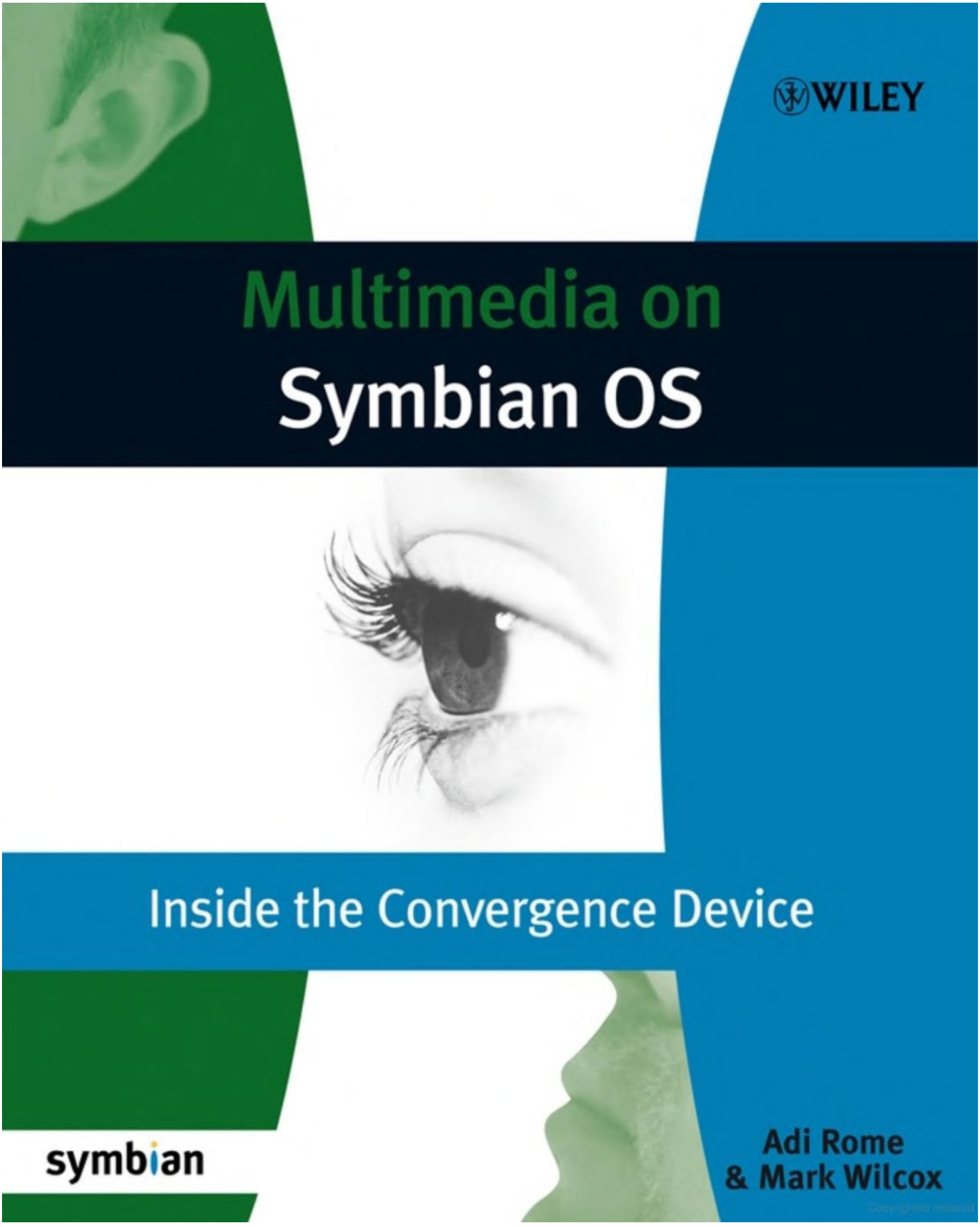

Meta Exhibit 1008 - Meta v. Weple
Page 001

Multimedia on Symbian OS

Inside the Convergence Device

Lead Authors
Adi Rome and Mark Wilcox

With
Kevin Butchart, John Forrest, Robert Heal, Kostyantyn Lutsenko, James Nash

Foreword
Andrew Till (Motorola)

Reviewers
Fadi Abbas (Scalado), Phyllisia Adjei, Les Clark, Patrick Cumming, James Curry, Brian Evans, Magnus Ingelsten (Scalado), Amanda Kamin, Petteri Kangaslampi, Roman Kleiner, Pamela Lopez, Anthony Newpower, Mike Roshko, Jo Stichbury, Gábor Török

Head of Symbian Press
Freddie Gjertsen

Managing Editor
Satu McNabb

A John Wiley and Sons, Ltd., Publication

Copyright © 2008

John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester,
West Sussex PO19 8SQ, England

Telephone (+44) 1243 779777

Email (for orders and customer service enquiries): cs-books@wiley.co.uk
Visit our Home Page on www.wileyeurope.com or www.wiley.com

All Rights Reserved. No part of this publication may be reproduced, stored in a retrieval system or transmitted in any form or by any means, electronic, mechanical, photocopying, recording, scanning or otherwise, except under the terms of the Copyright, Designs and Patents Act 1988 or under the terms of a licence issued by the Copyright Licensing Agency Ltd, 90 Tottenham Court Road, London W1T 4LP, UK, without the permission in writing of the Publisher. Requests to the Publisher should be addressed to the Permissions Department, John Wiley & Sons Ltd, The Atrium, Southern Gate, Chichester, West Sussex PO19 8SQ, England, or emailed to permreq@wiley.co.uk, or faxed to (+44) 1243 770620.

Designations used by companies to distinguish their products are often claimed as trademarks. All brand names and product names used in this book are trade names, service marks, trademarks or registered trademarks of their respective owners. The Publisher is not associated with any product or vendor mentioned in this book.

This publication is designed to provide accurate and authoritative information in regard to the subject matter covered. It is sold on the understanding that the Publisher is not engaged in rendering professional services. If professional advice or other expert assistance is required, the services of a competent professional should be sought.

Other Wiley Editorial Offices

John Wiley & Sons Inc., 111 River Street, Hoboken, NJ 07030, USA

Jossey-Bass, 989 Market Street, San Francisco, CA 94103-1741, USA

Wiley-VCH Verlag GmbH, Boschstr. 12, D-69469 Weinheim, Germany

John Wiley & Sons Australia Ltd, 42 McDougall Street, Milton, Queensland 4064, Australia

John Wiley & Sons (Asia) Pte Ltd, 2 Clementi Loop #02-01, Jin Xing Distripark, Singapore 129809

John Wiley & Sons Canada Ltd, 6045 Freemont Blvd, Mississauga, Ontario, L5R 4J3, Canada

Wiley also publishes its books in a variety of electronic formats. Some content that appears in print may not be available in electronic books.

British Library Cataloguing in Publication Data

A catalogue record for this book is available from the British Library

ISBN 978-0-470-69507-4

Typeset in 10/12 Optima by Laserwords Private Limited, Chennai, India
Printed and bound in Great Britain by Bell & Bain, Glasgow
This book is printed on acid-free paper responsibly manufactured from sustainable forestry in which at least two trees are planted for each one used for paper production.

Contents

Foreword ix

About this Book xiii

Authors' Biographies xvii

Authors' Acknowledgements xxi

Symbian Press Acknowledgments xxiii

Code Conventions and Notations xxv

1 Introduction 1
- 1.1 The Convergence Device 1
- 1.2 Transformation of the Media Industry 3
- 1.3 Symbian OS 5
- 1.4 The Cutting Edge 6
- 1.5 Evolution of the Multimedia Subsystem in Symbian OS 8
- 1.6 A Peek into the Future 12

2 Multimedia Architecture 15
- 2.1 The ECOM Framework 15
- 2.2 Platform Security 16

2.3 The Content Access Framework 21
2.4 Multimedia Subsystem 23
2.5 Future Multimedia Support 31

3 The Onboard Camera 35
3.1 Introduction 35
3.2 Accessing the Camera 36
3.3 Camera Control 39
3.4 Displaying the Viewfinder 43
3.5 Capturing Still Images 46
3.6 Capturing Video 49
3.7 Error Handling 52
3.8 Advanced Topics 53

4 Multimedia Framework: Video 55
4.1 Video Concepts 55
4.2 Symbian OS Video Architecture 57
4.3 Client API Introduction 60
4.4 Identifying Video Controllers 61
4.5 Controlling Video Playback 62
4.6 Controlling Screen Output 71
4.7 Getting Video Information 81
4.8 Accessing Metadata 83
4.9 Controlling the Audio Output 84
4.10 Streaming Playback 87
4.11 Recording Video 88
4.12 Controlling the Video that Is Recorded 93
4.13 Controlling the Audio that Is Recorded 97
4.14 Storing Metadata 100
4.15 Custom Commands 101
4.16 Examples and Troubleshooting 102

5 Multimedia Framework: Audio 105
5.1 Introduction 105
5.2 Audio Input and Output Streams 107
5.3 Audio Player Utility 117
5.4 Audio Recorder Utility 122
5.5 File Conversion 127
5.6 Tone Utility 129
5.7 DevSound 134
5.8 Audio Policies 135
5.9 Priority Settings 137
5.10 Miscellaneous 138

6 Image Conversion Library 141
6.1 Introduction 141
6.2 Decoding Images 144
6.3 Encoding Images 174
6.4 Displaying Images 181
6.5 Transforming Images 185
6.6 Native Bitmap Operations 189
6.7 Miscellaneous APIs 193

7 The Tuner API 197
7.1 Introduction 197
7.2 Getting Started 198
7.3 Basic Use Cases 201
7.4 Future Technologies 206
7.5 Sample Code 207

8 Best Practice 209
8.1 Always Use an Active Scheduler 209
8.2 Use APPARC to Recognize Audio and Video 210
8.3 Don't Use the Video Player to Open Audio Files 212
8.4 Know that MMF and ICL Cannot Detect Some Formats 212
8.5 Don't Use `CMdaAudioOutputStream` for Network Streaming 213
8.6 Use `CMdaAudioPlayerUtility` to Play Tone Sequence Files 215
8.7 Use `CMdaAudioPlayerUtility` to Play Clips 215
8.8 Don't Hardwire Controller UIDs in Portable Code 216
8.9 Set Controller Thread Priorities Before Playing or Recording 217
8.10 Recognize that Behavior Varies when Several Clients Are Active 218
8.11 Understand that the System is Based on Plug-ins 219
8.12 Use `RFile` and `TMMSource` Instead of Passing a File Name 221
8.13 Consider the Volume Control 222
8.14 Know When to Use Threaded Requests on ICL 222
8.15 Don't Have too many Clients 223
8.16 Understand the Multi-heap Problem 224

References and Resources 229

Index 233

4

Multimedia Framework: Video

In this chapter, we look at the Symbian OS video architecture and how you can use it. We begin with details about some general video concepts and look at the various levels of the software architecture. We then describe the client-side APIs that allow you to perform various video playback and recording operations.

4.1 Video Concepts

Before explaining the Symbian OS video architecture, let's get familiar with a few important video concepts.

A video is a series of still images which can be displayed one after another in sequence to show a scene which changes over time. Typically, the images have an accompanying audio track which is played at the same time as the images. The term 'video' is often taken to mean this combination of images and associated audio.

4.1.1 Delivering Video to a Device

There are two main ways that a video can be delivered to a mobile device:

- The video data could be stored in a file that is downloaded to the device. When playback is requested, the file is opened and the data is extracted and played.
- The video data could be streamed over a network to the device, which plays it as it is received.

4.1.2 Video Recording

On Symbian smartphones with a built-in camera and microphone, it is possible to record a video. The video and audio data can be recorded to a file for subsequent playback or could be streamed off the device for storage on a server elsewhere.

4.1.3 Video Storage Formats

Video and audio data stored in a file, or received over a streaming link, is often encoded in order to reduce the size of video files or the amount of data that must be streamed. The job of encoding and decoding the data is handled by a device or program called a codec. Encoding and decoding video is a very processor-intensive operation, so the codec may reside on a separate dedicated processor or on a general-purpose Digital Signal Processor (DSP) to ensure timely processing of the video without impacting the performance of the rest of the system.

There are a number of common encoded data formats that you may come across. For video image data, you may see MPEG2, H.263, MPEG4 and H.264. Each of these formats define a set of methods that can be used to encode the image data. Generally the newer formats, such as H.264, provide better levels of compression than older formats, such as MPEG2, especially as they define sets of levels and profiles which mean different encoding methods can be used depending on the type of image data being encoded.

For audio data you may come across formats such as MP3, AAC and AMR. MP3 is a widely used format for music files. AAC is intended to be its successor and provides better compression at lower bit rates. AMR is specifically designed for encoding speech data.

The encoded video image and audio data for a video are often stored or streamed together in a container. For video files, there are a number of different container formats that are commonly used, such as MP4, 3GP, and AVI. They differ in how the encoded data is stored within the file and in what encoded data formats they can store. MP4 and 3GP keep the information about the size and position of the encoded data units separate from the encoded data itself, allowing easier access to specific units. AVI groups all the information together.

Video image and audio data types are often described using a four-character code, known as a 'FourCC code'. For instance, video image data encoded using the XVID codec would have a FourCC code of 'XVID'.

In addition, the container file format and the video image data format can be described using a notation called a MIME type. For instance, the AVI container file format is described by the MIME type `video/msvideo` and the MPEG4 video image data format has the MIME type `video/mp4v-es`.

4.2 Symbian OS Video Architecture

Figure 4.1 shows an overview of the Symbian OS video architecture. In this section, we briefly describe each of the architectural layers.

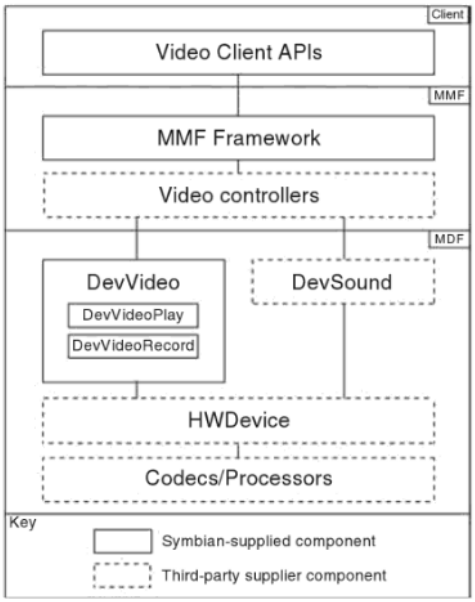

Figure 4.1 The Symbian OS video architecture

4.2.1 Client APIs

Access to the video functionality is provided through the client APIs. Applications use the APIs directly to perform the required video operations. One important point to note is that even though the client APIs may support an operation, there is no guarantee that the lower layers of the architecture will do so. Most of the functionality that performs video operations is handled by the video controllers (see Section 4.2.3) and some controllers may not handle certain operations. The client APIs are described in more detail in the following sections.

Video Playback

The client APIs support playback from file, stream, and descriptor. So you can open a file by specifying the file name or handle, specify a URL to stream from, or pass a buffer that contains the video data. Other basic video operations such as play, pause, and stop are also supported. There are methods which allow you to manipulate how the video is displayed on the screen and to perform operations such as rotating the video. There

are also methods which allow you to get information about the video, such as frame rate and frame size.

Video Recording

As with video playback, you can record to a file, to a stream, or to a descriptor. You can specify a file name, a stream URL, or a buffer. In addition you are required to give the handle of the camera that you would like to record from. See Section 3.6 for details about preparing the camera for recording. You can start, stop and pause the recording. There are also methods which allow you to modify various characteristics for the stored video such as the frame rate, or audio sample rate.

4.2.2 MMF Framework

The MMF framework provides methods to identify and load the video controller that should be used to playback or record a video. The framework also provides the mechanism for passing client requests to the controller and receiving responses and errors from the controller.

4.2.3 Video Controllers

While this book does not cover how to write a video controller, it is important to understand what they do. Video controllers are normally supplied preinstalled on a mobile phone, although, because the controllers use a plug-in mechanism, it is possible for developers to write and install their own controllers if they would like to.

A video controller controls the flow of video data from an input to one or more outputs. For example, a controller might open an MP4 file containing video and audio, use a video codec to decode the video frames and then write them to the display, while using an audio codec to decode the audio frames and then play them through the loudspeaker.

Each installed video controller advertises the container formats it can deal with. So for instance you might find one controller that can open and play only AVI files, while another controller, such as RealPlayer, might be able to support a whole range of container formats. This allows the correct controller to be selected based on what type of file or stream is being used.

For video playback from file, the chosen controller is responsible for opening the file and parsing any metadata in order to determine the types of video and audio data streams; the relevant video and audio codecs are then loaded to decode those streams. The controller extracts the video and audio frames into buffers and passes them to the relevant codecs for decoding. For video, the controller may then pass the decoded frames