



Technisch-Naturwissenschaftliche  
Fakultät

# **Network Media Content Aggregator for DLNA Server**

## **Master's Thesis**

to obtain the academic degree

## **Master of Science**

in the master's study programmes

## **Internationaler Universitätslehrgang Informatics: Engineering & Management<sup>\*</sup> and Open Informatics<sup>†</sup>**

Submitted by:

**Ing. Jan Kubový**

(Martikel Nr.: 1057096)

Institute:

**Institute for Application Oriented Knowledge Processing (FAW)**

Advisors:

**1<sup>st</sup> Advisor: A.Univ.-Prof. Dr. Josef Küng**

**2<sup>nd</sup> Advisor: Doc. Ing. Zdeněk Kouba CSc.**

**3<sup>rd</sup> Advisor: DI Franz Wieshofer**

**Linz, July, 2011**

---

<sup>\*</sup>Johannes Kepler University in Linz

<sup>†</sup>Czech Technical University in Prague

Date: 2011-07-13T08:42 Revision: 87457296d1a4faec6e27394a9f230e1fb5abc981



---

This research project was funded and supported by the **International Studies in Informatics Hagenberg**<sup>\*</sup> and the **Quanmax AG**<sup>†</sup> company.

Living expenses were partially covered by the Student Mobility Scholarship, Double-Degree Support Program of the Ministry of Education, Youth and Sport of the Czech Republic (the financial agreements no. DD10/10 and DD004/11) and from interest-free loan from the **Kompakt spol. s r.o.**<sup>‡</sup> company.

---

---

<sup>\*</sup><http://www.isi-hagenberg.at>

<sup>†</sup><http://www.quanmax.ag>

<sup>‡</sup><http://www.kompakt-cr.cz>



# Acknowledgements

Foremost, I would like to express my sincere gratitude to all my advisers: A.Univ.-Prof. Dr. Josef Küng from Institute for Application Oriented Knowledge Processing, Johannes Kepler University in Linz <sup>\*</sup>, Doc. Ing. Kouba Zdeněk CSc. from Department of Cybernetics, Czech Technical University in Prague <sup>†</sup> and DI Franz Wieshofer from Quanmax AG company for their interest, support, background and time they devoted to my thesis.

Besides my advisers, I would like to thank Prof. Dr. Michal Pěchouček, MSc. from Department of Cybernetics, Czech Technical University in Prague who made possible for me to take part on a Double-Degree programme between Czech Technical University in Prague and Johannes Kepler University in Linz for his motivation. O.Univ. Prof. Dr.phil. Dr.h.c.mult. Bruno Buchberger from Research Institute for Symbolic Computation, Johannes Kepler University in Linz <sup>‡</sup> for his enthusiasm, and immense knowledge, who devoted his time to help me with my thesis, prepared me for the final presentation and for his wise advices.

My sincere thanks also goes to PaedDr. Miroslav Káninský and Pavel Bažant from Kompakt spol. s r.o. company who provided me with financial support during my studies.

I would like to thank many people who have taught and motivated me: my high school math and physics teachers, especially Mgr. Marie Dekojová and Mgr. Alexej Bezděk, my undergraduate teachers at Czech Technical University, especially doc. Dr. Ing. Michal Bednařík for their kind assistance, giving wise advices, helping with various applications, and so on.

I also thank Leona Svobodová and Betina Curtis who were always there to help out on faculty and other issues.

I am grateful for the stimulating discussions and the proof reading of this thesis to my fellow colleague Mariam Wael Hassan Mohamed Mansour Rady and for the motivation and help to my fellow colleague Ing. Marek Sacha.

Last but not the least, I would like to thank my family, especially my dear mother Ing. Alena Kubová for giving birth to me at the first place, raising me and supporting me both spiritually and financially throughout my studies and the whole life.

---

<sup>\*</sup><http://www.faw.uni-linz.ac.at>

<sup>†</sup><http://cyber.felk.cvut.cz>

<sup>‡</sup><http://www.risc.jku.at>



I hereby declare under oath that the submitted Master's thesis has been written solely by me without any third-party assistance. Additional sources or aids are fully documented in this paper, and sources for literal or paraphrased quotes are accurately credited.

A handwritten signature in blue ink, consisting of stylized cursive letters, likely representing 'Jan Kubový'.

**Ing. Jan Kubový**  
Wednesday 13<sup>th</sup> July, 2011





# Abstract

This thesis engages in solving problem of collecting media files from different sources in the network, identify them, sort them and make them available to other devices in the network. Identification and removal of duplicates and gathering additional information such as author, album, date, etc.

This thesis is divided into two parts. In the first part a technology and market overview of existing devices, software and similar solutions can be found.

The second part introduces one solution and shows the implementation of such solution. Further possibilities of the implementation will be also discussed.

**Keywords:** Digital Living Network Alliance (DLNA), DLNA server, content aggregator, media files meta data



# Contents

<b>I Research</b>	<b>1</b>
<b>1 Introduction</b>	<b>3</b>
1.1 Goal	4
1.2 Tasks	4
<b>2 Technologies</b>	<b>5</b>
2.1 Digital Living Network Alliance (DLNA)	5
2.2 DLNA Certified Device Classes	7
2.2.1 Home Network Devices (HND)	7
2.2.2 Mobile Handheld Devices (MHD)	8
2.2.3 Home Infrastructure Devices (HID)	9
2.2.4 Digital Media Aggregator (DMA)	9
2.3 Universal Plug and Play (UPnP)	10
2.3.1 UPnP Audio and Video standards	11
2.3.2 UPnP ContentDirectory Service Template	12
2.4 Container Formats	12
2.5 Audio Codecs	16
2.6 Video Codecs	17
2.7 Image Formats	18
2.8 Metadata	20
<b>3 Market overview</b>	<b>27</b>
3.1 Devices Overview	27
3.2 Media Server Software Overview	28
3.3 Apple AirPlay	29
3.4 Summary	36
<b>4 Home Environment Example</b>	<b>37</b>
<b>II Implementation</b>	<b>41</b>
<b>5 Shared Media Types</b>	<b>43</b>
5.1 Audio	43
5.1.1 ID3 Tag	43
5.1.2 Xiph Comments	45
5.2 Image	46

5.2.1	EXIF & IFD . . . . .	46
5.3	Video . . . . .	49
<b>6</b>	<b>Collecting Media Files</b>	<b>51</b>
6.1	Sources . . . . .	51
6.2	Metadata Gathering . . . . .	51
6.3	Media File Identification . . . . .	54
6.4	Media Files Duplicates Identification . . . . .	56
6.5	Sharing Media Information . . . . .	58
<b>7</b>	<b>Topology</b>	<b>59</b>
7.1	Server-Based Topology . . . . .	59
7.2	Distributed Topology . . . . .	60
7.3	Conclusion . . . . .	60
<b>8</b>	<b>Prototype Architecture</b>	<b>61</b>
8.1	Application Model . . . . .	61
8.2	Aggregators Package . . . . .	61
8.2.1	AbstractAggregator Class . . . . .	61
8.2.2	Aggregator Class . . . . .	62
8.2.3	ContentDirectoryAggregator Class . . . . .	62
8.2.4	FileSystemDirectoryAggregator Class . . . . .	63
8.2.5	SambaAggregator Class . . . . .	63
8.3	Samba Subpackage . . . . .	63
8.3.1	SambaConnection Class . . . . .	64
8.3.2	SambaTicker Class . . . . .	64
8.4	GUI Package . . . . .	65
8.4.1	AggregatorGUI Class . . . . .	65
8.4.2	PreferencesDialog Class . . . . .	66
8.5	Media Models Package . . . . .	67
8.5.1	AbstractObject Class . . . . .	67
8.5.2	DeviceObject Class . . . . .	67
8.5.3	ContentObject Class . . . . .	67
8.5.4	MediaObject Class . . . . .	67
8.5.5	AudioObject Class . . . . .	68
8.5.6	ImageObject Class . . . . .	68
8.5.7	VideoObject Class . . . . .	68
8.5.8	ItemFactory Class . . . . .	68
8.5.9	ObjectFactory Class . . . . .	68
8.6	Servers Package . . . . .	69

8.6.1	ContentDirectoryCollector Class . . . . .	69
8.6.2	ContentDirectoryServer . . . . .	70
8.6.3	ContentDirectoryServiceClass . . . . .	71
8.7	Libraries package . . . . .	71
8.7.1	DMALib Class . . . . .	71
8.7.2	FileSystemExporter Class . . . . .	71
8.8	Testing of the Application . . . . .	72
<b>9</b>	<b>Application User Guide</b>	<b>73</b>
9.1	Aggregation Process . . . . .	74
9.2	Aggregators Configuration . . . . .	74
9.3	Browsing Aggregated Media . . . . .	74
9.4	Making Media available to the Network . . . . .	77
9.5	Saving and Loading Snapshots . . . . .	78
<b>10</b>	<b>Conclusion</b>	<b>79</b>
10.1	Application Proposal . . . . .	80
<b>11</b>	<b>References</b>	<b>85</b>
<b>A</b>	<b>Prototype UML Diagrams</b>	<b>87</b>
<b>B</b>	<b>List of Figures</b>	<b>102</b>
<b>C</b>	<b>List of Tables</b>	<b>103</b>
<b>D</b>	<b>List of Abbreviations</b>	<b>105</b>
<b>E</b>	<b>Curriculum Vitae</b>	<b>109</b>



# Part I

## **Research**

This part focuses on the prerequisite knowledge and technologies needed to the actual implementation of the prototype software.

First, a brief introduction of the Digital Living Network Alliance (DLNA) and the Universal Plug and Play (UPnP), as representatives of the core technologies, is present.

Next, commonly used and for the purpose of this thesis suitable image formats, audio and video codecs and container formats are described.

Finally, a look at the current market, collection and description of devices and software competitive to the DLNA Aggregator is done. Also a home environment example is introduced for testing purpose of the the implementation of the prototype software.





*All media exist to invest our lives with artificial perceptions and arbitrary values.*

– Marshall McLuhan

# 1

## Introduction

This thesis introduces a Network Content Aggregator for DLNA Server and related technologies, media container formats, media codecs and available 3<sup>rd</sup> party software and libraries. From those some of the 3<sup>rd</sup> party software and libraries are selected and used in the implementation of the software prototype.

An overview of the current market state and the existing devices and software competitive to the DLNA Server or the DLNA Aggregator is present in the first part of this thesis.

On the end of the first part, before the software prototype will be introduced, a typical home environment example is given. This example is used as a specimen of a typical household and a target environment of such software.

The second part of this thesis deals with the implementation of the software prototype and related requirements such as 3<sup>rd</sup> party libraries. A description of the different media types and the way how to gather additional information from them (e.g. author, album, date, etc.) is present in Chapter 5 ‘Shared Media Types’ on page 43.

The way how the media files are collected from the different devices in the network and with that related problems (e.g. identification, duplicates, etc.) is described in Chapter 6 ‘Collecting Media Files’ on page 51.

One of the tasks given by the Quanmax AG company was to elaborate about two different network topologies and find out their usability in such software. Those topologies and their possibilities are elaborated in Chapter 7 ‘Topology’ on page 59.

The prototype architecture description and the application user guide are present on the end of the second part in Chapter 8 ‘Prototype Architecture’ on page 61 and Chapter 9 ‘Application User Guide’ on page 73. The Unified Modeling Language (UML) diagrams of the implementation can be found in the Appendix A ‘Prototype UML Diagrams’ on page 87.

The conclusion and proposal of an extension and an improvement of the software prototype is present on the end of this document in Chapter 10 ‘Conclusion’ on page 79.

## 1.1 Goal

The goal of this thesis is to implement a prototype of a DLNA Aggregator, which is able to aggregate media files from different devices in the network. The prototype introduces an architecture, which makes it easy to implement additional device types and extend the ability of the DLNA Aggregator.

The prototype application includes a simple Graphical User Interface (GUI) which allows to control the DLNA Aggregator, go through the aggregated media files and display the details of those files.

Some of the aggregators may require configuration. In the current implementation only the Samba (SMB) aggregator and the Local File System (LocalFS) aggregator do need this. The application prototype allows to configure such aggregators using the GUI.

The GUI will be implemented only for testing and presentation purposes. Any eventual release of the application will not include this particular GUI as the DLNA Aggregator will be part of a DLNA Server, fully controlled by that DLNA Server and the aggregated media files will be presented to the user through GUI implemented in that DLNA Server.

## 1.2 Tasks

1. Get familiar with the current market around DLNA devices and server software.
2. Analyze current used technologies i.e. codecs, container formats for gathering meta-data.
3. Get familiar with the DLNA (class devices, technology components, software).
4. Elaborate about server-based and distributed topology and the possible usage.
5. Define a media content gathering proposal.
6. Implement a software prototype and evaluate the proposal.

*One machine can do the work of fifty ordinary men. No machine can do the work of one extraordinary man.*

– Elbert Hubbard

# 2

## Technologies

In this chapter i will go through technologies the DLNA Aggregator is concerned with. In Section 2.4 ‘Container Formats’, Section 2.5 ‘Audio Codecs’, Section 2.6 ‘Video Codecs’, Section 2.7 ‘Image Formats’ and Section 2.8 ‘Meta-data’ the goal is not to compare the individual technologies and make conclusions about their advantages and disadvantages but to show the diversity of them and the complexity the DLNA Aggregator has to deal with.

### 2.1 Digital Living Network Alliance (DLNA)

The DLNA is a non-profit cross-industry organization of leading companies in the customer electronics, computing, mobile devices and service provider industries. The goal of this alliance is using a standards-based technology to make easier for customers to use and share media content (i.e. videos, music, photos, etc.) [11].

One of the key technology of the DLNA is the **UPnP Device Control Protocol Framework (DCP)**, which simplifies the device networking and is the selected device discovery and control solution for digital home devices [11]. This technology will be described more in detail in Section 2.3 ‘Universal Plug and Play (UPnP)’ on page 10.

Next key technology component used through UPnP technology is **Internet Protocol (IP)**. IP is supported by a wide range of devices and is based on industry standard specifications. IP can connect any device to the internet and allows application to communicate transparently [11].

**Media Format and Transport Model** is another key technology intended to achieve a baseline for the network interoperability. While improvements in media codecs and formats are being encouraged, the DLNA media format support applies for media content that is being transported between server device (Digital Media Server (DMS) or Mobile Digital Media Server (M-DMS)) and player/renderer device (Digital Media Player (DMP), Mobile Digital Media Player (M-DMP), Digital Media Renderer (DMR), etc.) [11]. The DLNA devices can be divided to two categories: The Home Network Devices (HND) and the Mobile

Handheld Devices (MHD) (both will be described in detail in Section 2.2 ‘DLNA Certified Device Classes’ on page 7). For both of those categories a set of required and a set of optional media formats is defined for each of the three media classes (audio, video and image). Required and optional formats for this purpose for both HND and MHD can be seen in Table 2.2 ‘Supported DLNA Media Formats[11]’ on page 21.

The *Media Container Formats* are discussed in detail in Section 2.4 ‘Container Formats’ on page 12. A list with the description of the *Audio Codecs* is available in Section 2.5 ‘Audio Codecs’ on page 16. In Section 2.6 ‘Video Codecs’ on page 17 is a detailed list of *Video Codecs*. *Image Formats* are described in Section 2.7 ‘Image Formats’ on page 18.

Every DMS, DMP, DMR, Digital Media Printer (DMP<sub>r</sub>), M-DMS and M-DMP device has to support all the required media formats listed in Table 2.2 ‘Supported DLNA Media Formats[11]’ on page 21. Every DMS, DMP, M-DMS, M-DMP and Mobile Digital Media Downloader (M-DMD) device may support any of the optional format listed in that table. Also any of DMP, M-DMP, DMR, M-DMD and DMP<sub>r</sub> has to be able to receive content from any DMS or M-DMS [11].

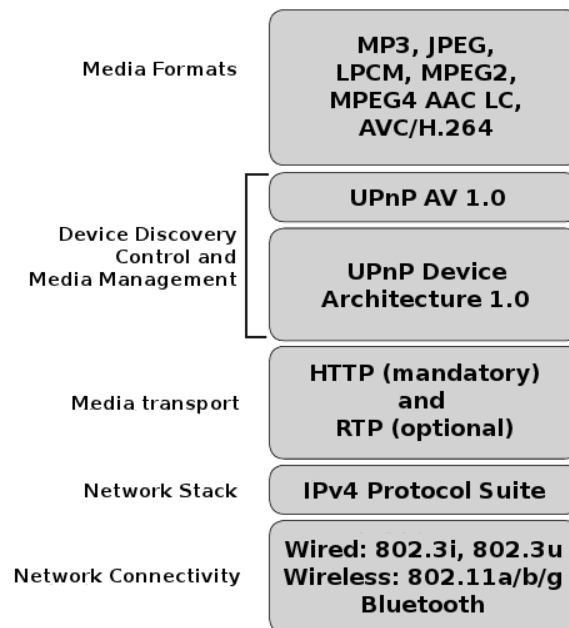


Figure 2.1: DLNA Interoperability Guidelines Building Blocks[11]

The DMS or M-DMS has on the other hand be able to decode as many *Codecs* and *Container Formats*<sup>1</sup> as possible and transcode them at least to the set of *Required Formats*. However the Digital Media Aggregator (DMA) needs knowledge about those *Codecs* and *Formats* only for the purpose of metadata (e.g. author, album, year, etc.) extraction.

**Media Management, Distribution, and Control** is the next key technology. It allows the devices to identify, manage and distribute the media content in the network and is provided by the UPnP Audio/Video technology (see Section 2.3.1 ‘UPnP Audio and Video standards’ on page 11)[11].

There are other technologies used by the DLNA, however they are out of the scope of this thesis. The building blocks of the DLNA interoperability guidelines can be seen in Figure 2.1 ‘DLNA Interoperability Guidelines Building Blocks[11]’.

## 2.2 DLNA Certified Device Classes

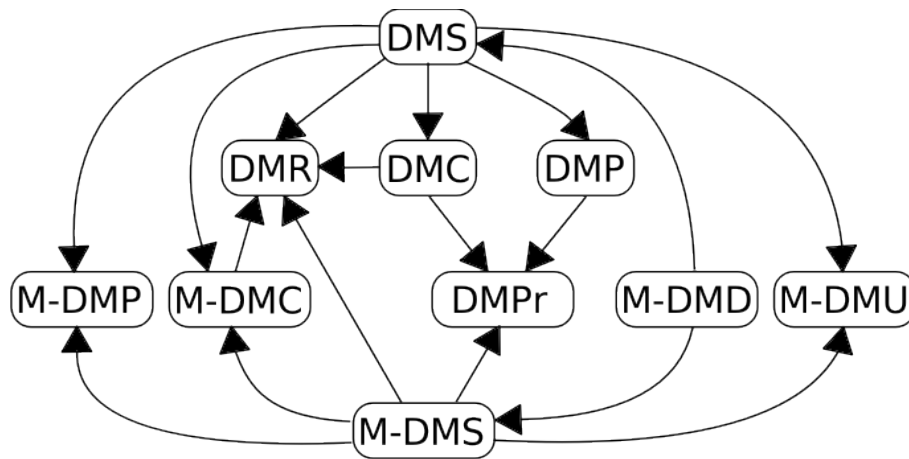


Figure 2.2: DLNA Classes Relations

The DLNA divides devices into 12 different classes in 3 main device categories[11]. The relations between the *DLNA Certified Device Classes* can be seen in Figure 2.2 ‘DLNA Classes Relations’.

### 2.2.1 Home Network Devices (HND)

**Digital Media Server (DMS)** stores the media content (e.g. music, video, images, etc.) and makes it available to all the devices in the wired and/or wireless network like DMR or DMP. These devices are for example computers or Network

<sup>1</sup>The difference between *Codec* and *Container Format* will be explained in Section 2.4 ‘Container Formats’ on page 12

Attached Storage (NAS) [11]. The DMS can also have transcoding<sup>1</sup> capabilities and can also include a DMA.

**Digital Media Controller (DMC)** plays content, which it finds on DMS or DMR. Digital Media Controller (DMC) can be a Tablet, Personal Digital Assistant (PDA), etc.[11].

**Digital Media Player (DMP)** provides rendering of content which it finds on a DMS. A DMP can be for example a TV, home theatre, stereo, projector, PCs, monitors, etc.[11]. A DMP can provide one or more of the rendering capabilities: image and/or video rendering and/or audio playback. For example an image frame provides only image rendering. A HiFi provides only audio playback. But a TV provides both audio and video rendering and additionally may provide also image rendering.

**Digital Media Renderer (DMR)** plays content received from a DMC. This can be also a TV, audio/video receiver etc.[11]. The DMC does not provide any actual content to the DMR, it just tells the DMR to pull certain media content from a DMS and play/render it. The difference between DMP and DMR is that a DMR needs to be told what to play by a DMC, but DMP can control itself alone without a DMC.

**Digital Media Printer (DMPr)** provides printing of media like pictures or documents. Generally, a DMP or DMC with print capability can print on a DMPr [11].

## 2.2.2 Mobile Handheld Devices (MHD)

**Mobile Digital Media Server (M-DMS)** is a wireless device, which stores media content (i.e. music, video, images) and makes it available to all devices in the wired/wireless network like M-DMP, DMR, DMP or DMPr. These devices are for example cell phones or portable music players[11].

**Mobile Digital Media Controller (M-DMC)** is a wireless devices that can find content on a DMS or M-DMS and send it to a DMR. Examples: PDA and cell phones [11].

---

<sup>1</sup>Transcoding is *digital-to-digital* converting method[24] of potentially unsupported media format into supported one for the DMP or DMR on the fly. This functionality has big advantage because there are lot of different, not always supported, media formats circulating around but only certain of them are supported by the amount of devices on the market.

**Mobile Digital Media Player (M-DMP)** can find and play content from other DLNA devices in DMS or M-DMS class. A M-DMP can be for example a cell phone, tablet, netbook, etc. [11].

**Mobile Digital Media Uploader (M-DMU)** is a wireless devices able to send/upload content to a DMS or M-DMS. A Mobile Digital Media Uploader (M-DMU) is for example a digital camera or a cell phone [11].

**Mobile Digital Media Downloader (M-DMD)** is a wireless devices, which downloads and stores content from a DMS or M-DMS. A M-DMD can be for example a cell phone or a portable music player[11].

### 2.2.3 Home Infrastructure Devices (HID)

**Mobile Network Connectivity Function (M-NCF)** is a device, which provides a bridge between a mobile handheld device network connectivity and home network connectivity[11].

**Media Interoperability Unit (MIU)** is a device, which provides content transformation between required media formats for home network and mobile handheld devices[11].

### 2.2.4 Digital Media Aggregator (DMA)

*Digital Media Aggregator* or *DLNA Aggregator* is not a specified *DLNA certified class*. It is a part of a DMS and therefore it has strong relation to that class. In the document DMS will be used if the context is related to the whole server and DMA if the context is related only to the aggregator part of the DMS only.

The DMS or M-DMS is in general responsible for:

- Content acquisition, recording and storing.
- Content protection enforcement.
- Content distribution to other DLNA devices in the network.
- Content aggregation (DMA).
- Device & media management.
- ... *in addition a DMS may provide DMP capabilities, media transcoding and a user interface*

A DMA as a part of a DMS is expected to provide:

- Knowledge about available devices in the network.
- Knowledge about relevant content in the network.
- Access to content on available devices in the network.
- Announce appeared devices in the network.

- Announce disappeared devices from the network.
- Announce content changes on the devices in the network.
- Metadata gathering (e.g. author, album, date, etc.).

## 2.3 Universal Plug and Play (UPnP)

The UPnP technology is a set of networking protocols for network devices supported by the UPnP FORUM. The UPnP enables seamless discovery and establishing of services for data sharing to devices in the network and is independent of any programming language, operating system or network technology (i.e. Ethernet, FireWire, IrDA, Bluetooth, Wi-Fi, etc.). The target of the UPnP are home networks, proximity networks, small businesses and commercial buildings[53].

The UPnP Device Control Protocol Framework (DCP) allows to the devices in the network discovering each other and communicate seamlessly. It is a distributed and open technology based on standard such as Transmission Control Protocol (TCP)/IP, User Datagram Protocol (UDP), Hypertext Transfer Protocol (HTTP), Extensible Markup Language (XML), etc. The UPnP architecture supports automatic network configuration, which means that every UPnP compatible device from any manufacturer can dynamically connect to the network, obtain its IP address, announce its abilities on request and discover the abilities of other devices in the network. The Dynamic Host Configuration Protocol (DHCP) and the Domain Name System (DNS) servers are optional and are used only if they are available in the network. The UPnP devices can leave the network without leaving any undesirable state informations behind[53].

The DCP defines the protocol for communication between the devices in the network. The DCP uses the stack shown in Table 2.1 ‘UPnP Architecture (DCP) stack [22]’ for discovery, control, eventing and presentation. On the top of the stack, messages contain only vendor specific information which are supplemented by the UPnP Forum information and hosted in the UPnP specific protocol such as Simple Service Discovery Protocol (SSDP), General Event Notification Architecture (GENA) and Simple Object Access Protocol (SOAP). Those messages are delivered by the HTTP, either multicast or unicast over the UDP or standard HTTP over the TCP. The HTTP messages are then delivered over the IP [22].

The DCP defines two device classes: the *Controlled Devices* and the *Control Points*, where the role of a *Controlled Device* is responding to requests from *Control Point*. In one network endpoint multiple *Controlled Devices* and/or *Control Points* can run simultaneously[22].

The DCP works in couple steps. In the so called **Step 0**, named *Addressing*, the device will try to receive an address from a DHCP server if such server is present in the network. If no DHCP server is present the device will assign itself



UPnP vendor
UPnP Forum
UPnP Device Architecture (DCP)
SSDP, SOAP, GENA
HTTP
UDP, TCP
IP

Table 2.1: UPnP Architecture (DCP) stack [22]

an IP address using *AutoIP* in the 169.254/16 range except of the first and last 256 addresses which are reserved and must not be used[22].

A successfully assigned IP address enables **Step 1**, which is called *Discovery*. In this step *Control Points* can find other *Controlled Devices* in the network[22].

If an "interesting" *Controlled Device* is found the *Control Point* will need to know the abilities of such device, which can be done in **Step 2** called *Description*. By sending a HTTP request to the *Descriptor URL* of the targeted *Controlled Device* a XML file with the description of all *Embedded Devices*, services, vendor-specific information, etc. will be sent back in the response [22].

**Step 3** in the UPnP networking is *Control*. A *Controlled Device* can be controlled by a *Control Point* if that *Control Point* has discovered the *Controlled Device* and has received the XML description file from it. From the XML description file the *Control Point* knows how to control the *Controlled Device* and can send control requests to the *Controlled Device*. Such requests can change the values of state variables of the *Controlled Device* depending on the concrete implementation[22].

*Eventing* is **Step 4** in the UPnP networking which allows listening for changes of state variables of a *Controlled Device*. Any *Control Point* device may subscribe to receive event messages with these information. An event message contains the name of one or more state variables and the current value of those variables expressed in a XML using GENA [22].

Last, **Step 5** is about *Presentation* which does not have to be present on every device. But if the *Controlled Device* will provide a URL for presentation, it can be loaded into a browser. The page can show state of the device and/or its state variables and it can allow to control the *Controlled Device* from that page. What the presentation page shows and allows to control, depends on the capabilities of that presentation page [22].

### 2.3.1 UPnP Audio and Video standards

The DCP divides devices into couple categories such as Audio/Video, Basic, Device Management, Home Automation, Networking, Printer, Remote Access, Re-

moting and Scanner. For the purpose of this thesis a closer look on the *UPnP Audio and Video* category[56] will be introduced.

The UPnP A/V category is one of the DLNA key technologies. It includes following device and service templates:

**UPnP MediaServer** is a device template that provides media content (using *Content Directory*) to other UPnP devices in the network[35].

**UPnP MediaRenderer** is a template that defines a general-purpose device template that can be used to instantiate any device that is capable of rendering AV content from the network. It provides also a set of controls for the *Control Point* to control specified rendering options of AV content (e.g. brightness, contrast, volume, etc.)[34].

**UPnP ConnectionManager** is a service template that provides modelling of streaming capabilities of an AV devices and binding of those capabilities between devices[20].

**UPnP ContentDirectory** provides a mechanism to browse content on the server and obtain detailed information about the content objects[36].

**UPnP Rendering Control** is a service template, which provides *Control Points* with the ability to query and/or adjust any rendering attribute that the device supports[37].

**UPnP AVTransport** is a service template, which provides control over the transport of audio and video streams[15].

### 2.3.2 UPnP ContentDirectory Service Template

The most important template from the list of UPnP Audio and Video standards for the DMA is the *ContentDirectory* template. This service allows the *Control Points* to enumerate the list of content (i.e. audio, video and still image files) on the device. It provides detailed information about each item, such as author, album, year, etc. It also enables searching and filtering capabilities.

Most commonly is this service present on a UPnP MediaServer and allows to access files from a remote UPnP device. This allows the user to play or render media content from a variety of devices without direct interaction with the device containing the content.

## 2.4 Container Formats

The difference between *Container Format* and *Codec* is that *Container Format* (i.e. AVI, MKV, MOV, WMV, etc.) specifies how the data streams (one or more audio, video and/or other streams like subtitles or other data) inside a file are

organized, but how the actual data are represented is specified by the concrete *Codec*. *Codec* describes how video or audio data are encoded (compressed) and how they should be decoded (decompressed). Theoretically any *Codec* can be used in any *Container Format* but sometime there are license restriction for a certain *Codec* to a certain *Container Format* (e.g. Windows Media Video (WMV) is only used in Windows Media files). List of most commonly used and supported *Container Formats* will follow further in this section.

Table 2.4 ‘Media Container Formats Comparison’ on page 23 shows a comparison of the chosen *Container Formats* with focus on the relevant data according to this thesis. For better orientation Table 2.5 ‘Media Container Formats and Audio Codecs Support’ on page 24 and Table 2.6 ‘Media Container Formats and Video Codecs Support’ on page 25 shows the possible usage of the different *Codecs* in chosen *Container Formats*. In those tables the symbol “X” stands for present support and the symbol “-” stands for absent support. If the support is composed of distinct values, those values are represented in the table cells as comma-separated values.

**Audio Interchange Format File (.AIFF, .AIF)** is a *Container Format* co-developed by APPLE INCORPORATED and based on more general *Interchange File Format (IFF)* used on *Amiga* systems. Audio Interchange Format File (AIFF) is popular on *MacOS* and *UNIX* systems. It can contain only one, by default uncompressed Pulse-Code Modulation (PCM) audio stream. There is also a variant supporting compression using various *Codecs* such as *MPEG-1,2 Audio Layer III (MP3)*. This *Container* is divided into a number of chunks each identified by *Four Character Code (FourCC)*:

- Common Chunk (required)
- Sound Data Chunk (required)
- Marker Chunk
- Instrument Chunk
- **Comment Chunk**<sup>1</sup>
- **Name Chunk**<sup>1</sup>
- **Author Chunk**<sup>1</sup>
- **Copyright Chunk**<sup>1</sup>
- **Annotation Chunk**<sup>1</sup>
- Audio Recording Chunk
- MIDI Data Chunk
- Application Chunk
- **ID3 Chunk**<sup>1</sup>

**Advanced Systems Format (.WMV, .WMA, .ASF, .ASX)** is a MICROSOFT CORPORATION proprietary digital audio/digital video container format, which typically

<sup>1</sup>Chunks containing useful metadata for the DMA

uses *WMV Codec* for video encoding and *Windows Media Audio (WMA) Codec* for audio streams, but in general can use any codec for both stream types. It supports streaming video over network and can contain one or more media (i.e. audio and/or video) streams. It uses own tagging format for metadata called *ASF Tags*[18].

**Audio Video Interleave (.AVI)** introduced by MICROSOFT as a part Video for Windows (VfW) technology and can contain multiple audio and video streams. The file is divided into chunks identified by FourCC code. Metadata can be stored in an optional *INFO* chunk[1] or in an embedded Extensible Metadata Platform (XMP) [28]. This *Container Format* may contain any audio/video data inside the chunks using virtually any *Codec*. This *Container Format* was not intended for streaming[64].

**DivX Media Format (.DIVX)** developed by DIVX, INCORPORATED includes following features:

- Interactive video menus.
- Multiple subtitles (XSUB).
- Multiple audio tracks.
- Multiple video streams.
- Chapter points.
- **Other metadata (XTAG)**<sup>1</sup>.
- Multiple format.
- Partial backwards compatibility with *Audio Video Interleave (AVI)*.

Video streams are encoded using one of two *DivX Codecs*; the regular *MPEG-4 Part 2 DivX codec* and the *H.264/MPEG-4 AVC DivX Plus HD codec*. Audio streams can be encoded using various audio codecs like *MP3*, *Audio Codec 3 (AC-3)*, *MPEG-2,4 Advanced Audio Coding (AAC)*, etc.[19].

**Flash Video (.FLV, .F4V)** was originally developed by MACROMEDIA INCORPORATED and now by ADOBE SYSTEMS INCORPORATED. The Flash Video (FLV) is used to deliver video over the internet using *Adobe Flash Player*. The FLV can be also embedded within a *ShockWave Flash (SWF)* file. This *Container Format* uses *Sorenson Spark*, *VP6* or *H.264 Codecs* for video stream and *MP3*, *Adaptive Differential Pulse-Code Modulation (ADPCM)*, *AAC*, *Nellymoser* or *High-Efficiency Advanced Audio Coding (HE-AAC)* for audio stream. FLV has own proprietary support for metadata[29].

**Matroska (.mkv)** is an open multimedia format which use *Extensible Binary Meta Language (EBML)* to provide extendibility for future format changes. The

---

<sup>1</sup>Feature containing useful metadata for the DMA

*Matroska File Format (MKV)* can use various *Codecs* for encoding video and audio streams and implement own proprietary tagging system using the EBML [52].

**MPEG-1, MPEG-2 (.MPG, .MPEG, .MPE)** are a formats designed by the INTERNATIONAL STANDARD ORGANIZATION (ISO) and used in *Video CDs (VCD)*. This container is protected by a patent and cannot store any metadata or tags. Video stream is encoded with *MPEG-1* or *MPEG-2* codec and audio stream with *MP3* codec[2].

**MPEG-4 (.MP4)** is based on *ISO Base Media File Format* [8] developed by ISO and can contain video and audio streams, subtitles and still images. This container format allows to stream media over the internet. Video streams can be encoded using *MPEG-4*, *MPEG-2* or *MPEG-1* codec and audio can be encoded using *AAC*, *MP3*, etc. Metadata may be stored using the format defined by the standard or using *XMP* [9].

**Ogg (.ogg)** is maintained by the XIPH.ORG FOUNDATION and can multiplex a number of independent streams for audio, video, text and metadata. In this container format *Free Lossless Audio Codec (FLAC)*, *OggPCM*, *Vorbis* or *Speex* codec can be used to encode audio streams, *MPEG-4*, *DivX*, *Xvid*, *RealVideo*, *WMV* or *Dirac* codec can be used to encode video streams. Metadata can be stored using *Continuous Media Markup Language (CMML)*, *Ogg Skeleton* or *Xiph Comments*[61].

**QuickTime (.MOV, .QT)** was introduced by APPLE and contains one or more tracks, each of which stores a particular type of data such as audio, video, text, etc. Track can either contains an encoded media stream or a reference to a media stream in another file. The advantage of the fact that the track can be just a reference to a stream in another file makes this container format more suitable for editing media content than the others. *QuickTime (QT)* uses the same *Codecs* for encoding video and audio streams as *MPEG-4* container but has less support especially on hardware devices because it is not an International Standard like *MPEG-4 Container Format*. It uses own proprietary tagging system[14].

**RealMedia (.RM, .RMVB)** was introduced by REALNETWORKS INCORPORATED and uses *RealVideo* codec for video and *RealAudio* codec for audio streams. *RealMedia (RM)* is supported on many platforms (e.g. *Windows*, *Mac*, *Linux*, *Solaris*, etc.)[48] however could be played in the past only on extremely proprietary *RealPlayer*. But nowadays, for example the open-source *ffmpeg*[21] library can play *RealVideo* without *RealPlayer* or any parts thereof. This format is suitable for use as a streaming media format and supports both *Constant Bitrate*

(*CBR*) and *Variable Bitrate (VBR)* encoding.

**Video Object (.VOB)** is a container format designed for DVD Video and can contain video stream, multiple audio streams, subtitles, menu and navigation content. It is based on *MPEG Program Stream* (specified in MPEG-1 Part 1 [2] and MPEG-2 Part 1 [5]). Video stream can be encoded using *H.262/MPEG-2 Part 2* or *MPEG-1 Part 2* codec and audio streams using *MPEG-1 Audio Layer II*, *MPEG-2 Audio Layer II*, *PCM*, *AC-3* or *Digital Theatre Systems (DTS)* codec. Information about the location of audio and video streams, chapters, etc. in a Video Object (VOB) are stored in separate info (.IFO) and info backup (.BUP) file.

**Windows WAVE audio (.WAV)** was developed by MICROSOFT and INTERNATIONAL BUSINESS MACHINES (IBM). Windows Wave Audio (WAV) is similar to *AIFF* and contains usually uncompressed Linear Pulse-Code Modulation (LPCM) audio stream. This format is popular on *Windows* systems.

## 2.5 Audio Codecs

A list of audio codecs will be introduced in this section. Because of the large number of available codecs, only commonly supported audio codecs will be included in this overview.

**MPEG-2,4 Advanced Audio Coding (AAC)** is a standard developed by the MOVING PICTURES EXPERT GROUP (MPEG). AAC allows to encode five full-bandwidth channel audio signals at data rates of 320kbps for ITU-R indistinguishable quality [7]. There is also a HE-AAC version.

**Audio Codec 3 (AC-3)** is a digital compression algorithm described by the UNITED STATES ADVANCED TELEVISION SYSTEMS COMMITTEE (ATCS) and can encode from one to five full-bandwidth audio channels, along with a low frequency enhancement channel. Data rates can be between 32kbps and 640kbps [55].

**Apple Lossless Audio Codec (ALAC)** is a proprietary lossless audio compression scheme introduced by APPLE.

**MPEG-4 Audio Lossless Coding (ALS)** was described by ISO as lossless coding for digital audio signals with up to 65535 channels support [10].

**Digital Theatre System (DTS)** was developed by DTS INCORPORATED and is a lossless audio codec with variable data rates up to 24.5Mbps. It can encode up to 7 channels and one low frequency enhancement channel [33].

**Free Lossless Audio Codec (FLAC)** was developed by XIPH.ORG and as the name suggest its a lossless audio codec. It can encode up to 8 channels and it uses *Xiph Comments* tags for storing metadata [60].

**MPEG-1,2 Audio Layer 3 (MP3)** is a lossy audio codec described by ISO as part of *MPEG-1* [4] and *MPEG-2*. Metadata can be stored using *ID3* tags or *APE* tags.

**Pulse-Code Modulation (PCM)** is an uncompressed, header-less audio format.

**RealAudio** is a proprietary audio codec developed by REALNETWORKS [48].

**MPEG-4 Scalable to Lossless (SLS)** was described by ISO . MPEG-4 Scalable to Lossless (SLS), is an extension to *ISO 14496-3*[10] standard and allows lossless audio compression to lossy *MPEG-4 General Audio coding methods*.

**Speex** was developed by XIPH.ORG. It is a patent-free, lossy audio codec optimized for *VoIP* [57].

**Vorbis** is a lossy audio codec developed by XIPH.ORG which can encode up to 255 discreet channels. Metadata are stored in the comment header of the file [62].

**Windows Media Audio (WMA)** is a proprietary technology developed by MICROSOFT and consists of four distinct codecs[50]:

- **Windows Media Audio:** a lossy codec able to encode one or two channels
- **Windows Media Audio Professional:** improved *Windows Media Audio* codec, which is able to encode theoretically an unlimited number of channels
- **Windows Media Audio Lossless:** can encode up to 6 discreet channels
- **Windows Media Audio Voice:** a lossy audio codec designed for low-bandwidth, voice application

## 2.6 Video Codecs

List of commonly used and supported video codecs follows:

**MPEG-1** is a lossy video codec introduced by ISO . It supports resolutions up to 4095x4095 and bitrates up to 100 Mbps [3]. Most common usage of this codec was on *VideoCD*.

**MPEG-2** is a lossy video codec, a successor of *MPEG-1* with some enhancements e.g. support for interlaced video. *MPEG-2* decoders are able to playback *MPEG-1* files [6].

**MPEG-4** was intended for very high compression of audio and video data and aims applications such as broadcast video over internet, Local Area Network (LAN), Wireless Local Area Network (WLAN), video databases, video email, home movies, games, etc. [47].

**RealVideo** is a lossy video codec. It is also suitable for streaming and was developed by REALNETWORKS [49].

**Theora** is a lossy codec developed by XIPH.ORG and based on the *VP3* video codec. Additional metadata can be stored in the header [63].

**VP6** is a lossy video codec developed by ON2 TECHNOLOGIES, INC.[45].

**WMV** is a proprietary lossy video codec developed by MICROSOFT. It consist of three distinct codecs[50]:

- **Windows Media Video**
- **Windows Media Video Screen**, which is a high efficient engine used to capture computer desktop for presentation purposes.
- **Windows Media Video Image**, which enables to encode still images with transition effects.

## 2.7 Image Formats

In comparison to audio codecs, video codecs and container formats the number of image formats is not so large. At least if we speak about actually used formats in the consumer electronics. Also speaking about images the border between *Codecs* (the actual representation of an image) and *Container Format* (the image file structure) is beginning to disappear. *Image Format* is used for both because most of the time the *Image Format* also specifies both. One exception is for example the Tag Image File Format (TIFF) which can embed other *Image Formats* like Joint Photographic Experts Group (JPEG) or others. If the mentioned name specifies only a certain *Codec* or *Container Format* but not both, it will be told explicitly.



**Graphics Interchange Format (GIF)** is a bitmap *Image Format* developed by COMPU SERVE. Graphics Interchange Format (GIF) supports animation, Lempel–Ziv–Welch (LZW) compression and palette of up to 256 distinct colors which can be chosen from 24bit space[32].

**Portable Network Graphics (PNG)** is an image format using lossless compression. It supports palette based images and RGB color images both with or without alpha channel. Portable Network Graphics (PNG) was created to replace and improve GIF [58].

**Joint Photographic Experts Group (JPEG)** uses lossy compression method (*Codec*), which is using the fact, that by selective neglect of certain information included in an image, that a much better compression ratio can be achieved than in lossless compression. Only information which do not cause visible damage by human observation can be neglected[54].

**JPEG File Interchange Format (JFIF)** is a minimal *Container Format* for exchanging JPEG encoded files commonly used for images on the internet. [26].

**Exchangeable Image File Format (EXIF)** is a *Container Format*, which can contain JPEG or TIFF images and metadata. Exchangeable Image File Format (EXIF) is used by cameras to store captured images and additional information such as rotation, shutter speed, focal length, metering mode, aperture, ISO speed information, time and date, GPS coordinates, copyright information, etc. [17].

**Tag Image File Format (TIFF)** was originally developed by ALDUS CORPORATION and is an unofficial standard for the use of saving images for publishing purposes. TIFF format is widely supported by image manipulating, publishing, scanning, faxing, word processing, etc. application. TIFF can contain both multiple images and additional data in one file. TIFF can also contain a wide range of image types and compression schemas both lossy and lossless (e.g. JPEG, LZW, etc.), both vector and bitmap based. The advantage over JPEG files is the possibility to use lossless (LZW) or none compression in TIFF images what allows images to be edited and re-saved without losing image quality.

The TIFF specification [13] is divided into two parts.

1. **Baseline TIFF** is the core of TIFF, the essentials that all mainstream TIFF developers should support in their applications. It describes features such as *multiple subfiles*, three basic *compression* schemes (None, *PackBits* and *Modified Huffman* compression), *image types* (BW, grayscale, palette-color, and RGB full-color images), *byte order*, etc. [13]

2. **TIFF Extensions** are TIFF features that may not be supported by all TIFF readers such as: *CCIT Bilevel Encoding*, *LZW Copression*, *JPEG Compression*, *CMYK images*, *Associated Alpha Handling*, etc. [13]

## 2.8 Metadata

Metadata systems will be introduced in this section. Only shared metadata systems between more than one *Codec* or *Container Format* will be mentioned here. Audio, video and image formats which implement own metadata system separately for each distinct *Codec* or *Container Format* will not be included in this list.

Metadata systems are shown in Table 2.3 ‘Metadata Systems Overview’ on page 22. The support of a concrete tagging system is indicated with symbol “X” in the corresponding table cell. Only codecs and container formats with metadata support are included in that table. Information about the content such as author, title, year, producer, etc. is meant by metadata in this context.

**IDentify an MP3 (ID3)** is a tagging system for audio files to store information about the origin and content of the audio within the file itself [44]. It is used for *MP3*, *AIFF* (inside *IFF* chunk called *ID3*), *Advanced Systems Format (ASF)* as attributes and *MP4 Container Format*.

**APE tag** are unstructured key/value pairs, originally designed for *MONKEY’S AUDIO* but can be used nowadays also for *MP3* files[12].

**eXtensible Metadata Platform (XMP)** is a standard created by *ADOBE* [28] and used in various file formats (e.g. *PDF*, *JPEG*, *GIF*, *PNG*, *TIFF*, *MP3*, *MP4*, *AVI*, *WAV*).

**Xiph Comments** is a metadata system [59] which is used in *Vorbis*, *FLAC*, *Theora* and *Speex* formats.

<b>Media Format</b>	<b>Required Format Set</b>	
	<b>Home Devices</b>	<b>Mobile Devices</b>
<b>Imaging</b>	JPEG	JPEG
<b>Audio</b>	LPCM (2 channel)	MP3 and MPEG4 AAC LC
<b>Video</b>	MPEG2	MPEG4 AVC (AAC LC Assoc Audio)
<b>Media Format</b>	<b>Optional Format Set</b>	
	<b>Home Devices</b>	<b>Mobile Devices</b>
<b>Imaging</b>	GIF, TIFF, PNG	GIF, TIFF, PNG
<b>Audio</b>	MP3, WMA 9, AC-3, AAC, ATRAC3plus	MPEG4 (HE-AAC, AAC LTP, BSAC), AMR, ATRAC3plus, G.726, WMA, LPCM
<b>Video</b>	MPEG1, MPEG4, WMV 9	VC1, H.263, MPEG4 part 2, MPEG2, MPEG4 AVC (BSAC or other for Assoc. Audio)

Table 2.2: Supported DLNA Media Formats[11]

	APE	ID3	Ogg Skel	Vorbis Comment	XMP	XTAG	Other
<b>Theora</b>				X			
<b>WMV</b>							X <sup>1</sup>
<b>FLAC</b>				X			
<b>MP3</b>	X	X			X		
<b>RealAudio</b>							X <sup>2</sup>
<b>Speex</b>				X			
<b>Vorbis</b>				X			
<b>WMA</b>							X <sup>1</sup>
<b>AIFF</b>		X					
<b>ASF</b>		X					
<b>AVI</b>					X		
<b>DivX</b>						X	
<b>FLV</b>							X
<b>MKV</b>							X <sup>3</sup>
<b>MPEG-4</b>		X			X		
<b>Ogg</b>			X	X			
<b>QT</b>							X
<b>RM</b>							X
<b>WAV</b>					X		

Table 2.3: Metadata Systems Overview

<sup>1</sup>Inside ASF container<sup>2</sup>Real Audio Metadata (RAM)<sup>3</sup>Matroska Tags [41]

Container Format	Video Formats	Audio Formats	Subtitles	Metadata	Menu Support
<b>AIFF</b>	-	PCM, AIFFC	-	-	-
<b>ASF</b>	any	any	X	X	-
<b>AVI</b>	any	any	X	X	-
<b>DivX</b>	MPEG-4	MP3, PCM, AC-3	X	-	X
<b>FLV</b>	Sorenson, VP6, Video, Screen H.264/MPEG-4 AVC	MP3, Nellymoser, ADPCM, Linear PCM, AAC, Speex	-	X	-
<b>MKV</b>	any	any	X	X	X
<b>MPEG-1</b>	MPEG-1	MPEG-1 Layers 1 - 3	-	-	-
<b>MPEG-4</b>	MPEG-2, MPEG-4, H.263, VC-1, Dirac	AAC, MP3, AC-3, ALS, SLS, Vorbis	X	X	X
<b>Ogg</b>	Theora, Dirac, OggUVS, MNG, Vfw, ...	Vorbis, FLAC, Speex, CELT, OggPCM, ACM, ...	X	X	-
<b>QT</b>	MPEG-2, MPEG-4, H.263, VC-1, Dirac, Sorenson	AAC, MP3, AC-3, ALS, SLS, Vorbis	X	X	X
<b>RM</b>	RealVideo 8, 9, 10	AAC, Cook Codec, Vorbis, RealAudio	X	-	-
<b>VOB</b>	MPEG-1, MPEG-2	AC-3, Linear PCM, DTS, MP3	X	-	X
<b>WAV</b>	-	PCM	-	-	-

Table 2.4: Media Container Formats Comparison

Format	Lossy Compression								
	AAC	AC-3	DTS	MP3	Nelly moser	Real Audio	Speex	Vorbis	WMA
ASF	X	X	X	X	-	X	X	-	X
AVI	X	X	X	X	-	X	X	-	X
DivX	X	X	-	X	-	-	-	-	-
FLV	X	-	-	X	X	-	X	-	-
MKV	X	X	X	X	X	X	X	X	X
MPEG-1	-	-	-	-	-	-	-	-	-
MPEG-4	X	X	X	-	-	-	-	X	-
Ogg	-	-	-	-	X	X	-	-	-
QT	X	X	X	-	-	-	-	-	-
RM	-	-	-	-	-	-	-	-	-
VOB	-	X	X	X	-	-	-	-	-
Format	Lossless Compression								
	ALAC	ALS	FLAC	PCM	SLS				
ASF	X	X	X	X	X			X	
AVI	X	X	X	X	X			X	
DivX	-	-	-	-	-			-	
FLV	-	-	-	-	X			-	
MKV	X	X	X	X	X			X	
MPEG-1	-	-	-	-	-			-	
MPEG-4	X	X	X	-	-			X	
Ogg	-	-	-	-	X			-	
QT	X	X	X	-	-			-	
RM	-	-	-	-	-			-	
VOB	-	X	X	X	-			-	

Table 2.5: Media Container Formats and Audio Codecs Support

Format	MPEG 1	MPEG 2	MPEG 4	WMV	Real Video	Theora	Sorenson	VP6
ASF	X	X	X	X	-	X	-	-
AVI	X	X	X	X	-	X	-	-
DivX	-	-	X	-	-	-	-	-
FLV	-	-	X	-	-	-	X	X
MKV	X	X	X	X	X	X	-	-
MPEG-1	X	-	-	-	-	-	-	-
MPEG-4	X	X	X	X	-	-	-	-
Ogg	X	X	X	X	-	X	-	-
QT	X	X	X	X	-	X	X	-
RM	-	-	-	-	X	-	-	-
VOB	X	X	-	-	-	-	-	-

Table 2.6: Media Container Formats and Video Codecs Support





*We get paid for bringing value to the market place.*

– Jim Rohn

# 3

## Market overview

### 3.1 Devices Overview

Because of the huge amount of devices available on the market there is no way to show them all. In this section only a small sample will be shown to demonstrate the market diversity and functionality differences or similarities.

Most of the devices are not pure DMS. It is common that a device combines more DLNA device classes together (e.g. DMP, DMC, etc.). But all listed devices contain at least the DMS DLNA class.

#### Boxie Box

*Boxie Box* is a *Digital Media Player* from D-LINK. This player contains High-Definition Multi-media Interface (HDMI) output which allows to play video content in FullHD (1080p). It supports following file formats:

**Video** AVI, DivX, MPEG-4, QuickTime, Xvid

**Audio** MP3, ACC, FLACK, OGG, WMA, WAV

**Image** BMP, GIF, JPEG, PNG, TIF

*Boxie Box* enables to play content from different sources. This can be external disk attached to build-in USB 2.0 port, internet stream video portals (e.g. Netflix, YouTube, Last.fm, Pandora, Flickr, Picasa) or SD card.

This player can be attached to the network via RJ-45 100Mbit Ethernet cable or WiFi 802.11 (incl. WEP, WPA, WPA2 encryption).

Vendor page: <http://www.dlink.com/boxee/>

#### Aspire Revo RV100

*Digital Media Player* with internal 1.5TB HDD (SATA II). Additional external HDD can be attached via USB 2.0 port. The device also

includes internal memory card reader and support of following file formats is available:

**Video** MPEG-1, MPEG-2, MPEG-4, H.264, XviD, WMV9, RV

**Audio** MP3, PCM, WMA, AAC, FLAC, WAV, OGG, AAC

**Image** JPEG, BMP, GIF, TIFF, PNG

This player can be attached to a TV via HDMI, YPbPr or A/V cable and to audio player via digital optical cable. The device can be connected to the network via RJ-45 100Mbit Ethernet or WiFi 802.11. Content from internet stream video portals (e.g. YouTube, Flickr, Picasa) can be also played by this device.

Because the important part of the physical devices for the purpose of this thesis is only the DMS software, comparison of the available software in more detail on the market will follow. The DMS software has the advantage over the physical devices that it can be used by more than one vendor in more than one device. Also the important parameters of a physical device for the purpose of this thesis are only parameters describing the DMS software.

## 3.2 Media Server Software Overview

There are many media servers on the market and each has different advantages and disadvantages. The media servers listed in Table 3.1 ‘Media servers: Media support comparison [25]’ on page 30, Table 3.2 ‘Media Servers: Operating Systems and License Comparison [25]’ on page 31, Table 3.3 ‘Media Servers: Vendor’s Product Pages[25]’ on page 32, Table 3.5 ‘Media Servers: Supported Audio Formats’ on page 34, Table 3.4 ‘Media Servers: Supported Video Formats’ on page 33 and Table 3.6 ‘Media Servers: Supported Image Formats’ on page 35 are in alphabetical order and all of them are UPnP<sup>1</sup> compliant. The tables are based on the *Comparison Chart* made by Robert Green[25]. All existing data were rechecked because of the quick development in this field.

In Table 3.1 ‘Media servers: Media support comparison [25]’ on page 30 playable media types are shown. The *Video* column indicates that the device is able to play at least one type of video *Container Format* and decode at least one video *Codec*. The *Music* column indicates that at least one type of music *Container Format* and one audio *Codec* can be streamed. The *Pictures* column indicates that at least one type of *Image Format* can be served. The *Transcoding* column indicates that the software can convert media content at least from one *Codec* to one another. The Operating System support and license is shown

<sup>1</sup>Universal Plug and Play permits networked devices in residential networks to seamlessly discover each other’s presence in the network and establish functional network services for data sharing, communications, and entertainment. This set of networking protocols is described in detail in Section 2.3 ‘Universal Plug and Play (UPnP)’ on page 10

in Table 3.2 'Media Servers: Operating Systems and License Comparison [25]' on page 31. Vendor's product web pages are listed in Table 3.3 'Media Servers: Vendor's Product Pages[25]' on page 32.

Main Audio/Video/Image formats supported by those servers are listed in Table 3.5 'Media Servers: Supported Audio Formats' on page 34, Table 3.4 'Media Servers: Supported Video Formats' on page 33 and Table 3.6 'Media Servers: Supported Image Formats' on page 35. If the particular server supports none of the media type formats than it is indicated by "*None*". If no source (i.e. documentation, sourcecode, webpage, etc.) providing the information was found then this fact is indicated by "*Information N/A*".

In some cases the support of formats is dependent on other libraries. In that case this fact is indicated by keyword "*Depends on LIB*", where **LIB** is the library the support depends on. If the list of supported formats is not complete then this fact is indicated by "*and more ...*". The list of supported formats can be incomplete because of two possible reasons. First the list of supported formats is too big so it was cut and only key formats are shown. Second, the support of a format can be extended by plugin or extension.

Most of the listed servers provide not only DMS but also DMP, DMR or DMC functionality.

### 3.3 Apple AirPlay

The *Apple AirPlay* is a replacement for the DLNA by APPLE. This is a new feature added to devices like *iPhone*, *iPad* and *iPod Touch* with *iOS* 4.2 and above. To play media using *AirPlay* you will need an *iOS* device and at least one of *AirPort Express*, *AppleTV* v2, third-party *AirPlay*-ready speaker, or a Bluetooth audio device [43]. Media controls on a Bluetooth audio device will allow to play, pause or skip the music.

*iTunes* can be also used with *AirPlay*. Though there are some differences between playing media from *iTunes* and playing media from *iOS* 4.2+ device. For example *iTunes* will allow you to send the media to multiple different devices and play for example the video on an *AirPlay* ready TV and audio stream on a HiFi set, while *iOS* devices are able to send the media just to one place at a time. However, *iTunes* are not able to use Bluetooth audio devices.

Now a closer look on *Apple AirPlay* devices and software follows:

**Airport Express** is a WiFi access point, which allows connection to USB printer, modem or LAN over 10/100BASE-T Ethernet cable, and/or speaker over 3.5mm Audio Jack. Speakers connected to *Airport Express* appear in the network and audio can be played on them by *iTunes* or *iOS* device[31].

<b>Name</b>	<b>Video</b>	<b>Music</b>	<b>Pictures</b>	<b>Transcoding</b>
Allegro Media Server	-	X	-	-
Cyber Media Gate (Java)	X	X	X	-
Cyberlink Digital Home Enabler Kit	X -	X -	X -	- -
Elgato Eyeconnect	X	X	X	-
Enna	X	X	X	-
Fuppes	X	X	X	X
Geexbox	X	X	X	-
GMediaServer	X	X	X	-
JRiver Media Center	X	X	X	-
MediaTomb	X	X	X	X
Mezzmo	X	X	X	X
MiniDLNA	X	X	X	-
MythTV	X	X	X	X
Nero MediaHome	X	X	X	X
Nullriver Medialink	X	X	X	X
On2Share	X	X	X	-
PS3 Media Server	X	X	X	X
Rhapsody	X	X	X	-
SimpleCenter Premium	-	X	-	X
Serviio	X	X	X	X
Tversity	X	X	X	X
TwonkyMedia	X	X	X	X
uShare	X	X	X	-
Wild Media Server	X	X	X	X
Winamp Remote	X	X	-	-
Windows Media Connect	X	X	X	X
Yahoo Music Jukebox	-	X	-	-

Table 3.1: Media servers: Media support comparison [25]

Name	Windows	MacOS	Linux	License
Allegro Media Server	X	X	-	Comercial
Cyber Media Gate (Java)	X	X	X	BSD
Cyberlink Digital Home Enabler Kit	X	-	-	Comercial
Elgato Eyeconnect	-	X	-	Comercial
Enna	-	-	X	LGPL, GPLv2
Fuppes	X	-	X	GPL
Geexbox	<i>full-featured OS</i>			GPL
GMediaServer	-	-	X	GPL
JRiver Media Center	X	-	-	Comercial
MediaTomb	-	X	X	GPL
Mezzmo	X	-	-	Comercial
MiniDLNA	-	-	X	BSD, GPL
MythTV	-	-	X	GPL
Nero MediaHome	X	-	-	Comercial
Nullriver Medialink	-	X	-	Comercial
On2Share	X	-	-	Comercial
PS3 Media Server	X	X	X	GPLv2
Rhapsody	X	X	X	Comercial
SimpleCenter Premium	X	-	-	Comercial
Serviio	X	X	X	Freeware
Tversity	X	-	-	Comercial
TwonkyMedia	X	X	X	Comercial
uShare	-	-	X	GPL
Wild Media Server	X	X	X	Comercial
Winamp Remote	X	-	-	Comercial
Windows Media Connect	X	-	-	Comercial
Yahoo Music Jukebox	X	-	-	Comercial

Table 3.2: Media Servers: Operating Systems and License Comparison [25]

Name	Product page
Allegro Media Server	<a href="http://www.allegrosoft.com/ams.html">http://www.allegrosoft.com/ams.html</a>
Cyber Media Gate (Java)	<a href="http://www.cybergarage.org/twiki/.../bin/view/Main/MediaGateForJava">http://www.cybergarage.org/twiki/.../bin/view/Main/MediaGateForJava</a>
Cyberlink Digital Home Enabler Kit	<a href="http://www.cyberlink.com/multi/.../products/main_111_ENU.html">http://www.cyberlink.com/multi/.../products/main_111_ENU.html</a>
Elgato Eyeconnect	<a href="http://www.elgato.com">http://www.elgato.com</a>
Enna	<a href="http://enna.geebox.org">http://enna.geebox.org</a>
Fuppes	<a href="http://fuppes.ulrich-voelkel.de">http://fuppes.ulrich-voelkel.de</a>
Geebox	<a href="http://www.geebox.org">http://www.geebox.org</a>
GMediaServer	<a href="http://www.gnu.org/software/gmediaserver">http://www.gnu.org/software/gmediaserver</a>
JRiver Media Center	<a href="http://www.jrmediacenter.com">http://www.jrmediacenter.com</a>
MediaTomb	<a href="http://mediatomb.cc">http://mediatomb.cc</a>
Mezzmo	<a href="http://www.conceiva.com/products/.../mezzmo/default.asp">http://www.conceiva.com/products/.../mezzmo/default.asp</a>
MiniDLNA	<a href="http://sourceforge.net/projects/minidlna">http://sourceforge.net/projects/minidlna</a>
MythTV	<a href="http://www.mythtv.org">http://www.mythtv.org</a>
Nero MediaHome	<a href="http://www.nero.com/enu/mediahome4-introduction.html">http://www.nero.com/enu/mediahome4-introduction.html</a>
Nullriver Medialink	<a href="http://www.nullriver.com/products/medialink">http://www.nullriver.com/products/medialink</a>
On2Share	–
PS3 Media Server	<a href="http://ps3mediaserver.blogspot.com">http://ps3mediaserver.blogspot.com</a>
Rhapsody	<a href="http://www.real.com/rhapsody">http://www.real.com/rhapsody</a>
SimpleCenter Premium	–
Serviio	<a href="http://www.serviio.org">http://www.serviio.org</a>
Tiversity	<a href="http://tiversity.com/home">http://tiversity.com/home</a>
TwonkyMedia	<a href="http://www.twonkyvision.de">http://www.twonkyvision.de</a>
uShare	<a href="http://ushare.geebox.org">http://ushare.geebox.org</a>
Wild Media Server	<a href="http://www.wildmediaserver.com">http://www.wildmediaserver.com</a>
Winamp Remote	<a href="https://winamp.orb.com/orb/html/login.html">https://winamp.orb.com/orb/html/login.html</a>
Windows Media Connect	<a href="http://www.microsoft.com/windows/.../windowsmedia/devices/wmconnect/default.aspx">http://www.microsoft.com/windows/.../windowsmedia/devices/wmconnect/default.aspx</a>
Yahoo Music Jukebox	<a href="http://new.music.yahoo.com/">http://new.music.yahoo.com/</a>

Table 3.3: Media Servers: Vendor's Product Pages[25]

Name	Video Formats
Allegro Media Server	<i>None</i>
Cyber Media Gate (Java)	<i>Information N/A</i>
Cyberlink Digital Home Enabler Kit	MPEG2 PS, MPEG2 TS, WMV, H.264 (MP4)
Elgato Eyeconnect	MPEG1, MPEG2, MPEG4, 3IVX, DIVX, XVID (incl. DIV3, DX50)
Enna	<i>Depends on <b>libplayer</b></i>
Fuppes	<i>Depends on <b>ffmpeg</b> instalation</i>
Geexbox	AVI, MPEG, DIVX, OGM <i>Depends on <b>libplayer</b></i>
GMediaServer	<i>Information N/A</i>
JRiver Media Center	VideoCD, AVI, MPEG, MPEG4, WMV, DIVX, DVD, QT, RV, SWE, TiVo, FLV
MediaTomb	<i>Depends on <b>ffmpeg</b> instalation</i>
Mezzmo	AVI, H264, SWE, MPEG, DV, RM, MPEG4, and more...
MiniDLNA	<i>Depends on <b>ffmpeg</b> instalation</i>
MythTV	<i>Big amount of formats.</i>
Nero MediaHome	<i>Information N/A</i>
Nullriver Medialink	MPEG1, MPEG2, MPEG4, H.264, DIVX, XVID, AVI, WMV, ASF, MOV, MKV, FLV
On2Share	<i>Information N/A</i>
PS3 Media Server	AVI, MPEG-4, TS, M2TS, MPEG-2, DVD, MKV, FLV, OGM, AVI
Rhapsody	<i>None</i>
SimpleCenter Premium	<i>Information N/A</i>
Serviio	MPEG1, MPEG2, MPEG4, AVI, VMW, MKV, FLV
Tversity	WMV, MJPEG, DVR-MS, AVI, DIVX(3,4,5,6), XVID, MPEG1, MPEG2, MPEG4, MOV, RT, FLV, MKV
TwonkyMedia	MPEG1, MPEG2, MPEG2-TS, MPEG4, AVI, WMV, VOB, DivX, 3GP, VDR, ASF, MPE, DVR-MS, XVID
uShare	ASF, AVI, DV, DIVX, WMV, MJPEG, MPEG1, MPEG2, MPEG4, DVD, MKV, MOV, QT, ...
Wild Media Server	3GP, ASF, AVI, DIVX, EVO, FLV, MPEG1, MPEG2, MPEG4, MKV, MOV, VDR, DVD, WMV, XVID, and more...
Winamp Remote	<i>Information N/A</i>
Windows Media Connect	<i>Information N/A</i>
Yahoo Music Jukebox	<i>None</i>

Table 3.4: Media Servers: Supported Video Formats

Name	Audio Formats
Allegro Media Server	AAC, MP3, WAV, AIF, <i>and more...</i>
Cyber Media Gate (Java)	<i>Information N/A</i>
Cyberlink Digital Home Enabler Kit	MP3, LPCM, WMA, AAC_ADTS_320 (3GP), AAC_ISO_320 (3GP)
Elgato Eyeconnect	AIF, MP1, MP2, MP3, WAV, AAC (unprotected), Ogg, WMA (unencrypted), PLS (Internet Radio)
Enna	<i>Depends on <b>libplayer</b></i>
Fuppes	MP2, MP3, WAV, PCM, OGG, MPC, FLAC, AAC
Geexbox	RM, MP3, OGG, CDA <i>Depends on <b>libplayer</b></i>
GMediaServer	AAC, RIFF WAVE, <i>and more...</i>
JRiver Media Center	APE, MPC, MP3, OGG, WAW, WMA, AAC, AIF, AU, AA, CDA, MIDI, RA, SHN, AC3, FLAC, DTS WAV
MediaTomb	MP3, FLAC, OGG, <i>and more...</i>
Mezzmo	AC3, AMR, ASE, AU, MP3, OGG, WAV, AAC, <i>and more...</i>
MiniDLNA	MP3, OGG, FLAC
MythTV	<i>Big amount of formats.</i>
Nero MediaHome	<i>Information N/A</i>
Nullriver Medialink	MP3, AAC, WMA, WAV
On2Share	<i>Information N/A</i>
PS3 Media Server	MP3, AC3, DTS, LPCM, OGG, FLAC, MPC, APE
Rhapsody	<i>Information N/A</i>
SimpleCenter Premium	<i>Information N/A</i>
Serviio	MP3, WMA, ACC, OGG, FLAC
Tversity	WMA, MP3, ACC, RT, OGG, FLAC, APE, MPC, WAV,
TwonkyMedia	MP3, WMA, WAV, 3GP, M4A, MP4, LPCM, OGG, FLAC, MP2, AC3, MPA, MP1, AIF
uShare	AAC, AC3, AIF, AU, SND, DTS, RMI, MP1, MP2, MP3, MP4, MPA, OGG, WAV, PCM, LPCM, WMA MKA, RM, FLAC, <i>and more...</i>
Wild Media Server	AC3, AMR, APE, DTS, FLAC, MP1, MP2, MP3, OGG, WAV, WMA, <i>and more...</i>
Winamp Remote	<i>Information N/A</i>
Windows Media Connect	<i>Information N/A</i>
Yahoo Music Jukebox	<i>Information N/A</i>

Table 3.5: Media Servers: Supported Audio Formats



<b>Name</b>	<b>Image Formats</b>
Allegro Media Server	<i>None</i>
Cyber Media Gate (Java)	<i>Information N/A</i>
Cyberlink Digital Home Enabler Kit	JPG, PNG, BMP
Elgato Eyeconnect	JPEG, BMP, GIF, PNG, TIFF
Enna	<i>Depends on <b>libplayer</b></i>
Fuppes	<i>Depends on <b>ImageMagick</b> instalation</i>
Geexbox	<i>Depends on <b>libplayer</b></i>
GMediaServer	<i>Information N/A</i>
JRiver Media Center	JPEG, TIFF, BMP, GIF, PNG, RAW
MediaTomb	JPEG, ...
Mezzmo	BMP, PNG, GIF, JPEG, TIFF
MiniDLNA	JPEG
MythTV	<i>Big amount of formats.</i>
Nero MediaHome	<i>Information N/A</i>
Nullriver Medialink	JPEG, PNG, GIF, TIFF, BMP, RAW, PDE, PS, EPS, TGA
On2Share	<i>Information N/A</i>
PS3 Media Server	JPG, PNG, GIF, TIFF
Rhapsody	<i>None</i>
SimpleCenter Premium	<i>Information N/A</i>
Serviio	JPEG, GIF, PNG
Tiversity	<i>Information N/A</i>
TwonkyMedia	JPEG, PNG, TIF, BMP
uShare	BMP, ICO, GIF, JPEG, PCD, PNG, PNM, PPM, QTI, QTIF, QTIF, TIFF
Wild Media Server	BMP, EPS, GIF, JPG, PCD, PCX, PIC, PNG, PSD, SCR, TGA, TIFF, ...
Winamp Remote	<i>Information N/A</i>
Windows Media Connect	<i>Information N/A</i>
Yahoo Music Jukebox	<i>None</i>

Table 3.6: Media Servers: Supported Image Formats

**Apple TV** is a box which can be connected to a High-definition television (HDTV) via HDMI cable. To the home network it can be connected via 10/100BASE-T Ethernet or Wi-Fi 802.11b, g or n wireless network. An additional optical audio output is present and the box is supplied with a simple remote control[31]. Supported media formats are shown in Table 3.7 ‘*Apple TV* supported formats [30]’:

<b>Video Formats</b>	H.264, MPEG 4, M-JPEG
<b>Audio Formats</b>	HE-AAC, AAC, MP3, AIFC, WAV
<b>Image Formats</b>	JPEG, GIF, TIFF

Table 3.7: *Apple TV* supported formats [30]

**iOS devices** starting with *iOS* version 4.2, such as *iPad*, *iPhone* (3GS or later) or *iPod touch* (2nd generation or later), are able to stream video, audio or images using *AirPlay* to *Apple TV*, *Airport Express* or compatible third-party device. There are still some limitations, e.g. that only one media can be played on only one *AirPlay* device[31].

**iTunes** can stream media files to *AirPlay* devices. Advantage over *iOS* devices is that media can be played on multiple *AirPlay* devices[31].

### 3.4 Summary

The market evolves very quickly. New servers are showing up. Existing ones are always adding new features, supporting more formats and/or standards. The purpose of this overview is to have a view on the current state of the market and decide which features/formats have to be supported by a DMS and what technologies can be used to achieve that approach.

Commonly used *Container Formats* and *Codecs* in the devices and software on the market were introduced. This gives the picture of the diversity of the software and the devices and accompanying difficulty of universal media support.

*The aim of marketing is to know and understand the customer so well the product or service fits him and sells itself.*

– Peter F. Drucker

# 4

## Home Environment Example

In this section an example of a home environment, which will be used as an example of typical run environment and for testing of the implementation in the Part II of this thesis, will be shown. The example should also outline the diversity and amount of device in a typical household.

In this example a four member household as a typical sample will be speculated, which gives us enough diversity in the device classes. The household consist of:

- Middle-aged man
- Middle-aged woman
- Adult in his twenties
- Adolescent

There are following rooms in the house: Livingroom, Kitchen, Room 1, Room 2, Workroom and Garage.

Devices available in the household are shown in Table 4.1 ‘Home Environment Example: Available Devices’ on page 39. The column *Class* indicates the DLNA class if any. Column *Location* indicates the location of the device in the house. If a device is connected to the DLNA network using wireless connection and its position is not fixed in one of the rooms, then this fact is indicated by "-". The column *Connection* indicates how the devices are connected together. The whole scheme of the household example can be seen in Figure 4.1 ‘Home Environment Example’ on page 40.

**Media Content in the Network** is following:

- **PC**
  - 1321 GB in 3747 video files
  - 54 GB in 11513 music files
  - 49 GB in 28125 image files
- **Notebook 1**

- 12 GB in 2058 music files
- **Notebook 2**
  - 5GB 2105 image files
  - 10GB 16 video files
- **NAS Storage 1**
  - 1116 GB in 2557 video files
- **NAS Storage 2**
  - 386 GB in 2018 video files
- **NAS Storage 3**
  - 25 GB in 4869 music files
- **USB External HDD 1**
  - 466 GB in 2690 video files
- **USB External HDD 2**
  - 855 GB in 1057 video files
- **USB External HDD 3**
  - 34 GB in 5569 music files
- **Cellphone 1**
  - 5 GB in 1258 music files
- **Cellphone 2**
  - 3 GB in 908 music files
- **Cellphone 3** - *N/A for the testing of the implementation*
- **Camera 1**
  - 3GB 1210 image files
- **Camera 2** - *N/A for the testing of the implementation*
- **Video Camera** - *N/A for the testing of the implementation*

Name	Class	Location	Connection
Central Server	DMS	–	LAN, WiFi
NAS Storage 1	–	–	<i>Central Server</i>
PC	DMS DMP	Workroom	LAN
NAS Storage 2	–	Workroom	<i>PC</i>
USB External HDD 1	–	Workroom	<i>PC</i>
Notebook 1	DMS DMP	Room 1	WiFi
NAS Storage 3	–	Room 1	<i>Notebook 1</i>
USB External HDD 2	–	Room 1	<i>Notebook 1</i>
Notebook 2	DMS DMP	Room 2	WiFi
USB External HDD 3	–	Workroom	<i>Notebook 2</i>
Kitchen player	DMP	Kitchen	WiFi
Garage player	DMP	Garage	WiFi
DLNA TV	DMP	Livingroom	LAN
DLNA Audio Receiver	DMP	Livingroom	LAN
DLNA Controller	DMC	Livingroom	WiFi
Cellphone 1	–	–	WiFi, USB, BT
Cellphone 2	–	–	WiFi, USB, BT
Cellphone 3	–	–	WiFi, USB, BT
PDA	M-DMU M-DMD	– –	WiFi, USB, BT
Camera 1	–	–	USB, SD Card
Camera 2	–	–	USB, SD Card
Video Camera	–	–	USB, SD Card, DVD

Table 4.1: Home Environment Example: Available Devices

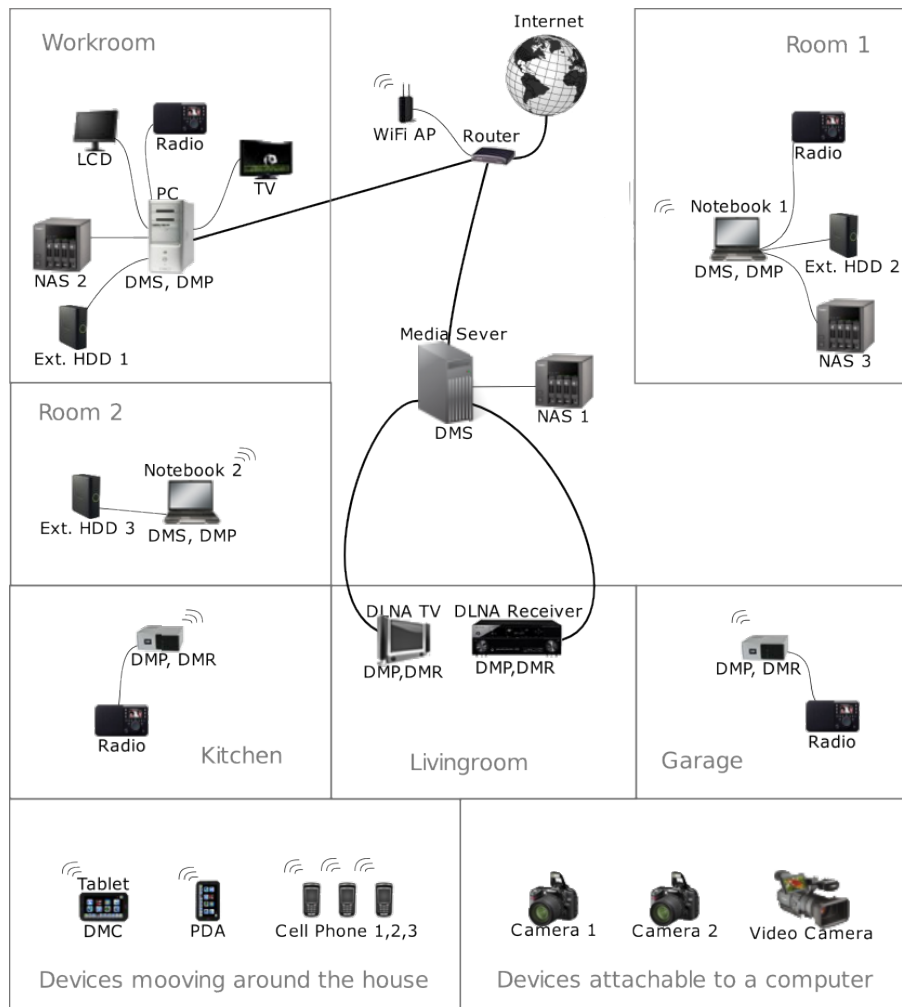


Figure 4.1: Home Environment Example

# Part II

## **Implementation**

This part contains description how the technologies, formats and techniques mentioned in Part I are used in implementation of the prototype software. Description of the architecture and design of the software developed as part of this thesis, used libraries and other 3rd party software used in the actual implementation is present.





*I think that you will see different types of content emerging, just the same as new media generates new content in the physical world. TV created new content, but it didn't mean that radio disappeared.*

– Donna Dubinsky

# 5

## Shared Media Types

This chapter describes selected libraries used to extract desired information from the different *Container Formats* and/or *Codecs* mentioned in Part I. Because of the huge number of the different *Container Formats*, *Codecs* and meta-data formats, only those actually used in the implementation of prototype software are mentioned. The goal of this thesis is not to try to support as much different formats as possible but to show an example of a DMA with such abilities and sketch the increasing implementation difficulty with increasing number of supported *Container Formats* and *Codecs*.

### 5.1 Audio

#### 5.1.1 ID3 Tag

The ID3 Tag is a data container within the audio file containing related text and/or graphical information about the audio such as artist name, song title, genre, year, cover image etc. It was created first for MP3 audio files. There are two versions of the ID3 Tag by now.

**ID3v1** is the first version, which allowed to store informations like song title, author, album, year and comment (see Figure 5.1 ‘Internal layout of an ID3v1 tagged file[27]’ on page 44 and Table 5.1 ‘Fields in ID3[27]’). In ID3v1 was not enough room for improvement or extension. The only improvement made to this version was ID3v1.1 which added an album track field containing the number of the song on the CD the music comes from (see Figure 5.2 ‘Internal layout of an ID3v1.1 tagged file[27]’ on page 44)[27].

**ID3v2** was introduced because of the insufficient room for improvement in ID3v1. ID3v2 is focused on flexibility and extensibility. It is a chunk of data before the audio data in the file and can contain one or more smaller chunks called frames. A frame in the ID3v2 tag can contain any kind of data, not just the ones mentioned in the ID3v1. Additionally it can contain a cover image, author

Field Name	Size
Song Title	30 characters
Artist	30 characters
Album	30 characters
Year	4 characters
Comment	30 characters
Genre	1 byte

Table 5.1: Fields in ID3[27].



Figure 5.1: Internal layout of an ID3v1 tagged file[27].



Figure 5.2: Internal layout of an ID3v1.1 tagged file[27].

or producer website URL, etc. (see Figure 5.3 ‘Internal layout of an ID3v2 tagged file[27]’)[27].

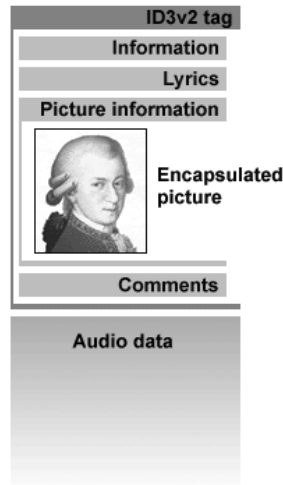


Figure 5.3: Internal layout of an ID3v2 tagged file[27].

**jID3lib** library[38] is used in the implementation of prototype software to support ID3 tags. This library covers following tags:

- ID3v1
- ID3v1.1
- ID3v2.2
- ID3v2.3
- ID3v2.4
- *Lyrics3v1 - not used*
- *Lyrics3v2 - not used*

### 5.1.2 Xiph Comments

*Xiph Comments* are embedded into *Ogg container*, basically used for *Ogg Vorbis*, *FLAC* and *Speex* and are meant for short text comments. The comment header is a list of vectors and each vector can have the maximum length of 32 bytes. Vendor vector is the only obligatory vector in the list. Comment headers are encoded as follows[62]:

- VENDOR\_LENGTH - *32bit unsigned integer*
- VENDOR\_STRING - *UTF-8 vector as vendor\_lenght octets*
- COMMENT\_LIST\_LENGTH - *32bit unsigned integer*
- COMMENT\_1\_LENGTH - *32bit unsigned integer*

- COMMENT\_1\_STRING - *UTF-8 vector as comment\_1\_lenght octets*
- COMMENT\_2\_LENGTH - *32bit unsigned integer*
- COMMENT\_2\_STRING - *UTF-8 vector as comment\_2\_lenght octets*
- ... -
- COMMENT\_N\_LENGTH - *32bit unsigned integer*
- COMMENT\_N\_STRING - *UTF-8 vector as comment\_N\_lenght octets*
- FRAMING\_BIT - *boolean*

The comment vector structure looks like[62]:

- `comment[0]="ARTIST=me"`
- `comment[1]="TITLE=The sound of Vorbis"`
- ...

This is the minimal list of standard field names[62]:

- TITLE
- VERSION
- ALBUM
- TRACKNUMBER
- ARTIST
- PERFORMER
- COPYRIGHT
- LICENSE
- ORGANIZATION
- DESCRIPTION
- GENRE
- DATE
- LOCATION
- CONTACT
- ISRC

The field names are not required to be unique within a comment header, i.e. more authors can be specified using multiple AUTHOR comment fields.

**J-Ogg** is a library that is used in the implementation to access the Xiph comments [39].

## 5.2 Image

### 5.2.1 EXIF & IFD

Image File Directory (IFD) is a recurring data structure within the EXIF. According to [17] IFD consists of 2Bytes indicating the number of fields, 12Bytes per

field and 4Bytes indicating the offset to the next IFD [13]. Every 12Byte field consist of:

**Bytes 0-1** Tag identified by 2Byte unique number.

**Bytes 2-3** Type identifying the value type:

- 1: BYTE** 8bit unsigned integer
- 2: ASCII** 8bit Byte containing one 7bit ASCII code. The final Byte is terminated with NULL.
- 3: SHORT** 16bit unsigned integer.
- 4: LONG** 32bit unsigned integer.
- 5: RATIONAL** 2 LONGs, first is numerator and second is denominator.
- 7: UNDEFINED** 8bit Byte.
- 9: SLONG** 32-bit (4Byte) signed integer.
- 10: SRATIONAL** 2 SLONGs, first is numerator and second is denominator.

**Bytes 4-7** Count (number) of values.

**Bytes 8-11** Value offset records the offset from the start of the header to the position where the value itself is recorded.

Some important IFD are listed in Table 5.2 ‘Selection of relevant TIFF Rev. 6.0 IFD Attributes [17]’ on page 48, Table 5.3 ‘Selection of relevant EXIF IFD Attributes [17]’ on page 48, Table 5.4 ‘Selection of relevant GPS IFD Attributes [17]’ on page 49.

In Section 2.7 ‘Image Formats’ on page 18, image formats which are considered in this thesis are discussed. A look on some libraries, which allow us to extract metadata from those files, follows.

**Sanselan** is a pure-Java image library which is used to retrieve additional metadata from images in the implementation of the prototype software. It supports following image formats (library supports more image formats than the ones listed here):

- TIFF
- JPEG/JFIF EXIF metadata
- JPEG/JFIF IPTC metadata

Although lot of image libraries are available out there, this library was chosen because it is a pure-java library, its portability and the number of supported image formats[51].

Tags related to the image data structure			
Tag ID	Field Name	Type	Count
256	ImageWidth	SHORT / LONG	1
257	ImageLength	SHORT / LONG	1
258	BitsPerSample	SHORT	3
259	Compression	SHORT	1
274	Orientation	SHORT	1
282	XResolution	RATIONAL	1
283	YResolution	RATIONAL	1
296	ResolutionUnit	SHORT	1
Other tags			
Tag ID	Field Name	Type	Count
306	DateTime	ASCII	20
270	ImageDescription	ASCII	<i>Any</i>
271	Make	ASCII	<i>Any</i>
272	Model	ASCII	<i>Any</i>
305	Software	ASCII	<i>Any</i>
315	Artist	ASCII	<i>Any</i>
33432	Copyright	ASCII	<i>Any</i>

Table 5.2: Selection of relevant TIFF Rev. 6.0 IFD Attributes [17]

Tags Relating to User Information			
Tag ID	Field Name	Type	Count
36864	ExifVersion	UNDEFINED	4
Tags Relating to User Information			
Tag ID	Field Name	Type	Count
37500	MakerNote	UNDEFINED	<i>Any</i>
37510	UserComment	UNDEFINED	<i>Any</i>
Tags Relating to Date and Time			
Tag ID	Field Name	Type	Count
36867	DateTimeOriginal	ASCII	20
36868	DateTimeDigitized	ASCII	20
Other Tags			
Tag ID	Field Name	Type	Count
42016	ImageUniqueID	ASCII	33
42032	CameraOwnerName	ASCII	<i>Any</i>
42033	BodySerialNumber	ASCII	<i>Any</i>

Table 5.3: Selection of relevant EXIF IFD Attributes [17]

Tags Relating to Global Positioning System (GPS)			
Tag ID	Field Name	Type	Count
0	GPSTimeStamp	RATIONAL	3
1	GPSLatitudeRef	ASCII	2
2	GPSLatitude	RATIONAL	3
3	GPSLongitudeRef	ASCII	2
4	GPSLongitude	RATIONAL	3
5	GPSAltitudeRef	BYTE	1
6	GPSAltitude	RATIONAL	1
7	GPSTimeStamp	RATIONAL	3
29	GPSTimeStamp	RATIONAL	3
	GPSDateStamp	ASCII	11

Table 5.4: Selection of relevant GPS IFD Attributes [17]

### 5.3 Video

Metadata extraction from video files was not implemented in the prototype software. Mostly every video *Container Format* and/or *Codec* introduces its own tagging system. Implementation of a library which would be able to extract desired information from at least one video *Container Format* and/or *Codec* is not in the scope of this thesis. However, the usage of such library would be similar to the usage of metadata extraction libraries for audio or image media. The architecture of the implementation allows the addition of such library. The proposal of this improvement is placed in Section 10.1 ‘Application Proposal’ on page 80.

The title possibly author and other information about the video media file are obtained only from the name of the file and the directory structure as described in Section 6.2 ‘Metadata Gathering’ on page 51.





*Life is made up of small pleasures. Happiness is made up of those tiny successes. The big ones come too infrequently. And if you don't collect all these tiny successes, the big ones don't really mean anything.*

– Norman Lear

# 6

## Collecting Media Files

The DMA should be able to collect media files such as audio, video and images from different sources and gather additional information about those files e.g. author, album, date, description etc.

### 6.1 Sources

Different sources can be used to obtain media files. In the first place UPnP Content Directory Service (CDS) will be used. This is also the way how aggregated media files are made available to other devices in the network. The source types (also called aggregation methods further in this document) included in the prototype are:

- UPnP ContentDirectory:1 Service
- LocalFS
- SMB Share<sup>1</sup>

### 6.2 Metadata Gathering

For each source type a different method for gathering metadata has to be used. The description of Metadata Gathering methods for the three implemented source types follows:

**UPnP Content Directory Services** requires no special method to be used as all needed metadata are available through the CDS properties.

**Local File System** needs usage of 3rd party libraries for metadata extraction (e.g. *ID3*, *EXIF*, etc.). Those libraries and their possibilities are described in Chapter 5 'Shared Media Types' on page 43.

---

<sup>1</sup>Samba Share is a reimplementatation of SMB/CIFS protocol for Unix-like systems. It allows Unix-like system to access Microsoft Windows shared files and printers.

MIME Type	File Extension	ContentDirectory Class
audio/mpeg	mp3	musicTrack
audio/x-wav	wav	audioItem
image/jpeg	jpe jpeg jpg	photo
image/png	png	imageItem
image/tiff	tif tiff	photo
video/mpeg	mp2 mpa mpe mpeg mpg mpv2	movie
video/quicktime	mov qt	movie
video/x-matroska	mkv	movie
video/x-ms-asf	asf asr asx	movie
video/x-msvideo	avi	movie

Table 6.1: MIME Type - File Extension - ContentDirectory Class Mapping

**Samba Share** and other similar media source types are having more restrictions. They have mostly access only to the name of the media file, extension, source type dependent path and size of the media file. To gather additional information using similar libraries like in LocalFS, the media files would need to be downloaded first to the LocalFS. This is not possible for all such files because of their size, the network capacity and the related processing time of gathering of those metadata. For the purpose of sources like SMB, a proposal of one possible solution will be introduced. This solution is implemented in the prototype software and should be considered as one possible solution of many. The solution is inspired by the directory structure used in CDS of Windows Media Player 11 [42].

The metadata will be based only on the available information mentioned above. In the first step the type (ContentDirectory Class) of the media will be recognized from the file extension. To this purpose `javax.activation.MimetypesFileTypeMap` class from the Java SE is used. A modified file with the extension to mime-type mapping is used. A subset considering the media mime-types and their mapping to extension and media types is shown in Table 6.1 'MIME Type - File Extension - ContentDirectory Class Mapping'.

The second step requires a directory structure compliance. This directory structure differs slightly for each media type. For audio files the directory expected structure is shown in Figure 6.1 'Audio Files Directory Structure Sample' and the directory structure for image files is shown in Figure 6.2 'Image Files Directory Structure Sample' on page 53. For video files no particular directory structure is needed and a different algorithm is used instead.

Video files are divided into 3 subtypes: **Movie** (*movie* UPnP ContentDirectory Class), **Serial Episode** (*videoItem* UPnP ContentDirectory Class) and **Music Video Clip** (*musicVideoClip* UPnP ContentDirectory Class). If the duration of the

```
[Root]
|- [Author]
|   |- [Album]
|       |- [Music Track]
|       | -...
|- [Album]
|   |- [Music Track]
|   | -...
```

Figure 6.1: Audio Files Directory Structure Sample

```
[Root]
|- [Album Part 1]
|   |- [Image]
|   | -...
|- [Album Part 1]
|   |- [Album Part 2]
|       |- [Image]
|       | -...
|- [Album Part 1]
|   |- [Album Part 2]
|       | -[...]
|           |- [Album Part X]
|               |- [Image]
|               | -...
```

Figure 6.2: Image Files Directory Structure Sample

footage is available the video file will be classified by following rules:

- < 15 minutes  $\Rightarrow$  Music Video Clip
- $\geq$  15 minutes and < 1 hour  $\Rightarrow$  Serial Episode
- $\geq$  1 hour  $\Rightarrow$  Movie

If the duration of the footage is not available but the file size is available than the video file will be classified by following rules:

- < 100 MB  $\Rightarrow$  Music Video Clip
- $\geq$  100 MB and < 500 MB  $\Rightarrow$  Serial Episode
- $\geq$  500 MB  $\Rightarrow$  Movie

If neither the duration of the footage nor the size of the file is available the video file will be classified as a **Movie**.

In case a video file is classified as **Music Video Clip** the directory structure for audio files shown in Figure 6.1 ‘Audio Files Directory Structure Sample’ on page 53 will be used for that video file for gathering additional metadata. If a video file is classified as a **Serial Episode** the parent directory name (if any) will be used as *channelName* property, which identifies the name of the serial in this case.

In all cases media types and the *title* property will be obtained from the file name by omission of the file extension.

The *date* property will be obtained from the file creation date, if this is available.

The *album* property for *photo* class will be constructed by concatenating of all *Album Parts* showed in Figure 6.2 ‘Image Files Directory Structure Sample’ on page 53 using "-" as a separator.

If any of the properties cannot be obtained it will not be set as all of them are not required by the UPnP ContentDirectory:1 specifications[36].

## 6.3 Media File Identification

The CDS presents the shared files in a tree structure where every file and container is identified by an *id* property, which must be unique with respect to the CDS. The tree structure is realized by using a *parentID* property, which is the *id* value of the parent container. The *parentID* of the root container must be set to the reserved value "-1"[36].

The CDS is following the directory structure used by Windows Media Player 11[42] shown in Table 6.2 ‘ContentDirectory Service Directory Structure[42]’. All other subcontainers (i.e. genre container, author container, album container, etc.), not listed in that table will have an auto-generated *id* property. The generated *id* property of such sub-containers will follow following rules:

ID	ParentID	Title	Child Type
0	-1	<b>Root</b>	object.container
1	0	<b>Music</b>	object.container
4	1	<b>All Music</b>	<i>object.item.audioItem</i>
5	1	<b>Genre</b>	object.container.genre.musicGenre
6	1	<b>Artist</b>	object.container.person.musicArtist
7	1	<b>Album</b>	object.container.album.musicAlbum
100	1	<b>Contributing Artists</b>	object.container.person.musicArtist
107	1	<b>Album Artist</b>	object.container.person.musicArtist
108	1	<b>Composer</b>	object.container.person.musicArtist
2	0	<b>Video</b>	object.container
8	2	<b>All Video</b>	<i>object.item.videoItem</i>
9	2	<b>Genre</b>	object.container.genre.videoGenre
A	2	<b>Actor</b>	object.container.person.movieActor
E	2	<b>Series</b>	object.container.album.videoAlbum
3	0	<b>Pictures</b>	object.container
B	3	<b>All Pictures</b>	<i>object.item.imageItem</i>
C	3	<b>Date Taken</b>	object.container.album.photoAlbum
D	3	<b>Albums</b>	object.container.album.photoAlbum
D2	3	<b>Keyword</b>	object.container.album.photoAlbum

Table 6.2: ContentDirectory Service Directory Structure[42]

- The length will be 8 characters.
- The *id* will contain only numbers and upper case letters.
- The *id* will be unique with respect to the CDS.

For every media file a Globally Unique Identifier (GUID) will be generated. The GUID has to be unique but may change for a same media file each time the application restarts. A problem regarding the uniqueness of the *id* has to be solved when a media file will appear in more than one container in the directory structure. This will actually happen quite often. An example could be an audio file which will appear in:

- Music/All Music
- Music/Artist/[Particular Artist]
- Music/Genre/[Particular Genre]
- ...

Every media file will first appear in the default container. For audio files the default container is "Music/All Music", for video files the default container is "Video/All Video" and for picture files the default container is "Pictures/All Pictures". The format of the *id* property of a media file is "{[GUID]}.0.[container\_id]", where *[container\_id]* is the *id* of the container the file is in. For example, an audio file in a default container (Music/All Music which has *id* = 4) the *id* will be "{[GUID]}.0.4". All other occurrences of the file in some other sub-containers will have the same GUID part but different *[container\_id]* part. In addition except the occurrence in the default container all other occurrences of the same media file will have specified a *refID* property which will contain the *id* of the occurrence of the media file in the default container.

## 6.4 Media Files Duplicates Identification

Next problem to be solved in this chapter is regarding duplicates of the media files in the network. Basically there are two potential chances how this can happen:

- The media file on one machine is accessible through two or more different aggregation methods e.g.: CDS, SMB, LocalFS etc.
- The same media file is present on two or more machines and accessible through at least one aggregation method on each machine.

For simplification a new term is defined: Gathering Source (GS). GS is a unique combination of a machine and aggregation method. For example:

- Machine A through SMB  
Machine A through CDS  
⇒ are **two different GS**

- Machine A through CDS  
Machine B through CDS  
⇒ are **two different GS**
- Machine A through LocalFS  
Machine B through CDS  
⇒ are **two different GS**

To distinguish that two media files obtained from two different GS are same is a difficult task. On a LocalFS a checksum could be calculated using Message-Digest algorithm 5 (MD5) or some other algorithm suitable for this purpose. But because we are expecting a huge number of files and those files can have size around several GB (e.g. HD movies), the computation of such checksum for all the files on the LocalFS could take quite a long time. For files on other devices in the network, using SMB, CDS, etc., this is even harder because the media file would have to be downloaded before calculating the checksum. Connection speed, network load, latency would have to be taken into consideration in that case.

A problem are also two media files with the same content but using different representation (e.g. two same music tracks stored using different codec). This can easily happen if one of the GS has transcoding abilities. In such case the calculation of a checksum would not help at all.

Those reasons lead to the decision to identify duplicate media files by matching ContentDirectory properties or metadata of those files. A small subset of properties for each media type (audio, video and pictures) is defined. Also a method for media files which are missing one, more or all of those properties is defined.

Property subsets on what the uniqueness identification is based on are:

**Music Track** - the *musicTrack* UPnP Class will be used[36].

- Title
- Album
- Alphabetically sorted Artists

**Music Video Clip** - the *musicVideoClip* UPnP Class will be used[36].

- Title
- Album
- Alphabetically sorted Artists

**Movie / Series Episode** - the *videoItem* UPnP Class will be used with all sub-classes except the *musicVideoClip* Class [36].

- Title

**Photo** - the *photo* UPnP Class will be used[36].

- Title
- Album

- Date

**Image** - the *imageItem* UPnP Class will be used[36].

- Title
- Date

If a property is missing it will be replaced by "Unknown [X]", where X is the name of the property which is missing. For example if artist property is missing, it will be replaced by "Unknown Artist" or if an album property is missing, it will be replaced by "Unknown Album", etc. One exception is the title property. It will be replaced with "Unknown Title [I]" where, I is a counter increased each time such replacement is used for a media file. Every media type (audio, video and pictures) has its own counter.

## 6.5 Sharing Media Information

Collected media are exposed to other devices in the network using a *CDS*. There are two main libraries for Java available on the internet: *CyberLink for Java*[16] and *Cling - Java/Android UPnP library*[40]. The reasons that *Cling* library was chosen were: documentation, examples and efficiency of the development of the *ContentDirectory* service prototype. The project contains reference open-source implementation of a renderer and open-source application called *Workbench*, which allow discovering UPnP devices in the network, showing their capabilities like actions and state variables values and allows invocation of service actions with arguments etc.

**Cling - Java/Android UPnP library** is a UPnP-compatible stack for Java EE, Java SE and Android[16]. The project is divided into two parts:

**Core** implements the UPnP Device Architecture 1.0[23].

**Support** classes for developing and controlling UPnP Services with **Core**. Simplification of working with UPnP media servers, renderers, etc. (*optional*)



*The white man knows how to make every-  
thing, but he does not know how to distribute  
it.*

– Sitting Bull

# 7

## Topology

One of the given task by QUANMAX company was to research about was the topology which should be used. There are theoretically two possible options: *Server-Based Topology* and *Distributed Topology*.

### 7.1 Server-Based Topology

The *Server-Based Topology* means that there has to be at least one server (DMS) available in the network (see Figure 7.1 ‘Server Based Topology’). All the devices like DMP, DMC, etc. can detect the server (using UPnP) and ask about available media (using CDS). If the wished media is found on the server it can be downloaded or streamed upon request to the rendering device.

The advantage of this topology is, that we have defined servers, which are taking care of the aggregation of the media files.

The disadvantage is that we have to have a server device present in the network what increase the costs.

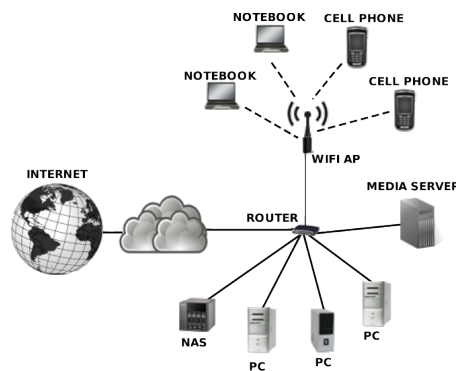


Figure 7.1: Server Based Topology

## 7.2 Distributed Topology

The idea of *Distributed Topology* is that the server device has not to be necessarily present. The devices in the network would keep track about other devices and media in the network on their own. This means that the software part of the DMA has to be present in such devices and this devices starts to behave as a DMS. Otherwise, DLNA devices such as DMP, DMC, etc. without the DMA capabilities will not find any content in the network since they expect the DMS to be present.

The *Distributed Topology* makes sense in the manner between two or more devices with DMA. Those DMA could exchange the media databases between each other, find duplicate media, redirect request from devices like DMP or DMC to the closest server etc. In some cases the aggregation process on one DMA could be improved by just using the data from another DMA if both have access to the same GS. Or if one DMA discovers any changes, it can populate those changes to others, so the other DMA devices does not have to repeat the job again.

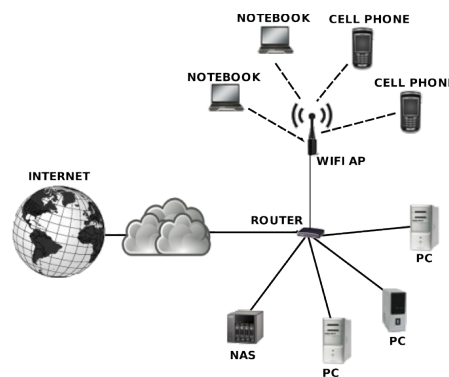


Figure 7.2: Distributed Topology

## 7.3 Conclusion

Since the whole concept of DLNA build on top of UPnP is a *Server Based Topology*, there is no choice to not support that. The additional *Distributed Topology* between two or more devices with DMA introduces some improvements. But those improvements will start to be interesting in bigger networks with lot of devices and media files. In those cases the *Distributed Topology* could bring speed improvements and more efficient usage of the network capacity.

Because the implementation of such topology is out of the time constrains of this thesis only a proposal was introduced in Section 10.1 'Application Proposal' on page 80.

*The process is always the same, it doesn't really matter what the object is, whether it's a prototype or whether it's for production.*

– Marc Newson

# 8

## Prototype Architecture

The application prototype architecture will be shown in this section. The UML class diagrams of the application are present in Appendix A 'Prototype UML Diagrams' on page 87. Public packages, classes and methods of the application are described in this section. The list is not complete and shows only packages, classes and methods, which are important to understand the architecture of the prototype software.

### 8.1 Application Model

This section will introduce and describe the application model of the prototype software. There are five main packages in the project:

- Aggregators package [at.quanmax.dlna.aggregators]
- GUI package [at.quanmax.dlna.gui]
- Models package [at.quanmax.dlna.media.models]
- Servers package [at.quanmax.dlna.servers]
- Libraries package [at.quanmax.libraries]

### 8.2 Aggregators Package

This package contains implementation of all the different aggregator types.

#### 8.2.1 AbstractAggregator Class

Is is the super class of all future aggregators. The purpose of this class is to guide the developer of future aggregators and to provide connection to the application. The UML of this class can be seen in Figure A.2 'Aggregators Class Diagram' on page 88. Description of public methods of this class follows:

**public final AbstractAggregator getInstance()** is a lazy singleton constructor. It returns the AbstractAggregator singleton instance.

**public void interrupt()** sends an interrupt signal to the aggregator thread. This will cause that the aggregator thread will stop as soon as it will be safe.

**public final boolean isInterrupted()** checks if the interrupt signal was already sent to the aggregator thread. This doesn't mean that the aggregator thread has already been stopped. It will run until it will be safe to stop it without leaving anything unwanted behind. This method returns a boolean value telling whether the signal was already sent.

**public void run()** is a method, which will be called after the aggregator thread is instantiated. It should be overwritten in the subclasses and it should implement the aggregator type specific code.

### 8.2.2 Aggregator Class

This class contains the main method which starts the application GUI (see Section 8.4 'GUI Package' on page 65). The UML diagram of this package can be seen in Figure A.2 'Aggregators Class Diagram' on page 88.

**public static void main(String[] args)** is the main method. Directly after the GUI is started, the existence of previous snapshots of the aggregated data will be checked. If such a snapshot is found it will be loaded. To load the snapshots the `at.quanmax.libraries.FileSystemExporter` class is used (see Section 8.7.2 'FileSystemExporter Class' on page 71).

**public static void start(List<AbstractAggregator> aggregators)** will start all aggregators listed in `aggregators` parameter. Every aggregator will be started in a separate thread. The given aggregators will be saved to the *aggregator list* and an *aggregation indicator* indicating the running state of the aggregation will be switched on. This indicator forbids to start the aggregation more than once at the time.

**public static void stop()** interrupts all running aggregator threads and waits till all of them are stopped. After all the aggregator threads are stopped it will clear the *aggregator list* and switch off the *aggregation indicator*. This will allow to start the aggregation again using the **start** method.

### 8.2.3 ContentDirectoryAggregator Class

This class is an aggregator for UPnP ContentDirectory:1 sources. It uses the *Cling Library*[16] and inherits the `at.quanmax.dlna.aggregatorsAbstractAggregator` class described in

Section 8.2.1 ‘AbstractAggregator Class’ on page 61. This aggregator allows to discover and aggregate media files from UPnP devices in the network. Its UML diagram is shown in Figure A.2 ‘Aggregators Class Diagram’ on page 88.

#### 8.2.4 FileSystemDirectoryAggregator Class

This class inherits the `at.quantmax.dlna.aggregators.AbstractAggregator` class described in Section 8.2.1 ‘AbstractAggregator Class’ on page 61 and implements support of aggregating media files from directory structure on a local filesystem. The list of directories which have to be searched through is retrieved from `at.quantmax.dlna.gui.PreferencesDialog#getFSDirs` described in Section 8.4.2 ‘PreferencesDialog Class’ on page 66. The UML diagram of this class is shown in Figure A.2 ‘Aggregators Class Diagram’ on page 88.

#### 8.2.5 SambaAggregator Class

This class implements aggregation of media files from Samba shares in the network using the `jCIFS` library[46]. It also inherits the `at.quantmax.dlna.aggregators.AbstractAggregator` class described in Section 8.2.1 ‘AbstractAggregator Class’ on page 61 and its UML diagram can be seen in Figure A.2 ‘Aggregators Class Diagram’ on page 88. It allows to aggregate media files from both public and private shares. This aggregator can be setup using the `at.quantmax.dlna.gui.PreferencesDialog` described in Section 8.4.2 ‘PreferencesDialog Class’ on page 66. The implementation is too complex for one class and because of that an additional `at.quantmax.dlna.aggregators.samba` subpackage described in Section 8.3 ‘Samba Subpackage’ on page 63 was added.

**public static ArrayList<String> getWorkGroups()** gets all available workgroups in the network. First it will try to query the network with “`smb://`” query to obtain all the available groups then the retrieved list will be added to user defined workgroups (see `at.quantmax.dlna.gui.PreferencesDialog#getSMBWorkgroup()` described in Section 8.4.2 ‘PreferencesDialog Class’ on page 66). The delimiter for more workgroups is comma, semicolon or space.

### 8.3 Samba Subpackage

Is a package used by the `at.quantmax.dlna.aggregators.SambaAggregator`, described in Section 8.2.5 ‘SambaAggregator Class’ on page 63, to aggregate media files from Samba Shares.

### 8.3.1 SambaConnection Class

Is a class that performs the actual search through samba sources. Can perform both public (anonymous) and user defined (password based authentications) searches. The UML diagram of this class and its dependencies can be seen in Figure A.2 ‘Aggregators Class Diagram’ on page 88.

**public String getIP()** is a method that returns the IP address of the host.

**public boolean isReachable()** determines whether the host is reachable or not. Returns TRUE if the host is reachable, otherwise it returns FALSE.

**public void start()** method will start the search in a separate thread. While the search thread is running this method will not do anything. The search has to be stopped by calling the **stop** method before starting a new search, by calling this method, again.

**public static void startPublicSearch()** will start the public search in a separate thread. Search through all machines in all available workgroups (see `at.quanmax.dlna.aggregators.SambaAggregator#getWorkGroups()` described in Section 8.2.5 ‘SambaAggregator Class’ on page 63) will be performed.

Only one public search thread can be started at the time. If another call of this method will be performed before the currently running search thread stops it will be remembered and a new public search will be executed after the current will finish.

**public void stop()** is a method that will stop currently running search if any. If no search is running this method will not do anything.

### 8.3.2 SambaTicker Class

Samba ticker is a samba observer. If any samba server appears/becomes available in the network this class will perform the search on it using `at.quanmax.dlna.aggregators.samba.SambaConnection` class described in Section 8.3.1 ‘SambaConnection Class’ on page 64.

Also if any of the sources disappears this class is responsible for stopping the search and removing the device from the tree.

The UML diagram of this class can be seen in Figure A.2 ‘Aggregators Class Diagram’ on page 88.

**public static void restart()** is a method that restarts currently running ticker thread if any or starts a new one if no one was running before.

**public static void start()** will start the ticker thread. Only one ticker thread can be running at the time.

**public static void stop()** stops currently running ticker thread if any.

## 8.4 GUI Package

The package at `.quanmax.dlna.gui` contains the GUI part of the application. There are two dialogs whose UML diagrams are shown in Figure A.1 ‘GUI Dialogs Diagram’ on page 87. Some screenshots of the application can be seen in Chapter 9 ‘Application User Guide’ on page 73.

### 8.4.1 AggregatorGUI Class

The main dialog *AggregatorGUI*, whose UML diagram can be seen in Figure A.1 ‘GUI Dialogs Diagram’ on page 87 and Figure A.3 ‘Aggregator GUI Detail Diagram’ on page 89, is the control point of the application. From it all the different aggregators (*CDS Aggregator*, *LocalFS Aggregator* and *SMB Aggregator*) or *CDS Server* can be started. Also the whole file structure of the aggregated media files can be browsed and details of the aggregated files can be displayed.

**public static void disableDialog(String message)** method will disable the control elements (i.e. buttons, checkboxes, ...) of the dialog and the text message, given in the message parameter, will be shown in the status bar. The message describes the reason of disabling the dialog.

**public static void display()** method shows the Aggregator GUI dialog using a separate AWT thread.

**public static void enableDialog()** will enable the control elements (i.e. buttons, checkboxes, ...) of the dialog. It will also clear the status message in the status bar.

**public static AggregatorGUI getInstance()** method is a lazy singleton constructor of the Aggregator GUI dialog class.

**public void setInfoText(String text)** sets the info text on the right side of the dialog to the value of the text parameter.

**public static void setStatus(String message)** method sets the status message in the status bar to value given by the message parameter.

**public static void setStatusDone()** sets the status message in the status bar to "Done." value.

**public static void tick(Short id)** pokes a ticker defined by id identification number. This will cause that the corresponding indicator in the bottom area of the dialog will blink.

### 8.4.2 PreferencesDialog Class

The second dialog called *PreferencesDialog* (UML diagram can be seen in Figure A.1 'GUI Dialogs Diagram' on page 87) is used to configure the different parts of the application (i.e. the different aggregators). Each aggregator has a separate tab in this dialog and only aggregators which need to be configured have a tab present.

**public static String[] getFSDirs()** gets the list of local file system directories to search through and returns them as an array.

**public static PreferencesDialog getInstance()** is a lazy singleton constructor of the PreferencesDialog class.

**public static boolean getSMBSearchPublic()** determines if Samba public search should be performed or not. This method will return TRUE if public search should be performed and FALSE if not.

**public static String[] getSMBShares()** method returns an array of samba share URIs which should be searched through.

**public static String getSMBWorkgroup()** gets Samba default workgroup(s). More than one workgroup can be specified by creating a symbol separated list. The separator symbol between two workgroups can be comma, semicolon or space. From this method the whole *string* as entered is returned. The separation of the workgroups is done in `at.quanmax.dlna.aggregators.SambaAggregator#getWorkGroups()` defined in Section 8.2.5 'SambaAggregator Class' on page 63.



## 8.5 Media Models Package

The package `at.quanmax.dlna.media.models` contains object representation of all the different media file types and factories. The structure is based on the UPnP ContentDirectory:1 AV Class Definition[36] and the UML diagram can be seen in Figure A.4 ‘Media Models Class Diagram’ on page 90. Detailed UML diagram of the classes in this package are shown in Figure A.5 ‘AbstractObject Class Detail’ on page 91, Figure A.6 ‘DeviceObject Class Detail’ on page 92, Figure A.7 ‘ContentObject Class Detail’ on page 92, Figure A.8 ‘MediaObject Class Detail’ on page 93, Figure A.9 ‘AudioObject Class Detail’ on page 94, Figure A.10 ‘ImageObject Class Detail’ on page 95 and Figure A.11 ‘VideoObject Class Detail’ on page 96.

### 8.5.1 AbstractObject Class

This is the top-level abstraction of every object in the system file structure (see Figure A.5 ‘AbstractObject Class Detail’ on page 91). It implements the storage back-end and common methods. This class has two successors: *DeviceObject* (described in Section 8.5.2 ‘DeviceObject Class’) and *ContentObject* (described in Section 8.5.3 ‘ContentObject Class’).

### 8.5.2 DeviceObject Class

Is a class that is meant as the root for different devices (see Figure A.6 ‘DeviceObject Class Detail’ on page 92). By different devices is not meant a different physical machine but different server application on different or same machines. For example a *DeviceObject* will be created for a SMB share on PC1, UPnP CDS on PC1 and also for SMB on PC2, etc. Basically a *DeviceObject* will be created for each GS as described in Section 6.4 ‘Media Files Duplicates Identification’ on page 56.

### 8.5.3 ContentObject Class

It is a superclass for all files on a device. Those files could possibly be also non-media files in the future (see Figure A.7 ‘ContentObject Class Detail’ on page 92). The only successor of this class in the current implementation is *MediaObject* class described in Section 8.5.4 ‘MediaObject Class’.

### 8.5.4 MediaObject Class

It is a superclass of all media files on a device (see Figure A.8 ‘MediaObject Class Detail’ on page 93). The successor are: *AudioObject* class (described in Section 8.5.5 ‘AudioObject Class’), *ImageObject* (described in Section 8.5.6 ‘ImageObject Class’ on page 68) and *VideoObject* (described in

Section 8.5.7 ‘VideoObject Class’ on page 68) as the three main media types. The structure starting here is based on the UPnP ContentDirectory:1 AV Class Definition[36].

### 8.5.5 AudioObject Class

Is a class meant for audio media files. The successors are: *AudioBookObject*, *AudioBroadcastObject* and *MusicTrackObject* (see Figure A.9 ‘AudioObject Class Detail’ on page 94). All the successor classes are implemented in the prototype but only *MusicTrackObject* is actually used.

### 8.5.6 ImageObject Class

It is a class meant for image files. This class has only one successor in the current implementation: *PhotoObject* (see Figure A.10 ‘ImageObject Class Detail’ on page 95).

### 8.5.7 VideoObject Class

Is a class handling video files. The successors are: *VideoBroadcastObject*, *MusicVideoClipObject* and *MovieObject* (see Figure A.11 ‘VideoObject Class Detail’ on page 96). *VideoBroadcastObject* is implemented but not actually used in the current implementation.

### 8.5.8 ItemFactory Class

Factory class to create `org.teleal.cling.support.model.item.Item`[16]. The UML diagrams of this class can be seen in Figure A.12 ‘Factories’ on page 97 and Figure A.13 ‘Item Factory Detail’ on page 98.

**public static Item get(MediaObject object)** method, which constructs `org.teleal.cling.support.model.item.Item` object from `at.quanmax.dlna.media.models.MediaObject` or subclasses. Param `object` specifies the `at.quanmax.dlna.media.models.MediaObject` to construct the `org.teleal.cling.support.model.item.Item` from.

### 8.5.9 ObjectFactory Class

This class is a factory class that is able to construct `at.quanmax.dlna.media.models.MediaObject` from different objects. The UML diagrams can be seen in Figure A.12 ‘Factories’ on page 97 and Figure A.14 ‘Object Factory Detail’ on page 99.

**public static MediaObject getMediaObject(Item item)** method constructs a `at.quanmax.dlna.media.models.MediaObject` from given `org.teleal.cling.support.model.item.Item`. The parameter `item` specifies the `org.teleal.cling.support.model.item.Item` to construct the `at.quanmax.dlna.media.models.MediaObject` from.

**public static MediaObject getMediaObject(FileObject fo)** constructs a `at.quanmax.dlna.media.models.MediaObject` from given `org.openide.filesystems.FileObject`. The parameter `fo` specifies the `org.openide.filesystems.FileObject` to construct the `at.quanmax.dlna.media.models.MediaObject` from. The method returns the constructed `at.quanmax.dlna.media.models.MediaObject`.

**public static AbstractObject getObject(FileObject fo)** is a method that construct a `at.quanmax.dlna.media.models.AbstractObject` from `org.openide.filesystems.FileObject`. The parameter `fo` specified the `org.openide.filesystems.FileObject` to construct the `at.quanmax.dlna.media.models.AbstractObject` from. It returns the constructed `at.quanmax.dlna.media.models.AbstractObject`.

## 8.6 Servers Package

The package `at.quanmax.dlna.servers` contains classes necessary to run the server part of the aggregator. This part consists of three classes: *ContentDirectoryCollector* (described in Section 8.6.1 ‘ContentDirectoryCollector Class’ on page 69), *ContentDirectoryServer* (described in Section 8.6.2 ‘ContentDirectoryServer’ on page 70) and *ContentDirectoryService* (described in Section 8.6.3 ‘ContentDirectoryServiceClass’ on page 71).

### 8.6.1 ContentDirectoryCollector Class

This class collects all the aggregated files from all the different sources/devices, sorts them (by artist, album, etc.) and stores them in the memory so the *ContentDirectoryService* (described in Section 8.6.3 ‘ContentDirectoryServiceClass’ on page 71) can use them.

The *ContentDirectoryCollector* has 2 separated storages where it can put the collected files: *Storage A* and *Storage B*. Only one is published to the *ContentDirectoryService* and the second one is ready if another collecting job is performed. Till the collecting is finished *ContentDirectoryService* has access to the old collection and after the collecting job finishes the storages will switch roles. Lets show this on an example:

In the beginning both of the storages are empty and the *Empty Storage Indicator* is set to TRUE. The *Current Storage Pointer* points to *Storage A*.

When the first collecting job starts it uses the *Storage A*. But no data are available to the application because the *Empty Storage Indicator* is still set to TRUE. After the first collecting job finishes the *Empty Storage Indicator* will be set to FALSE and remains in that state until the application exits.

When a next collecting job is started it will use the *Storage B*. The application has access to *Storage A* till the collecting job is running. When the collecting job finishes it will switch the *Current Storage Pointer* to *Storage B*.

Next time the collecting job will use *Storage A* again and after it finishes the *Current Storage Pointer* will be switched to point to *Storage A*. This way it will go on every time a collecting job will be started.

**public static void check()** method performs media files check. This will start a separate thread, which will collect all the media files to a separate storage (the other one than the one currently selected) in memory.

After the collection is done the actually used storage and the one where the new data were collected will switch.

**public static Container get()** returns the root container from the current storage.

**public static Container get(String id)** method, which returns the container from the current storage identified by the *id* parameter.

**public static boolean isLoaded()** checks if any collection is loaded into the storage. This will return TRUE after the first collection is loaded.

### 8.6.2 ContentDirectoryServer

In this class the whole UPnP stack will be setup. The service will announce itself in the network, start the first collecting job using *ContentDirectoryCollector* and start the *ContentDirectoryService*. Any UPnP compliant device in the network will be able to browse through the collected media files.

**public static UDN getUDN()** method, which returns the Unique Device Name (UDN) of this UPnP ContentDirectory server.

**public static boolean isInterrupted()** checks if the server thread was interrupted. This does not have to mean that the server thread was stopped, just that it will stop as soon as it will be safe. The method returns TRUE if the server thread was already interrupted.

**public static boolean isRunning()** method check if the server thread is running or not. Even if the server thread was interrupted this can return TRUE, meaning that it was not safe yet to stop the thread yet. If the server thread is still running, this method returns TRUE, otherwise it returns FALSE.

**public static void start()** starts the server thread. Only one server thread can be started at the time. Calling this method when one server thread is already running will not do anything.

**public static void stop()** tries to stop the currently running server thread, if any, by interrupting it. The server thread will not be stopped immediately.

### 8.6.3 ContentDirectoryServiceClass

This class allows to make the collected media files available to the network.

**public BrowseResult browse(String objectID, BrowseFlag browseFlag, String filter, long firstResult, long maxResults, SortCriterion[] orderBy)** implements browsing of the aggregated content. This is a required action defined by ContentDirectory:1 [16].

**public BrowseResult search(String containerId, String searchCriteria, String filter, long firstResult, long maxResults, SortCriterion[] orderBy)** method implements searching through the aggregated content [16].

## 8.7 Libraries package

### 8.7.1 DMLib Class

The DLNA Aggregator library contains project specific code shared across the whole project.

### 8.7.2 FileSystemExporter Class

The *FileSystemExporter* class allows to export `org.openide.filesystems.FileObject` into a XML file and compress the output XML file using ZIP compression.

**public static void exportFileToXML(String filename)** exports all found devices with aggregated media files into a XML file. The name of the file is specified by `filename` parameter.

**public static void exportFileToZip(String filename)** method, which exports all found devices with aggregated media files into a XML file compressed using ZIP compression. Parameter `filename` specifies the name of the file.

**public static void importXMLFile(String filename)** method, which imports a XML file into the tree of devices and aggregated media files. The name of the file to import is specified by `filename` parameter.

**public static void importZIPFile(String filename)** imports a XML compressed using ZIP compression into the tree of devices and aggregated media files. Parameter `filename` specifies the name of the file to import.

**public static void importSnapshots()** imports all existing snapshots of the devices and aggregated media files. Snapshots are XML files beginning with “snapshot\_” in the path of the application.

## 8.8 Testing of the Application

The application was tested using data, device arrangement and media files distribution described in Chapter 4 ‘Home Environment Example’ on page 37.

*Logic is a poor guide compared with custom.*

– Winston Churchill

# 9

## Application User Guide

This section will describe how the application works from the user's point of view. When the application is started the application main window, shown in Figure 9.1 'Application: Main Window', will appear. All the aggregators (i.e. *UPnP ContentDirectory Aggregator*, *LocalFS Aggregator* and *Samba Share Aggregator*) and the *CDS Server* can be controlled from this window.

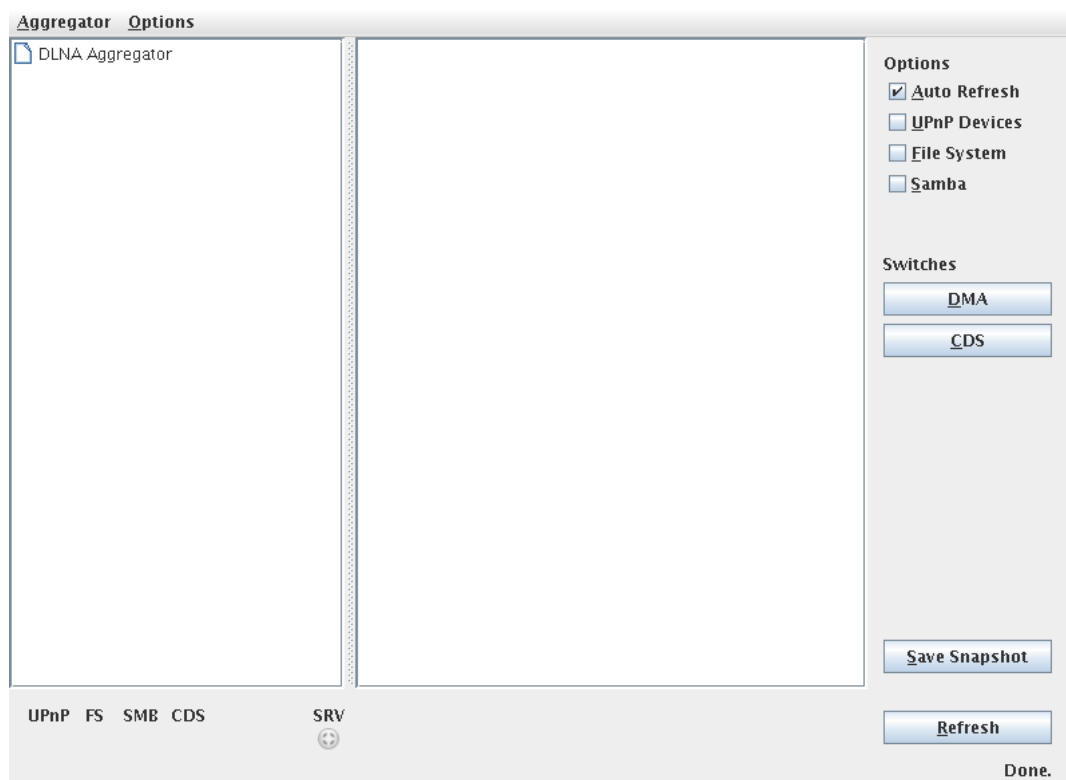


Figure 9.1: Application: Main Window

## 9.1 Aggregation Process

To start aggregating the media files from the network, first the aggregators, which will be used, have to be selected by checking the corresponding checkbox in the **Option** area or from the **Options** menu.

After the desired aggregators are selected the DMA has to be switched on by clicking the **DMA** button in the **Switches** area or in the **Aggregator** menu. The **DMA** button in the **Switches** area will remain pushed as an indication that the DMA is running. The aggregation can be interrupted any time by clicking on the **DMA** button in the **Switches** area or in the **Aggregator** menu again. Clicking on the button will start/stop the aggregation process again. Already aggregated files will not be deleted before but overridden, if necessary, during the aggregation process. Only files which do not exist anymore will be removed from the list of aggregated files.

During the aggregation process the devices and media files will appear/disappear in the list as they will be connected/disconnected from the network. This behaviour can be turned off by unchecking the **Auto Refresh** checkbox in the **Options** area or in the **Options** menu. A manual refresh of the whole tree can be performed by clicking on the **Refresh** button.

In the bottom left part of the main window (see Figure 9.1 ‘Application: Main Window’ on page 73), following aggregator indicator labels can be seen:

**UPnP** for UPnP ContentDirectory aggregator

**FS** for local file system aggregator

**SMB** for Samba aggregator

## 9.2 Aggregators Configuration

If the **File System** aggregator in the *Options* menu was checked the aggregation process will also search through LocalFS. Directories which should be searched on the LocalFS can be defined in the Preferences in the **Local File System** tab (see Figure 9.2 ‘Preferences: Local File System’ on page 75).

**Samba Shares** tab in the Preferences window shown in the Figure 9.3 ‘Preferences: Samba Shares’ allows to setup the *SMB Aggregator*. It allows to set the default workgroup, to check whether also public folders should be searched and to set-up User Defined Shares.

## 9.3 Browsing Aggregated Media

The directory tree can be browsed in the left part of the window and the details of a device or media file will appear in the right side as soon as the corresponding



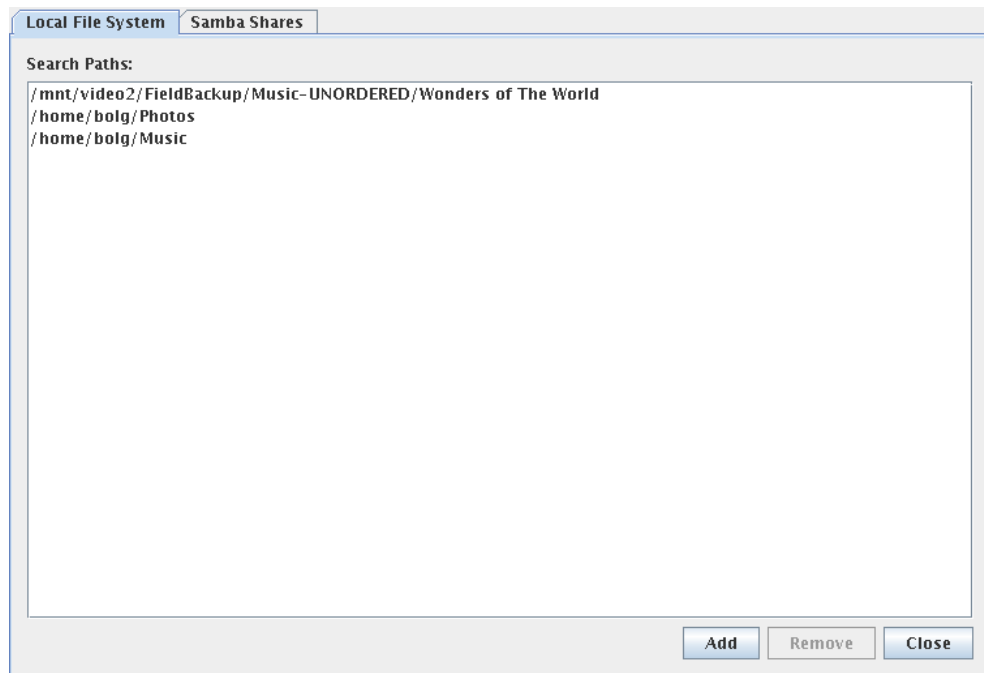


Figure 9.2: Preferences: Local File System

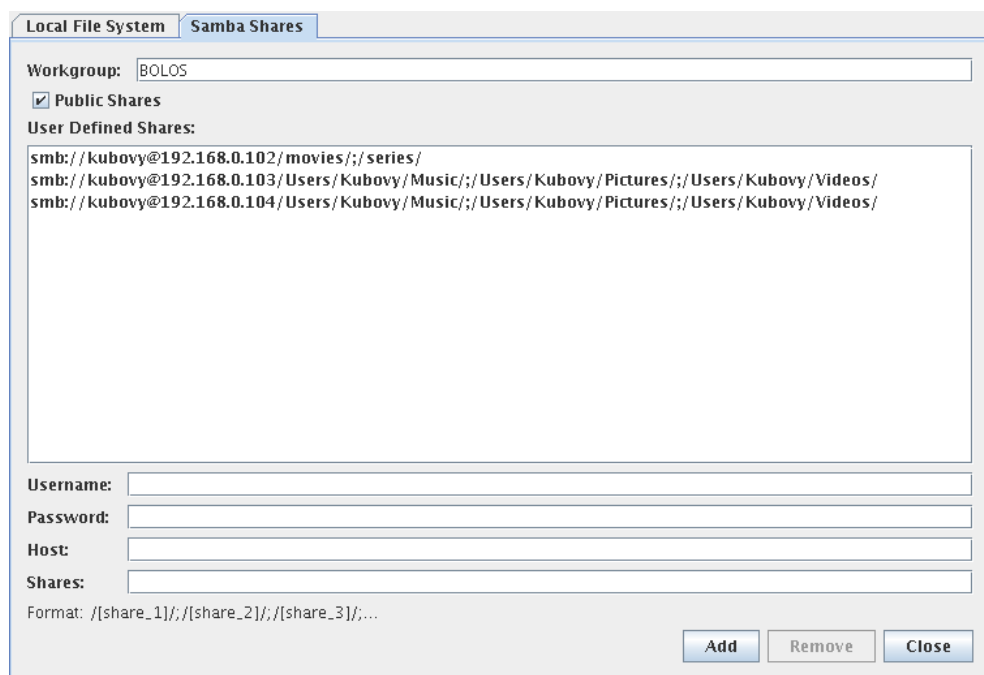


Figure 9.3: Preferences: Samba Shares

item in the tree is selected (see Figure 9.5 ‘Application: Media File Detail’ on page 78).

Each device will have its own folder in the root of the tree structure. The name of the folder will be the name of the device and its type in parentheses. For example a UPnP ContentDirectory Service on a computer called QUANMAXPC will appear as “QUANMAXPC: Username: (UPnP Device)”, samba share on the same computer will appear as “QUANMAXPC (Samba Share)”, local filesystem will appear as "localhost (Local Machine)", etc. (see Figure 9.4 ‘Application: Aggregation in progress’ on page 77).

Every device will contain following folders in the tree structure:

**AUDIO** containing all audio files on that device.

**IMAGE** containing all image files on that device.

**VIDEO** containing all video files on that device.

Each device can also contain additional folder for the purpose of showing the original file structure on the concrete device. Those folders are just for debugging, development and presentation purposes. The *ContentDirectory* server is not taking them into consideration.

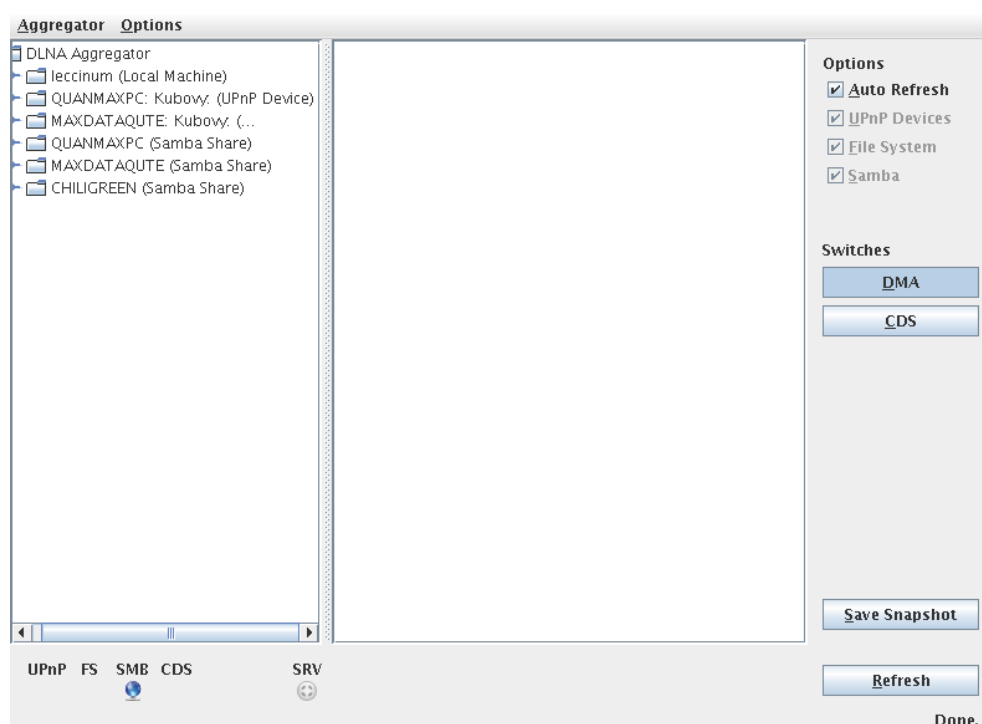


Figure 9.4: Application: Aggregation in progress

## 9.4 Making Media available to the Network

To make the aggregated files available to other devices in the network, the UPnP ContentDirectory Server has to be started. This can be done by clicking the **CDS** button in the **Switches** area or in the **Aggregator** menu. The **CDS** button will remain pressed while the server is running. The server status indicator can be also seen in the bottom left area of the main window in the indicator's area under the label **SRV** (see Figure 9.1 'Application: Main Window' on page 73). The activity

indicator of the UPnP ContentDirectory Server is in the same are under the **CDS** label.

## 9.5 Saving and Loading Snapshots

The aggregated files can be saved to a snapshot any time by clicking the **Save Snapshot** button. If a snapshot is saved it will be automatically loaded before the application starts next time. This allows a faster startup (we don't need to wait till all the files are aggregated again the first time). The files loaded from such snapshot will be overwritten by the next aggregation process as soon as the particular aggregator comes to the particular file.

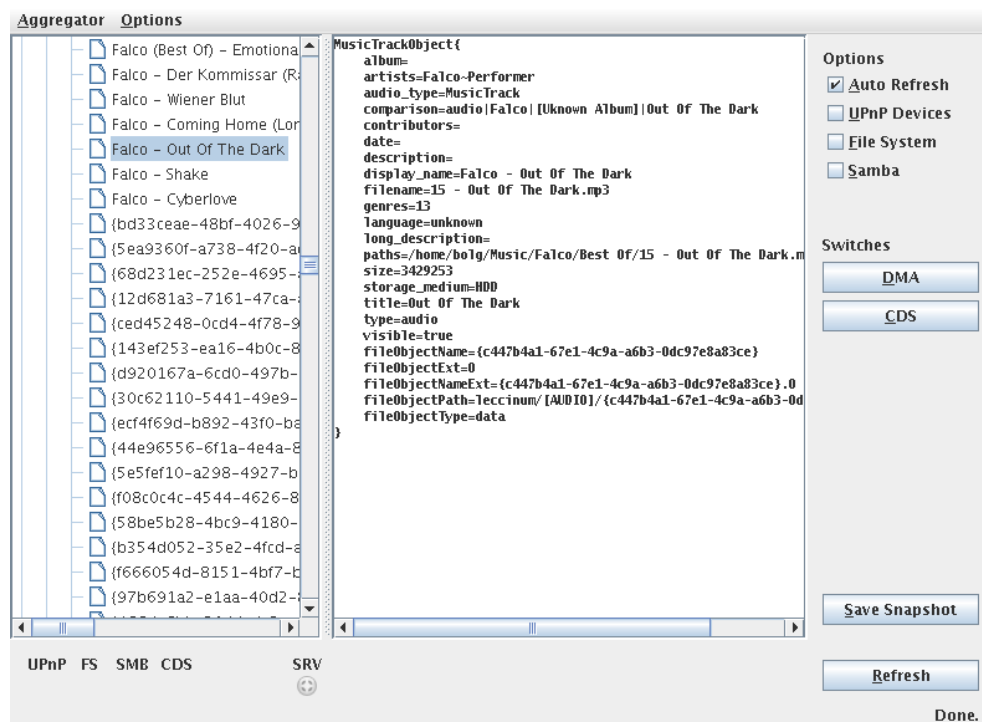


Figure 9.5: Application: Media File Detail

*A conclusion is the place where you got tired thinking.*

– Martin H. Fischer

# 10

## Conclusion

Research about technologies related to the problem of aggregating media files in the network, gathering additional information such as author, album, date, etc. from them and make them available to other DLNA (UPnP) devices in the network was done in this thesis. A market overview was made about existing devices, software and similar solutions of this problem. A *Home Environment Example* was introduced as a specimen of a typical usage environment of such a problem and its solution.

In the second part of this thesis a solution was introduced to solve problems such as collecting media files from different sources in a network, gathering and normalizing metadata/tags from media files, identification of those files, media duplicate identification and elimination and disposing collected files further to the network.

Two usage cases were introduced based on different topology: *Server Based* and *Distributed* topology, where the distributed topology allows saving network connection resources in larger networks. It also allows to complete certain media files with additional metadata gathered by one DMA but not by another.

A prototype software was implemented to try out the solution and demonstrate the possibilities of such implementation. The advantage of such solution is that the files are aggregated from different sources like LocalFS, SMB, CDS, etc. The introduced architecture of the implementation allows to add more types of such sources (e.g. File Transfer Protocol (FTP), OBject EXchange (OBEX), etc.) easily. The implemented types are aggregating media files from the particular sources properly.

Also the media files duplicates identification, described in Section 6.4 ‘Media Files Duplicates Identification’ on page 56, was tested and works across the different implemented aggregators.

The software prototype met all goals defined in Section 1.1 ‘Goal’ on page 4 and all tasks listed in Section 1.2 ‘Tasks’ on page 4.

## 10.1 Application Proposal

Further to this thesis I would like to propose some improvements and/or ideas, which are out the scope of this thesis:

- Implementation of more aggregator's types such as Bluetooth (OBEX), (S)FTP, WebDAV, etc.
- Dividing aggregator types into two different categories. *Stable* and *Mobile*, where the second one will include aggregators with low bandwidth and/or unreliable access (i.e. OBEX) and the aggregated files will be also downloaded to the DMS device, where the DMA is running. The assumption in this case is that the files available through such devices will be relatively small, can be downloaded in reasonable time and there is enough space on the DMS device with the DMA to store them.
- Implementation of *Distributed Topology* support between two or more devices with DMA as described in Chapter 7 'Topology' on page 59.
- Implementation of metadata extraction from not yet supported *Codecs* and *File Formats*.

# 11

## References

- [1] Multimedia programming interface and data. Specification, IBM Corporation; Microsoft Corporation, August 1991. version 1.0. 14
- [2] ISO 11172-1:1993. *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbit/s - Part 1: Systems*. ISO, Geneva, Switzerland, 1993. 15, 16
- [3] ISO 11172-2:2006. *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbit/s - Part 2: Video*. ISO, Geneva, Switzerland, 2006. 17
- [4] ISO 11172-3:2009. *Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5Mbit/s - Part 3: Audio*. ISO, Geneva, Switzerland, 2009. 17
- [5] ISO 13818-1:2000. *Information technology - Generic coding of moving pictures and associated audio information - Part 1: Systems*. ISO, Geneva, Switzerland, 2000. 16
- [6] ISO 13818-2:2004. *Information technology - Generic coding of moving pictures and associated audio information - Part 2: Video*. ISO, Geneva, Switzerland, 2004. 18
- [7] ISO 13818-7:2004. *Information technology - Generic coding of moving pictures and associated audio information - Part 7: Advanced Audio Coding (AAC)*. ISO, Geneva, Switzerland, 2004. 16
- [8] ISO 14496-12:2008. *Information technology - Coding of audio-visual objects - Part 12: ISO base media file format*. ISO, Geneva, Switzerland, 2008. 15
- [9] ISO 14496-14:2003. *Information technology - Coding of audio-visual objects - Part 14: MP4 file format*. ISO, Geneva, Switzerland, 2003. 15
- [10] ISO 14496-3:2009. *Information technology - Coding of audio-visual objects - Part 3: Audio*. ISO, Geneva, Switzerland, 2009. 16, 17
- [11] Digital Living Network Alliance. Dlna overview and vision whitepaper 2007. Whitepater, Digital Living Network Alliance, 2007. 5, 6, 7, 8, 9, 21, 101, 103

- [12] Apev2 specification. [http://wiki.hydrogenaudio.org/index.php?title=APEv2\\_specification](http://wiki.hydrogenaudio.org/index.php?title=APEv2_specification). Last modified: 24th February 2008, Read: 20th February 2011. 20
- [13] Adobe Developers Association. Tiff. Specification, Adobe Systems Incorporated, 1585 Charleston Road, P.O. Box 7900, Mountain View, CA, June 1992. Revision 6.0. 19, 20, 47
- [14] E. Hoffert; M. Krueger; L. Mighdoll; M. Mills; J. Cohen; D. Camplejohn; B. Leak; J. Batson; D. Van Brink; D. Blackketter; M. Arent; R. Williams; C. Thorman; M. Yawitz; K. Doyle; S. Callahan. Quicktime: an extensible standard for digital multimedia. In *Compcon Spring '92. Thirty-Seventh IEEE Computer Society International Conference, Digest of Papers*, pages 15–20, San Francisco, CA, February 1992. Apple Comput. Inc., Cupertino, CA. 15
- [15] L. Buerk (Microsoft Corporation); J. Moonen (Philips Electronics); D. Sather (Microsoft Corporation); J. Kai Fu (Pioneer Research Center). Avtransport:1 service template version 1.01. Standardized dcp, UPnP Forum, June 2002. 12
- [16] Cling - java/android upnp library and tools. <http://teleal.org/projects/cling/>. Read: 18th March 2011. 58, 62, 68, 71
- [17] Standardization Committee. Exchangeable image file format for digital still cameras: Exif version 2.3. Specification, Camera & Imaging Products Association, 1585 Charleston Road, P.O. Box 7900, Mountain View, CA, April 2010. CIPA DC-008-Translation-2010. 19, 46, 47, 48, 49, 103
- [18] Microsoft Corporation. Advanced systems format (asf) specification. Specification, Microsoft Corporation, June 2010. Revision 01/20/05. 14
- [19] Divx website. <http://www.divx.com/>, 2011. Read: 15th March 2011. 14
- [20] S. Chan (Microsoft); A. Dara-Abrams (Sony Electronics); M. Dawson (OpenGlobe); J. Kai Fu (Pioneer); F. Matsubara (Mitsubishi Electric); J. Moonen (Philips Electronics); Y. Rasheed (Intel); D. Sather (Microsoft); E. Shteyn (Philips Electronics). Connectionmanager:1 service template version 1.01. Standardized dcp, UPnP Forum, June 2002. 12
- [21] Ffmpeg. <http://www.ffmpeg.org>. Read: 30th March 2011. 15
- [22] UPnP Forum. Upnp device architecture 1.0. Specification, UPnP Forum, April 2008. Revision 04/24/2008. 10, 11, 103
- [23] UPnP Forum. Upnp device architecture 1.0. Standardized dcp, UPnP Forum, October 2008. 58



- [24] J. Sánchez; M. C. R. Gancedo. Sixth framework programme priority 2 – search on audio-visual content using peer-to-peer information retrieval (safir). Prototype, Information Society Technologies, January 2009. Contract no.: 45128. 8
- [25] R. Green. How to choose a dlina media server for windows, mac os x or linux. <http://www.rbgrn.net/content/21-how-to-choose-dlina-media-server-windows-mac-os-x-or-linux>, August 2007. Read: 4th February 2011. 28, 29, 30, 31, 32, 103
- [26] E. Hamilton. Jpeg file interchange format. Technical report, C-Cube Microsystems, 1778 McCarthy Blvd. Milpitas, CA 95035, September 1992. version 1.02. 19
- [27] Id3 website. <http://www.id3.org>. Read: 28th March 2011. 43, 44, 45, 101, 103
- [28] Adobe Systems Incorporated. Xmp specification. Specification, September 2005. 14, 20
- [29] Adobe Systems Incorporated. Adobe flash file format specification. Specification, 345 Park Avenue, San Jose, California 95110, USA, August 2010. version 10.1. 14
- [30] Apple Incorporated. Apple tv. [http://store.apple.com/us/browse/home/shop\\_ipod/family/apple\\_tv?mco=MTY3ODQ5OTY](http://store.apple.com/us/browse/home/shop_ipod/family/apple_tv?mco=MTY3ODQ5OTY). Read: 18th February 2011. 36, 103
- [31] Apple Incorporated. Using airplay. <http://support.apple.com/kb/HT4437>. Read: 18th February 2011. 29, 36
- [32] CompuServe Incorporated. Graphics interchange format(sm) version 89a. <http://www.w3.org/Graphics/GIF/spec-gif89a.txt>, July 1990. Read: 2nd April 2011. 19
- [33] DTS Incorporated. Dts-hd audio - customer white paper for blu-ray disc and hd dvd applications. White paper, November 2006. 16
- [34] J. Ritchie (Intel). Mediarenderer:1 device template version 1.01. Standardized dcp, UPnP Forum, June 2002. 12
- [35] J. Ritchie (Intel). Mediaserver:1 device template version 1.01. Standardized dcp, UPnP Forum, June 2002. 12
- [36] K. Debique (Microsoft); T. Igarashi (Sony); S. Kou (Sony); J. Moonen (Philips); J. Ritchie (Intel); G. Schults (HP); M. Walker (Intel). Contentdirectory:1 service template version 1.01. Standardized dcp, UPnP Forum, June 2002. 12, 54, 57, 58, 67, 68

- [37] S. Kou (Sony); T. Matsui (EIZO NANA0); J. Moonen(Philips); Y. Rasheed (Intel); J. Ritchie (Intel). Renderingcontrol:1 service template version 1.01. Standardized dcp, UPnP Forum, June 2002. 12
- [38] Java id3 tag library. <http://javamusictag.sourceforge.net/>. Read: 29th March 2011. 45
- [39] J-ogg website. <http://www.j-ogg.de>. Read: 28th March 2011. 46
- [40] S. Konno. Cyberlink for java. <http://www.cybergarage.org/twiki/bin/view/Main/CyberLinkForJava/>, February 2011. Read: 18th March 2011. 58
- [41] Matroska. Tag specifications. <http://www.matroska.org/technical/specs/tagging/index.html>. Read: 6th March 2011. 22
- [42] Building a network device compatible with microsoft windows media player 11. Specification, Microsoft Corporation, November 2007. 52, 54, 55, 103
- [43] Ware Ground IT News. The hidden secrets of apple's airplay. [http://www.wareground.com/articles/the\\_hidden\\_secrets\\_of\\_apples\\_airplay](http://www.wareground.com/articles/the_hidden_secrets_of_apples_airplay), November 2010. Read: 19th February 2011. 29
- [44] M. Nilsson. Id3 tag version 2.3.0. Informal Standard id3v2.3, February 1999. 20
- [45] Advanced of truemotion vp6 technology. White Paper v1.2, On2 Technologies, Inc., February 2004. 18
- [46] The JCIFS Project. The java cifs client library. <http://jcifs.samba.org/>, October 2010. Read: 6th February 2011. 63
- [47] A. Puri. Mpeg-4: an object-based multimedia coding standard supporting mobile applications. *Mobile Networks and Applications - Special issue: mobile multimedia communications*, 3(1):5–20, June 1998. 18
- [48] Realnetworks website. <http://www.realnetworks.com>. Read: 27. March 2011. 15, 17
- [49] Realvideo 10. Technical overview, RealNetworks, 2003. version 1.0. 18
- [50] J. Ribas-Corbera. Windows media 9 series. Technical report, Microsoft Windows Digital Media Division, January 2003. 17, 18
- [51] Sanselan: a pure-java image library. <http://commons.apache.org/sanselan/>. Read: 29th March 2011. 47
- [52] Matroška. Codec specs. <http://www.matroska.org/technical/specs/codecid/index.html>. Read: 25th February 2011. 15

- [53] S. Tarkoma. *Mobile Middleware: Architecture, Patterns and Practice*. John Wiley & Sons, Ltd, Chichester, UK, March 2009. 10
- [54] Information technology - digital compression and coding of continuous-tone still images - requirements and guidelines. Recommendation T.81, The International Telegraph and Telephone Consultative Committee (CCITT), September 1992. 19
- [55] Digital audio compression standard. Standard A/52:2010, United States Advanced Television Systems Committee (ATSC), 1776 K Street, N.W., Suite 200, Washington, D.C. 20006, November 2010. 16
- [56] Universal plug and play forum website. <http://www.upnp.org>, 2011. Read: 10th March 2010. 12
- [57] J. M. Valin. The speex codec manual - version 1.2 beta 2. <http://www.speex.org/docs/manual/speex-manual/manual.html>, 2007. Read: 3rd March 2011. 17
- [58] M. Adler; T. Boutell; J. Bowler; C. Brunschen; A. M. Costello; L. Daniel; A. Dilger; O. Fromme; J. Gailly; C. Herborth; A. Jakulin; N. Kettler; T. Lane; A. Lehmann; C. Lilley; D. Martindale; O. Mortensen; K. S. Pickens; R. P. Poole; G. Randers-Pehrson; G. Roelofs; W. Schaik; G. Schalnat; P. Schmidt; M. Stokes; T. Wegner; J. Wohl. Portable network graphics (png) specification (second edition). <http://www.w3.org/TR/2003/REC-PNG-20031110>, November 2003. Read: 2nd April 2011. 19
- [59] Xiph.Org. Ogg vorbis i format specification: comment field and header specification. <http://www.xiph.org/vorbis/doc/v-comment.html>. Read: 5th March 2011. 20
- [60] Xiph.Org. Flac free lossless audio codec documentation. <http://flac.sourceforge.net/documentation.html>, 2008. Read: 2nd March 2011. 17
- [61] Xiph.Org. The ogg container format. <http://www.xiph.org/ogg/>, 2008. 15
- [62] Vorbis i. Specification, Xiph.org Foundation, February 2010. 17, 45, 46
- [63] Theora specification. Specification, Xiph.Org Foundation, September 2010. 18
- [64] R. Zimmermann. Streaming of divx avi movies. In *SAC '03 Proceedings of the 2003 ACM symposium on Applied computing*, pages 979–982. University of Southern California - Department of Computer Science, 2003. 14





## Prototype UML Diagrams

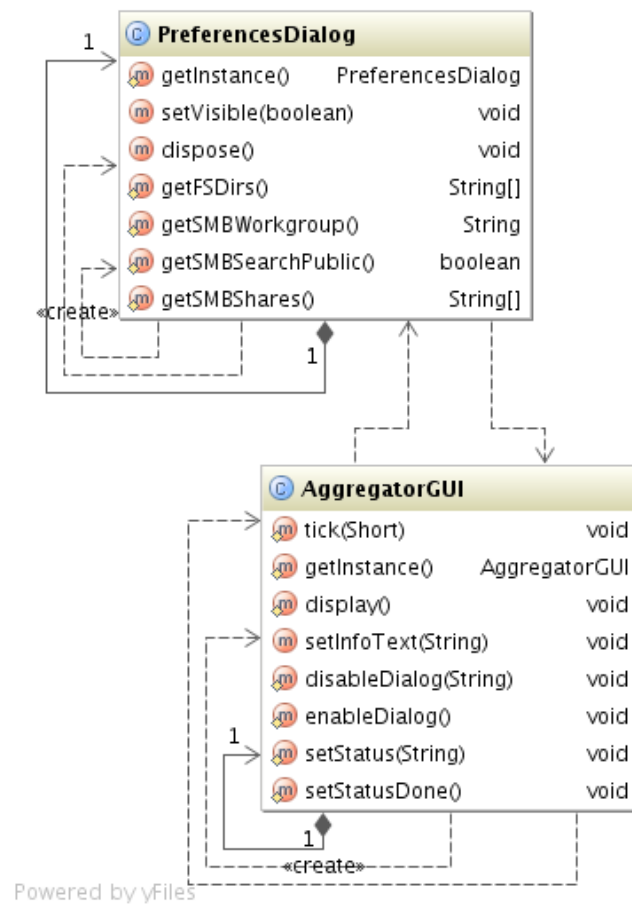
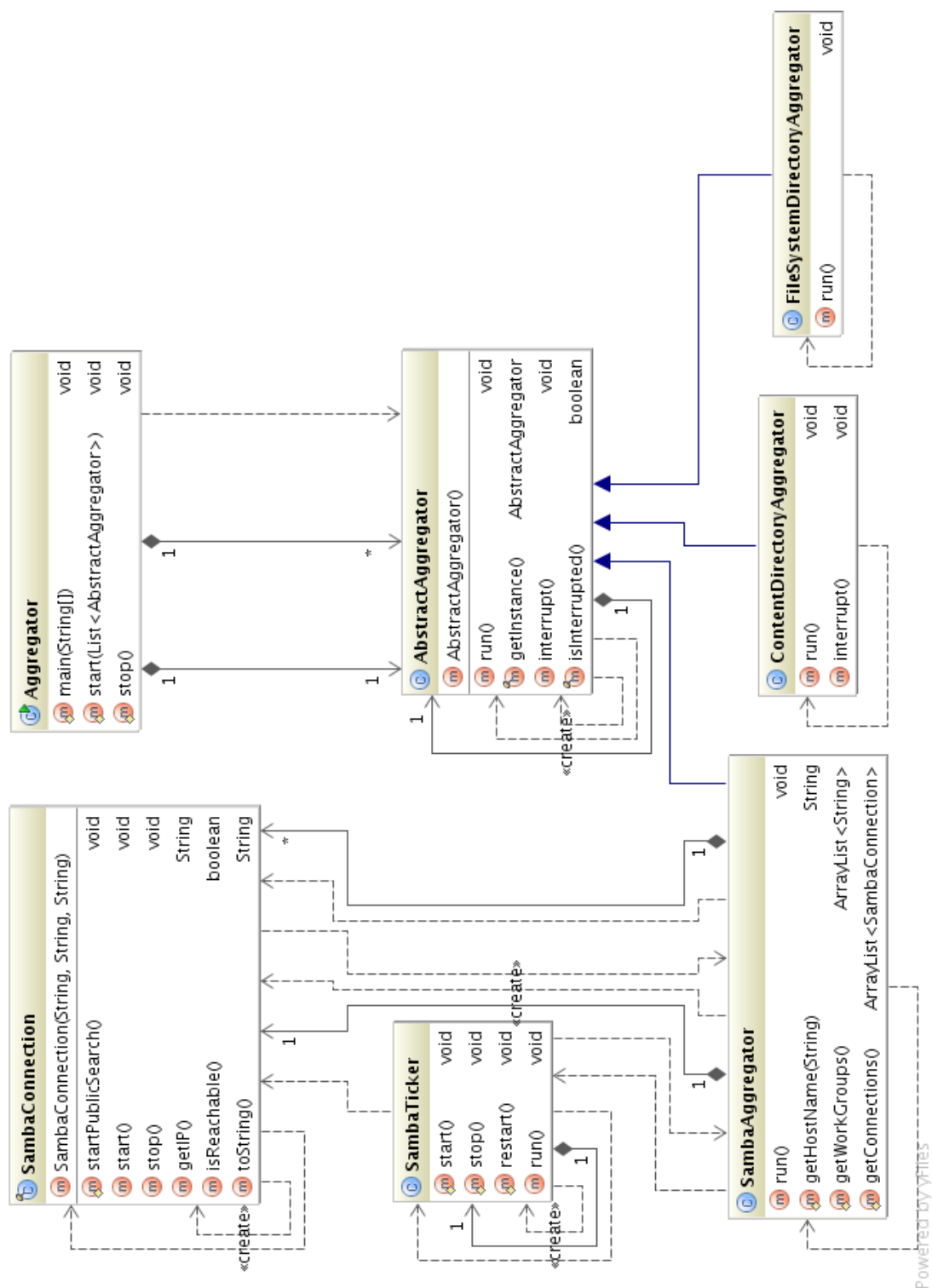


Figure A.1: GUI Dialogs Diagram





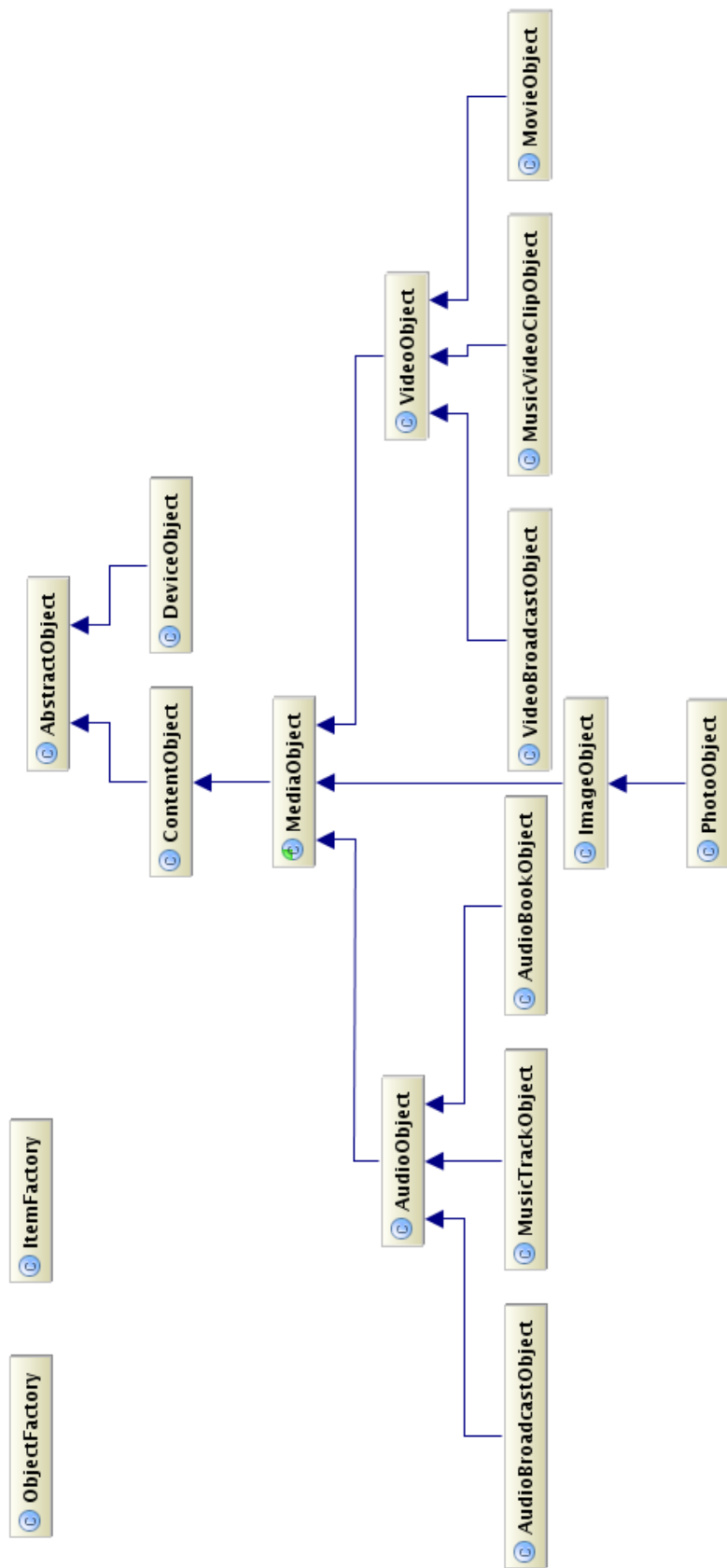





























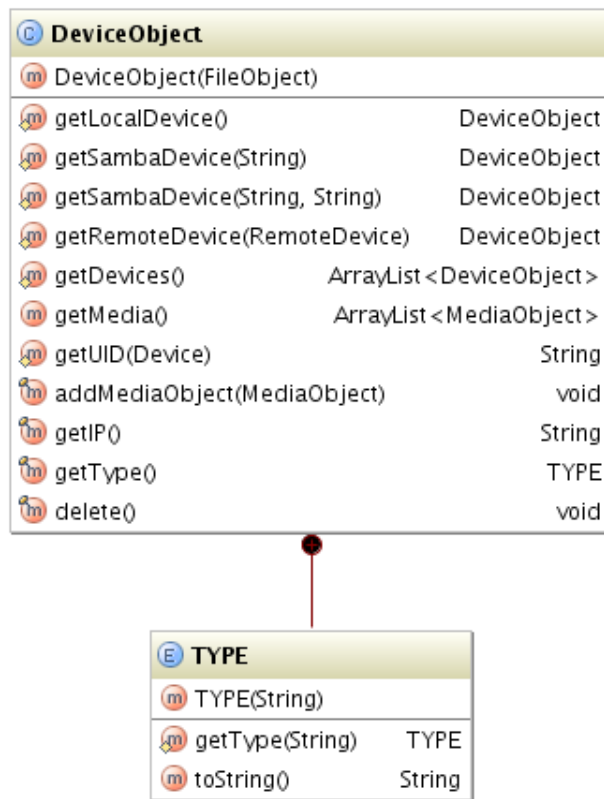
Figure A.4: Media Models Class Diagram



AbstractObject		
	AbstractObject(FileObject)	
	AbstractObject(String)	
	AbstractObject(String, String)	
	getUID()	String
	getDisplayName()	String
	setDisplayName(String)	void
	setFileName(String)	void
	addPath(String)	void
	setPaths(String[])	void
	getPaths()	String[]
	getAttribute(String)	Object
	getStringAttribute(String)	String
	getStringsAttribute(String)	String[]
	getPersonsAttribute(String)	Person[]
	getPersonsWithRoleAttribute(String)	PersonWithRole[]
	getURIsAttribute(String)	URI[]
	getAttributes()	String[]
	setAttribute(String, Object)	void
	setAttribute(String, String)	void
	setAttribute(String, String[])	void
	setAttribute(String, PersonWithRole[])	void
	setAttribute(String, Person[])	void
	setAttribute(String, URI[])	void
	isVisible()	boolean
	setVisible()	void
	toString()	String
	getFileObjectPath()	String
	getChildren()	AbstractObject[]
	copyTo(FileObject)	void
	compareTo(AbstractObject)	int

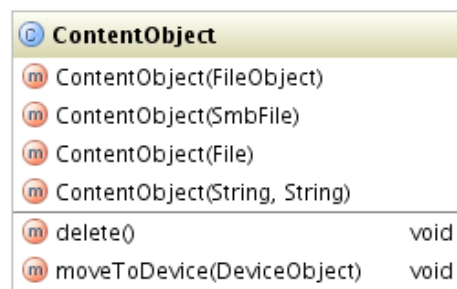
Powered by yfiles

Figure A.5: AbstractObject Class Detail



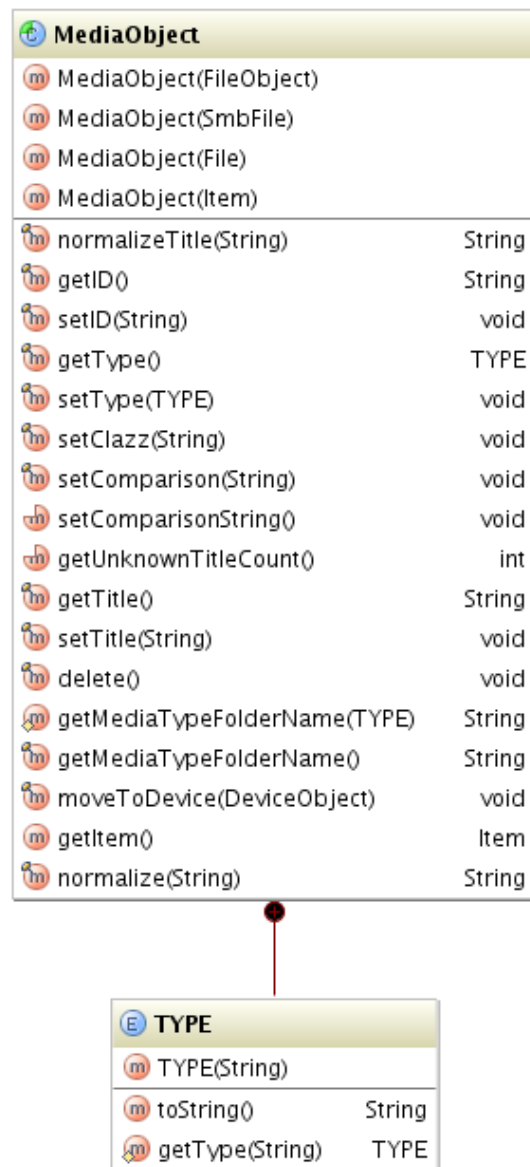
Powered by yFiles

Figure A.6: DeviceObject Class Detail



Powered by yFiles

Figure A.7: ContentObject Class Detail



Powered by yFiles

Figure A.8: MediaObject Class Detail

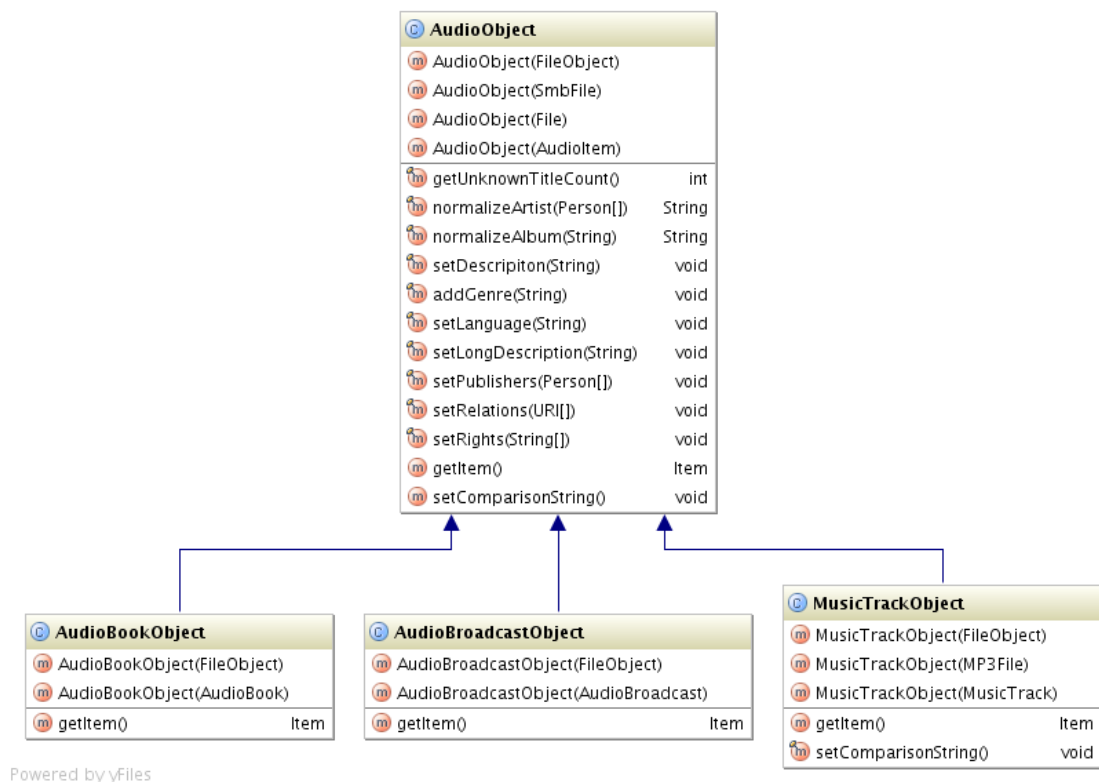


Figure A.9: AudioObject Class Detail

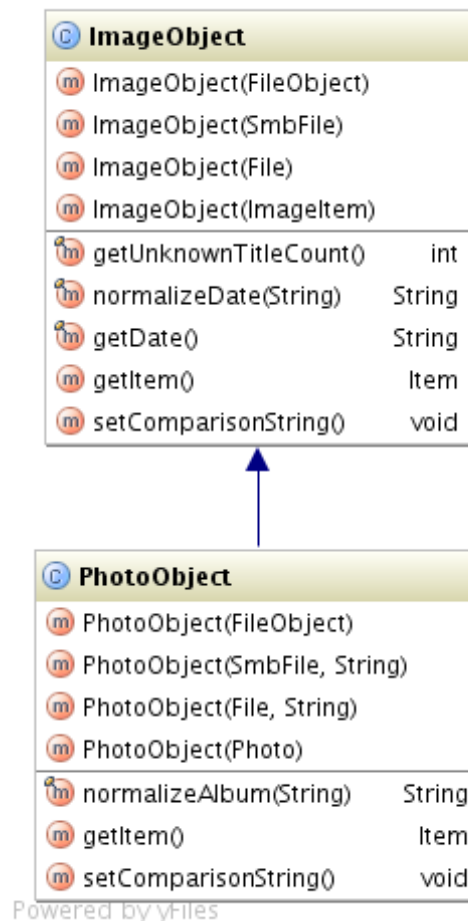


Figure A.10: ImageObject Class Detail

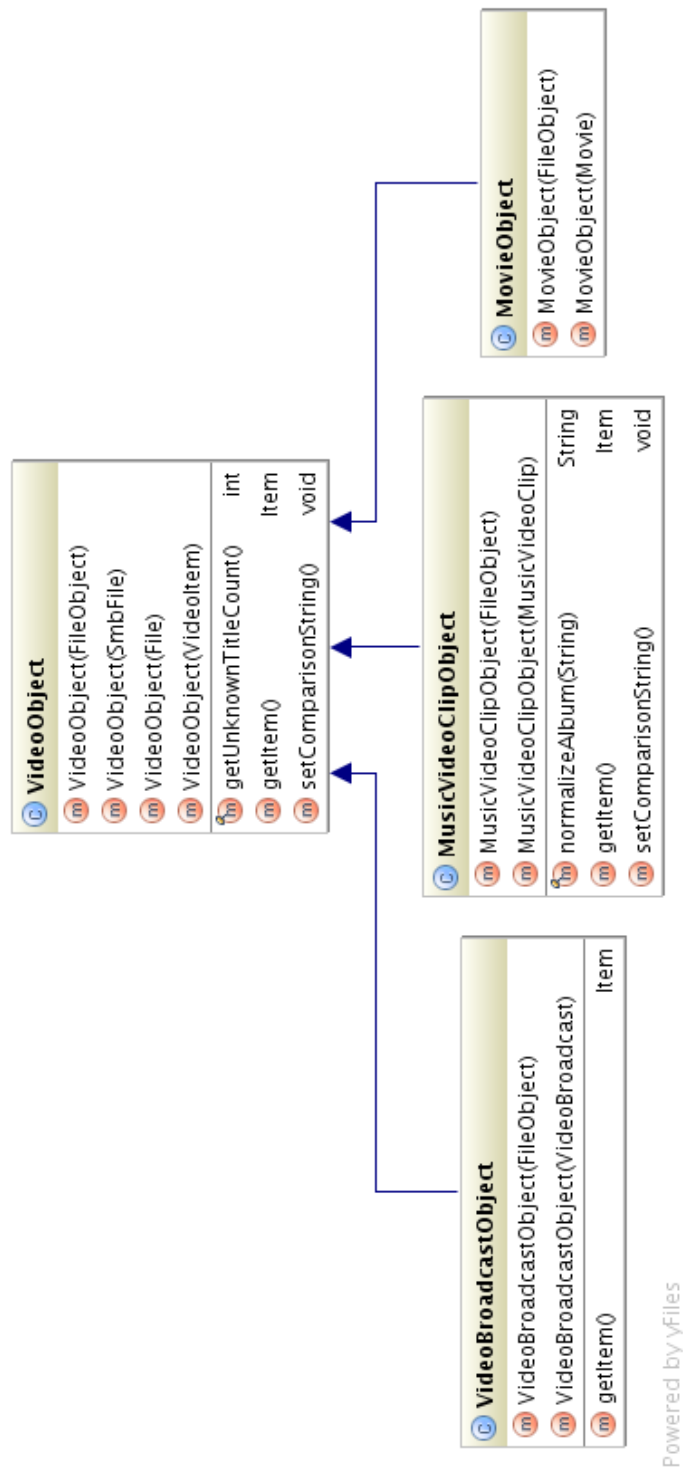


Figure A.11: VideoObject Class Detail

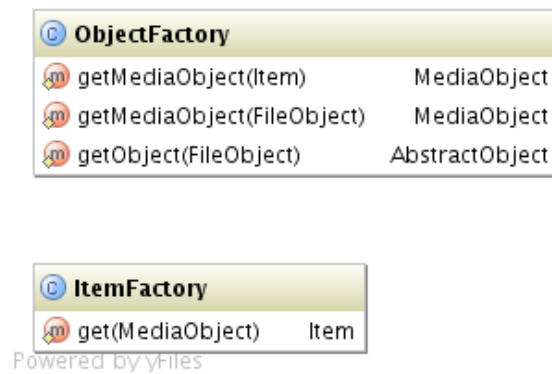


Figure A.12: Factories

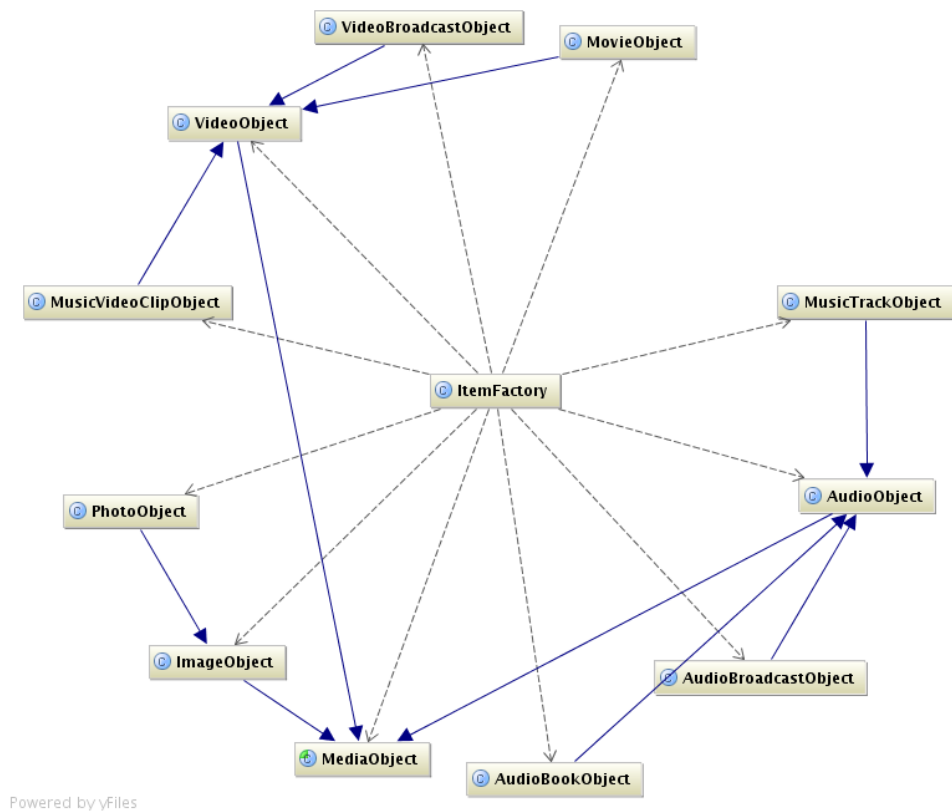


Figure A.13: Item Factory Detail



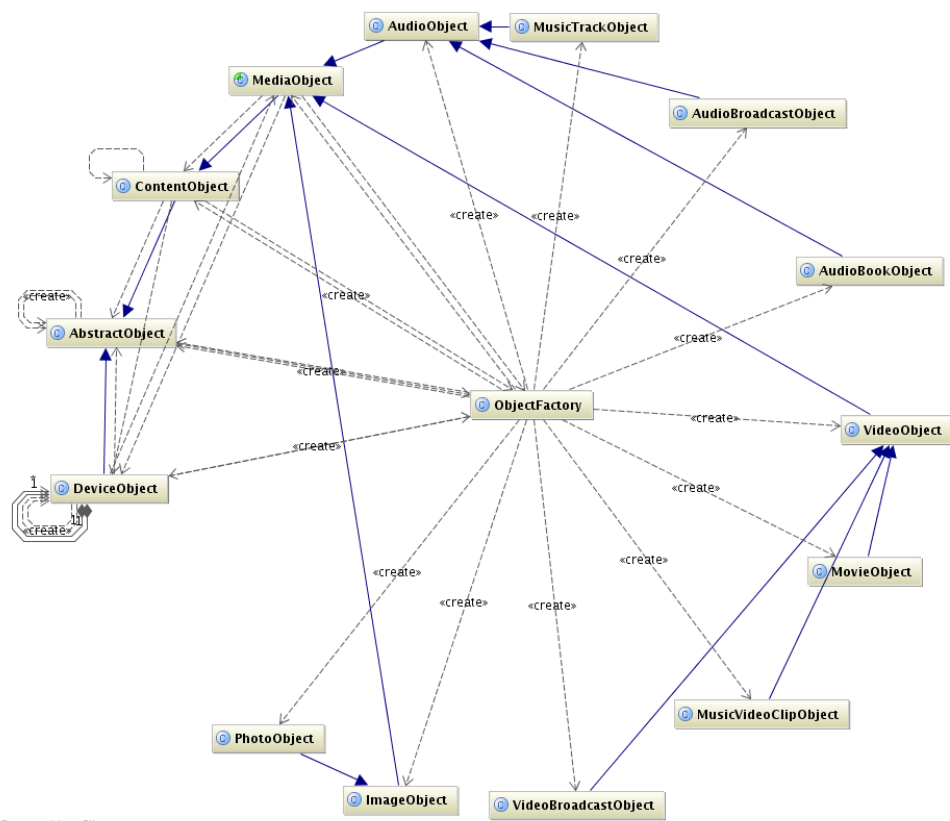


Figure A.14: Object Factory Detail



# B

## List of Figures

2.1	DLNA Interoperability Guidelines Building Blocks[11]	6
2.2	DLNA Classes Relations	7
4.1	Home Environment Example	40
5.1	Internal layout of an ID3v1 tagged file[27].	44
5.2	Internal layout of an ID3v1.1 tagged file[27].	44
5.3	Internal layout of an ID3v2 tagged file[27].	45
6.1	Audio Files Directory Structure Sample	53
6.2	Image Files Directory Structure Sample	53
7.1	Server Based Topology	59
7.2	Distributed Topology	60
9.1	Application: Main Window	73
9.2	Preferences: Local File System	75
9.3	Preferences: Samba Shares	75
9.4	Application: Aggregation in progress	77
9.5	Application: Media File Detail	78
A.1	GUI Dialogs Diagram	87
A.2	Aggregators Class Diagram	88
A.3	Aggregator GUI Detail Diagram	89
A.4	Media Models Class Diagram	90
A.5	AbstractObject Class Detail	91
A.6	DeviceObject Class Detail	92
A.7	ContentObject Class Detail	92
A.8	MediaObject Class Detail	93
A.9	AudioObject Class Detail	94
A.10	ImageObject Class Detail	95
A.11	VideoObject Class Detail	96
A.12	Factories	97
A.13	Item Factory Detail	98

A.14 Object Factory Detail . . . . .	99
--------------------------------------	----



## List of Tables

2.1	UPnP Architecture (DCP) stack [22] . . . . .	11
2.2	Supported DLNA Media Formats[11] . . . . .	21
2.3	Metadata Systems Overview . . . . .	22
2.4	Media Container Formats Comparison . . . . .	23
2.5	Media Container Formats and Audio Codecs Support . . . . .	24
2.6	Media Container Formats and Video Codecs Support . . . . .	25
3.1	Media servers: Media support comparison [25] . . . . .	30
3.2	Media Servers: Operating Systems and License Comparison [25] . .	31
3.3	Media Servers: Vendor's Product Pages[25] . . . . .	32
3.4	Media Servers: Supported Video Formats . . . . .	33
3.5	Media Servers: Supported Audio Formats . . . . .	34
3.6	Media Servers: Supported Image Formats . . . . .	35
3.7	<i>Apple TV</i> supported formats [30] . . . . .	36
4.1	Home Environment Example: Available Devices . . . . .	39
5.1	Fields in ID3[27]. . . . .	44
5.2	Selection of relevant TIFF Rev. 6.0 IFD Attributes [17] . . . . .	48
5.3	Selection of relevant EXIF IFD Attributes [17] . . . . .	48
5.4	Selection of relevant GPS IFD Attributes [17] . . . . .	49
6.1	MIME Type - File Extension - ContentDirectory Class Mapping . . .	52
6.2	ContentDirectory Service Directory Structure[42] . . . . .	55



# D

## List of Abbreviations

AAC	MPEG-2,4 Advanced Audio Coding
ACIII	Audio Codec 3
ADPCM	Adaptive Differential Pulse-Code Modulation
AIFF	Audio Interchange Format File
ALAC	Apple Lossless Audio Codec
ALS	MPEG-4 Audio Lossless Coding
ASF	Advanced Systems Format
ATSC	United States Advanced Television Systems Committee
AVI	Audio Video Interleave
CBR	Constant Bitrate
CDS	UPnP Content Directory Service
CMML	Continuous Media Markup Language
CMYK	Cyan-Magenta-Yellow-Key (black) color model
DCP	UPnP Device Control Protocol Framework
DHCP	Dynamic Host Configuration Protocol
DLNA	Digital Living Network Alliance
DMA	Digital Media Aggregator
DMC	Digital Media Controller
DMP	Digital Media Player
DMP <sub>r</sub>	Digital Media Printer
DMR	Digital Media Renderer
DMS	Digital Media Server
DNS	Domain Name System
DTS	Digital Theatre Systems
EBML	Extensible Binary Meta Language
EXIF	Exchangeable Image File Format
FLAC	Free Lossless Audio Codec
FLV	Flash Video
FourCC	Four Character Code
FS	Local File System
FTP	File Transfer Protocol
GENA	General Event Notification Architecture

GIF	Graphics Interchange Format
GPS	Global Positioning System
GS	Gathering Source
GUI	Graphical User Interface
GUID	Globally Unique Identifier
HDMI	High-Definition Multi-media Interface
HDTV	High-definition television
HEAAC	High-Efficiency Advanced Audio Coding
HID	Home Infrastructure Devices
HND	Home Network Devices
HTTP	Hypertext Transfer Protocol
ID3	IDentify an MP3
IFD	Image File Directory
IFF	Interchange File Format
ID3	IDentify an MP3
IP	Internet Protocol
ISO	bibliography refInternational Standard Organization
JFIF	JPEG File Interchange Format
JPEG	Joint Photographic Experts Group
LAN	Local Area Network
LZW	Lempel–Ziv–Welch
MDMC	Mobile Digital Media Controller
MDMD	Mobile Digital Media Downloader
MDMP	Mobile Digital Media Player
MDMS	Mobile Digital Media Server
MDMU	Mobile Digital Media Uploader
MDV	Message-Digest algorithm 5
MHD	Mobile Handheld Devices
MIU	Media Interoperability Unit
MKV	Matroska File Format
MNCF	Mobile Network Connectivity Function
MP3	MPEG-1 or MPEG-2 Audio Layer III
MPEG	Moving Pictures Expert Group
NAS	Network Attached Storage
OBEX	OBject EXchange
PCM	Pulse-code modulation
PDA	Personal Digital Assistant
PNG	Portable Network Graphics
QT	QuickTime
RM	RealMedia
SLS	MPEG-4 Scalable to Lossless
SMB	Samba
SOAP	Simple Object Access Protocol
SSDP	Simple Service Discovery Protocol



SWF	ShockWave Flash
TCP	Transmission Control Protocol
TIFF	Tag Image File Format
UDN	Unique Device Name
UDP	User Datagram Protocol
UML	Unified Modeling Language
UPnP	Universal Plug and Play
VBR	Variable Bitrate
VfW	Video for Windows
VOB	Video Object
WAV	Windows Wave Audio
WLAN	Wireless Local Area Network
WMA	Windows Media Audio
WMV	Windows Media Video
XML	Extensible Markup Language
XMP	Extensible Metadata Platform



# E

## Curriculum Vitae

Jan Kubový

### Affiliation

Institute for Application Oriented Knowledge Processing (FAW)  
Johannes Kepler University Linz  
Altenberger Straße 69, A-4040 Linz, Austria

### Personal Information

Date of birth:	December 12, 1984
Nationality:	Czech
Gender:	Male
Phone:	(+43) (0) 699 104 291 12
Email:	jan@kubovy.eu
Homepage:	<a href="http://jan.kubovy.eu">http://jan.kubovy.eu</a>
Professional Profile:	<a href="http://www.linkedin.com/in/kubovy">http://www.linkedin.com/in/kubovy</a> <a href="http://www.xing.com/profile/Jan_Kubovy">http://www.xing.com/profile/Jan_Kubovy</a>
Address:	Goethova 1201/20 35002 Cheb Czech Republic

## Education

### **Johannes Kepler University in Linz**

M.Sc., Software Engineering, Computer Science, 2010 - 2011 (expected)  
Institute for Application Oriented Knowledge Processing (FAW)  
International Studies in Informatics

### **Czech Technical University in Prague**

Ing. (eq. to M.Sc.), Computer Science, Cybernetics, 2009 - 2011  
Open Informatics  
Faculty of Electrical Engineering

### **Czech Technical University in Prague**

Bc., Computer Science, Electrical Engineering, 2005 - 2009  
Electrical Engineering and Communication Engineering  
Faculty of Electrical Engineering

## Honours, Awards & Scholarships

### **Student Mobility Scholarship** 2010-2011

Faculty of Electrical Engineering, Czech Technical University in Prague  
Support for Double-Degree Programme

### **Merit Scholarship (awarded 2 times)** 2009-2010, 2010-2011

Faculty of Electrical Engineering, Czech Technical University in Prague  
Open Informatics

### **Google Developer Competition Winner** 2010

Faculty of Electrical Engineering, Czech Technical University in Prague  
Open Informatics