prevent any misunderstandings - not the successor of RSS 1.0, but a completely different standard.)

There are basically three practical ways to make a compatible feed for a podcast or video vlog that contains the proper media enclosure elements:

²⁹ [AUDACITY]

³⁰ [JAHSHAKA]

³¹ cp. FELIX, p. 196

³² HERRINGTON, p. 380

1. *Have your blogging application generate it.*
2. *Use a third-party service, such as FeedBurner.*
3. *Write it yourself or use an application, such as FeedForAll.* ³³

FeedBurner, which is mentioned in this citation of Lionel Felix, is indeed great to optimize and maximize compatibility of already existent feeds as well as to transform RSS 1.0 or ATOM feeds into RSS 2.0 format.

Unfortunately, it is not suitable for building podcasts from the scratch.

If you do not use a blogging application which offers podcasting support (such as WordPress ³⁴), you will have to use FeedForAll or a comparable solution for this purposes. Additionally, there exist a lot of useful scripts in different programming languages which are able to automatically generate RSS feeds as well as other applications with a graphical user interface.

However, it is not difficult to manually create a RSS feed, if an example (see Chapter 3.4.4) is available to be modified. Often, it is even faster to do it that way then using software offering plenty of form fields for metadata of which only a few will fit the needs of a podcast's author.

3.4.2 Subscription and Download

Once the podcast has successfully been published on a web server it is ready to be received by interested listeners.

To subscribe to the podcast, they will have to add the accordant URL to the podcast list in their RSS reader. Software tools of this category which have integrated specific functions for podcasts and vodcasts are also called "podcatchers". Popular podcatchers also supporting vodcasting are for

³³ FELIX, p. 112

³⁴ [WORDPRESS]

example FireAnt³⁵, PodSpider³⁶ or FeedDemon³⁷. (FireAnt is the only one of these three products which is free of license costs.)

However, the most convenient way to get podcasts and vodcasts is to use Apple iTunes, Apple's media jukebox:

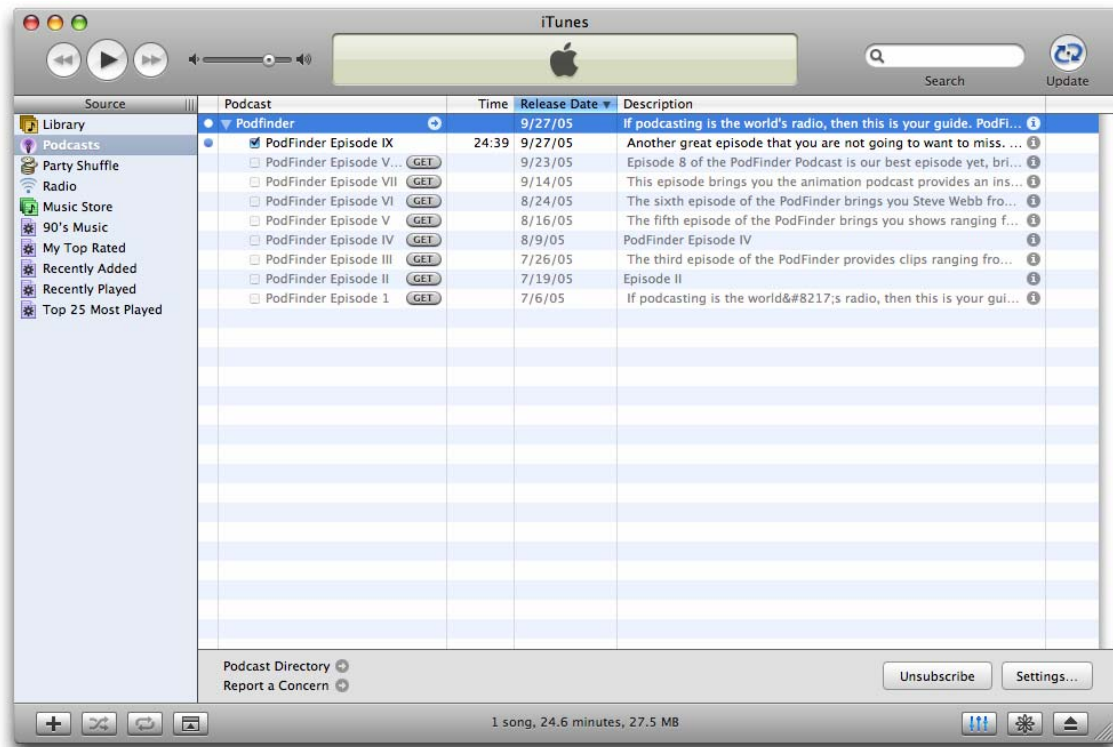


Figure 7 - Receiving Podcasts in iTunes ³⁸

It is free to use, but a commercial application: On the one hand, users are tempted to buy music in the integrated extensive music store with their credit cards and on the other hand, some great syncing features are only available if an iPod is connected to the computer.

³⁵ [FIREANT]

³⁶ [PODSPIDER]

³⁷ [FEEDDEMON]

³⁸ <http://www.apple.com/de/itunes/overview/>

Against iTunes is also well usable with other mobile media devices, though. As it saves all podcast episodes in a specific folder on the user's hard drive, it is possible to transfer new files to any connected mobile media player.

Whether iTunes or another podcatcher is used, the principle of receiving a podcast is always the same: As soon as a new feed is added, the software will look up existing episodes on the server where the feed has been placed, identified by the <title> tag in the corresponding RSS file.

After these initial entries have been downloaded, this procedure will repeat on every start-up of the tool or at regular intervals according to user settings. Whether only metadata is transferred or also enclosed media files depends on each product. Downloaded files can be viewed in a built-in media player (in iTunes, FireAnt and PodSpider) or with the application system-wide associated with the according file type (in FeedDemon).

3.4.3 Schematic Overview – Podcasting and Vodcasting

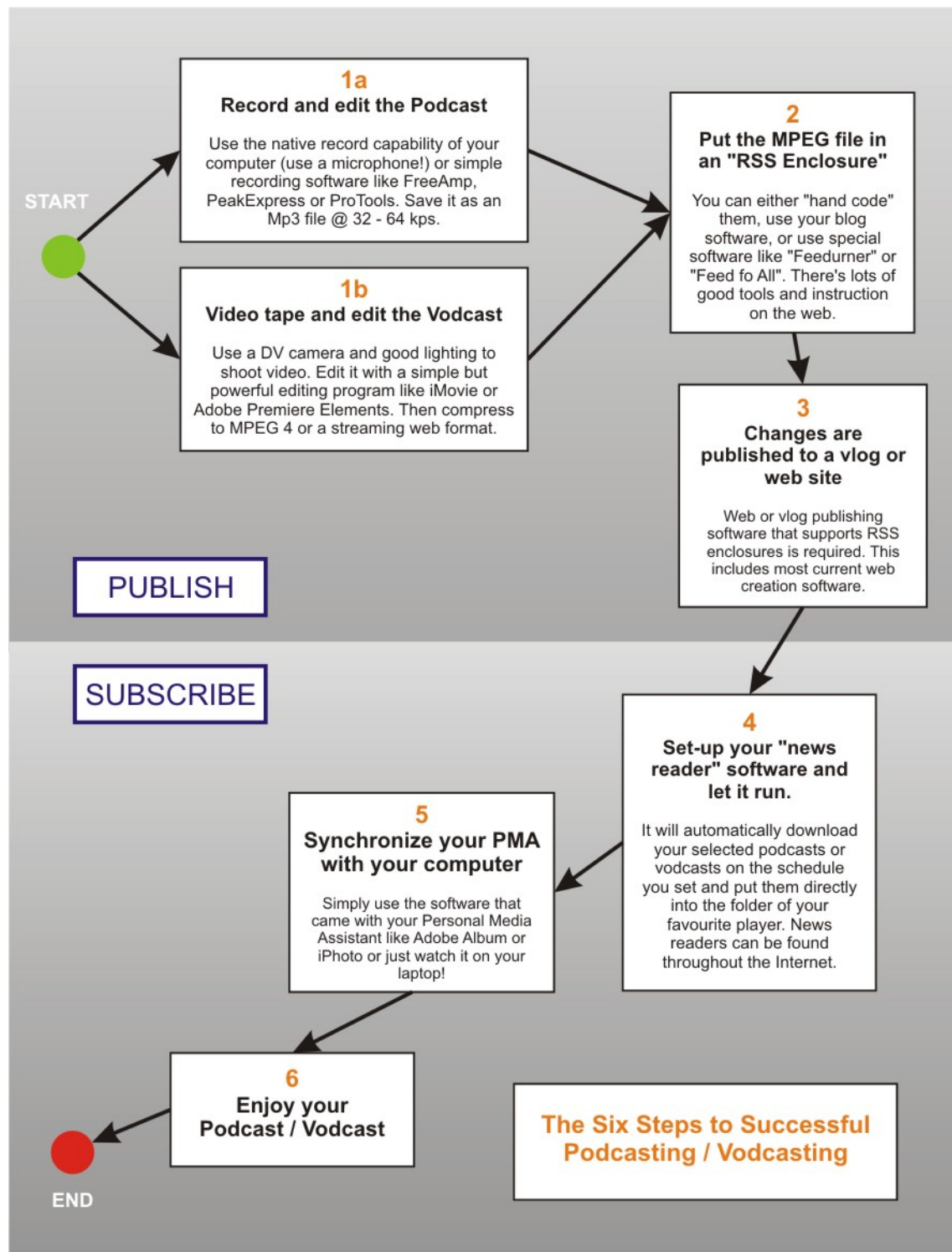


Figure 8 - Podcasting/Vodcasting Workflow ³⁹

³⁹ cp. MENG, p. 11f

3.4.4 A Vodcast Example Feed

Let's have a look on how exactly a vodcast RSS 2.0 is compound. (In the following example ⁴⁰, the tags beginning with "itunes:" are in fact not necessary for a video podcast to work, but permit full compatibility with Apple iTunes.)

```
<?xml version="1.0" encoding="UTF-8"?>
<rss xmlns:itunes="http://www.itunes.com/dtds/podcast-1.0.dtd" version="2.0">
  <channel>
    <title>Vodcast Title</title>
    <itunes:author>Your Name</itunes:author>
    <link>http://www.sitename.com</link>
    <description>A description of your vodcast channel</description>
    <itunes:subtitle>A subtitle for your vodcast channel</itunes:subtitle>
    <itunes:summary>A summary of your vodcast channel</itunes:summary>
    <language>EN</language>
    <lastBuildDate>Wed, 19 Apr 2006 10:00:00 GMT</lastBuildDate>
    <copyright>(c) 2005 Your Name</copyright>
    <itunes:owner>
      <itunes:name>Your Name</itunes:name>
      <itunes:email>youremail@example.com</itunes:email>
    </itunes:owner>
    <category>Technology</category>
    <itunes:category text="Technology"></itunes:category>
    <itunes:explicit>no</itunes:explicit>
    <item>
      <title>Your Movie Title</title>
      <itunes:author>Your Name</itunes:author>
      <description>A description of this movie</description>
      <itunes:subtitle>A subtitle about this movie</itunes:subtitle>
      <itunes:summary>A summary of your movie</itunes:summary>
      <enclosure url="http://ab.com/xy.mp4" length="1024" type="video/mov"/>
      <guid>http://www.yoursitename.com/movienam.mp4</guid>
      <pubDate>Wed, 19 Apr 2006 10:00:00 GMT</pubDate>
      <itunes:duration>00:01:35</itunes:duration>
      <itunes:keywords>keyword1, keyword2, keyword3</itunes:keywords>
    </item>
  </channel>
</rss>
```

⁴⁰ cp. BREEN

Most of the used tags are self-explaining. However, a few important details should be mentioned:

The child elements of the `<channel>` tag describe the vodcasting channel as a whole and may contain many single `<item>` sections.

As already mentioned, the most important field for podcasting of any kind is the `<enclosure>` tag. It must contain the exact URL of the embedded media file as "url", the file size in Bytes – and not the duration in seconds! – as "length" and finally the appropriate MIME type as "type".

There are two date tags: `<lastBuildDate>` for the channel as a whole and `<pubDate>` for each item. It is important that these dates are indicated in compliance with the RFC 2822 standard ⁴¹ as it can be seen in the example feed. Short forms like "06-07-18" are not acceptable.

As RSS is a XML variant, HTML entities are not understood. For example, if a non-breakable space is needed, " " has to be used instead of " ".

3.5 Mobile Video Devices

During the past few years, plenty of modern technological gadgets have conquered the markets around the world. Especially, various kinds of MP3 players with continuously rising memory capabilities have enjoyed great popularity. In consequence, mobile devices which are also able to play back video have been released some of which have been chosen to be described and compared in this chapter.

However, this is the place to repeat that these special devices are not necessarily required for vodcasting as a normal PC can equally be used for it.

⁴¹ [RFC]

3.5.1 iPod Video



Figure 9 - iPod Video ⁴²

The by far most specialized device in terms of vodcasting is without a doubt Apple's iPod Video which has been released in October 2005 as an advancement of the original (audio) iPod which had been introduced to the market in autumn 2001.

The iPod Video requires one of the following video formats:

H.264 video: up to 768 Kbps, 320 x 240, 30 frames per sec., Baseline Profile up to Level 1.3 with AAC-LC up to 160 Kbps, 48 Khz, stereo audio in .m4v, .mp4 and .mov file formats.

MPEG-4 video: up to 2.5 mbps, 480 x 480, 30 frames per sec., Simple Profile with AAC-LC up to 160 Kbps, 48 Khz, stereo audio in .m4v, .mp4 and .mov file formats. ⁴³

Additionally, a variety of audio only formats are supported as well.

Together with iTunes, which was already mentioned in Chapter 3.4.2, it is very easy to subscribe to podcasts and automatically download them on an iPod as soon it has been connected via USB.

⁴² <http://www.apple.com/ipod/ipod.html>

⁴³ cp. [APPLE]

With either 30 or 60 Gigabytes of memory space, the iPod Video offers unequalled opportunities and is therefore also one of the most expensive mobile media players on the market.

3.5.2 Cell Phone



Figure 10 - Nokia 6230i ⁴⁴

The Nokia 6230i was one of the first phones on the European market which offered an integrated video player.

A selection of its features, taken from the manufacturer's product specification ⁴⁵:

- *Integrated video player for download and playback or for streaming: 3GPP, H.263 video, MPEG-4 and AMR*
- *Music player supports AAC, MP3 and M4A formats*
- *Video encoding and playback in QCIF format with sound*
- *AMR, the 3GPP speech codec, provides improved sound quality*

The Nokia 6230i has been used as a testing device during the development process of the StreamOnTheFly Video Extension.

⁴⁴ http://www.nokia.at/german/about_nokia/press/fotoarchiv_aktuell.html

⁴⁵ cp. [NOKIA]

3.5.3 Personal Digital Assistant (PDA)



Figure 11 - T-Mobile MDA Compact ⁴⁶

Personal Digital Assistants – abbreviated as PDAs – have existed for a long time on the market and can be regarded as the first mobile video devices.

As many PDAs run on Windows Mobile, which had formerly been known as Windows CE, WMV is the video format of choice when it comes to playback videos without having to install additional third-party tools.

The PDA used for testing purposes in the context of the Video Extension was T-Mobile's MDA Compact.

3.5.4 Sony PlayStation Portable



Figure 12 - Sony PlayStation Portable ⁴⁷

The PlayStation Portable (also known as PSP) is more than a mobile gaming console. It is further able to playback music, to display photos and to show

⁴⁶ http://www.theunwired.net/media/news/t-mobile_mda_compact2_stylus.jpg

⁴⁷ http://eu.playstation.com/iw_images/assets/images/news/05Apr/psp_announcement_01.jpg

video clips on a relatively large screen with great resolution. It has been sold over more than a half million times on its very first days on the market. ⁴⁸

According to Sony, the PlayStation Portable is able to playback the following video formats:

on "UMD": H.264/MPEG-4 AVC Main Profile Level3

on "Memory Stick": MPEG-4 SP, AAC ⁴⁹

Nevertheless, although it has been tried many times using different FFmpeg settings, no fully PSP compatible video could have been created. Finally, it was managed to transcode a video which was correctly displayed on the PlayStation Portable by using FFmpeg's "-f psp" function. Nevertheless, the title of the video was still not shown after several different trials to achieve it.

This is an especially unsatisfying solution in the case when many videos are downloaded from StreamOnTheFly and are present on the screen to choose from, but none of it can be identified as the titles are missing. Therefore, PSP support could not have been integrated in the StreamOnTheFly Video Extension so far. The day FFmpeg will provide better results concerning this problem it will however be easy to add this additional format in the node's configuration file by entering the appropriate parameters.

⁴⁸ cp. [SONY_PRESS]

⁴⁹ cp. [SONY_PSP]

3.6 Relevant Video Formats

3.6.1 MPEG-1

MPEG-1 was the first video compression algorithm developed by the ISO. The driving application was storage and retrieval of moving pictures and audio on digital media such as video CDs using SIF resolution (352x240) at 30 fps. The targeted output bitrate was 1.15 Mbps, which produces effectively 25:1 compression. ⁵⁰

For the StreamOnTheFly Video Extension, MPEG-1 was finally not used, but compared with FLV in terms of quality and transcoding performance. Although MPEG-1 achieved slightly better results in these tests, we finally decided to use a flash player for our video preview facility (see Chapter 4.5).

As such Flash based solutions have become more and more widespread, the popularity of MPEG-1 throughout the Internet has significantly decreased over the past years.

Nevertheless, it is still a serious alternative to modern file formats, especially in terms of compatibility as almost every media player software is able to play back MPEG-1 files.

3.6.2 MPEG-4

MPEG-4 was initiated by the ISO as a follow-on to the success of MPEG-2. Some of the early objectives were increased error robustness supporting wireless networks, better support for low bitrate applications, and a variety of new tools to support merging graphic objects with video. Most of the graphics features have not gained significant traction yet in products and implementations have focused primarily on the improved low bitrate compression and error resiliency. ⁵¹

⁵⁰ GOLSTON, p. 7

⁵¹ same, p. 9

For StreamOnTheFly, especially MPEG-4 ASP (Advanced Simple Profile) is relevant. This MPEG-4 variant has also served as base for such popular codec algorithms as DivX or XviD.

An even more efficient codec is its successor: MPEG-4 (Part 10) Advanced Video Coding, also known as H.264/AVC. This compression technology offers unprecedented quality at even low bitrates. It is described to be three times as efficient as MPEG-2.⁵²

Although H.264/AVC is the codec of choice also for new DVD technologies such as HD-DVD and Blu-ray Disc, it has not become a default video format in StreamOnTheFly for mainly two reasons: In the first place, the encoding process is extensive in terms of time and memory space and secondly, videos encoded using MPEG-4 ASP provided better compatibility with our test devices. Those clips could be played back equally on Sony's Playstation Portable, Apple's iPod Video and in the Quicktime media player.

3.6.3 3GP

The container format 3GP, which today is mainly implemented in mobile phones, is often used in connection with the H.263 codec although it also supports MPEG-4.

*H.263 was developed after H.261 with a focus on enabling better quality at even lower bitrates. One of the major original targets was video over ordinary telephone modems that ran at 28.8 Kbps at the time. The target resolution was from SQCIF (128x96) to CIF. The basic algorithms are similar to H.261 [...].*⁵³

However, as it has been developed to be used for video conferences, H.263 is not very suitable for movie sequences containing spontaneous, fast movements.

⁵² cp. SULLIVAN, p. 1

⁵³ GOLSTON, p. 6

The audio formats supported by 3GP are AMR (Adaptive Multi-Rate) and AAC (Advanced Audio Coding). Preferably, AMR should be used as tests have shown that AAC causes sound problems on some phones.

3.6.4 WMV

Precisely, Windows Media Video (WMV) is not a video file format but a group of video codecs having been developed by Microsoft since. Mostly, ASF (Advanced Streaming Format) or AVI are used as container formats.

In the context of the StreamOnTheFly Video Extension, WMV-2 was used. This version of WMV was introduced with Microsoft's Windows Media Player 8 in 2001.

The advantage of WMV videos is that they can be shown on a Windows PC without installing additional software as well as on PDAs running on "Windows Mobile" – formerly known as "Windows CE".

3.6.5 FLV

*Since the 2002 introduction of Flash video, Macromedia Flash Player has become the most widely installed Internet video client, running on over 96% of all Internet-connected personal computers. Also, Flash Player runs on a wide variety of platforms and operating systems. The ubiquity of Flash Player ensures that most visitors can view Flash video without downloading additional plug-ins, so you can reach more people with lower development, testing, and support costs.*⁵⁴

The Flash Video Format (FLV) is very important in the context of the StreamOnTheFly Video Extension as it is used to facilitate a flexible video preview solution. FLV is based on an implementation of the H.263 codec.

⁵⁴ [FLASH]

4 Development Process

This chapter contains a definition of evolutionary prototyping together with a justification why it has been chosen as appropriate development concept for the StreamOnTheFly Video Extension. The following main part however consists of six subchapters that describe the most important stages of the implementation process:

- Video Upload and File Check
- Video Conversion
- Video Metadata
- Video Preview
- New Advanced Search
- Vodcasting

The changes in the code are however not explained line by line, rather especially interesting parts of the implementation process have been selected to be described.

4.1 Evolutionary Prototyping

Often prototyping is an iterative process, involving a cyclic multi-stage design/modify/review procedure. This procedure terminates either when sufficient experience has been gained from developing the prototype (in the case of throw-away prototyping), or when the final system is complete (in the case of evolutionary prototyping).⁵⁵

The latter is the model which has been selected for the development of the StreamOnTheFly Video Extension as it fits well the given conditions: Evolutionary prototyping is suitable for projects of small or medium size

⁵⁵ GORDON

without concrete specification that are based on working methods allowing many iterations within a short period.

The life cycle of a project using evolutionary prototyping can be illustrated as follows:

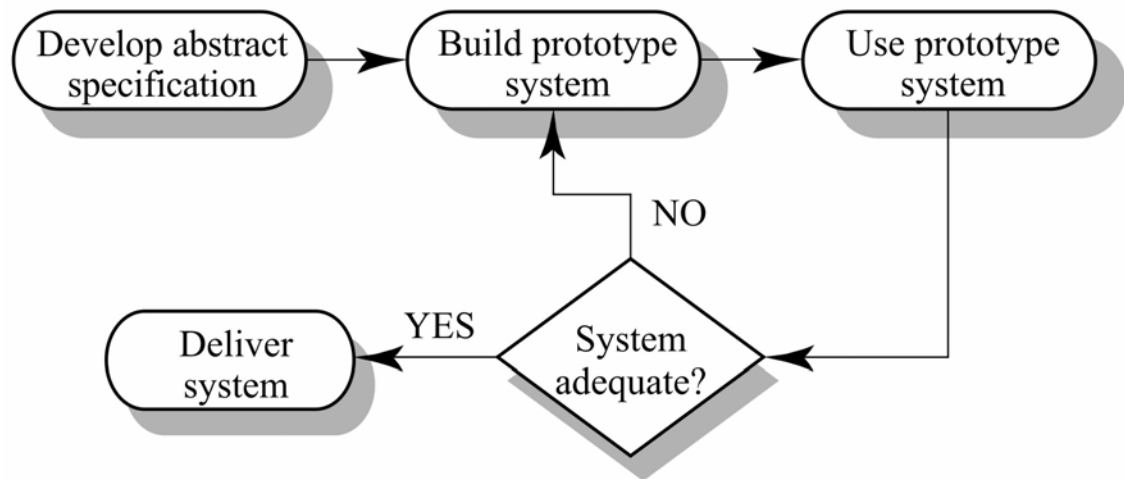


Figure 13 - Evolutionary Prototyping ⁵⁶

Each of the following chapters describing some important stages of the development will be divided into two parts: First, a few paragraphs state the requirements of each project stage, many of which are based on a kick-off meeting having taken place at the University of Applied Sciences Dornbirn in December 2005. In a second part significant problems having occurred during the realization are described as well as the solutions having been found.

4.2 Video Upload and File Check

4.2.1 Requirements

As videos are naturally much larger than audio files, HTML upload is only suitable for short clips. Transferring the files to the server via FTP is a solution but in fact not the most usable one. Therefore, the creation of a rich

⁵⁶ SOMMERVILLE, p. 142

client has been discussed. If FFmpeg was integrated into this client, this application could equally be used for processing transcoding tasks directly on the user's computer and thus for crucially reducing the server load. Additionally, it could contain forms for entering metadata. Finally, the completely prepared programme would then be transferred at once to a StreamOnTheFly node as a XBMF ("Exchange Broadcast Binary and Metadata Format") container.

*XBMF can be used for transfer and archival of radio programmes coupled with metadata. This is a simple container format, in which metadata (in XML format), content files and other associated files can be packaged together.*⁵⁷

The import routine for such XBMF files is already part of StreamOnTheFly and therefore could easily be adapted to integrate the file returned from the rich client into the system.

Checking the videos for their validity and integrity is another task which has to be performed before adding files to the system. Such functionality would have to be integrated in a "Rich Client" but in any case, the existent file check in the Editor's Console has to be extended for video content, too.

4.2.2 Implementation

In the first half of 2006, the primary focus of the development was a seamless integration of video functionality in the already existent source code of StreamOnTheFly audio nodes. Therefore, the idea of a rich client was not put into practice.

However, the video check before offering a file for selection in the Editor's Console has been integrated. Therefore getID3, the PHP library included in StreamOnTheFly and responsible for extracting relevant information from media files, was updated from version 1.6.2 to version 1.7.5. This resulted in a more stable and mature solution available as many bugs were fixed and some modules for new file formats were added between the releases of

⁵⁷ ALTON-SCHIEDL, p. 2

these two versions. The greatest change above all was a class-based restructuring of the whole code in version 1.7.0b1, which made the use of `getID3` more flexible and transparent and resulted in a better structured return array containing all possible information about the file given.

In version 1.6.2, the appropriate line for getting required data about a MP3 file was the following:

```
$ThisFileInfo = GetAllFileInfo($file, 'mp3', false, false, false); 58
```

In comparison, the new call procedure which has been introduced in StreamOnTheFly across the whole code looks like this:

```
$getID3 = new getID3();  
$ThisFileInfo = $getID3->analyze($file);  
getid3_lib::CopyTagsToComments($ThisFileInfo); 59
```

Now, additional settings can be applied by simply setting the corresponding property of the `getID3` object (e.g. `$getID3->option_max_2gb_check = false;`). Thus, no more complicated parameters have to be used.

If the `$ThisFileInfo['video']` is contained in the array returned from `getID3`, a video file has successfully been identified. This is the base to decide the creation of a video object in StreamOnTheFly. Once this has been done, the uploaded file stands ready for selection in the “Editor’s Console” and can be used throughout the system.

4.3 Video Conversion

4.3.1 Requirements

The most important aim in terms of the video conversion is to select only as few formats as possible not to strain server resources too much. However,

⁵⁸ [FILE] `./code/www/editMeta.php`, SVN revision 558

⁵⁹ [FILE] `./code/www/editMeta.php`, SVN revision 560

these should still fit the requirements of a wide variety of video players and mobile devices (as mentioned in Chapter 3.5). The node administrator should nevertheless be able to modify transcoding settings and to add new formats to the configuration, too.

The conversion of video files into the selected formats will be placed where it has already been on pure audio nodes: It should be the first step of adding a programme to StreamOnTheFly.

Because of video transcoding being a time-consuming process when it comes to large files, it is very important that the procedure should not be aborted if a user preliminarily closes the browser window. Additionally, the current status of the process should continuously be displayed either in the front page of the Editor's Console or on the conversion page (editFiles.php).

4.3.2 Implementation

After an extended period of research and testing, the following formats have been selected to be added by default to StreamOnTheFly's configuration file:

	MPEG-4	3GP	WMV	FLV
Size	320x240 px	176x144 px	320x240 px	176x144 px
Video Codec	MPEG-4 ASP	H.263	WMV 2	FLV
Frame rate	25 fps	14.985 fps	25 fps	15 fps
Audio Codec	AAC	AMR	MP3	MP3
Sample Rate	22,050 Hz	8,000 Hz	22,050 Hz	22,050 Hz

Table 1 - Used transcoding settings

MPEG-4 ASP files are suitable for Quicktime Players as well as for iTunes and iPods while 3GP fits the need of mobile phones with integrated video player. (In this context, it is important to use AMR as an audio codec to avoid sound problems!) WMV has been added to the list of default formats because on many users' computers, there is still only Windows Media player available as media playback application. And finally, FLV encoding is necessary to provide

the source clip for the Video Extension's integrated preview player on a programme's detail page (see Chapter 4.5 for details).

To realize the conversion, the following FFmpeg parameters are used ⁶⁰:

-ab <i>bitrate</i>	Set the audio bitrate in kbit/s (default = 64).
-ac <i>channels</i>	Set the number of audio channels (default = 1).
-acodec <i>codec</i>	Force audio codec to <i>codec</i> .
-ar <i>freq</i>	Set the audio sampling frequency (default = 44100 Hz).
-b <i>bitrate</i>	Set the video bitrate in kbit/s (default = 200 kb/s).
-bt <i>tolerance</i>	Set video bitrate tolerance (in kbit/s).
-bufsize <i>size</i>	Set rate control buffer size (in kbit).
-f <i>fmt</i>	Force format.
-g <i>gop_size</i>	Set the group of pictures size.
-hq	Activate high quality settings.
-i <i>filename</i>	Set input filename.
-maxrate <i>bitrate</i>	Set max video bitrate tolerance (in kbit/s).
-qmax <i>q</i>	Set maximum video quantiser scale (VBR).
-qmin <i>q</i>	Set minimum video quantiser scale (VBR).
-r <i>fps</i>	Set frame rate (default = 25).
-s <i>size</i>	Set frame size.
-ss <i>position</i>	Seek to given time position in seconds. <i>hh:mm:ss[.xxx]</i> syntax is also supported.
-t <i>duration</i>	Set the recording time in seconds.
-vcodec <i>codec</i>	Force video codec to <i>codec</i> .
-y	Overwrite output files.

Table 2 - FFmpeg Conversion Parameters

⁶⁰ cp. [FFMPEG]

The conversion starts by a call of the `convert()` method belonging to the `sotf_VideoCheck` object which is used for identifying needed formats and converting them subsequently. This method contains the following line where a system command is composed:

```
$cmd = "nohup nice -n 15 ".$config['ffmpeg'].' -i '.$source.' ' .  
      $config['videoFormats'][$index]['ffmpeg_params'].' ' . $target .  
      " 1>".$target.".txt 2>&1 &"; 61
```

It will be executed afterwards by using PHP's `exec()` function and consists of the following elements:

- `nohup` for uncoupling the process from its parents in the process hierarchy. As the FFmpeg command is called via PHP, its parent process is the Apache web server. If Apache's process thread was closed because of a user closing his browser window, the process would be aborted. By using `nohup`, this unwanted behaviour is avoided.
- `nice -n 15` for associating a lower priority to the process. Thus, only a part of the available system CPU and memory resources are used which prevents the server from slowing down other tasks (such as Apache). Anyway, this is only a temporary solution. When it comes to production use of the system where a lot of conversions are performed at the same time a conversion queue should be considered to be implemented.
- `$config['ffmpeg']` as the system specific FFmpeg command defined in the configuration file (`config.inc.php`).
- `-i '.$source` for setting the source file.
- `$config['videoFormats'][$index]['ffmpeg_params']` contains all specific settings for the required file format identified by `$index` (see Table 2 for used FFmpeg parameters). To fit the requirements, these parameters can be edited by the node administrator in the `config.inc.php`

⁶¹ [FILE] `./code/classes/sotf_VideoCheck.class.php`, SVN revision 578

file. Thus, new formats might also be added to the system as long as FFmpeg is able to produce them.

- `$target` for setting the target file. It is temporarily stored in StreamOnTheFly's tmp directory. After successful conversion and file validation, this file will be added to the repository. Else it will be automatically deleted when the cron.php file, where such cleanup functions are integrated, is executed next. To facilitate fast identification of the temporary file, the name of the target file begins with the id of the currently converted programme.
- `1>".$target.".txt 2>&1 &` for saving FFmpeg's output to a temporary text file which possesses the same name as the target file, only supplemented by the extension ".txt". These files are used to calculate the current status of the conversion process which is then displayed in editFiles.php.

Therefore, the method `getPercentageOrError` of the `sotf_VideoCheck` object is required to be called on every reload of the conversion page (which is done automatically every five seconds via JavaScript). This method scans the produced FFmpeg output for the following information specified in `config.inc.php` as an "Advanced Video Options" section:

```
// Perl regexp for parsing ffmpeg output during transcode.
$config['ffmpegRegexp'] = "/frame=(.*?) q=/";

// Perl regexp for parsing ffmpeg errors occuring
// BEFORE CONVERSION EVEN STARTS.
$config['ffmpegErrorsBeforeConversion'] = array(
    "/\n\[.*@ 0x.*\n/",
    "/Unsupported codec/"
);

// Perl regexp for parsing ffmpeg errors occuring DURING THE CONVERSION
// (frame progress already displayed).
$config['ffmpegErrorsDuringConversion'] = array(
    "/Color spaces other than 420p/"
);
```

```
// String in ffmpeg output when CONVERSION IS FINISHED.
$config['ffmpegFinishMessage'] = "muxing overhead";

// String in ffmpeg output when CONVERSION IS FINISHED,
// BUT NO USABLE VIDEO HAS BEEN PRODUCED.
$config['ffmpegEmptyVideo'] = "video:0kB"; 62
```

By offering the modification of these search strings in the node configuration to administrators, it is warranted that differently compiled or future FFmpeg versions which may produce slightly different output are equally supported.

If transcoding is still running, the percentage of the progress is calculated by taking the current rendered frame and setting it in relation to the total sum of frames. These are represented as the product of the target frame rate and the duration of the source video. As this information is not always precisely read out by getID3 – especially when large files are processed – the percentage is displayed on the conversion page as an approximate value:

Files and links associated with programme: Spacerace
Edit metadata

Programme files

Filename	Format	Size	Last modified	Length	Download access
Converting ~ 76%	flash_preview.flv				
Converting ~ 52%	768kbps_2chn_22050Hz.wmv				
Converting ~ 46%	768kbps_2chn_22050Hz.mp4				
missing	176kbps_1chn_8000Hz.3gp				Convert from others
spacerace_7000kbps_0chn_0Hz.mpg 7000kbps_0chn_0Hz.mpg 90507302 2006-03-13 14:01:38+01 103 s <input checked="" type="checkbox"/> Delete					
Add other formats of content					

Associated files

Other files:

Filename	Caption	Size	Last modified	MIME type	Public	
still_881pr7_1.gif		2249	2006-05-29 15:45:23+02	image/gif	<input type="checkbox"/>	Change caption Delete
still_881pr7_2.gif		5201	2006-05-29 15:45:33+02	image/gif	<input type="checkbox"/>	Change caption Delete
still_881pr7_3.gif		5616	2006-05-29 15:45:38+02	image/gif	<input type="checkbox"/>	Change caption Delete
still_881pr7_4.gif		2765	2006-05-29 15:45:41+02	image/gif	<input type="checkbox"/>	Change caption Delete

Add files

Figure 14 - Video Conversion running in Editor's Console

If the conversion succeeds, the new file is copied to the programme directory it belongs to and database entries are updated. The check for such

⁶² [FILE] ./www/config.inc.php.template, SVN revision 581

newly transcoded files is done on each display of the conversion page (automatically reloaded via JavaScript or manually called by the editor). In the case the conversion page is no more displayed after the conversion has been completed, the check will be performed when the first user opens the programme's detail page (get.php). So, a station's editor does not have to wait until the transcoding has finished.

4.4 Video Metadata

4.4.1 Requirements

Video properties should be stored in the database to be available throughout the system without extensive access times. Such fields may include the codec, the bit rate and the resolution of a file.

In this context, it is important to make syncing with pure audio nodes equally possible. András Micsik pointed the problem out as follows:

*Imagine a network: $A \leftrightarrow B \leftrightarrow C$ where A and C are video-capable, and arrows represent neighborhood. Then how would node C ever get video data from node A ? There are some basic problems with this: you have to completely think over the peer-to-peer network in this mixed case.*⁶³

As part of a programme's metadata, the roles of associated contributors have to be adapted, as the list currently does not cover professions in the context of video production such as actor or camera operator.

4.4.2 Implementation

Technical information about a video file provided by getID3 is now inserted into the `sotf_media_files` table in the database.

Therefore, the following SQL statements were applied:

⁶³ Mail by András Micsik on May 2, 2006.

```

BEGIN;
ALTER TABLE sotf_media_files ADD COLUMN "codec" varchar(40) NULL;
ALTER TABLE sotf_media_files ADD COLUMN "frame_rate" float NULL;
ALTER TABLE sotf_media_files ADD COLUMN "lossless" boolean NULL;

ALTER TABLE sotf_media_files ADD COLUMN "resolution_x" float NULL;
ALTER TABLE sotf_media_files ADD COLUMN "resolution_y" float NULL;
ALTER TABLE sotf_media_files ADD COLUMN "pixel_aspect_ratio" float NULL;
[...]
ALTER TABLE sotf_media_files ADD COLUMN kbps_2 int;
UPDATE sotf_media_files SET kbps_2 = CAST(kbps AS int);
ALTER TABLE sotf_media_files DROP COLUMN kbps;
ALTER TABLE sotf_media_files RENAME COLUMN kbps_2 TO kbps;
COMMIT; 64

```

In the last five lines, it can be seen how it is possible to change the data type of fields in PostgreSQL. This has become necessary as the field for storing the data rate of a file in kB/s was on audio nodes designed for containing SMALLINT values which has turned out to be insufficient for some uncompressed video files.

A similar update script was produced and committed to SVN in order to integrate these fields into existing audio nodes as well. Thus, syncing of two video nodes over an audio node has become possible without any problems. The only thing left to do was to modify the source code of StreamOnTheFly audio nodes in a way that they only display audio programmes as they self-evidently do not dispose of specific video functionality. Mainly, it was sufficient to add the string "... WHERE type='audio' ..." to existing database queries.

Finally, in order to provide video-related professions when it comes to add contribution details to programmes, the following entries have been added in the database's role_names table:

⁶⁴ [FILE] ./code/share/update.sql

[...]

25 Actor

26 Animator

27 Art Director

28 Camera Operator

29 Director of Photography

30 Dubber

31 Graphic Designer

32 Lighting Technician

33 Make-Up Artist

34 Photographer

35 Post-Production Editor

36 Rigger

37 Set Designer

38 Set Dresser

39 Sound Designer

40 Sound Mixer

41 Special Effects

42 Stunts

43 Subtitles

44 Visual Editor

45 Visual Effects

46 Wardrobe ⁶⁵

All these professions are now also available for selection on the page where a programme's contributions are edited as part of the metadata (editMeta.php).

⁶⁵ [FILE] ./code/share/roles_eng.txt, SVN revision 562

4.5 Video Preview

4.5.1 Requirements

On a programme's detail page (get.php), the possibility to preview corresponding video content before downloading it should be offered. This may be implemented by embedding a MPEG1 encoded video into the page which then would be displayed in a user's browser by a plug-in. Another way would be to transcode video files into FLV format which can be read by a video player solution realized with Flash.

Additionally, preview stills should be generated and displayed to give a fast overview over the video without obliging the user to view it as a whole.

4.5.2 Implementation

First of all, it had to be decided whether to use MPEG1 or a Flash based solution for preview purposes. After some research and comparisons carried out concerning this subject, it turned out that MPEG1 provides slightly better quality than FLV at same bit rates. On the other hand, using a Flash player allows to adapt the appearance to fit the user interface of the node. Additionally, the required Flash plug-in is already available on most computers connected to the Internet, so no additional software would have to be installed on the user side. Prominent examples like Google Video or YouTube, which both use Flash players for web-based viewing of their video content, finally confirmed our choice to use a similar solution.

Searching for an already existing player fitting our needs, soon an exemplar with very basic but sufficient functionality was found, a solution created by Klaus Rechert.⁶⁶ An advantage of that Flash video player was that it could be generated dynamically using MING.

⁶⁶ RECHERT

MING is an open source library which is able to create flash files with almost all recent flash features, including Action Script, sound and video support. The library also has language bindings for a bunch of script and programming languages. ⁶⁷

In order to dynamically load requested video files into the player, a very practicable way has been found: A new variable was introduced in the player's ActionScript code containing the path of the FLV file which should be displayed. Additionally, Klaus Rechert's player was even more simplified for the use in StreamOnTheFly. Finally, the modified source code was recompiled to a ready-to-use SWF file using PHP's MING module.

Now, the accordant path could be given over to the player easily by using a GET-parameter in the Smarty template of the get.php page like this:

```
[...] videoplayer.swf?flvpath={ $FLV_PATH } [...]
```

⁶⁸

In the ActionScript part of the Flash file, the FLV file then was referred to as `"_root.flvpath"` and integrated as a NetStream object.

Unfortunately it has turned out that getID3 – even in version 1.7.5 – has not been able to efficiently analyze FLV files. There was even a serious performance problem when a preview page was opened in a browser, because getID3 always read out the whole FLV file line by line which was a very time-consuming process. The user had sometimes to wait for twenty seconds or even longer, depending on the duration of the displayed video.

Therefore, a config variable `$config['skipGetID3FileTypes']` has been introduced. This array contains any file extensions for which getID3 analyzing will be skipped. As FLV files are only used in the Flash player and are not offered for download, it is not necessary to show additional metadata such as duration or bit rate to the user. This being, getID3 can be skipped without problems.

⁶⁷ RECHERT

⁶⁸ [FILE] ./code/templates/get.htm, SVN revision 580


For rendering the preview images, once again FFmpeg was used as transcoding tool. To initiate their creation, a new button “Create Preview Stills” was added on the conversion page.

Now, a video programme’s detail page is shown as follows:




Programme:
Cars

[Metadata](#), [Content](#), [Statistics and feedback](#)

RSS

Preview


▶ ⏸ ⏹ 1:40 100% loaded


Metadata

Station: [sunny videos \(881\)](#)
Title: Cars
Language: German
Abstract:
Keywords: Flo Schödl, Computeranimation, FH-Projekt
Production date: 2006-05-14
Entry date: 2006-05-14
Last modification: 2006-05-15

Content

Video Files:









	Programme video	3gp	270 kbps	03:18	6675907 bytes	
	Programme video	mp4	415 kbps	03:17	10244376 bytes	
	Programme video	mpg	7000 kbps	04:05	214720550 bytes	
	Programme video	wmv	768 kbps	03:15	16171393 bytes	

Figure 15 - Programme Detail Page with Video Preview

4.6 New Advanced Search

4.6.1 Requirements

Above all, the usability of StreamOnTheFly's original "Advanced Search" has to be improved. As already mentioned in Chapter 2.2.3, the search interface is too complicated to use for users who are not familiar with SQL. For example, it is not necessary for the user to set the linking word AND or OR for each field.

Additionally, a new field "content type" has to be integrated to facilitate a precise search for audio or video files.

4.6.2 Implementation

Technically, a search request in StreamOnTheFly is performed according to a "SQLquerySerial" containing all required information like the following:

```
entry_date%20DESC/Bstation/  
AAND/Bentry_date/Bbigger/B1150322400/Bdate/  
AOR/Btitle/Bcontains/Bvideo/Bstring
```

At the beginning, the two sorting criteria are stated, followed by a chain of groups each containing five pieces of information:

1. "A" + the linking word placed before in the SQL query statement
2. "B" + field name
3. "B" + comparison expression
4. "B" + search term
5. "B" + data type

So, the example above represents a search for programmes entered into the system before June 15, 2006 (shown as UNIX timestamp) or the title of which contains the string "video". The result will first be sorted in

descending order by the programme's entry date and then by the station name.

On existing audio nodes, a new group as described above is added to the "SQLquerySerial" each time a new field is selected for the query (see Figure 3). The serial is written into a session variable, so when a user comes back to the search form, the last selected fields are still visible.

As a part of the StreamOnTheFly Video Extension, the Advanced Search has been revised and looks now like this:

Advanced search

Combine one or more criterias to perform an advanced search on programmes

<input checked="" type="checkbox"/> Title	contains	Vodcasting
<input type="checkbox"/> Abstract	contains	
<input type="checkbox"/> Keywords	contains	
<input checked="" type="checkbox"/> Station name	is	sunny videos (881)
<input type="checkbox"/> Series Title	contains	
<input type="checkbox"/> Series Description	contains	
<input type="checkbox"/> Language	is	English
<input type="checkbox"/> Genre	is	Documentary
<input type="checkbox"/> Topic	contains	
<input checked="" type="checkbox"/> Content Type	is	Video
<input type="checkbox"/> Length	is longer	Seconds
<input checked="" type="checkbox"/> Rating	bigger	3
<input type="checkbox"/> Person	contains	
<input type="checkbox"/> Production date	after	2006 08 11
<input type="checkbox"/> Broadcast date	after	2006 08 11
<input checked="" type="checkbox"/> Entry date	after	2006 05 31
<input type="checkbox"/> Modify date	after	2006 08 11
<input type="checkbox"/> Expiry date	after	2006 08 11

Sort results by

Entry date ☒ Desc

Second sort by

Station name ☐ Desc

Run query

Create new query

Figure 16 - Renewed Advanced Search Form

As it can be seen, all possible fields are now shown as once. The user has only to check the fields he wants to query for, enter the search criteria and then run the query. It might still not be the most logical solution, but already poses a significant improvement compared with the old “Advanced Search” page.

To realize the new layout without having to modify the code too much, the new linking word IGNORE has been introduced in addition to AND and OR.

Thus, a “SQLquerySerial” on a video node looks like this:

```
entry_date%20DESC/Bstation/
AAND/Btitle/Bcontains/BVodcasting/Bstring/
AIGNORE/Babstract/Bcontains/B/Bstring/
AIGNORE/Bkeywords/Bcontains/B/Bstring/
AAND/Bstation/Bis/Bsunny%20videos%20(881)/Bstation/
AIGNORE/Bseriestitle/Bcontains/B/Bstring/
AIGNORE/Bseriesdescription/Bcontains/B/Bstring/
AIGNORE/Blanguage/Bis/Beng/Blang/
AAND/Bgenre_id/Bis/B8/Bgenre/
AIGNORE/Btopic/Bcontains/B/Btopic/
AAND/Bcontenttype/Bis/Bvideo/Bcontenttype/
AIGNORE/Blength/Bbigger/B0/Blength/
AAND/Brating/Bbigger/B3/Brating/
AIGNORE/Bperson/Bcontains/B/Bstring/
AIGNORE/Bproduction_date/Bbigger/B1155247200/Bdate/
AIGNORE/Bbroadcast_date/Bbigger/B1155247200/Bdate/
AAND/Bentry_date/Bbigger/B1147298400/Bdate/
AIGNORE/Bmodify_date/Bbigger/B1155592800/Bdate/
AIGNORE/Bexpiry_date/Bbigger/B1155247200/Bdate
```

All terms beginning with AIGNORE are not regarded when building the SQL query. However, this solution finally leads to the fact that a search page always appears with all possible entry fields visible.

4.7 Vodcasting

4.7.1 Requirements

Equivalently to StreamOnTheFly's already existing podcasting functions vodcasting feeds has to be offered on result pages of the advanced search (advsearchresults.php) as well as on pages containing programmes of a given station (showStations.php) or the episodes of a certain series (showSeries.php).

4.7.2 Implementation

It turned out that the implementation of vodcasting into StreamOnTheFly was less complicated than originally thought. In fact, the most important thing having to be modified was the content of the enclosure tag in the case of a video file. As it has been pointed out in Chapter 3.4, this is the only difference between a podcast and a vodcast. The most significant changes in StreamOnTheFly's podcast.php file therefore have been the following:

```
if ($prg->isVideoPrg() && $f['format']=="mp4") {  
    $f['url'] = $baseUrl.'/getFile.php/'.$fid__.$f['id'].'__'.  
        $fname . ".mp4";  
    return $f;  
} else if ($prg->isAudioPrg()) {  
    $f['url'] = $baseUrl.'/getFile.php/'.$fid__.$f['id'].'__'.  
        $fname . ".mp3";  
    return $f;  
} 69
```

This was the required conditional clause for selecting the appropriate filename. If the video programme has not been transcoded into a MP4 file, no enclosure is available for that programme.

Apart from the file itself, its type has also to be adapted:

⁶⁹ [FILE] ./www/podcast.php, SVN revision 558

```
if($prog->isVideoPrg()) $type = 'video/mov';  
else $type = 'audio/mpeg';70
```

Much more time than on modifying the podcast routine had to be spent on generating video files equally suitable for Apple iTunes, the iPod Video and the current Quicktime Player. Finally, the following FFmpeg parameters lead to the best results (in alphabetical order):

```
-ab 96 -ac 2 -acodec aac -ar 22050 -b 672 -bufsize 4096 -g 300  
-maxrate 900 -qmax 5 -qmin 3 -s 320x240 -vcodec xvid
```

For a detailed explication of each parameter see Table 2 (FFmpeg Conversion Parameters).

⁷⁰ [FILE] ./www/podcast.php, SVN revision 558

5 Final Conclusion

On June 8, 2006, the preliminary beta version of the StreamOnTheFly Video Extension could be committed to SVN.

This having been an important step of the StreamOnTheFly project, the system is now able to import and transcode a wide variety of video formats along with all other features described in the previous chapter.

However, there are a few pieces of functionality having not been implemented so far which would be required for the system to fit the requirements of a multi-user production system.

In this context, the probably most important feature would be a transcoding queue which is able to guarantee fast answer times of the system even if multiple transcoding tasks have to be performed at the same time.

To provide better usability also for technically less experienced user groups, a thorough modernization of the whole user interface would have to be performed as well. Especially important would be the implementation of validation routines on pages containing forms. Additionally, the whole information architecture would need simplification.

Another known bug is the fact that getID3 is not able to provide accurate duration times for all video formats it supports. As it can be seen in Figure 15, the given times differ even when relatively short clips are analyzed. As getID3 also caused severe performance problems in connection with FLV videos, it might be assumed that there might exist a better, more elaborate solution for parsing media files.

Speaking of different video formats: One issue which is still to be solved is the incompatibility of the system with Sony's Playstation Portable, especially the fact that a programme's title is not displayed on this device. Maybe, a member of the developer team will one day find a solution for this problem.

These improvements having been realized, the system will need to be thoroughly evaluated using well defined testing scenarios. As such a clear

test period involving a representative selection of users has so far been omitted, there may exist yet unknown bugs throughout the system.

This phase should then be followed by a final iteration of development work to fix major bugs before the probably most important step of all – the transformation of the whole StreamOnTheFly system into a network of video capable nodes. As tests proved that the Video Extension's source code is fully compatible with the existing audio nodes, this upgrade should easily be performed.

"Why is the StreamOnTheFly Video Extension so important?" was the question having been posed at the very beginning. This diploma thesis might have answered it, but can never respond to the final one, being "What serves the newest technology if no one uses it?"

In this spirit, I hope that some developers will make the whole thing perfect so it can be made publicly available one day. May the system then facilitate many people to enjoy themselves sharing and viewing videos – either on their PCs at home or on the way to work by taking advantage of StreamOnTheFly's vodcasting features.

6 Appendix

6.1 Bibliography

6.1.1 Books

FELIX, Lionell / STOLARZ, Damien

"Hands-On Guide to Video Blogging and Podcasting"

Focal Press, April 2006

HAMMERSLEY, Ben

"Developing Feeds with RSS and Atom"

O'Reilly Media, April 2005

HERRINGTON, Jack D.

"Podcasting Hacks: Tips and Tools for blogging out loud."

O'Reilly Media, October 2005

SOMMERVILLE, Ian

"Software Engineering" (5th edition)

Addison-Wesley, October 1996

6.1.2 Conference Proceedings

ALTON-SCHEIDL, Roland / MICSIK, András / PATAKI, Máté / REUTZ, Wolfgang / SCHMIDT, Jürgen / THURNER, Thomas

"StreamOnTheFly: a Peer-to-peer Network for Radio Stations and Podcasters"

First International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'05)

Florence, 2005

ALUMBAUGH, Scott E.

"Providing Content on Demand: Step-by-Step Resource Guide"

Supplement to Conference Session: "The Future is Digital" at the
Alliance for Community Media, Annual Conference 2005
Monterey, 2005

GOLSTON, Jeremiah

"Comparing Media Codecs for Video Content"

Embedded Systems Conference 2004
San Francisco, 2004

SULLIVAN, Gary J. / TOPIWALA, Pankaj / Luthra, Ajay

"The H.264/AVC Advanced Video Coding Standard "

SPIE Conference on Applications of Digital Image Processing 2004
Denver, 2004

WINKLER, Stefan / FALLER, Christof

"Audiovisual Quality Evaluation of Low-Bitrate Video"

SPIE/IS&T Electronic Imaging, Annual Symposium 2005
San Jose, 2005

6.1.3 University Material

GOGA, Thanas

"Podcasting: The Emerging Business of Nanocasting"

Handout Course TCOM 563 - "New Technology"
Ohio University, International Development Studies
Ohio, 2005

MENG, Peter

"Podcasting & Vodcasting - A White Paper: Definitions, Discussions & Implications"

University of Missouri, IAT Services
Columbia, 2005

ZODL, Alexander

"Open Source Medientechniksoftware in der Anwendung – Adobe Photoshop vs. The GIMP"

Diploma Thesis at the University of Applied Sciences in St. Pölten

St. Pölten, 2004

6.1.4 Articles

BOLD, Max

"Auf Sendung"

Internet Professionell, July 2006, p 34-36

Munich, 2006

CAMPBELL, Gardner

"There's Something in the Air: Podcasting in Education"

EDUCAUSE Review, November/December 2005, p 32-46

Boulder, 2005

GORDON, V. Scott / BIEMAN, James M.

"Reported effects of Rapid Prototyping on Industrial Software Quality"

Software Quality Journal, June 1993, p 93-108

Dordrecht, 1993

KOENEN, Rob

"MPEG-4 – Multimedia for our time"

IEEE Spectrum, February 1999, p 26-33

New York, 1999

RANADA, David:

"Facing the codec challenge"

Sound & Vision, July 2002, p 98-100

New York, 2002

RENSMANN, Jörg:

"Content à la carte"

Internet Professionell, July 2006, p 24-33

Munich, 2006

WAGENKNECHT, Achim

"Film ab!"

Internet Professionell, July 2006, p 76-80

Munich, 2006

6.1.5 Internet Ressources

[APPLE]

"Apple – iPod – Technical Specifications"

<http://www.apple.com/ipod/specs.html>

July 26, 2005

[AUDACITY]

"Audacity: Free Audio Editor and Recorder"

<http://audacity.sourceforge.net/?lang=en>

July 26, 2005

[BERLIOS]

"BerliOS Developer: Project Info - StreamOnTheFly"

<http://developer.berlios.de/projects/sotf>

July 26, 2005

BREEN, Christopher

"How to create a Vodcast: Steps for offering video on demand"

<http://playlistmag.com/features/2005/07/howtoVodcast/index.php>

July 26, 2005

[BUNDESKANZLERIN]

"Bundestkanzlerin – VIDEO-PODCAST"

<http://www.bundestkanzlerin.de/Webs/BK/DE/Aktuelles/VideoPodcast/video-podcast.html>

July 26, 2005

CURRY, Adam

"Usernum 1014" (Weblog Entry of October 21, 2002)

<http://radio.weblogs.com/0001014/2002/10/21.html>

June 5, 2006

CURRY, Adam

"Adam Curry's Weblog" (Weblog Entry of October 12, 2003)

<http://radio.weblogs.com/0001014/2003/10/12.html>

June 5, 2006

EVA (pseudonym)

"Google Video's New Year" (Google Video Blog Entry of January 9, 2006)

<http://googlevideo.blogspot.com/2006/01/google-videos-new-year.html>

June 5, 2006

[FEEDDEMON]

"Bradbury Software – Creators of TopStyle and FeedDemon"

<http://www.feeddemon.com>

July 26, 2005

[FIREANT]

"FireAnt.tv"

<http://www.fireant.tv>

July 26, 2005

[FFMPEG]

"FFmpeg Documentation"

<http://ffmpeg.mplayerhq.hu/ffmpeg-doc.html>

July 26, 2005

[FLASH]

"Adobe – Developer Center: Flash Video Learning Guide"

http://www.adobe.com/devnet/flash/articles/video_guide.html

July 26, 2005

GARFIELD, Steve

"Steve Garfield's VideoPodcast."

http://stevegarfield.blogs.com/videopodcast/2004/11/videopodcast_20.html

June 5, 2006

GERVAIS, Ricky

"Ricky Gervais 'surfs' into the Guinness World Records book."

<http://www.rickygervais.com/guinnessrecord.php>

June 5, 2006

[GETID3]

"getID3() – The PHP media file parser"

<http://www.getid3.org>

July 26, 2005

[GNU]

"The Free Software Definition"

<http://www.gnu.org/philosophy/free-sw.html>

June 5, 2006

GREEN, Heather

"Is the Web the new Hollywood?"

http://businessweek.com/technology/content/jan2006/tc20060123_227536.htm

June 5, 2006

GREGOIRE, Dannie J.

"How to handle getting past episodes?"

<http://groups.yahoo.com/group/ipodder-dev/message/41>

June 5, 2006

HAMMERSLEY, Ben

"Audible Revolution" (in: "The Guardian Technology Blog", February 12, 2004)

<http://technology.guardian.co.uk/online/story/0,3605,1145689,00.html>

June 12, 2005

HAMMOCK, Rex

"Rex Hammock's Weblog" (Weblog Entry of October 12, 2003)

<http://www.rexblog.com/2005/02/20#a5901>

June 12, 2005

[JAHSHAKA]

"jahshaka.org – Powering the New Hollywood"

<http://www.jahshaka.org>

July 26, 2005

LOUIS, Tristan

"Some suggestions for RSS .92"

<http://groups.yahoo.com/group/syndication/message/698>

May 15, 2006

MARKS, Kevin

"Epeus' epigone – Kevin Marks Weblog" (Weblog Entry of October 01, 2003)

http://epeus.blogspot.com/2003_10_01_epeus_archive.html

June 5, 2006

[MOD_SQL]

"ProFTPD module mod_sql"

http://www.castaglia.org/proftpd/modules/mod_sql.html

July 26, 2005

[NOKIA]

"Nokia – Phone Features – Nokia 6230i"

<http://www.nokia.co.uk/nokia/0,8764,72154,00.html>

July 26, 2005

PERENS, Bruce

"The Open Source Definition"

<http://www.opensource.org/docs/osd.pdf>

May 19, 2006

[PODCAST_NET]

"Podcast.net – The Podcast Directory"

<http://www.podcast.net/show/29642>

July 26, 2005

[PODSPIDER]

"PodSpider – Podcast Client und Podcast Verzeichnis"

<http://www.podspider.com>

July 26, 2005

RECHERT, Klaus

"Video and Audio Streaming with Flash and Open Source Tools"

<http://klaus.geekserver.net/flash/streaming.html>

February 9, 2006

[RFC]

"RFC2822"

<http://rfc.net/rfc2822.html#s3.3>

July 26, 2005

SCHMIDT, Jürgen / DOSSER, Michael

"StreamOnTheFly - Technical Documentation"

http://sotf.berlios.de/documents/sotf_tech_doc_1.pdf

May 11, 2006

[SONY_PRESS]

"PlayStation.com – News – Press Releases"

<http://www.us.playstation.com/News/PressReleases/268>

July 26, 2005

[SONY_PSP]

"PlayStation.com – PlayStation Portable – The PSP"

<http://www.us.playstation.com/PSP/System>

July 26, 2005

[SOTF_ORG]

"Running Nodes – StreamOnTheFly.org"

http://streamonthe-fly.org/about/running-nodes?set_language=en

July 26, 2005

[TAGESSCHAU]

"Video-Podcast / tagesschau.de"

<http://www.tagesschau.de/export/video-podcast>

July 26, 2005

[THUNDERSTONEMEDIA]

"Thunderstone Media"

http://thunderstonemedia.com/name_change

July 26, 2005

[VODCAST_NL]

"VODcast.nl – Pioneering Vodcasting. Subscribe, Choose, Play."

<http://www.vodcast.nl>

July 26, 2005

[WIKIPEDIA]

"History of podcasting"

http://en.wikipedia.org/wiki/History_of_podcasting

May 11, 2006

WINER, Dave

"Scripting News" (Weblog Entry of January 11, 2001)

<http://www.scripting.com/2001/01/11.html>

May 11, 2006

[WORDPRESS]

"WordPress > Free Blog Tool and Weblog Platform"

<http://www.wordpress.org>

July 26, 2005

[YOUTUBE]

"YouTube – Broadcast Yourself."

<http://www.youtube.com>

July 26, 2005

6.2 List of Figures

Figure 1 - StreamOnTheFly's Home Page	12
Figure 2 - Schema of the StreamOnTheFly Node Network	14
Figure 3 - The Original "Advanced Search" Screen	15
Figure 4 - Extract of the "Topic Tree" Screen	16
Figure 5 - "Editor's Console" Screenshot	17
Figure 6 - editFiles.php on an Audio Node	18
Figure 7 - Receiving Podcasts in iTunes	30
Figure 8 - Podcasting/Vodcasting Workflow	32
Figure 9 - iPod Video	35
Figure 10 - Nokia 6230i	36
Figure 11 - T-Mobile MDA Compact	37
Figure 12 - Sony PlayStation Portable	37
Figure 13 - Evolutionary Prototyping	43
Figure 14 - Video Conversion running in Editor's Console.....	50
Figure 15 - Programme Detail Page with Video Preview	56
Figure 16 - Renewed Advanced Search Form	58

6.3 Extracts from the Source Code

This section contains parts of StreamOnTheFly's current source code (SVN branch "video_test", revision 581). It shall provide a deeper insight into the modifications done throughout the system during the development of the StreamOnTheFly Video Extension.

The code extracts belong to these following six important files:

- config.inc.php (the StreamOnTheFly configuration file)
- functions.inc.php (useful functions for the whole project)
- sotf_Programme.class.php (where programme objects are managed)
- sotf_VideoFile.class.php (class for handling video files)
- sotf_VideoCheck.class.php (video conversion and verification functions)
- editFiles.php (containing the logic for the transcoding page)

However, the files are not completely shown as this would go beyond the scope of this diploma thesis. Left out passages are substituted by [...].

6.3.1 config.inc.php

<?php

[...]

```
/* Formats for which to skip getID3()
 *
 * Array containing the extensions of files for which NOT to run getID3 on.
 * This must include 'flv' as long as getID3 does not provide a more performant
 * solution for checking flash videos.
 * Notice that no details (as duration, bitrate, etc.) of the converted file
 * will be available to display!
 */
$config['skipGetID3FileTypes'] = array('flv');
```

```
////////////////////////////////////
// VIDEO OPTIONS -----
////////////////////////////////////

//Path to FFmpeg directory
$config['ffmpeg'] = $config['helperdir']. "/ffmpeg";

//Required video formats. (FLV format is necessary for preview on get.php!)
$config['videoFormats'] = array(
    array(
        'format' => 'flv',
        'size' => '176x144',
        'framerate' => '15',
        'video_bitrate' => '336',
        'audio_bitrate' => '64',
        'audio_channels' => '2',
        'audio_samplerate' => '22050',
        'ffmpeg_params' => '-f flv -vcodec flv -s qcif -r 15 -b 336 -bt 16 -ac 2
                           -ar 22050 -ab 64'
    ),
    array(
        'format' => 'wmv',
        'size' => '320x240',
        'framerate' => '25',
        'video_bitrate' => '672',
        'audio_bitrate' => '96',
        'audio_channels' => '2',
        'audio_samplerate' => '22050',
        'ffmpeg_params' => '-vcodec wmv2 -s 320x240 -r 25 -b 672 -acodec MP3
                           -ac 2 -ar 22050 -ab 96'
    ),
    array(
        'format' => 'mp4',
        'size' => '320x240',
        'framerate' => '25',
        'video_bitrate' => '672',
        'audio_bitrate' => '96',
        'audio_channels' => '2',
        'audio_samplerate' => '22050',
        'ffmpeg_params' => '-b 672 -maxrate 900 -vcodec xvid -qmin 3 -qmax 5
                           -bufsize 4096 -g 300 -s 320x240 -acodec aac -ab 96
                           -ac 2 -ar 22050'
    ),
)
```

```

    array(
        'format' => '3gp',
        'size' => '176x144',
        'framerate' => '14.985',
        'video_bitrate' => '336',
        'audio_bitrate' => '64',
        'audio_channels' => '2',
        'audio_samplerate' => '22050',
        'ffmpeg_params' => '-f 3gp -vcodec mpeg4 -s 176x144 -r 14.985 -b 336
                            -acodec aac -ac 2 -ar 22050 -ab 64'
    )
);

//FFmpeg parameters used for preview still creation
$config['ffmpeg_params_stills'] = '-f image2 -img gif -t 1 -r 1 -s sqcif';
[...]

////////////////////////////////////
// ADVANCED VIDEO OPTIONS -----
////////////////////////////////////

//Perl regexp for parsing ffmpeg output during transcode.
$config['ffmpegRegexp'] = "/frame=(.*?) q="/;

//Perl regexp for parsing ffmpeg errors occuring BEFORE CONVERSION EVEN STARTS.
$config['ffmpegErrorsBeforeConversion'] = array(
    "/\n\[.*@ 0x.*\n/",
    "/Unsupported codec/"
);

//Perl regexp for parsing ffmpeg errors occuring DURING THE CONVERSION
//(frame progress already displayed).
$config['ffmpegErrorsDuringConversion'] = array(
    "/Color spaces other than 420p/"
);

// String in ffmpeg output when CONVERSION IS FINISHED.
$config['ffmpegFinishMessage'] = "muxing overhead";

// String in ffmpeg output when CONVERSION IS FINISHED,
// BUT NO USABLE VIDEO HAS BEEN PRODUCED.
$config['ffmpegEmptyVideo'] = "video:0kB";

?>

```

6.3.2 functions.inc.php

```
<?php
[...]
function isUnicolorImage($file) {
    global $config;
    $isOneColor = false;

    // getting file information with ImageMagick
    $cmd = $config['magickDir'] . "/identify -verbose $file 2>&1 &";
    debug("check unicolor command", $cmd);
    $fp = popen($cmd, 'r');
    while(!feof($fp)) {
        $output = fread($fp, 2096);
        if (strpos($output, 'Colors: 1 ') $isOneColor = true;
    }
    pclose($fp);
    return $isOneColor;
}
[...]
```

6.3.3 sotf_Programme.class.php

```
<?php
[...]
```

```
function setAudio($filename, $copy=false) {
    global $page;

    $source = $filename;
    if(!is_file($source)) raiseError("no such file: $source");

    $srcFile = new sotf_AudioFile($source);

    if($srcFile->isVideo()) $srcFile = new sotf_VideoFile($source,
        $srcFile->allInfo);

    $target = $this->getAudioDir() . '/' . $this->get('track') . '_' .
        $srcFile->getFormatFilename();
    if(!$srcFile->isAudio() && !$srcFile->isVideo()) {
        raiseError("$source is neither an audio nor a video file");
    }
    [...]
```

```
    if($copy) $success = copy($source,$target);
```

```
else $success = rename($source,$target);
if(!$success) raiseError("could not copy/move $source");

// save file info into database
$this->saveFileInfo($target, true);
}
```

[...]

```
function saveFileInfo($filepath, $mainContent = false) {
    global $db;
    [...]
    // get properties of file
    $file = new sotf_AudioFile($filepath);

    // create videofile object if getid3 return video info
    if($file->isVideo()) $file = new sotf_VideoFile($filepath);

    // find if this is an existing file
    if($file->isAudio() || $file->isVideo()) { //CHANGED BY BUDDHAFLY 06-02-20
        $fileInfo = new sotf_NodeObject('sotf_media_files');
    } else {
        $fileInfo = new sotf_NodeObject('sotf_other_files');
    }
    [...]
    // save file info into database
    if($file->isVideo()) {
        $fileInfo->set('play_length', round($file->duration));
        $fileInfo->set('type', $file->type);
        if(is_numeric($file->bitrate)) {
            // constant bitrate
            $fileInfo->set('kbps', round($file->bitrate));
            $fileInfo->set('vbr', 'f');
        } else {
            // variable bitrate
            $fileInfo->set('kbps', round($file->average_bitrate));
            $fileInfo->set('vbr', 't');
        }
        $fileInfo->set('format', $file->getFormatFilename());
        $fileInfo->set('main_content', $mainContent);
        $fileInfo->set('codec', $file->codec);
        $fileInfo->set('frame_rate', $file->frame_rate);
        $fileInfo->set('resolution_x', $file->resolution_x);
    }
}
```

```

        $fileInfo->set('resolution_y', $file->resolution_y);
        $fileInfo->set('pixel_aspect_ratio', $file->pixel_aspect_ratio);

        //manage access rights for video
        $fileInfo->set('download_access', 't');
        $fileInfo->set('stream_access', 'f');
    }

    else if($file->isAudio()) {
        $fileInfo->set('play_length', round($file->duration));
        $fileInfo->set('type', $file->type);
        if(is_numeric($file->bitrate)) {
            // constant bitrate
            $fileInfo->set('kbps', round($file->bitrate));
            $fileInfo->set('vbr', 'f');
        } else {
            // variable bitrate
            $fileInfo->set('kbps', round($file->average_bitrate));
            $fileInfo->set('vbr', 't');
        }
        $fileInfo->set('format', $file->getFormatFilename());
        $fileInfo->set('main_content', $mainContent);
    }

    $fstat = stat($filepath);
    $fileInfo->set('filesize', $fstat['size']);
    $fileInfo->set('last_modified', $db->getTimestampTz($fstat['mtime']));
    $fileInfo->set('mime_type', $file->mimetype);
    $success = $fileInfo->save();

    // change id3 tags in file
    $this->saveID3($file);

    return $fileInfo->id;
}
[...]
```

?>

6.3.4 sotf_VideoFile.class.php

```

<?php
[...]
```

```

function sotf_VideoFile($path, $fileinfo='') {
    global $config;
```

```
$parent = get_parent_class($this);
// call the constructor of the parent class. lk. super()
parent::$parent($path);

if(!in_array(sotf_File::getExtension($path),$config['skipGetID3FileTypes']))
{
    // get fileinfo with getid3
    $getID3 = new getID3();
    $fileinfo = $getID3->analyze($path);
    getid3_lib::CopyTagsToComments($fileinfo);
}
else $fileinfo['video']=true;

if($fileinfo) $this->allInfo = $fileinfo;

if(!in_array(sotf_File::getExtension($path),$config['skipGetID3FileTypes']))
{
    $videoinfo = $fileinfo['video'];
    $this->type = "video";

    $this->format = $videoinfo["dataformat"];
    if($this->format == "quicktime") $this->format = "mov";
    if($fileinfo['quicktime']['ftyp']['signature']=="3gp4") {
        $this->format = "3gp";
    }
    else if($this->format == "mpeg4") {
        $this->format = "mp4";
    }
    if($this->format == "asf") $this->format = "wmv";
    if($this->format == "mpeg") $this->format = "mpg";

    if ($videoinfo["bitrate_mode"] == 'vbr') $this->bitrate = "VBR";
    $this->bitrate = round($fileinfo["bitrate"]/1000);
    $this->average_bitrate = round($fileinfo["bitrate"]/1000);
    $this->duration = round($fileinfo["playtime_seconds"]);
    $this->mimetype = $this->determineMimeType($this->format);
    $this->codec = $videoinfo["codec"];
    $this->frame_rate = round($videoinfo["frame_rate"]);
    $this->resolution_x = $videoinfo["resolution_x"];
    $this->resolution_y = $videoinfo["resolution_y"];
    $this->lossless = $videoinfo["lossless"];
    $this->pixel_aspect_ratio = $videoinfo["pixel_aspect_ratio"];

    if(isset($fileinfo['audio'])) {
        $audioinfo = $fileinfo['audio'];
```



```

        if($audioinfo["sample_rate"])$this->samplerate =
        $audioinfo["sample_rate"];
        else $this->samplerate = 0;
        if($audioinfo["channels"]) $this->channels = $audioinfo["channels"];
        else $this->channels = 0;
    } else {
        $this->samplerate = 0;
        $this->channels = 0;
    }
}

else if (isset($fileinfo['video'])) {
    $this->type = "video";
    $this->format = sotf_File::getExtension($path);
    $this->mimetype = $config['mimetypes']['format'];
    $this->bitrate = 0;
    $this->samplerate = 0;
    $this->channels = 0;
}
}

function createStills($file, $length, $id) {
    global $config;
    $temppath=$config['wwwdir']. "/tmp";

    for($i=1;$i<=5;$i++) {
        $position = round(((($i+($i-1))/10)*$length);
        $target = $temppath."/still_". $id. "_". $i. ".gif";
        $cmd = "nohup nice -n 15 ".$config['ffmpeg']. "
            -i $file ".$config['ffmpeg_params_stills']. " -ss $position
            -y $target 1>$target.txt 2>&1 &";
        exec($cmd);
    }
}

function searchForStill($prg) {
    global $config;
    $id=$prg->get('id');
    $temppath=$config['wwwdir']. "/tmp/";
    $still_found=false;

    if ($directory = opendir($prg->getOtherFilesDir())) {

```

```

        while (false !== ($filename = readdir($directory))) {
            if(preg_match("/^still_". $id. "_[12345]\\.gif$/", $filename)) {
                $still_found=true;
            }
        }
        closedir($directory);
    }
    if ($tempdir = opendir($temppath)) {
        while (false !== ($filename = readdir($tempdir))) {
            if(preg_match("/^still_". $id. "_[12345]\\.gif$/", $filename)) {
                if(!isUnicolorImage($temppath.$filename)) {
                    $still_found=true;
                }
            }
        }
        closedir($tempdir);
    }
    return $still_found;
}

function processTranscodingQueue($repository, $prg) {
    global $config;
    $id=$prg->get('id');
    $temppath=$config['wwwdir']. "/tmp/";
    $list_changed = false;

    if ($tempdir = opendir($temppath)) {
        while (false !== ($filename = readdir($tempdir))) {
            if(preg_match("/^". $id. "_/", $filename)) {
                if(sotf_VideoCheck::fileOK($temppath.$filename)) {
                    if(is_file($temppath.$filename.".txt"))
                        unlink($temppath.$filename.".txt");
                    $prg->setAudio($temppath.$filename);
                    $list_changed=true;
                }
            }
        }
        if(preg_match("/^still_". $id. "_[12345]\\.gif$/", $filename)) {
            if(!isUnicolorImage($temppath.$filename)) {
                $obj_id=$prg->setOtherFile($temppath.$filename);
                $fileInfo = &$repository->getObject($obj_id);
                $fileInfo->set('public_access', 'f');
                $fileInfo->update();
                if(is_file($temppath.$filename.".txt"))

```

```
        unlink($temppath.$filename.".txt");
    } else {
        if(is_file($temppath.$filename.".txt")) {
            unlink($temppath.$filename.".txt");
            unlink($temppath.$filename);
        }
        $list_changed=true;
    }
}
closedir($tempdir);
}

return $list_changed;
}
[...]
```

```
?>
```

6.3.5 sotf_VideoCheck.class.php

```
<?php
```

```
[...]
```

```
function fileOK($file) {
    global $config;

    if(!in_array(sotf_File::getExtension($file),$config['skipGetID3FileTypes']))
    {
        $getID3 = new getID3();
        $fileinfo = $getID3->analyze($file);
        getid3_lib::CopyTagsToComments($fileinfo);
    }
    else $fileinfo['video']=true;

    if(is_file($file.".txt")) {
        $handle = fopen ($file.".txt", "r");
        $buffer="";

        while (!feof($handle)) {
            $buffer .= fgets($handle, 4096);
        }
        fclose ($handle);
        $finished = strstr($buffer,$config['ffmpegFinishMessage']) &&
            !strstr($buffer, $config['ffmpegEmptyVideo']);
    }
    else $finished = false;
}
```

```

if(!is_readable($file) || filesize($file)==0 ||
(!isset($fileinfo['audio'])&&!isset($fileinfo['video'])) || !$finished) {
    return false;
} else {
    return true;
}
}

function getTotalFrames($source, $index) {

    global $config;
    if(!in_array(sotf_File::getExtension($source),$config['skipGetID3FileTypes']
)) {
        $getID3 = new getID3();
        $fileinfo = $getID3->analyze($source);
        getid3_lib::CopyTagsToComments($fileinfo);
        $totalframes=round($fileinfo["playtime_seconds"]*$config['videoFormats']
            [$index]['framerate']);
        return $totalframes;
    }
    else return -1;
}

function getPercentageOrError($tempfile, $totalframes) {

    global $config;

    if(is_file($tempfile.".txt")) {
        $handle = fopen ($tempfile.".txt", "r");
        $buffer="";
        while (!feof($handle)) {
            $buffer .= fgets($handle, 4096);
        }
        fclose ($handle);

        $returnarray=array();

        preg_match_all($config['ffmpegRegexp'], $buffer, $results);
        $curframe=$results[1][count($results[1])-1];
        $timediff= time()-filemtime($tempfile);

        $errors_before=false;

```

```

$errors_during=false;

for($i=0;$i<count($config['ffmpegErrorsBeforeConversion']);$i++) {
    if(preg_match_all($config['ffmpegErrorsBeforeConversion'][$i],
        $buffer, $errors_1)) {
        $errors_before=true;
    }
}
    for($j=0;$j<count($config['ffmpegErrorsDuringConversion']);$j++) {
        if(preg_match_all($config['ffmpegErrorsDuringConversion'][$j],
            $buffer, $errors_2)) {
            $errors_during=true;
        }
    }
}

if (is_file($tempfile) && ((empty($results[1]) && $errors_before) ||
    $errors_during) && $timediff>3) {
    $errors=array_merge($errors_1, $errors_2);
    $returnarray['errors']=$errors;
    logError('conversion failed: '.$tempfile);
    logError('ffmpeg output: '. $buffer);
}
$percentage='';
if($totalframes!=-1) @$percentage=round($curframe/$totalframes*100);
$returnarray['percentage']=$percentage;
return $returnarray;
}
}

```

[...]

```

function convert($id, $index) {
    global $config, $page;

    debug('conversion started', $this->getFormatFilename($index));

    if ($this->reqs[$index][0] === true) {
        debug("We have this format already", $index);
        return;
    }

    $sourceindex = $this->reqs[$index][1];
    $audioFiles = & $this->list;
    $source = $audioFiles->list[$sourceindex]->getPath();
    debug("source", $source);
}

```

```
$target = $config['tmpDir'] . '/' . $id . '_' . time() . '_' .  
    $this->getFormatFilename($index);  
  
[...]  
$cmd = "nohup nice -n 15 ".$config['ffmpeg'].' -i '.$source.'  
    '.$config['videoFormats'][$index]['ffmpeg_params'].' '.$target."  
    1>".$target.".txt 2>&1 &";  
$this->transcodeWithFfmpeg($cmd, $totalframes); // == exec($cmd)  
  
[...]  
debug('conversion finished', $target);  
  
    return $target;  
}  
[...]  
?>
```

6.3.6 editFiles.php

```
<?php  
[...]  
require("init.inc.php");  
  
$id = sotf_Utils::getParameter('id');  
$new = sotf_Utils::getParameter('new');  
$okURL = sotf_Utils::getParameter('okURL');  
$videoconv = sotf_Utils::getParameter('videoconversion');  
$convertall = sotf_Utils::getParameter('convertall');  
$convertindex = sotf_Utils::getParameter('convertindex');  
$createstills = sotf_Utils::getParameter('createstills');  
[...]  
$prg = & new sotf_Programme($id);  
  
if(!$prg->isLocal()) {  
    raiseError("You can only edit programmes locally!");  
}  
  
if($prg->isVideoPrg()) $video=true;  
else $video = false;  
$converting=false;  
  
// admins or owners can change files  
checkPerm($id, 'change');
```

```
// delete link
$delLink = sotf_Utils::getParameter('dellink');
$linkid = sotf_Utils::getParameter('linkid');
if($delLink) {
    $link = new sotf_NodeObject("sotf_links", $linkid);
    $link->delete();
    $page->redirect("editFiles.php?id=$id#links");
    exit;
}

// delete file
$delFile = sotf_Utils::getParameter('delfile');
if($delFile) {
    $prg->deleteFile($delFile);
    $page->redirect("editFiles.php?id=$id#mfiles");
    exit;
}

[...]

// audio files for programme
$audioFiles = $prg->listAudioFiles('true');

for ($i=0;$i<count($audioFiles);$i++) {
    $mainAudio[$audioFiles[$i]['filename']] = $audioFiles[$i];
}

$prgAudiolist = & new sotf_FileList();
$prgAudiolist->getAudioVideoFromDir($prg->getAudioDir());

// check SQL validity
if($prgAudiolist->count() != count($mainAudio)) {
    $page->addStatusMsg("main_audio_count_mismatch");
}

$files = $prgAudiolist->getFiles();
debug('mainAudio', $mainAudio);
debug('prgaudiolist', $files);
for ($i=0;$i<count($files);$i++) {
    if(!$mainAudio[$files[$i]->name]) {
        // missing from SQL!
        $missing = 1;
        $prg->saveFileInfo($files[$i]->path, true);
        $msg = $page->getLocalizedWithParams("missing_from_sql",
            $files[$i]->name);
        $page->addStatusMsg($msg, false);
    }
}
```

```
    }
}

if($video) {
    $still_found=sotf_VideoFile::searchForStill($prg);
    if (!$still_found && $createstills) {
        sotf_VideoFile::createStills($files[0]->path, $files[0]->duration, $id);
    }
}

if($missing) {
    // there was a missing file description, so we must restart the whole thing
    $page->redirectSelf();
    exit;
}

$checker = & new sotf_ContentCheck($prgAudiolist);
$checker = $checker->selectType();

//check for recently converted files or transcoding in progress
if($video && $prgAudiolist->count()) {
    $list_changed=sotf_VideoFile::processTranscodingQueue($repository, $prg,
$checker);
    if($list_changed) {
        $page->redirectSelf();
        exit;
    }
}

// compare with required formats
$PRG_AUDIO = array();

// $config[$checker->prefix.'Formats'] is either "audio" or "video"
for ($i=0;$i<count($config[$checker->prefix.'Formats']);$i++) {
    $PRG_AUDIO[$i] = array(
        "format" => $checker->getFormatFileName($i),
        "index" => $i,
        "flv" => preg_match("/flv/", $checker->getFormatFileName($i))
    );
    if ($checker->reqs[$i][0]) {
        $fname = $prgAudiolist->list[$checker->reqs[$i][1]]->name;
        $PRG_AUDIO[$i] = array_merge($PRG_AUDIO[$i], $mainAudio[$fname]);
        unset($mainAudio[$fname]);
    } else {
        if($video) {
```



```

// if conversion in progress calculate percentage
$regexp_file="/^".$$id . '_.*_'. $checker->getFormatFilename($i)."$$/";
$source = $prgAudiolist->list[$checker->regs[$i][1]]->getPath();
$temppath = $config['wwmdir']. "/tmp/";

if ($tempdir = opendir($temppath)) {
    while (false != ($filename = readdir($tempdir))) {
        if(preg_match($regexp_file, $filename)) {
            $PRG_AUDIO[$i]['converting']=true;
            $totalframes = $checker->getTotalFrames($source, $i);
            $perc_error = $checker->
                getPercentageOrError($temppath.$filename,
                    $totalframes);
            $PRG_AUDIO[$i]['errors']=$perc_error['errors'];
            if($perc_error['percentage']) {
                if($perc_error['percentage']>100) {
                    $perc_error['percentage']=100;
                }
                $PRG_AUDIO[$i]['percentage'] = "~ ".
                    $perc_error['percentage'].
                    "%";
            }
            if(!empty($perc_error['errors'])) {
                $PRG_AUDIO[$i]['converting'] = false;
            }
        }
    }
    closedir($tempdir);
}

$PRG_AUDIO[$i]['missing'] = 1;
$missing = 1;
}
}

//check whether a file is converting
$converting=false;
for ($i=0;$i<count($config[$checker->prefix.'Formats']);$i++) {
    if($PRG_AUDIO[$i]['converting']==true) $converting=true;
}

[...]

// start converting required video formats
if($videoconv && $missing) {

```

```
$obj = $repository->getObject($id);
if(!$obj) throwError("object does not exist!");

checkPerm($obj->id, 'change');

$checker->console = false;

if($convertall) {
    $checker->convertAll($obj->id);
}
elseif($convertindex!="") {
    $checker->convert($obj->id, $convertindex);
}

$page->redirect("editFiles.php?id=$id");
exit;
}
[...]
```

?>