Multimedia Messaging Service Components for Web 2.0

KRISTOFER BORGSTRÖM

KTH Information and
Communication Technology

Master of Science Thesis
Stockholm, Sweden 2007

COS/CCS 2008-01

Multimedia Messaging Service Components for Web 2.0

Kristofer Borgström

Academic supervisor and examiner

Professor Gerald Q. Maguire Jr.

Industrial supervisor

Peter Yeung, Ericsson

Abstract

The purpose of this master's thesis is to simplify the exchange (in both directions) of multimedia content between mobile phones and network attached web servers. The solution proposed in this report specifically concerns displaying Multimedia Messaging Service (MMS) messages via a web browser connected to a network-attached web server and graphically authoring MMS messages via a web interface.

This thesis project is important because it brings multimedia content, in the form of MMS messages, from isolation in the telecommunication world closer to wide availability via the Internet. This transition is very important as the Internet is where media is shared with the world today. This approach brings added value to end users who want to share content generated using their phone on a web site. It also provides added value to operators who want to increase the amount of MMS traffic in their networks.

The solution is non-trivial because there are a number of complexities at both ends. This is because the MMS messages that are authored at mobile phones differ between both handset models and manufacturers. Moreover, the format used for MMS (*MMS SMIL*) is not widely used on the Internet, thus a transformation to an Internet browser supported format must be performed. The thesis examines to what extent this transformation can be completely automatic and how MMS messages can be authored through a web interface.

The results show that MMS messages can be successfully transformed to HTML and embedded directly in web pages, thus providing a seamless experience for viewing MMS messages. Depending on the content of the MMS message in question, the current browser and which media player plug-ins are available, the generated HTML will be displayed differently. The results also show that MMS messages can be composed in real time* through a web interface with good results.

* The MMS will be displayed as it is being composed, allowing the end user to have good idea of how the MMS message will be experienced when sent to a mobile terminal.

Sammanfattning

Syftet med detta examensarbete är att förenkla utbyte (i båda riktningarna) av multimedia mellan mobiltelefoner och nätverksuppkopplade webbservrar. Lösningen som föreslås i denna rapport behandlar specifikt hur Multimedia Messaging Service (MMS)-meddelanden kan visas i en webbläsare via nätverksuppkopplade webbservrar och hur MMS-meddelanden kan komponeras grafiskt via ett webbinterface.

Detta examensarbete är viktigt eftersom det för multimedia, i form av MMS-meddelanden, från isolering i telekommunikationsvärlden närmare en bred tillgänglighet via Internet. Denna övergång är viktig eftersom det är på Internet som multimedia görs tillgängligt för världen i dagens läge. Denna approach förbättrar upplevelsen för användare som vill dela med sig av innehåll genom sin telefon genom en hemsida. Den ökar också möjligheterna för mobiloperatörer att öka MMS-trafiken i sina nätverk.

Lösningen är inte trivial eftersom det existerar ett antal komplexiteter i båda ändarna. Detta beror på att MMS-meddelanden som skapas i mobiltelefoner skiljer sig åt mellan såväl tillverkare som modeller. Dessutom används inte MMS-formatet (*MMS SMIL*) på Internet. Således måste en transformering till ett format som stöds av webbläsare genomföras. Detta examensarbete undersöker i vilken utsträckning denna transformering kan automatiseras helt och även hur MMS-meddelanden kan skapas via ett webbinterface.

Resultaten visar att MMS-meddelanden framgångsrikt kan transformeras till HTML och bäddas in på en hemsida på ett sådant sätt att de upplevs som en del av hemsidan. Beroende på vilken typ av media som MMS-meddelandet innehåller, den aktuella webbläsarkonfigurationen och på vilka mediaspelar-plug-ins som finns tillgängliga, måste olika HTML genereras för att innehållet ska visas på ett bra sätt. Resultaten visar också att MMS-meddelanden kan skapas grafiskt, i realtid, direkt genom ett webbinterface med goda resultat.

Abbreviations

3GPP	3’rd Generation Partnership Project
AJAX	Asynchronous JavaScript and XML
Blog	Web log
CSS	Cascading Style Sheets
DOM	Document Object Model
DHTML	Dynamic HyperText Markup Language
GUI	Graphical User Interface
HTML	HyperText Markup Language
IPX	Internet Payment Exchange
JAXB	Java API for XML Binding
JAXP	Java API for XML Processing
JDK	Java Development Kit
JMF	Java Media Framework
JSF	Java Server Faces
JSON	JavaScript Object Notation
JSP	Java Server Pages
JSTL	Java Standard Tag Library
MMS	Multimedia Messaging Service
OMA	Open Mobile Alliance
SAX	Simple API for XML
SDK	Software Development Kit
SMIL	Synchronized Multimedia Integration Language
XHTML	Extensible HyperText Markup Language
XML	Extensible Markup Language
XPath	XML Path Language
XSL	Extensible Stylesheet Language
XSLT	XSL Transformations
VASP	Value Added Service Provider
W3C	World Wide Web Consortium

Contents

1. Introduction

1.1. Problem description

The goal of this master's thesis project is to simplify the exchange of multimedia content between mobile phones and Internet web sites. To achieve this goal, this project has three major parts:

- Java components are used to convert *Multimedia Messages Service* (MMS) *Synchronized Multimedia Integration Language* (SMIL) to HTML (see section 1.2.2). The resulting views can be displayed via traditional web browsers.

- Web components are introduced to display MMS content and to compose new *MMS SMIL*-based MMS messages on Internet web sites via traditional web browsers.

- An integration of web components into an existing web application is made in order to demonstrate how to use the proposed components in a meaningful way within a traditional web browser.

This master thesis contributes to the realization of Web 2.0 by bringing mobile multimedia content to the traditional Internet based web.

1.1.1. MMS transformation components

MMS messages received from a mobile phone may contain several different types of media, such as: images, text, video, and audio. Such a message also contains a SMIL document that describes how the message should be displayed. A SMIL document received from a mobile phone is formatted according to *MMS SMIL* as defined by the *Open Mobile Alliance* (OMA) [1].

In order to be able to display MMS messages on a web site, each *MMS SMIL*-formatted message has to be converted to a format that is supported by either the user's browser or by a multimedia player that is available on the user's computer, preferably through a browser plug-in (such plug-ins are described in section 2.1.3.2).

The best way to display the media would be to have it integrated directly in a web page. This is possible using the *Microsoft Internet Explorer* browser through an extension of HTML known as *HTML+TIME* [10], this is similar to *XHTML+SMIL*[*]. However, none of the other major browsers currently support SMIL directly, although the *Mozilla Firefox* team are currently considering SMIL support [2]. As a starting point, *HTML+TIME* should be supported by the MMS components; the means of doing so is described as part of this thesis project.

[*] XHTML+SMIL is defined in the SMIL 2.0 specification which is briefly discussed in section 2.2.1.

Ideally, this thesis project should implement a simple cross-browser *MMS SMIL* player using *Dynamic HyperText Markup Language* (DHTML)[*] in order to support other browsers. However, the first priority is support for major desktop browsers (specifically: *Microsoft's Internet Explorer*[3], *Mozilla's Firefox* [4], and *Apple's Safari* [5]). The second priority is porting the implementation to a *XHTML-Mobile Profile* (XHTML-MP) [6] and *ECMAScript* [7] solution in order to support mobile browsers.

Support for SMIL in an embedded SMIL player is the lowest priority, because initial testing indicated that the existing SMIL player implementations were quite different and do **not** display content in a satisfactory manner, see section 4.1.1.

1.1.1.1. Content transcoding support

Bringing the Internet and mobile phones together also presents some problems with regard to the content itself. This is because a given type of content is not always supported by different devices nor via the different media players. The problem is clearest with audio and video content. The formats traditionally used for video and audio in MMS messages are H.263 video and AAC/AMR/MIDI audio[†]. These formats are not traditionally used on the web; therefore they are not supported by the major web browsers. However, they are supported by *Apple's QuickTime* player and *RealNetworks' RealPlayer*.

Content transcoding should ideally be enabled through use of Java libraries, e.g. the *Java Media Framework (*JMF). Unfortunately, the most recent version of JMF (2.1.1) does not support most of the mobile media formats. Although, there is an open source project called *Jffmpeg* [9] which may be used to provide such transcoding. Unfortunately, there is currently no implementation that could be used directly. Thus it is unlikely that such transcoding support can be part of this thesis project. Instead, an *Apple QuickTime* plug-in or similar media player plug-in that can play these formats will be utilized. Additional details of such plug-ins will be given in section 2.1.3.2.

1.1.2. Web components

1.1.2.1. MMS presentation components

As stated in the previous section, different web browsers have different levels of support for SMIL. Furthermore, while one or more media players are able to play SMIL content, they may or may not be available as plug-ins to display content for a given web browser. Thus, depending on browser/media player capabilities which a given user has installed and on the type of media that a specific MMS message contains, a decision about what HTML view to generate and display on the page to be presented to the user must be made. As should be readily apparent, this greatly complicates the problem of presenting content to each user.

The best way to display MMS messages would be to integrate them directly into a web page, so that they are a seamless part of the web page. Ideally they should be displayed the way that the author of the content intended. This can be achieved with direct SMIL support in the browser (*HTML+TIME*), with a DHTML *MMS SMIL* player implementation, or with an embedded media player that can play SMIL presentations.

---

[*] DHTML is a collection of technologies used to create dynamic web pages. Such technologies include: HTML, *JavaScript*, *Cascading Style Sheets* (CSS) and the *Document Object Model* (DOM).
[†] These media formats are defined as being in the Core MM Content Domain in the *MMS Conformance Document* [8] as defined by the *3'rd Generation Partnership Project* (3GPP).

In some scenarios, it may not be desirable to present a given MMS message as it was composed, e.g. as a timed presentation. For example, in such scenarios, the web developer might want to control *how the content is displayed.* For these situations, the MMS content should also be available, through suitable web components, for example, as an ordered array of multimedia elements.

1.1.2.2. MMS composer components

The MMS composer components enable easy, web based creation of *MMS SMIL* presentations. The resulting presentations can then be sent to a mobile phone or displayed using the presentation components.

The Graphical User Interface (GUI) of the MMS composer components should be similar to MMS message composers available on mobile phones, although when running on a network-attached computer the interface should make use of the larger presentation area available.

1.2. Why is the thesis project significant?

1.2.1. Introduction

In many modern technologies, features have been offered to end users through different devices and different underlying technologies. Today we see these different devices converging. This is especially noticeable in the telecommunication industry where mobile phones now have the same features as mp3 players, web browsers, personal organizers, etc.

As the various technologies converge with respect to hardware, the need to ensure that these different devices can exchange information with each other increases. Thus providing the end user with full flexibility of accessing and sharing information from one device to another, despite using potentially different underlying technologies. This further accelerates the trend toward convergence.

1.2.2. Potential

The Internet is likely to be the major source of information that is available to most users today. Thus if you want to show something to the world, the Internet is the place to make it available. The sharing of pictures, music, and video is a very large part of the Internet as experienced by users today, for example, through services such as Facebook [11], YouTube [12], etc. Mobile phones have the advantage of being a device that most people always have with them and increasingly these devices have the ability to take photos or shoot videos, allowing them to capture activities that might otherwise not be readily captured. Therefore mobile phones have a great potential to be a major source of new multimedia content. We are starting to see this content on the web today (see for example CNN's *I-Report* [13]). However, the production of user-generated content is only starting to explode and we can expect much more such content in the future.

Unfortunately, it is currently quite complicated to share content from a mobile phone via a web site of your own choice. This often requires moving the content from the mobile phone to a computer in order to upload it. Even when web sites have a mobile version, these sites often do not have the full features of a regular web site and even if they do, then data traffic to and from this site is often quite expensive*. Further more, mobile surfing of the web is perceived as complicated. Hence the option of simply sending content as a message is very attractive, as many users are familiar with sending messages, thus sending a multimedia message is often perceived as being fast and easy (as compared to the alternative means of sending content). The result is that the user is able to readily display multimedia content generated by their mobile phones on a web site. This new source of significant volumes of user produced content creates a lot of new possibilities for both web masters and end users.

At the receiving site (or along the path to it) the MMS content must be transformed to one of the formats which are commonly used on Internet web sites. This MMS transformation can also be used to close the gap between the Internet and the mobile world in the other direction. If MMS transformation **from** web sites is made easy and wide-spread access to mobile networks is available, then multimedia sites all over the web could have a button to send an MMS message with multimedia content to a mobile phone†. Some examples of what this solution could be used to achieve include:

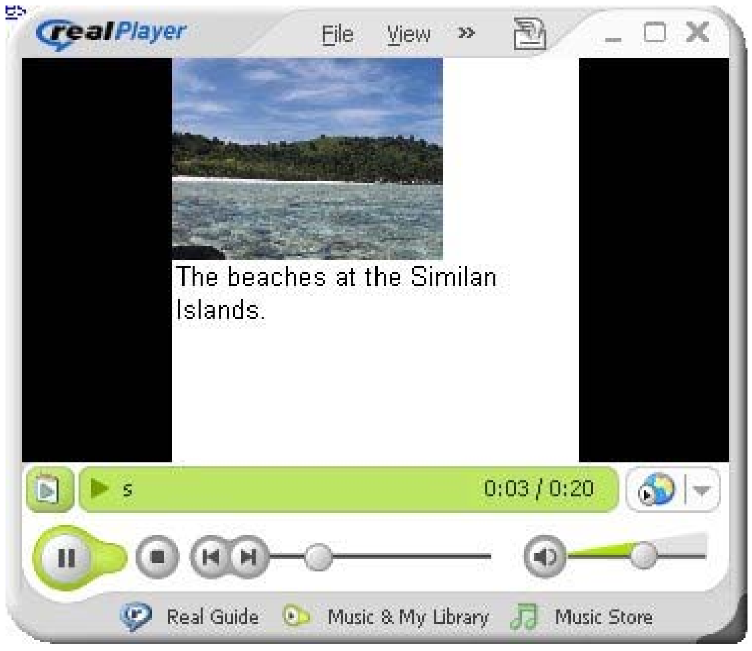
- Existing media services such as YouTube could allow MMS messages as a new way of posting content.
- Blogs could receive content directly from MMS messages and could generate content to be viewed by users as MMS messages.
- End users could increase traffic to their sites/blogs by sending the most interesting posts to friends as MMS messages.

1.2.3. Difficulties

It is quite complicated to display an MMS message in a web browser in the same layout it was composed. This is because MMS messages consist of a number of media elements and a SMIL file that defines how the media should be displayed. Unfortunately, it is the interpretation of the SMIL that presents the greatest problem. Currently, SMIL capable players show the message in very different ways. Figures 1 and 2 show how an MMS message sent from a mobile phone was displayed in two different media players.

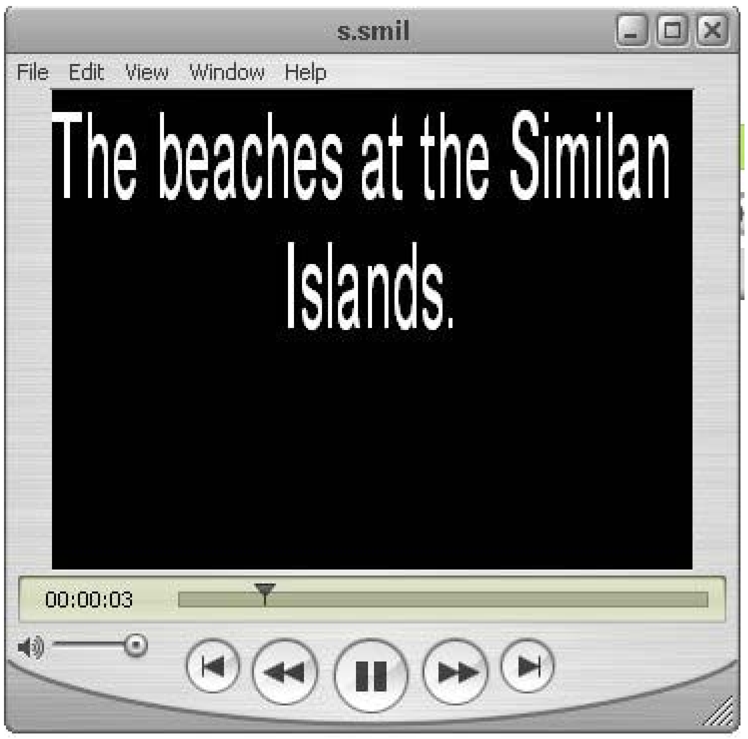
* Even though there are flat-rate options available, most Swedish operators (Telenor [14], Tele2 [15], and Tre [16]) charge 10-15kr/MB as the going rate for mobile surfing.

† There is an issue regarding who will pay for the sent MMS message. In some scenarios it may be acceptable that the creator of the service (web site) is charged (the creator can then charge users in turn). In other scenarios, when the receiver of the MMS message is to be charged, I believe that some sort of subscription service will be required. I.e. a user send a message specifying that it is ok to receive x messages per month at a cost of y each.



The beaches at the Similan Islands.

Figure 1. MMS SMIL displayed in RealNetworks' RealPlayer. All elements are displayed, but the display area is not fully utilized.



The beaches at the Similan Islands.

Figure 2. MMS SMIL displayed in Apple's QuickTime. Only the text appears and it does so both enlarged and with inverted colors.

Currently there is direct SMIL support in the Microsoft's *Internet Explorer* browser through *HTML+TIME* [10]. However, because this is not a direct subset of SMIL, but rather a different language that can be used to achieve a similar end result, *MMS SMIL* can not be displayed at all via *Internet Explorer* without transformation of the *MMS SMIL* to *HTML+TIME*. This illustrates the need for MMS transformation.

There are many reasons why we have not yet seen the types of services described in section 1.2.2 available all over the web. First of all, there is a knowledge gap between web developers and developers of mobile services, few web developers know how to connect to a mobile network and access the services available there. There are however ongoing efforts in the telecommunication industry to simplify this process through standardized interfaces and Web Services [18]. Such a standard is, for example, Parlay X which is standardized by the Parlay Group [19]. The Parlay X standard defines a set of telecommunication Web Services that can be used to access telecommunication network capabilities such as MMS. There are also Software Development Kits[*] (SDKs) to further simplify the access to such capabilities, for example, the Telecom Web Services SDK [17].

The MMS components, as presented in this thesis project, are needed to simplify the utilization of MMS capabilities by web developers. Examples of technologies that may be used to expose this type of functionality include Java Server Pages (JSP) (see sections 2.2.3.2 and 2.2.3.3), Java Server Faces (JSF) and even Java object representations of MMS messages. The purpose of such components is to make it very easy to display and create MMS content in any Java application, and especially in web applications. However, few web developers know of these technologies or how to utilize them to achieve their goals.

Traditionally, operators have posed a big problem by being very strict as to what services they offered end users. Often, the service creation process[†] from innovation to launch has been very tightly controlled by operators; even though applications have been developed by a third party these applications are typically hosted and branded by the operator. This tight control of all aspects of the service (and even the choice of available services) is known as the *walled garden* approach to service delivery, as opposed to the *open garden* approach where the operator exposes the capabilities of the mobile network to third parties that can create and host services themselves. Such capabilities may be exposed either within the operator's network, or publicly through the Internet. The latter approach seems likely enable a larger number of developers and service providers to make use of telecommunication network capabilities, thus providing users with a wider selection of services.

It appears to be increasingly difficult for operators to ignore the potential traffic which would be added to their network by allowing multimedia sites all over the world to be mobile-enabled as well as allowing third parties to create and host traditional mobile services. In short, operators are moving toward embracing the *open garden* model which has been so successful on the Internet. This trend was one of the key findings in a service creation study that was performed by *inCode* on behalf of *Ericsson* during the summer of 2007, see Appendix A.

---

[*] An SDK is typically a set of software development tools used by software developers to create applications that use a specific set of functionality.

[†] The service creation process in this context refers to the process of developing a new service/application that utilizes telecommunication network capabilities. The process covers the steps of innovation, development, testing, and launch of the service.

Further more, several operators have already taken steps towards enabling an open garden and are offering their developer communities proprietary SDKs which they can use to create and launch their own services. These operators include British *BT* with their *Web21C SDK* [20] and Swedish *Telenor* with their *Mobilstart* portal [21]. *BT*'s solution does not currently support MMS, but *Telenor's* solution does. In addition, the solution presented in this master's thesis could be used together with *Telenor's* solution in order to simplify sending and receiving MMS messages.

A web developer who wants to utilize MMS capabilities needs to have a basic understanding of the structure of MMS messages in order to know how best to display their content on the web and also to be able to create MMS messages that are displayed as intended when sent to mobile phones. The developer should also know how to transport the MMS messages to and from a mobile network. However, in an open garden scenario, when using available protocols and SDKs as described above, this is not necessarily a lot more difficult than performing a file upload.

2. Background

2.1. Previous work in the area

There has been very little work done to try to achieve what this thesis project attempts. That is, there is currently no open solution for showing MMS content as presentations on the web via a web browser. However, there are a number of sites that display MMS messages as blog entries, such as mms2web.com [22] and mobilblogg.nu [23]. Neither of these are based on open solutions for displaying MMS messages on the web, nor do they implement the slide-show-presentation effect of MMS messages when displaying content on the web. Furthermore, neither is able to send MMS content to a mobile phone.

There has been a lot of work done in related areas. This work can be summarized as follows:

- Displaying *MMS SMIL* presentations to make full use of the presentation area, see section 2.1.1.
- Transforming Extensible Markup Language (XML) documents, see section 2.1.2.
- Displaying timed multimedia on the web, see section 2.1.3.
- Connecting to service providers through telecommunication protocols such as MM7 and Parlay X, see section 2.1.4.

2.1.1. Displaying MMS SMIL

This section describes previous work done to display *MMS SMIL*; while making full use of the presentation area. To fully understand the details of this section, please refer to sections 2.2.1 and 2.2.2 for an introduction to SMIL and *MMS SMIL*.

2.1.1.1. Background

Because of the large differences in capabilities (e.g. screen size and processing capacity) between different mobile phones (even between models from a single vendor), the *MMS SMIL* specification [8] leaves a lot of room for the *MMS SMIL* player to decide how to display content based on device specific capabilities.

The *MMS SMIL* specification [8] states that:

> *…the messages that are produced should be valid and complete SMIL messages, and should be displayed properly on non-mobile terminals (e.g., PCs).* [4.1 Usage of SMIL, page 13]

While this is technically true in most cases and the *MMS SMIL* generated by mobile devices can be viewed in SMIL 2.0 compliant players, no real effort has been made by mobile device vendors to author SMIL that will be displayed in a user-friendly way. A study of how different handset models generate SMIL and what data can be used to determine how the message should be displayed has been made. This study is presented in section 4.1.3.

2.1.1.2. Handset SMIL players

In order for the experience when viewing MMS messages on the web to be similar to when these messages are displayed on a mobile device a set of rules have to be followed. These rules mainly concern the presentation area size and how to optimize the viewing experience when these messages do not conform to the SMIL specification (as noted in the *MMS SMIL* specification [8]). The rules proposed in this thesis project are based on how handset vendors have actually chosen to implement *MMS SMIL* players.

After examining how MMS messages are displayed on two mobiles phones (one Nokia model N70 and one *Sony Ericsson* phone model P990i), the following rules for displaying MMS on the web are proposed:

- The presentation area should have a fixed size that emulates the screen of a mobile phone.
- The regions defined in *MMS SMIL* will be ignored in order to make full use of screen size. In other words, if a slide contains only contains one element which according to the layout should be put in the lower half of the presentation area, it will be displayed in the top instead. Otherwise the top half of this slide would be empty.[*]
- Allow vertical scrolling while an element is playing. Scrolling is done across the whole presentation area.[†]
- Video will support the following scaling settings:
    - Use the original size of the video.
    - Set height to half the screen height and retain aspect ratio.
    - Set height to half the screen height and stretch width.

Note that the above rules sometimes do not follow layout attributes in the *MMS SMIL* file. This is in accordance with the *MMS SMIL* specification, as stated in section 2.1.1.1.

2.1.2. XML transformations

2.1.2.1. Background

There are a couple of different technologies that can be used to transform XML encoded content. These include, but are not limited to: XSLT, JAXB, DOM, JDOM [24] and SAX. Most of these technologies will be described later.

[*] This is how it behaves on the Sony Ericsson P990 test phone, but not on the Nokia N70 test phone. The layout employed by the Sony Ericsson handset was chosen because it was considered to provide a more natural view of an MMS message (as opposed to strictly following MMS SMIL as in the Nokia case).
[†] The Nokia N70 test phone required manually activating scrolling, which was viewed as a limitation.

The particular requirements for transforming *MMS SMIL* to *HTML+TIME* have been investigated and are explained in section 4.1.5. The objective of considering the technologies above was to identify a technology that can perform the transformation in a satisfactory manner. In establishing the exact requirements for the transformation, one must consider what data is available **and** meaningful in the *MMS SMIL*. Further more, consideration must also be given to the question of whether the data actually needs to be processed and consideration given to the composition of the end result.

The following requirements are suggested and explained in section 4.1.5. They are presented here to motivate the consideration of each of the potential Java technologies in terms of these requirements.

- Read and parse *MMS SMIL*.
- Generate output in XML.
- Perform string processing.
- Generate XML with mixed namespaces and processing instructions.
- Generate different output based on current settings of one or more variables.
- Ease of Java code integration.
- Ease of reading and inserting text files.

2.1.2.2. XSLT

XSL Transformation (XSLT) is part of the Extensible Stylesheet Language (XSL). It can be used to transform one XML document into another XML document based on one or more XSL style sheets. XSLT uses XPath to navigate through the elements and attributes of an XML document. An XSLT style sheet is used as a template to produce the resulting XML document. [25]

Advantages

- XSLT supports a wide variety of features that support complex transformations.
- Because XSLT is supported by major browsers, server side transformation can be avoided, thus reducing the amount of processing power required by the server (at the cost of processing at the client).
- XSLT provides a standard and widespread technique to transform XML documents.
- Almost no Java code is required to perform the transformation.
- XSLT does not require an XML schema.

Disadvantages

- One XSL template file is required for each combination of settings.
- Does not directly support inserting content from text files into the output document. This could be worked around by using file inclusion tags in the server side scripting language (JSP, JSF, etc.).

- Does not support reading of a document into a custom Java object representation as required in the Java code integration requirement.

2.1.2.3. JAXB

Java API for XML Binding (JAXB) provides an API that enables a developer to generate a set of classes that represents a specific XML schema and based on these classes; generate an XML document that conforms to the specified XML schema.

JAXB uses the *xjc compiler* (which is part of the Java Developers Kit (JDK) [26]) available as an *Apache Ant*[*][27] task and as a command line tool to generate java classes. These Java classes are then used to programmatically produce an XML document that can be written (marshalled) to a file. JAXB can also be used to read (unmarshal) XML documents that conform to a certain XML schema.

Advantages

- JAXB guarantees that the resulting XML document conforms to the XML schema.
- Provides a widespread and standardized way generate XML documents.
- The XML document is created from a *Java* context so that java expressions can be used to produce the output.
- Using *Java* I/O APIs any file can be read and its content inserted into the generated XML document.

Disadvantages

- Transformation must be done on the server.
- Requires a lot of java code.
- XML schemas must be written to define exactly what is legal and what is not in the output and input files.
- It is very complicated, though possible, to mix namespaces.

2.1.2.4. SAX

Simple API for XML (SAX) is an event based API that can be used to parse and generate XML encoded content. A SAX parser parses an XML file sequentially, invoking event methods of a handler as the file is processed. This results in a fast and memory efficient way to parse XML.

However, SAX can not be used to access an XML file randomly. It requires more code than programming using the previously mentioned technologies and it is also more difficult to visualize its operations because it is event based. [28], [29]

[*] Apache Ant is a software tool that enables platform independent software builds.

Advantages

- Provides a standard API to parse and generate XML.
- Fast and memory efficient [30].
- The XML document is created from a Java context so that Java expressions can be used to produce the output.
- Provides simple methods to generate XML documents with different namespaces.
- Using Java I/O APIs any file can be read and its content inserted into the generated XML document.

Disadvantages

- Requires more code than other technologies.
- Does not allow in-memory modification of the XML.

2.1.2.5. Other technologies

There are several other technologies that can be used to transform XML, although research into how to use these are outside the scope of this master's thesis.

The decision to exclude investigation of other technologies from the scope of this master's project was based on the following assumptions:

- JAXB, SAX, and XSLT are flexible, provide good performance, and are widely used. [30]
- JAXB and SAX provide all the required features, as put forth in section 2.1.2.1.
- There are too many different potential XML transformation technologies to investigate all of them within the scope of this master's thesis; thus this thesis has focused on the most widely used technologies.

2.1.3. Displaying multimedia on the Web

2.1.3.1. HTML+TIME

HTML+TIME is the basis for SMIL support that is built into Microsoft's *Internet Explorer*. Since the introduction of *HTML+TIME* in *Internet Explorer* 5, the specification has evolved and is now at version 2. [10] *HTML+TIME* provides timing much like that used in SMIL. In fact, *HTML+TIME* is actually based on the *XHTML+SMIL* profile of SMIL 2.0, see section 2.2.1. Although *HTML+TIME* has been around for a long time, it has not received been widely adopted by web developers. One major reason for this is that it is still only implemented by *Microsoft's Internet Explorer* and will not work with any other browser. This might change in the future as the *Mozilla Firefox* team wants to support parts of the forthcoming SMIL 3.0 specification [2].

Although there are few actual implementations of *HTML+TIME* sites on the web, there are some good references available, describing how to use *HTML+TIME*. These include tutorials, references, and examples, see the *HTML+TIME* tutorials at *W3Schools.com* [31] and on *Microsoft.com* [10]. These tutorials were used as a reference when creating the *HTML+TIME* templates described in section 4.1.4.

2.1.3.2. Embedding media players

To enable playback of media types that are not directly supported by a web browser, an external media player can be embedded as a part of a web page. This requires that the end user's browser has a plug-in for the specific media player that is to be embedded. Most media players expose playback control features through *JavaScript* so that the media can be started and stopped at any time.

The process of embedding media players is done slightly differently in different browsers and different versions thereof. There are two different tags that can be used, the embed tag and the object tag. To ensure cross-compatibility, the two can be combined by using an `object` tag with a nested `embed` tag. A simple example of how *Apple's QuickTime* player can be embedded in an HTML web page is given in the code example shown in Example 1.

```
<object id="audio1" height="0" width="0"
     classid="clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"
     codebase="http://www.apple.com/qtactivex/qtplugin.cab">

  <param name="src" value="mmsvoice.amr">
  <param name="autoplay" value="false">

  <embed name="audio1" height="0" width="0"
        src="mmsvoice.amr"
        type="video/quicktime"
        autostart="false"
        pluginspage="http://www.apple.com/quicktime/download/">
  </embed>
</object>
```

Example 1. HTML code example of embedding an *Apple QuickTime* media player element.

Using the `object` tag, all run-time parameters should be specified using nested `param` elements. Using the `embed` tag on the other hand all parameters are specified as attributes.

The `classid` attribute is different for different media players. Different players have a series of different parameters that define how the embedded element will be displayed. Different media players also define different *JavaScript* methods to control playback of media files. The parameters and functions for the different media player plug-ins that are interesting for this thesis project are listed tables 1, 2, and 3 below.

Table 1. Parameters and functions for *Apple's QuickTime* plug-in [32],[33].

Name	Description
`AUTOPLAY`	This parameter specifies whether playback should start automatically. Values: `true` Start playback automatically `false` Do not start playback automatically
`CLASSID`	This parameter specifies the unique class ID for this object as needed by *Microsoft's Internet Explorer*. The value of this attribute specifies that *Apple's QuickTime* should be used to play the the content. Value: `"clsid:02BF25D5-8C17-4B23-BC80-D3488ABDDC6B"`
`CONTROLLER`	This parameter specifies the visibility of the movie controller. Values: `true` The movie controller is visible `false` The movie controller is not visible
`PLUGINSPAGE`	This parameter specifies a Uniform Resource Locator (URL) from which the user can fetch the necessary plug-in if it is not installed.
`SCALE`	This parameter specifies the scaling to use for a movie. Values: `"tofit"` The movie is stretched to fit within the height and width of the element `"aspect"` The movie is stretched to fit within the height and width of the element, while maintaining the aspect ratio `<numeric>` A numeric value defining a scaling factor relative to the original size of the video clip. For example: "1.5"

Name	Description
`GetPluginStatus()`	This function gets the status of the current movie. Values: `“Waiting”` Waiting for the movie data stream to begin `“Loading”` Data stream has begun, not able to play/display the movie yet `“Playable”` Movie is playable, although not all data has been downloaded `“Complete”` All data has been downloaded `“Error: <#>”` The movie failed with the specified error number
`Play()`	This function starts playback of the movie
`Rewind()`	This function rewinds the movie
`Stop()`	This function stops playback of the movie

Table 2. Parameters and functions for RealNetworks’ Real Player plug-in [34].

Name	Description
`AUTOSTART`	This parameter specifies whether playback should start automatically. Values: `true` Start playback automatically `false` Do not start playback automatically

Name	**Description**
`CENTER`	This parameter specifies whether the presentation should be centered in the image window and displayed in its original, encoded size. Values: `true` The presentation is centered `false` The presentation is not centered
`CLASSID`	This parameter specifies the unique class ID for this object as needed by *Microsoft's Internet Explorer*. The value of this attribute specifies that *RealNetworks' RealPlayer* should be used to play the the content. Value: `“clsid:CFCDAA03-8BE4-11cf-B84B-0020AFBBCCFA”`
`CONTROLS`	This parameter specifies the controls to be displayed on the web page. Applicable value: `“ImageWindow”` `Show only the image window for video or animation playback`
`MAINTAINASPECT`	This parameter specifies whether the height-to-width (aspect) ratio of the clip should stay constant when the clip scales to fit the image window Values: `true` The aspect ratio remains constant `false` The aspect ratio does not remain constant
`DoPlay()`	This function starts playback of the movie
`DoStop()`	This function stops playback of the movie

Name	Description
`GetPlayState()`	This function gets the current playback state. Values: `0` Stopped `1` Contacting `2` Buffering `3` Playing `4` Paused `5` Seeking

Table 3. Parameters and functions for Microsoft's Windows Media Player plug-in [35][36].

Name	Description
`AUTOREWIND`	This parameter specifies whether the movie should automatically be rewound when it stops. Values: `true` Auto rewind `false` Don't auto rewind
`AUTOSTART`	This parameter specifies whether playback should start automatically. Values: `true` Start playback automatically `false` Do not start playback automatically
`CLASSID`	This parameter specifies the unique class ID for this object as needed by *Microsoft's Internet Explorer*. The value of this attribute specifies that *Microsoft's Windows Media Player* should be used to play the the content. Value: `"clsid:6BF52A52-394A-11D3-B153-00C04F79FAA6"`

Name	Description
`PLAYSTATE`	This parameter specifies the current playback state. Applicable values: `0` Undefined `1` Stopped `2` Paused `3` Playing
`SHOWCONTROLS`	This parameter specifies whether to show playback controls or not. Values: `true` Show playback controls `false` Do not show playback controls
`Play()`	This function starts playback of the movie
`Stop()`	This function stops playback of the movie

2.1.4. Connecting to service providers

Sections 2.1.4.1 and 2.1.4.2 describe two different protocols that can be used to send and receive MMS messages. Sending and receiving MMS messages requires that the application is connected to a service provider that in turn is connected to the mobile networks to which the messages should be sent.

2.1.4.1. MM7

MM7 is a telecommunication protocol used to handle multimedia messaging between Value Added Service Providers and mobile networks. The MM7 protocol is defined in 3GPP 23.140 [41] and is based on the SOAP protocol [42]. MM7 allows Value Added Service Providers to send, receive, cancel, and replace MMS messages when communicating directly with an MMS relay server; as well as to receive and/or retrieve status information about a certain message.

There are several MM7 SDKs available to simplify the use of the MM7 protocol by Java application developers. One such SDK is *Ericsson's MM7 SDK*[39] which can be downloaded from *Ericsson's Developer Program* [38].

There is also a standardized Java API that can be used for MMS messaging. This API is defined in Java Specification Request (JSR) 212 and is known as the *SAMS-M API*. There is currently no open *SAMS-M* implementation available. However, there is an implementation of *SAMS-M* included in the *J2EE Web Application Template* which is available from *Ericsson's Developer Program [54]*.

2.1.4.2. Parlay X

Parlay X is a set of telecommunication Web Services which provide application developers with an abstract interface to a mobile network. Parlay X Web Services can be used to access telecommunication network capabilities such as, for example, MMS messaging. Parlay X Web Services is defined by the Parlay Group [19] and is a SOAP-based [42] set of Web Services. The Parlay X Web Services that are defined for MMS allow an application developer to send and receive MMS messages as well as retrieve and receive the delivery status for an MMS message.

Parlay X Web Service calls are made to a Parlay X gateway that is responsible for exposing the Web Services (either within an operator's network or on the Internet), and for processing the Web Service requests and forwarding them to the mobile network using the appropriate protocol. In doing so, it allows application developers to access a number of different telecommunication network capabilities, without using complex telecom protocols (such as MM7), but rather using Web Services.

Furthermore, there are SDKs available that will simplify the use of Parlay X web services. One such SDK is *Ericsson's Telecom Web Services SDK [17]*. This SDK exposes the telecommunication capabilities as basic *JavaBeans*, which means the developer does not have to be familiar with Web Services in order to access such capabilities.

2.2. Underlying technologies

2.2.1. SMIL basics

The SMIL language is a text-based, standardized XML format developed to provide interactive multimedia as web content. The SMIL standard was released by the web standards organization: World Wide Web Consortium (W3C) [37]. [38]

The SMIL language combines different types of media formats into an interactive presentation. The SMIL 2.0 specification supports features such as selection, linking, activation control, synchronization, and transitions, which enables a very rich display of multimedia content. [38]

Because the SMIL 2.0 specification is very extensive, W3C defined a number of profiles that provide subsets of the functionality of the full SMIL 2.0 language, to allow thinner clients to support the more basic features of SMIL. SMIL 2.0 defines the following profiles: [38]

SMIL 2.0 Language	The full SMIL 2.0 language
SMIL 2.0 Basic	Subset for low-power devices
3GPP Mobile SMIL	SMIL subset designed for mobile phones
XHTML+SMIL	SMIL timing integrated into XHTML
SMIL 1.0	The original SMIL specification

Even though the 3GPP Mobile SMIL profile was designed for mobile phones, it has not been widely adopted by the telecommunication industry as of yet, although *Nokia* has added 3GPP SMIL support to some of their newer models [44]. Instead, the telecommunication industry uses another subset of SMIL known as *MMS SMIL*. *MMS SMIL* was not defined by *W3C*, but rather by *Open Mobile Alliance* (OMA) [45]. This subset is explained in more detail in section 2.2.2.

XHTML+SMIL was designed to be integrated into XHTML. Despite the release of the SMIL 2.0 specification in August 2001, none of the major browsers* currently support XHTML+SMIL. In fact, this specification has not progressed to W3C's recommendation status. However, *Microsoft's Internet Explorer* does offer similar functionality through *HTML+TIME* [10], which was the result of a collaboration between *Microsoft*, *Macromedia*, *Compaq/Digital*, and *Digital Renaissance*. It has not been adopted in browsers other than *Microsoft's Internet Explorer*. It was submitted to the W3C in 1998 [46] and it appears likely, though not specifically stated, that it was an important contribution to the work of defining the *XHTML+SMIL* profile [47].

2.2.2. MMS SMIL and its limitations

As briefly described in section 2.2.1, *MMS SMIL* [8] is a subset of the SMIL 2.0 specification. It was specified by the Open Mobile Alliance (OMA) [45]. OMA is an organization that was formed by nearly 200 companies involved in the mobile industry to ensure interoperability between vendors, operators, networks, and devices. *MMS SMIL* was designed to be a subset of SMIL suited for multimedia messaging. SMIL provides the capabilities to present multimedia content. At the time *MMS SMIL* was proposed there was no SMIL profile defined by the W3C [37] that was considered suitable for the mobile devices available at that time. Thus, OMA specified *MMS SMIL* [8].

There are a number of limitations in the *MMS SMIL* specification, most are related to the limited screen size and limited processing capabilities of the mobile devices available at that time. The most notable limitations on *MMS SMIL* are:

- Each SMIL "slide" may contain a maximum of two regions, of which one must be text. It may also contain background sound. [8]
- The size of the SMIL message is critical. Early devices supported a total message size of at least 100kB, while newer devices can handle messages up to 600kB. However, the size of the message may also be limited by networks it passes through. [8]
- Mobile devices are not guaranteed to display the content as it was specified in SMIL. If the content does not fit the display, then the mobile device may rearrange it so that it can be displayed. [8]
- The synchronization features are limited, for instance, nesting of time containers are not allowed. [8]

2.2.3. Web applications in Java

This section provides a brief introduction to web applications in Java, for more information, please refer to *Java Servlet Programming* [48], *Pro JSP 2* [49], and *J2EE Design Patterns* [50].

* This includes *Microsoft's Internet Explorer, Mozilla's Firefox, Apple's Safari and Opera's Opera*.

2.2.3.1. Introduction

The main difference between a web application and a traditional application is that for web applications, most of the logic performed by the application is not performed on the client's computer, but rather on a web server. A web application client (a browser) sends a request to view a resource or to perform an action, and then the server processes this request, returning as its result a web page. Typical actions performed by the web server include: reading data from disk, processing of data, and performing database lookups.

Web applications in Java are typically Servlet based. A Servlet is a Java class that handles an incoming HTTP requests, processes these requests and sends responses back to the clients. Servlets exist within a web container on the web server. Based on which URL was requested by the client, different Servlets will be responsible for handling the request. Which Servlet is responsible for serving a given URL is defined in a deployment descriptor. Figure 5 shows an overview of a typical Java web application.

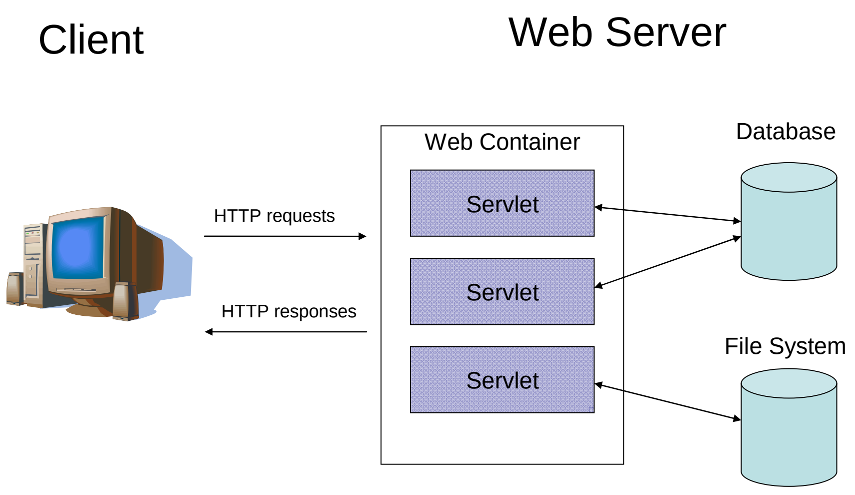


Figure 3. Web application overview.

Servlet-based web applications are highly maintainable and reusable [49]. A Java web application is packaged in a `.war` archive and can be deployed on any web container supporting Java Servlet technology, for example, *Apache's Tomcat* [51].

2.2.3.2. JSP

The markup of JSP is similar to HTML markup. The difference between HTML pages and JSP pages is that JSP pages can contain Java statements. In short, JSP adds the full functionality of the Java APIs to a web page. This enables the page to access databases, perform business logic, and display the result of these transactions on a web page.

The means by which this is accomplished is through a number of JSP tags, JSP expressions, and through embedded Java code. However, embedding Java code directly in a JSP page is considered bad practice because it often results in a lot of Java code mixed with HTML, making the code hard to follow and maintain. Using JSP 2, it is rarely necessary to embed Java code directly in a JSP page, because there is a large set of predefined tags, available through the Java Standard Tag Library (JSTL) that can be used to handle tasks such as iterating through collections and performing conditional statements. [49]

2.2.3.3. Custom tags

JSP custom tags provide a way to encapsulate reusable functionality on JSP pages. Custom tags can be used to add JSP code and logic that will be used on several different places within a web site, thus greatly improving reusability, readability, and maintainability.

3. Method

3.1. Goals

3.1.1. MMS transformation components

The first goal is to provide a minimal set of transformations for *MMS SMIL* to enable *MMS SMIL* to be displayed by a HTML-capable web browser. This includes supporting all media types and the commonly used slide compositions defined in *MMS SMIL* in order to display MMS messages as they would be displayed on mobile terminals. Another goal is to support MMS transformation from and to a Java object representation of an MMS message. This will greatly simplify editing of an MMS message as well as enabling MMS messages to be authored directly using Java code. Additionally it would be interesting to support 3GPP SMIL as an additional source format for transformation to HTML views; however, this is a very low priority activity.

The transformation components should support a set of rules that define how the HTML view of a given MMS message should be produced. The rules were determined based on functionality in the SMIL language and common media players. Such rules include for example the size of the presentation area for the MMS message, how many times the MMS message should be played, and whether or not to include a playback control bar.

The first priority for an *MMS SMIL* to HTML transformation is implementing a transformation that enables the resulting web content to utilize the timing capabilities of *HTML+TIME*. As discussed in section 2.1.3.1 *HTML+TIME* is currently only supported by *Microsoft's Internet Explorer* browser. Additionally, it would be very interesting to implement HTML output that will work in browsers that do not support *HTML+TIME*. Such an approach could make use of *JavaScript* and CSS to achieve the timing effects that MMS messages require. The first step in such an alternative solution would be to generate output that is supported by typical desktop-based browsers. However, it would also be interesting to implement a solution based on XHTML-MP and ECMAScript, such that it could also be displayed in mobile terminal browsers.

There should also be an automatic way of determining what media player plug-ins are available in the end user's browser. This is likely to improve the user's experience because the user will not have to manually determine which media player is installed and select this as their media player for this content.

The MMS transformation components should not limit the persistence strategy that can be chosen for media files. Thus it should be possible to serve media files that are required by the transformation components directly from a database or any other data source. Therefore, it should be possible to specify a Java class representation of a folder that contains media files. This Java class representation of a specific folder can then be used to serve the files from persistent (or even in-memory) storage.

It is also important to log important events and to implement the component code following a clear abstract design in order to simplify future development.

3.1.2. MMS presentation components

The MMS presentation components are based on the transformation components. The goal of the presentation components is to simplify the use of the transformation components from a web developer's perspective.

The presentation components should include a Servlet that performs real-time transformations from *MMS SMIL* to HTML. We will refer to the different forms of HTML output as views. The Servlet should also handle errors and based upon the cause of the error, the HTTP request should be dispatched to a predefined error page that will display a message to the end user. To further help developers integrate MMS messages into their web sites; a JSP custom tag that embeds the MMS view in a web page and passes the applicable parameters to the Servlet should be developed. Ideally, the web developer would only have to embed the JSP tag on a page and need not be aware of all the complexities involved concerning either *MMS SMIL* or the transformation process. Given sufficient time it would be interesting to implement a JSF tag with which could provide features similar to this JSP tag.

3.1.3. MMS composer components

The main goal of the MMS composer components is to make it as easy as possible to add MMS message authoring capabilities to a web site. A suitable solution should automate the process of composing an MMS message on a web page using *JavaScript* methods to add and remove different elements from an area in which the MMS is being graphically composed in real time based on input from the user. This area will henceforth be referred to as the *MMS canvas*. The content composition of the MMS message should also be verified, e.g. if a method is called to add an element that is not allowed on the current slide, then an appropriate error message will be returned.

The MMS composer components should also include a *JavaScript* method that will send the data of the composed MMS message to a given URL. This URL should in turn point to a Servlet that is responsible for creating a Java object representation of the MMS message and saving the resulting *MMS SMIL*. This Servlet should be one of the transformation components.

The MMS composer's HTML code should be exposed to the web developer through at least a JSP custom tag. Given sufficient time, it would be interesting to expose the HTML code as a JSF tag as well in order to simplify integration of the composer in a JSF application.

The composer components need **not** handle file uploads. The composer components should assume that all media files are already available from the web container. The Servlet responsible for creating and saving the *MMS SMIL* should not be aware of any paths specifying where the media files are located[*]. Instead, only the file name of a given media file should need to be added to the `src` attribute of a given media element in the resulting *MMS SMIL*. As such, the composer components are primarily *MMS SMIL* authoring tools, while the media file handling responsibilities belong to the surrounding web application.

The exception to this rule concerns text elements which should not be added to the *MMS canvas* as media files. Instead, the add-text *JavaScript* method (which takes one string containing the text) will be used to store this text in a text file that is generated during the transformation to *MMS SMIL*. As with the transformation components, the persistence strategy used to save text files should not be limited by the components.

[*] Note that during the authoring process, the composer components should support any valid HTML media paths. This functionality should be supported because it will greatly improve the user experience, as the user can see the media elements on the MMS canvas **during** the composing phase.

3.1.4. Additional deliverables

This thesis project is being conducted at *Ericsson* and as such there are some additional deliverables required to package this thesis so that it is interesting and valuable for *Ericsson* and by extension, the Web 2.0 community. One of the key components of this will be downloadable development information via *Ericsson's Developer Program* web site [52].

While this thesis will not be included in the download package that will be made available to developers through *Ericsson's* web site, a shorter and more focused user guide will be written and will be made available. This user guide is included as Appendix E.

To demonstrate to developers how to use the components in a meaningful way, the components will also be integrated into one or more example web applications, which should be included as a part of the download package. Parts of these examples are used in this thesis project.

3.2. Steps to be performed

3.2.1. Determining how SMIL is displayed in different SMIL players

As briefly discussed in section 1.2.3, SMIL presentations are not always displayed uniformly by different SMIL players. Thus, a study of how content is actually displayed in different media players was considered to be a very important part of this thesis project. SMIL is a very extensive specification, and few players support the whole SMIL specification. Further more, one should not assume that all players that can display the content do so in exactly the same way. Therefore, in order to present messages as uniformly as possible via different players, a study was conducted to determine how to format SMIL so that it is displayed very similarly via different media players; as well as to make sure it is displayed as was intended by the author. See section 4.1.1.

3.2.2. Setting up a test environment for MMS

A test environment was created in order to test how mobile networks and terminals handle MMS messages. This test environment is very important as it enables us to conduct experiments in a controlled environment. It was initially unknown whether all terminals and networks produce MMS messages formatted in the exact same way. Further more, different networks and terminals may modify the SMIL file and terminals may ignore certain parts of the layout that is defined within the SMIL file [8].

These differences in rendering and content changes can be very difficult and time consuming to predict from specifications alone. Therefore a central task throughout this project was to test, in reality, what the end results are for both incoming and outgoing messages.

The *Ericsson* department where this thesis project is taking place had access to an *Ericsson Internet Payment Exchange* (IPX) account [53], which enables sending and receiving MMS messages to more than 350 different mobile networks. The protocol used to communicate with *Ericsson* IPX is MM7. Thus, test applications could send and receive MMS messages, using an MM7 API (such as an implementation of *SAMS-M* (JSR-212) [40] or *Ericsson's* MM7 SDK 5.0 [39]).

The implementation of the test applications created for this test environment is described in section 4.1.2.

3.2.3. Interpreting MMS SMIL

It is desirable to use most of the information contained in the *MMS SMIL* file in order to present the content as it was intended by the author and as output by an MMS composer. This data could also help the transformation components decide how to transform the *MMS SMIL* to an HTML view.

However, it is not necessarily the case that all available data in each *MMS SMIL* file should be used in the transformation. As discussed in section 2.1.1, *MMS SMIL* players do not necessarily comply with the SMIL specification. Instead, a given *MMS SMIL* presentation should be displayed as the player deems best, given its available screen size and processing power. Although these limitations are typically not present on a desktop computer, this freedom has effected how *MMS SMIL* files are created by mobile phones in the sense that they are often not adapted to the content of a given MMS message. Thus, it may be preferable to ignore some data in the *MMS SMIL* in order to display it in user friendly way.

A qualitative study of how *MMS SMIL* messages are composed in mobile phones was carried out to determine which parts of an *MMS SMIL* file contain valid and/or valuable data which should be used during the transformation. This study is described in section 4.1.3.

3.2.4. Implementing static templates for HTML+TIME

Before the transformation components can be implemented it is necessary to identify what the output of the transformation should be. For our first set of transformation components we implement static templates that output *HTML+TIME* pages to display the content of an MMS message which has been formatted using *MMS SMIL*.

Some challenges to overcome are:

- Achieving a slide show effect using *HTML+TIME*
- Embedding media players for audio and video slides
- Performing scaling of images and video
- Controlling playback using *JavaScript* and *HTML+TIME* events
- Implementing playback control bar

The steps necessary in order to create static templates for conversion of *MMS SMIL* to *HTML+TIME* are described in section 4.1.4.

3.2.5. Implementing static templates for DHTML

Before the transformation components for DHTML can be implemented it is necessary to identify what the output the transformation should be. For our second set of transformation components we implement static templates that output DHTML pages to display the content of an MMS message.

Some challenges to overcome are:

- Achieving slide show effect using DHTML.
- Controlling playback using *JavaScript* and CSS.
- Controlling playback for audio and video files
- Performing scaling of images

The steps necessary in order to create static templates for conversion of *MMS SMIL* to DHTML are described in section 4.1.5.

3.2.6. Transformation requirements

Even when both the input and output of an XML transformation are known, the best way to perform a transformation between the two formats has to be determined. In order to determine which technology is best suited for performing this transformation, a set of requirements were used to evaluate a number of potential transformation technologies. These requirements are based on the input, output, and also the level of integration with Java code that is required.

The specific requirements and their respective motivations are discussed in section 4.1.5.

3.2.7. Transformation technology

As discussed in section 2.1.2 there are a number different of technologies that could be used to transform XML data. Determining which of these technologies should be used is very important, as it will affect most parts of the transformation components. To determine which XML technology to use, the following technologies were investigated and compared to the requirements – this is described in section 4.1.5.

The different technologies (introduced in section 2.1.2) are compared with the suggested requirements (described in section 4.1.5) in section 4.1.7.

3.2.8. Determine media player capabilities

In order for the transformation components to be able to embed an appropriate media player for a certain media type, it is important to know what media players support each of the different media formats. To determine this, a number of media players and their capabilities have been empirically tested. This testing and the results of these tests are described in section 4.1.8.

3.2.9. Designing and implementing the components

Much of the effort in this thesis project concerns how to design and implement the different components. The implementation details are briefly discussed in section 4.2. The Javadoc documentation for all classes can be found in Appendix F. Further more, the user guide is included in Appendix E. The complete package will be published during the first quarter of 2008, and can then be downloaded from *Ericsson's Developer Program* [52].

4. Analysis

4.1. Performed steps

4.1.1. Adapting SMIL content to specific players

After some very basic testing of how SMIL is displayed in the different supported players that were to be supported, it became very clear that both SMIL and the contents need to be generated differently depending on which player will be used by the receiver to display the message. Some of the inconsistencies among the players are:

- Embedded players require different amounts of extra space at the bottom of the embedded element's area on the web page, for playback controls.
- Text size varies.
- Text clipping is different in the different players.
- Some players support relative URIs to the media content (e.g. `mypic.jpg`), while others require an absolute URL[*].

Initial testing revealed that there was a need for an extensive study of how to create content to be displayed in more or less the same way via the different media players. Alternatively it would be necessary to generate multiple versions of the content (one version per player) and to know which player will be used for playback **before** sending the content to the device.

Due to the complexities involved in conducting this study and implementing transformation components to transform *MMS SMIL* content to the format needed for specific media players; the alternative solution of implementing a transformation solution from *MMS SMIL* to the subset of SMIL that was supported directly in media players was given a low priority.

4.1.2. Setting up a test environment for MMS

In order to easily send and receive MMS messages, two simple applications were implemented. These applications use SAMS-M to communicate with IPX. The SAMS-M implementation used was previously developed at *Ericsson*, but has not been released separately. To further simplify the implementation of the test applications, the application *J2EE Web Application Template* [54] from *Ericsson's Developer Program* was used as a base (as the above mentioned SAMS-M implementation is included in this template application).

[*] Real Network's RealPlayer requires absolute URLs if media is referenced directly from SMIL. It may be possible to work around this issue by referencing the SMIL file from a RealPlayer .ram file. For more information, see the e-mail correspondence with Eric Hyche from Real Networks collected in Appendix D.

4.1.2.1. Sending MMS

The send MMS application was designed to read files from a specific folder, attach them to an MMS message, and send the resulting message. This enables easy testing of any *MMS SMIL* message by simply copying the *MMS SMIL* file and the accompanying media files into this directory and invoking the sending class.

The send MMS application consists of a starter class with only a main method. The main method performs the following steps:

- Defines hard-coded parameters such to-number, from-number, and subject.
- Creates a new `MmsSendBean`.[*]
- Sets the parameters of the `MmsSendBean`.
- All files in the subfolder `mms` are read and set as attachments to the `MmsSendBean`.
- Use the business method of the bean to send the MMS message.

The code snippet in Example 2 shows this main method:

```
public static void main(String[] args) throws Exception {

     MmsSendBean sendMms = new MmsSendBean();

     sendMms.setMmsReceiver("<to-number>");
     sendMms.setMmsSender("<from-number>");
     sendMms.setSubject("<subject>");

     ArrayList<File> attachments = new ArrayList<File>();

     File mmsDir = new File("mms");
     File[] mmsFiles = mmsDir.listFiles();

     for (int i = 0; i < mmsFiles.length; i++) {
       attachments.add(mmsFiles[i]);
     }

     sendMms.setUploadedFiles(attachments);

     sendMms.setMimeFile("mime.types"); //Set mime types

     sendMms.sendMMS();
}
```

Example 2. Main method for send MMS application

[*] Note that a Java bean is simply a Java component, for more details about this particular Java bean, please refer to the *J2EE Web Application Template* [54].

The `MmsSendBean` was taken from the *J2EE Web Application Template* and modified slightly to use hard-coded parameters matching the particular IPX account which was used for testing. The resulting class is a Java bean wrapper around *SAMS-M* that enables sending of MMS messages using the MM7 protocol and this particular IPX account.

4.1.2.2. Receiving MMS

The receive MMS application was designed to listen for incoming MMS messages from IPX and save the attached files (both the *MMS SMIL* and the media files) in a new folder for each MMS message. This enables fast and easy collection of *MMS SMIL* sent from different mobile phones so that the variations between the resulting content can be easily studied.

Setting up the receive MMS application is a bit more complicated because it must be able to receive and act upon MM7 messages. Using the *SAMS-M* implementation, this is done in a web container by mapping a Servlet class to the URL that is used to listen for incoming MMS messages. This Servlet is responsible for parsing the MM7 message and notifying the listener that a new message has arrived.

In summary, the following steps were performed to create the receive MMS application:

- Implement the `javax.sams.MessageListener` interface that will receive messages and save them in a folder.
- Create an init Servlet that registers the `javax.sams.MessageListener` implementation with the *SAMS-M* API.
- Configure the init Servlet via the `web.xml` deployment descriptor and set it to be loaded on startup of the application.
- Configure the `com.ericsson.mm7.MmsReceiverServlet` Servlet that receives incoming MM7 messages in the `web.xml` file.

For more information about web applications, please refer to section 2.2.3.

The code snippet in Example 3 below shows part of the `MmsListener` class that implements the `javax.sams.MessageListener` interface. This class also defines a `startListener()` method that is called by the init Servlet to register the current instance with the SAMS-M API.

```
public class MmsListener implements javax.sams.MessageListener {

  //This method is called by the SAMS-M API when a message is
received
  public void onMessage(Message message) throws
InvalidArgumentException, ServiceException {
     if (message instanceof javax.sams.messaging.mms.MmsMessage) {
            MmsMessage mmsMessage = (MmsMessage)message;
            saveMms(mmsMessage);
        }
  }

  private void saveMms(MmsMessage mmsMessage) {
     try {
        ContentPart[] contentArray = mmsMessage.getContents();

        for (int i = 0; i < contentArray.length; i++) {
           byte[] content = contentArray[i].getContent();
           String name = contentArray[i].getLocation();
           File folder = new File(folderStr);
           FileOutputStream out = new FileOutputStream(new File(folder
+ name));
           out.write(data);
           out.flush();
           out.close();

        }
     } catch (Exception e) {
        e.printStackTrace();
     }
  }

  public void startListener(){
     //Register this MmsListener to listener handler
     try{

        MmsMessageListenerHandler mmsMessageListenerHandler = null;

        ServiceFactory factory = new ServiceFactory();
        Service service = factory.getService(MessagingSession.class,
"mms");

        MmsMessagingSession session =
           (MmsMessagingSession)service.openSession(new Hashtable());

          mmsMessageListenerHandler =
(MmsMessageListenerHandler)session.getMessageListenerHandler();

          mmsMessageListenerHandler.setDefaultMessageListener(this);
          System.out.println("Listening to new MMS!");
     } catch(Exception se){
        System.err.print(se);
     }
  }
}
```

Example 3. SAMS-M MmsListener implementation example.

The code snippet depicted in Example 4 is part of the init Servlet that registers the listener.

```
public class InitServlet extends javax.Servlet.http.HttpServlet
implements javax.Servlet.Servlet {

  @Override
  public void init() throws ServletException {
    super.init();

    MmsListener listener = new MmsListener();
    listener.startListener();

  }

}
```

Example 4. InitServlet code that is used to start listening to incoming MMS messages

Example 5 contains an XML code snippet which is part of the `web.xml` file that defines the application as a web application to a Java web container such as, for example, *Apache Tomcat* [51].

```
<Servlet>
  <Servlet-name>MmsReceiverServlet</Servlet-name>
  <Servlet-class>com.ericsson.mm7.MmsReceiverServlet</Servlet-class>
</Servlet>
<Servlet>
  <Servlet-name>InitServlet</Servlet-name>
  <Servlet-class>mms.receive.InitServlet</Servlet-class>
  <load-on-startup>1</load-on-startup>
</Servlet>
<Servlet-mapping>
  <Servlet-name>MmsReceiverServlet</Servlet-name>
  <url-pattern>/MmsReceiverServlet</url-pattern>
</Servlet-mapping>
```

Example 5. Servlet mappings for the receive MMS application.

4.1.3. Determining interesting data in MMS SMIL

In order to identify which elements and attributes are available in MMS messages sent from different mobile phones and whether they contain valid data, MMS messages were composed (containing roughly the same content) and sent from the following types of phones and the resulting *MMS SMIL* analyzed:

- *Motorola MotoRAZR V3*
- *Nokia N70*
- *Nokia 5140*
- *Samsung SGH-Z560*
- *Sony Ericsson K800*
- *Sony Ericsson P910*
- *Sony Ericsson P990*

Appendix B contains a selection of the resulting *MMS SMIL* files which were analyzed.

The main conclusion drawn after a closer look at the different SMIL files (for these messages) is that they differ a lot in terms of which attributes are set and what their values are. However, the following data is present in all of these samples:

- Order of the region elements, e.g. media or text region on top.
- Which media/text element should be displayed on which slide.
- File name of media/text element.[*]
- Duration of each slide and element.

The dimension attributes were not related to the actual size of the content included in the message. Instead, all of the terminals that were tested, used fixed height and width attributes. Thus, the dimension attributes will be disregarded in order to transform MMS messages to a HTML version that is displayed similarly to how they would be displayed on a mobile terminal.

Table 4 contains a list of elements and attributes that are defined in the *MMS SMIL* specification, the occurrence of these in *MMS SMIL* from different vendors, and how these elements and attributes are processed by the transformation components.

Table 4. MMS SMIL elements and attributes and how they should be processed by the transformation components.

Name	Occurrence	Processing notes
`smil`	Always	The root element. Occurs with or without a namespace declaration
`head`	Always	Contains no interesting data
`layout`	Always	Contains no interesting data
`root-layout`	Always	Contains `backgroundColor`, `height` and `width` attributes. The `backgroundColor` attribute contains background color information for the MMS message. This attribute will be fully supported. The `height` and `width` attributes will be ignored as discussed earlier in this section.

[*] Not true for the Nokia 5140 where the content type must be used to generate the actual file name.

Name	Occurrence	Processing notes
`region`	Always	Contains left, top, height, width, fit and id attributes. Order of occurrence of regions with IDs `“Text”` and `“Image”` respectively will be used to determine which type of layout applies to a certain MMS message (e.g. text on top or image/video on top) All attributes will be ignored and the layout determined by the transformation components.
`body`	Always	Contains no interesting data
`par`	Always	Contains `dur` attribute The `dur` attribute will be used to determine how long a slide is displayed.
`text`	When applicable	Contains `src`, `region`, `alt`, `begin`, `end` and `dur` attributes. The `src` attribute contains the content ID of the text file to which it refers. If the `src` attribute is not a valid file name, it must be replaced with a valid file name before the *MMS SMIL* is transformed to an HTML view. The text file referenced by the source attribute is read and inserted into the HTML view during transformation. All other attributes are ignored.
`img`	When applicable	Contains `src`, `region`, `alt`, `begin`, `end` and `dur` attributes. The `src` attribute contains the content ID of the image file to which it refers. If the `src` attribute is not a valid file name, it must be replaced with a valid file name before the *MMS SMIL* is transformed to an HTML view. All other attributes are ignored.

Name	Occurrence	Processing notes
`audio`	When applicable	Contains `src`, `region`, `alt`, `begin`, `end` and `dur` attributes. The `src` attribute contains the content ID of the audio file to which it refers. If the `src` attribute is not a valid file name, it must be replaced with a valid file name before the *MMS SMIL* is transformed to an HTML view. All other attributes are ignored.
`video`	When applicable	Contains `src`, `region`, `alt`, `begin`, `end` and `dur` attributes. The `src` attribute contains the content ID of the video file to which it refers. If the `src` attribute is not a valid file name, it must be replaced with a valid file name before the *MMS SMIL* is transformed to an HTML view. All other attributes are ignored.
`ref`	Never	Not supported.

4.1.4. Implementation of HTML+TIME static templates

The next step in creating the transformation components was defining what the end result should be in various scenarios, in order to transform the content into the correct *HTML+TIME* format necessary in order to achieve the desired output look and behavior.

4.1.4.1. Difficulties

While creating the HTML+TIME view templates, some problems were discovered:

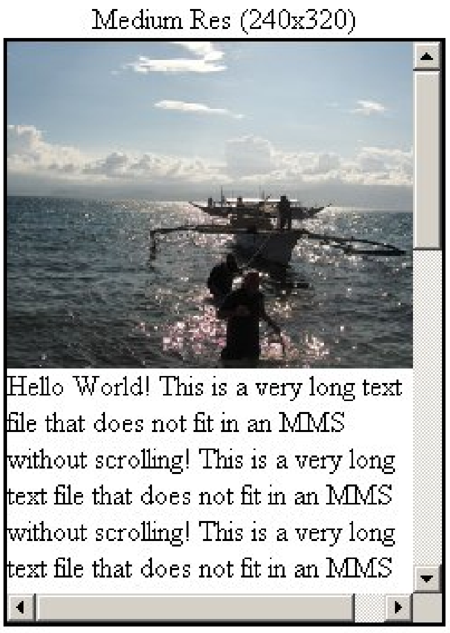
- If the content overflows vertically, a scrollbar is added on the right side to allow scrolling. However, this reduces the available width to display the content. Because the width of an image or video element is often set to the width of the presentation area, this will also add a horizontal scroll bar because the image/video element no longer fits in the width of the column. This problem is illustrated in Figure 4 below.

- Playback of a video or audio element needs to be started when a certain slide is played.

- Scaling settings have to be implemented for video in different media players. The supported scaling settings are defined in the Javadoc documentation; see Appendix F (class `TransformationConfiguration`).

- It is desirable to display a splash-screen* before and after MMS message playback.

- When a media player plug-in is embedded on a slide it can be started using *JavaScript* methods invoked by *HTML+TIME* events. However, if the media takes several seconds to load, then the playback of the embedded media player will not be synchronized with the timing of the slide. This means that the last few seconds of the media file will not be played because the slide change causes the media player to stop playback **before** it has played the final part of this media.



Hello World! This is a very long text file that does not fit in an MMS without scrolling! This is a very long text file that does not fit in an MMS without scrolling! This is a very long text file that does not fit in an MMS

Figure 4. Horizontal scroll bar automatically added because of the addition of the vertical scroll bar.

4.1.4.2. Solutions

Asynchronous JavaScript method calls

The problems concerning content overflow and starting playback of video or audio when a slide is played can be solved through an event in *HTML+TIME*, the `onbegin` event occurs when timed elements are started. This can be used to address the various difficulties noted above, specifically:

- We can resize all child elements of the presentation area to the current available width using *JavaScript*.

- We can start playback of an embedded media player using *JavaScript*.

* A splash-screen is an image that is displayed before an MMS has started playing and after playback finishes (unless `repeatCount` is set to `"indefinite"`). The splash screen can be customized to show, for example, advertisements.

However, using this event mechanism does not solve the entire problem. As the event occurs just *before* the browser "inserts" the timed element. This results in two difficulties:

- Resizing of the image fails because the elements are not yet in the presentation area, so there is not yet a scroll bar limiting the available width.

- The embedded media player does not yet exist, thus it can not be started.

Fortunately both these difficulties can be solved by calling the resizing/play methods **asynchronously** by declaring an inner function and starting it using the `setTimeout` method. An example of how this is technique is used to start playback of an embedded *Apple QuickTime* player is given in Example 6:

```
/**
 * Call play on QuickTime embedded element asyncronously
 */
function playQt(player) {

  //Define the function
  var theFunction = function() {
    player.Play();
  };

  //Do asynchronous call
  setTimeout(theFunction, 0);

}
```

Example 6. Asynchronous JavaScript method to start playback of an embedded QuickTime player.

Complete implementations of the *JavaScripts* that are used to start and stop playback for *Apple's QuickTime*, *Real Network's RealPlayer* and *Microsoft's Windows Media Player* and to resize elements are included in Appendix C.2..

Handling video scaling in different media players

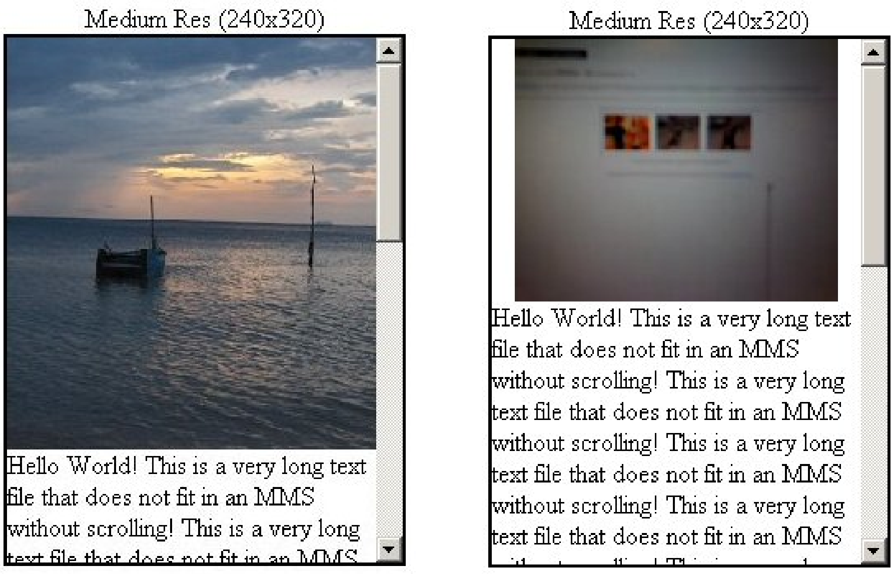
Different media players enable scaling functionality through different attributes. The attributes used in different media players to achieve these scaling settings are shown in Table 5 along with their respective values.

Table 5. Scaling settings for media players.

Scaling setting	Apple's QuickTime	RealNetworks' RealPlayer	Microsoft's Windows Media Player
Original size	`scale="1"`	`center="true"` `maintainaspect="true"`	Not supported
Stretch	`scale="tofit"`	`center="false"` `maintainaspect="false"`	always
Stretch but retain aspect ratio	`scale="aspect"`	`center="false"` `maintainaspect="true"`	Not supported

4.1.4.3. Result

The end result was successful and all the different slide compositions were displayed satisfactorily. Figure 5 illustrates two examples of transformed output during playback and Figure 6 shows a static template of the splash screen and the playback control bar. However, no means of dynamically adapting the duration of a slide to allow synchronization with the media player was discovered.



Hello World! This is a very long text file that does not fit in an MMS without scrolling! This is a very long text file that does not fit in an MMS

Hello World! This is a very long text file that does not fit in an MMS without scrolling! This is a very long text file that does not fit in an MMS without scrolling! This is a very long text file that does not fit in an MMS without scrolling! This is a very long text file that does not fit in an MMS

Figure 5. HTML+TIME template results. (left) Image + text and (right) video + text.

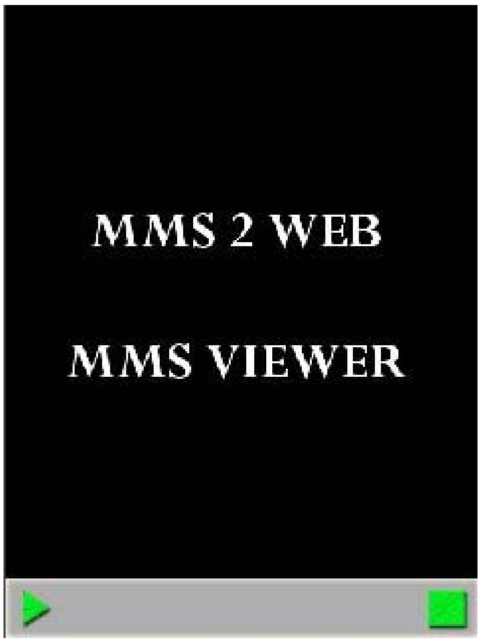


Figure 6. HTML+TIME template with splash screen and playback control.

4.1.5. Implementation of DHTML static templates

The next step in creating the transformation components was defining what the end result should be in a transformation to DHTML, in order to achieve the desired output look and behavior.

4.1.5.1. Difficulties

While creating the DHTML view templates, some problems were discovered (their solutions are addressed in the next section):

- Implementing a timed slide-show presentation using DHTML.
- Not starting the timeout of a slide until an embedded media player has started playing is necessary to avoid clipping the end of the media file.
- While implementing the static templates for DHTML, the templates were tested on more recent versions of *Microsoft Windows*[*] using several different browsers[†]. These tests revealed an issue with respect to embedding media players in recent versions *Microsoft's Internet Explorer* and *Opera's Opera*. The issue is that media player controls in these browsers have to be manually clicked by the user before they are activated and can be used.

[*] The templates were tested on Windows XP SP 2 and Windows Vista.

[†] Specifically: Microsoft's Internet Explorer 7.0, Mozilla's Firefox 2.0, Apple's Safari 3 public beta, and Opera's Opera 9

- *RealNetworks' RealPlayer* plug-in does not seem support *JavaScript* control of playback from: *Opera's Opera* and *Apple's Safari*.

- There are some compatibility issues between *Apple's QuickTime 7.1.6* plug-in and *Mozilla's Firefox* which randomly causes the browser to crash (other versions of *Apple's QuickTime*, specifically more recent versions, work without any problem)

- It requires a lot of time to support a number of media players, as new code has to developed for each one.

4.1.5.2. Solutions

Implementing the slide show effect

The slide show effect of MMS messages is achieved using a series of HTML `div` elements which are put on top of each other. To alternate between the slides (and the splash-screen) the CSS style attribute `visibility` is set to `visible` for the currently displayed `div` and to `hidden` for all other `div` elements. The timing of the slides is handled using *JavaScript* timers.

Waiting for media player playback to start

The browser plug-ins for both *Apple's QuickTime* and *RealNetworks' RealPlayer* include *JavaScript* methods to check the current state of the media player. These methods are used to determine if playback of a media file has started or not, thus instead of immediately starting the timer that invokes the method that will display the next slide: if playback has not started, the state of the media player is iteratively checked (after a small time delay) until playback has started and the timeout to show the next slide is started at this point.

Avoiding click-to-activate for embedded media players

The requirement for a user to click a media player element in order to activate it can be avoided by using *JavaScript* code to dynamically insert the elements of the embedded media player. This method is described in *Microsoft's Microsoft Developer Network* (MSDN) documentation [43] and requires that an external *JavaScript* file is used to insert the embedded media player. To enable the control to be dynamic with respect to different attributes and parameters, this requires that a *JavaScript* file is dynamically created for each MMS message.

Avoiding media player/browser incompatibilities

Because no means of working around the above mentioned media player/browser incompatibility issues were discovered, this will be handled by adding configuration parameters that define what media player/browser combinations will be allowed when transforming to an HTML view.

Removing support for Microsoft's Windows Media Player

There is significant work involved in creating static views that works with each individual media player. Due to this, the limited time available for this thesis project, and the fact that *Microsoft's Windows Media Player* does not support the formats that are typically used for audio and video in MMS messages (see section 4.1.8), support for this player was not included for the DHTML views.

4.1.5.3. Result

The end result was successful and all the different slide compositions were displayed satisfactorily. Exploiting the fact that the playback state of media elements can be checked, enables the created DHTML views to solve the problem of media being clipped at the end, unlike the situation when using an *HTML+TIME* view (section 4.1.4). The end result of the DHTML template with an updated playback control bar and splash screen are shown in Figure 7.

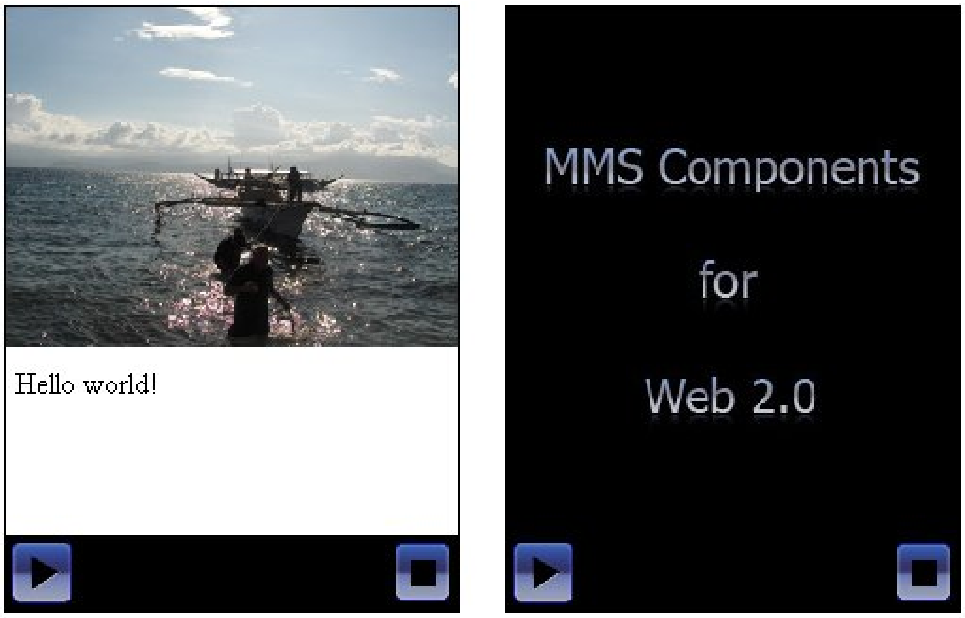


Figure 7. DHTML template results.

4.1.6. Transformation requirements

4.1.6.1. Introduction

To get an idea of what capabilities are required for the transformation components, the following examples show an *MMS SMIL* file and the same *MMS SMIL* file after it has been transformed to valid *HTML+TIME*, such that the content can be displayed in *Microsoft's Internet Explorer*. This is depicted in examples 7 and 8.

```
<smil>
  <head>
    <meta name="generator" content="SEMC-UIQSMARTPHONE-P990i" />
    <layout>
      <root-layout width="200px" height="200px"
background-color="white" backgroundColor="white" />
        <region id="Image" top="0%" height="50%" fit="meet"
background-color="white" backgroundColor="white" />
        <region id="Text" top="50%" height="50%" fit="meet"
background-color="white" backgroundColor="white" />
    </layout>
  </head>
  <body>
  <par dur="5000ms">
    <img src="similan.jpg" region="Image" end="5000ms" />
    <text src="text.txt" region="Text">
      <param name="foreground-color" value="black" />
      <param name="textsize" value="small" />
    </text>
  </par>
  </body>
</smil>
```

Example 7. Example MMS SMIL file.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<html xmlns:t="urn:schemas-microsoft-com:time">
  <head>

    <?import namespace="t" implementation="#default#time2">

    <style>.t {behavior: url(#default#time2)}</style>
  </head>
  <body>
    <div style="background-color: white; width: 200px; heigth:
200px;">
      <t:par>
        <t:seq>
          <img class="t" src="similan.jpg" style="width: 100%;
height: 50%" dur="5000ms"/><br/>
        </t:seq>
        <t:seq>
          <div class="t" style="width: 100%; height: 50%;
overflow: scroll; font-size: small; color: black;" dur="5000ms">
            The beaches at the Similan Islands.
          </div>
        </t:seq>
      </t:par>
    </div>
  </body>
</html>
```

Example 8. Example of MMS SMIL transformed to HTML+TIME.

The above examples make it possible to define some specific requirements upon the transformation capabilities. These capability requirements are listed and explained in section 4.1.6.2.

4.1.6.2. Requirements

The requirements for the MMS transformation technologies that will be utilized by the transformation components are listed in Table 6.

Table 6. Requirements for transformation technologies.

Requirement	**Description**
Read and parse *MMS SMIL*	In order to transform an *MMS SMIL* file to another format, it must be read and parsed so that data from it can be added to the resulting HTML view. Though this requirement seems very basic, there were some issues parsing incoming *MMS SMIL* files using JAXB, this is described in more detail in section 4.1.7.
Generate output XML	The transformation components must also be able to output XML data in the form of *HTML+TIME* web pages.
Enable string processing	Basic string processing including functions that can be used to concatenate and split strings is required. For instance, it must be possible to use data from several attributes in the input *MMS SMIL* file and concatenate it to a singe style attribute of an *HTML+TIME* element.
Generate XML with mixed namespaces and processing instructions	*HTML+TIME* web pages require a processing instruction to specify that the content on a given web page should be treated as *HTML+TIME*. *HTML+TIME* elements must also have a namespace prefix (the prefix “t” for time is used for all *HTML+TIME* tags throughout this master’s thesis).
Generate output based on settings	The output generated by the transformation must be flexible to enable a web site author to choose how to display the MMS messages in different scenarios. For more details, see section 4.2 and the Javadoc documentation in Appendix F (class `TransformationConfiguration`).
Java code integration	During implementation of the components it became apparent that it would greatly simplify transformation and generation to have a common Java object representation of a given MMS message. Thus, a requirement that the transformation technology can be integrated with Java was suggested.

Requirement	**Description**
Read and insert text files	In SMIL, text is specified in a separate text file. This text file is then referenced by a media tag with a `src` attribute that specifies the location of the text file. This is not the case for HTML where text is generally embedded in the code. Thus, the text files referenced by an *MMS SMIL* must be read and inserted into the output file.

4.1.7. Transformation technologies

XSLT

XSLT is a very flexible way to easily transform *MMS SMIL* documents to *HTML+TIME* that is supported by *Internet Explorer*. However, its intended use is not to parse XML files into Java object representations. Further more, supporting settings would require that a set of XSL templates be generated for each combination of settings.

JAXB

JAXB provides a way to interpret and generate XML documents using Java code. JAXB was ruled out during testing because it is not flexible and encountered parsing errors for some SMIL files that were received from actual terminals, because these did not match the schemas from which the JAXB classes were generated. Thus, several schemas would have to written and each input file may require several parsing attempts before the parsing will be successful.

SAX

SAX provides the fastest and most memory efficient way to parse and generate XML with great flexibility. Using SAX APIs will result in more code, which possibly will be harder to follow.

Conclusion

An object representation of an MMS message is suggested as the starting point. A properly designed set of classes to represent an MMS message provides a common object tree for different parts of the components. The suggested benefits of such an object tree are:

- To easily accept output from the parsing parts of the components
- To provide an easy to interpret input to the HTML view generation parts of the components
- To provide a simple to understand input format for a function for writing *MMS SMIL* to an output stream
- Enables run-time editing of an MMS message
- Enables easy MMS authoring through Java code

The decision to utilize this object representation of MMS messages immediately excludes XSLT. While JAXB requires many different complicated schemas to represent each input and output file format. This later approach quickly becomes difficult to handle and is likely to have poor transformation performance as a given input file may have to be parsed several times before it is successfully transformed. Thus, SAX is the suggested transformation technology, as it provides all required the features, is very flexible, and has the best performance for such a transformation-oriented application. However, while not directly related to transformation, JAXB is suggested as the XML technology to use when directly writing *MMS SMIL* to an output stream. The reason is that very little flexibility is required for this operation and that the *MMS SMIL* output should be well defined (hence only one schema is needed for output).

4.1.8. Determining media player capabilities

A number of tests were made by simply trying to play different media types in the different media players. The test was performed on a computer running Microsoft Windows 2000 with service pack 4. The reason that the tests were not performed on more recent operating system is that no such computer was available at that time. However, since that time, the views generated by the transformation have been tested successfully on both Windows XP SP2 and Windows Vista[*]. The following media players were tested:

- *Microsoft's Internet Explorer*
- *Microsoft's Windows Media Player*
- *Apple's QuickTime*
- *RealNetworks' RealPlayer*

While not all media formats have been tested, the ambition was to test media types that are likely to originate from a mobile phone, as well as some other widely used formats. The results for audio and video are given in tables 7 and 8 respectively.

[*] Not all media formats have been tested on these systems

Table 7. Media player audio support (D – direct support, A – automatic CODEC download support and N – Not supported).

Media player	**AAC**	**AMR**	**AU**	**M4A**	**MIDI**	**MP3**	**WAV**	**WMA**
Microsoft's Internet Explorer 6.0	N	N	D	N	D	D	D	D
Microsoft's Windows Media Player 9.0	N	N	D	N	D	D	D	D
Apple's QuickTime 7.1.6	D	D	D	D	D	D	D	N
RealNetwork's RealPlayer 10.5	D	A	D	D	A	D	D	D

Table 8. Media player video support (D – direct support, A – automatic CODEC download support and N – Not supported).

Media player	**H.263**	**H.264**	**MP4**	**AVI**	**WMV**
Microsoft's Internet Explorer 6.0	N	N	N	D	D
Microsoft's Windows Media Player 9.0	N	N	N	D	D
Apple's QuickTime 7.1.6	D	D	D	D	N
RealNetworks' RealPlayer 10.5	A	A	A	D	D

4.2. Proposed solution

4.2.1. Naming conventions

An important consideration during the design and implementation of the transformation components was to establish a naming convention for media players and browsers so that these can be communicated throughout the components. Thus, a naming convention for media players (Table 9) and browsers (Table 10) is proposed. This naming convention specifies that each media player and browser should have a name which is a case sensitive string that should match java-class-name case conventions.

Table 9. Media player names.

Media player	**Name**
Microsoft's Internet Explorer	`InternetExplorer`
Microsoft's Windows Media Player	`WindowsMediaPlayer`
Apple's QuickTime	`QuickTime`
RealNetworks' RealPlayer	`RealPlayer`

Table 10. Browser names.

Browser	**Name**
Microsoft's Internet Explorer	`InternetExplorer`
Mozilla's Firefox	`Firefox`
Apple's Safari	`Safari`
Opera's Opera	`Opera`
Any other browser without *HTML+TIME* support	Any other string

4.2.2. Transformation components

The transformation components allow developers to transform MMS messages between different formats. Transformation is typically done from *MMS SMIL* to an HTML view of the MMS message. However, to simplify editing of MMS messages and even authoring an MMS message using Java code, the proposed solution specifies a Java object representation of an MMS message. This object is the `MmsMessage` class. It is primarily designed to be an in-memory representation of an *MMS SMIL* file but, also contains additional information that is needed (or may be useful) during transformation.

The proposed solution also specifies the `MmsTransformer` class. This class is used to perform a set of different transformations such as parsing *MMS SMIL* and generating HTML views from an MMS message. The `MmsTransformer` class uses different implementations of the `HtmlViewGenerator` interface to generate different HTML views. Which implementation is used depends upon which browser is specified in the `TransformationConfiguration`. This is also passed to the transformation method. There are additional settings available through the `TransformationConfiguration`, for more details see the Javadoc in Appendix F.

Another class that is fundamental to understanding the proposed solution is the `DataSourceFolder` interface. Implementations of this interface are used to serve files to the transformation components. This interface is required because to perform a transformation to an HTML view, data from the media files is required (e.g. the text contents of a text file). All the data access is handled through this interface, rather than reading directly from the file system, this allows the media file data to originate from any data source. Such data sources include: hard-drives, databases, and even memory.

A brief description of the core classes and interfaces for the proposed solution are given in Table 11. For further details about the implementation, please refer to the Javadoc which is included as Appendix F.

Table 11. Core MMS transformation classes method overview.

Class/Interface	**Method**	**Description**
`MmsTransformer`	`newInstance()`	A static method used to create a new `MmsTransformer` instance. Because instantiation is resource expensive, it is recommended to use the `TransformerPool` class to obtain an `MmsTransformer`.
	`parseMmsSmil(InputStream smilStream)`	A SAX parser is used to parse the *MMS SMIL* to an `MmsMessage` object.

Class/Interface	Method	Description
	`transform(MmsMessage mms, OutputStream output, TransformationConfiguration config)`	Transforms the specified `MmsMessage` to an HTML view based on the settings provided in the `TransformationConfiguration`. After the transformation is completed, the resulting HTML is written to the specified `OutputStream`. This method requires that a `DataSourceFolder` has been set on the `MmsMessage`.
	`transform(InputStream smilStream, DataSourceFolder mmsFolder, OutputStream output, TransformationConfiguration config)`	Transforms the specified *MMS SMIL* specified as an `InputStream` to an HTML view based on the settings provided in the `TransformationConfiguration`. After the transformation is completed the resulting HTML is written to the specified `OutputStream`.

`HtmlViewGenerator` (interface)	`generate(MmsMessage message, OutputStream output, TransformationConfiguration config)`	This method is called by the transform-methods of the `MmsTransformer` class and used to transform an MMS message to a specific HTML view implementation.
		
		Included implementations:
		
		`HtmlTimeGenerator`
		
		`DhtmlGenerator`

`Transformation`
 	
`Configuration`
	-	This class extends `java.utils.Properties`. It defines a set of static fields that define property names for different settings that are processed during the transformation process.

Class/Interface	Method	Description
`DataSourceFolder` (interface)	`getDataSource(URI relativeUri)`	Returns a `javax.activation.DataSource` corresponding to the specified file name or null if the file was not found in the folder. Included implementations: `FileDataSourceFolder`

The sequence diagram in Figure 8 shows how the transformation components can be used by an application to generate an HTML view from *MMS SMIL*.

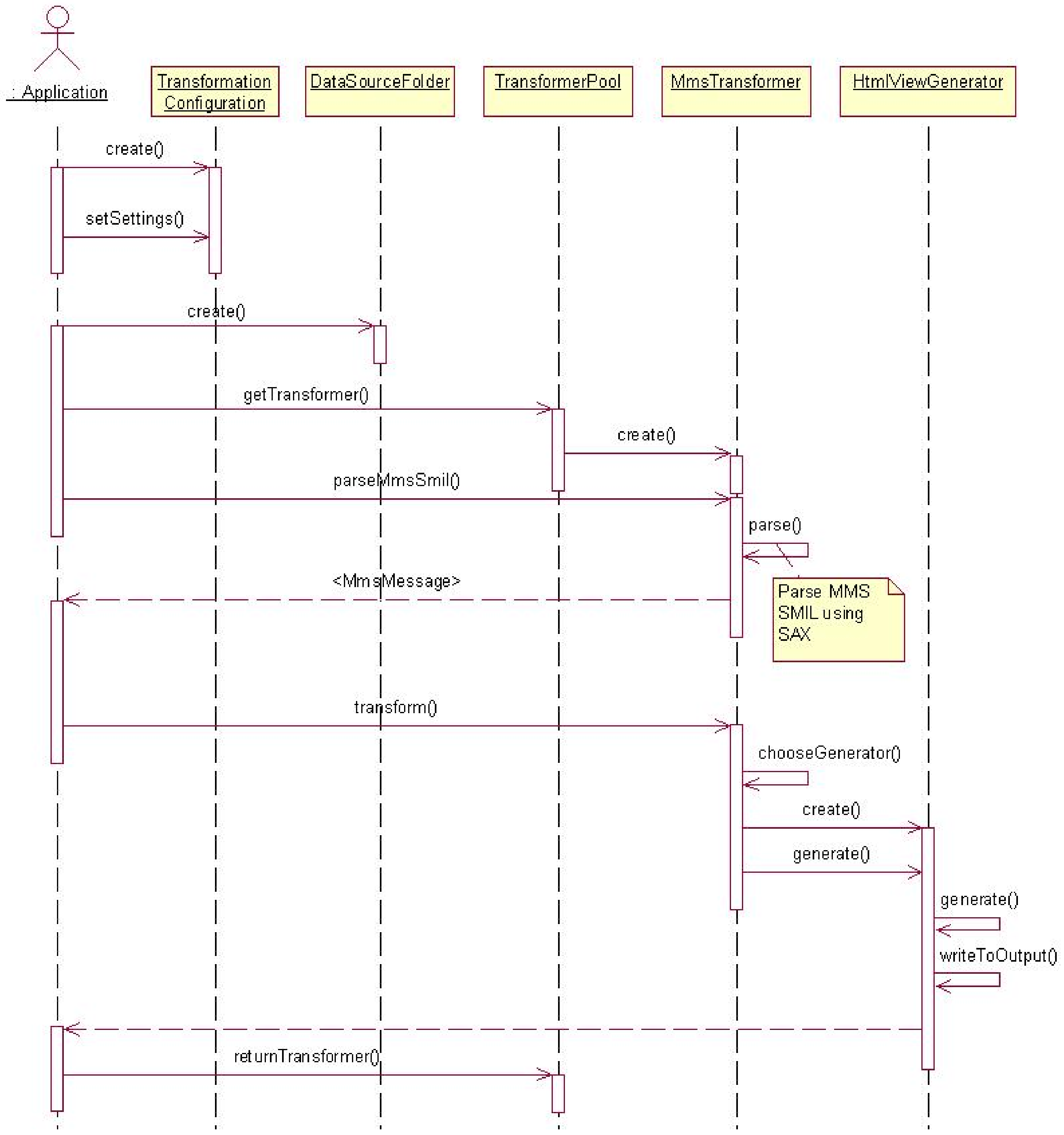


Figure 8. Transformation sequence diagram. This sequence diagram illustrates the interaction between an application and the different Java classes used to transform an MMS SMIL to an HTML view.

4.2.3. SMIL viewer components

The SMIL viewer components are used to display a transformed MMS message on a web page. The viewer components consist of the `MmsTransformationServlet` and the custom JSP tag `mms.tag`. The `MmsTransformationServlet` is responsible for calling the appropriate transformation components and returning an HTML view or an appropriate error page (if something went wrong during the transformation). The tag is responsible for adding the HTML code that requests the transformed MMS message from the `MmsTransformationServlet`. The tag implements this behavior by adding an IFrame on the page with `src` attribute set to point to the `MmsTransformationServlet`. The `src` attribute also contains the applicable parameters needed to display the MMS message.

Example 9 below shows a very simple example of how a JSP page could display an MMS message using the `viewer.tag` custom tag. The example assumes that the `viewer.tag` tag file is located in the `WEB-INF/tags` directory of the web application.

```
<%@ taglib prefix="mms" tagdir="/WEB-INF/tags" %>
<html>
  <head>
    <title>MMS Test</title>
  </head>
  <body>
    <mms:viewer width="240px"
                 height="320px"
                 mmsDir="/mms/mms1"
                 smil="s.smil"
                 servletPath="MmsTransformationServlet">
    </mms:viewer>
  </body>
</html>
```

Example 9. MMS viewer tag example code

Figure 9 shows a sequence diagram of the steps performed by the JSP page, the `mms.tag` custom tag and the `MmsTransformationServlet`:

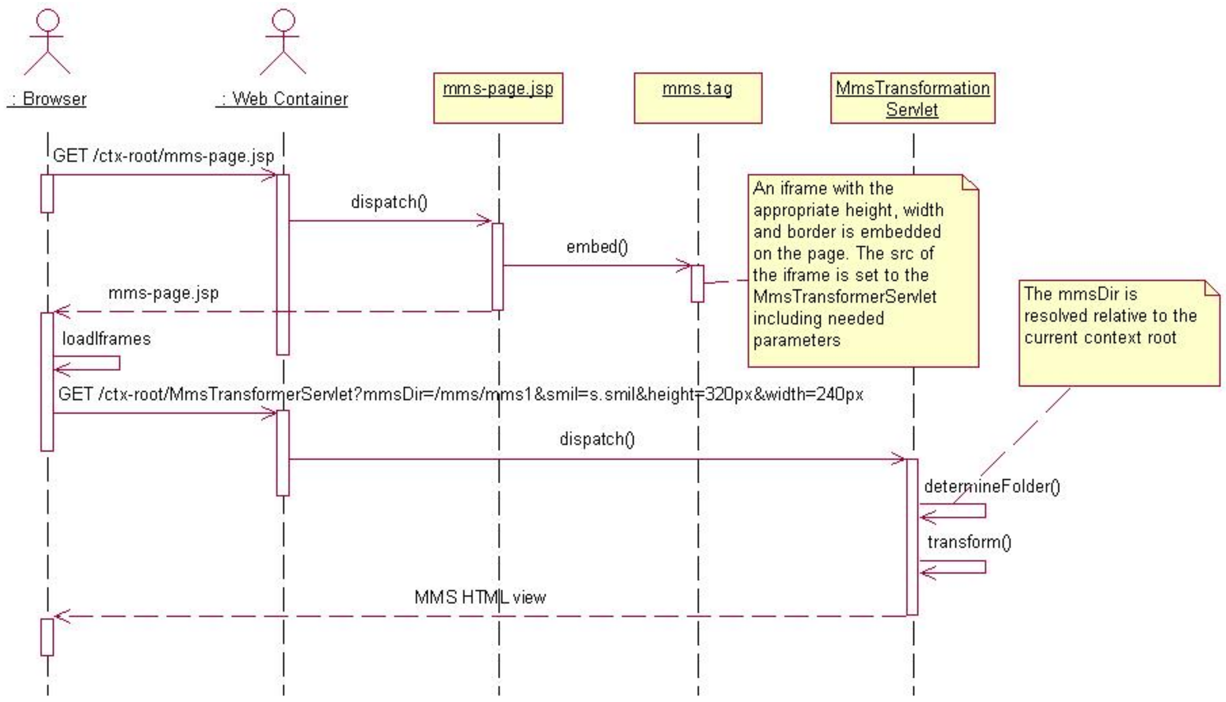

: Browser
: Web Container
mms-page.jsp
mms.tag
MmsTransformation Servlet
GET /ctx-root/mms-page.jsp
dispatch()
embed()
An iframe with the appropriate height, width and border is embedded on the page. The src of the iframe is set to the MmsTransformerServlet including needed parameters
mms-page.jsp
loadIframes
The mmsDir is resolved relative to the current context root
GET /ctx-root/MmsTransformerServlet?mmsDir=/mms/mms1&smil=s.smil&height=320px&width=240px
dispatch()
determineFolder()
transform()
MMS HTML view

Figure 9. Viewer components sequence diagram. This sequence diagram illustrates the interaction between a browser and different parts of the MMS components that are used to transform an MMS message to HTML.

The sequence diagram in Figure 9 shows the most trivial scenario where the MMS message contains no complex content and the transformation was successful. However, there are a number of reasons why a transformation may fail. In failure scenarios the `MmsTransformationServlet` will dispatch the view to a web page that is responsible for notifying the end user of what went wrong. There are three distinct error scenarios which have different error pages which could be presented to the end user. The relative URL of each error page can be customized as an init parameter to the `MmsTransformationServlet`. These errors, their causes, and which error page is to be shown to the user are described in Table 12.

Table 12. Error pages for the MmsTransformationServlet.

Cause	Error page	Responsibility of the error page
Transformation failed because the MMS message that was transformed contained audio or video (requires an embedded media player) and no information about what media players were available in the end user's browser was included in the request to the `MmsTransformationServlet`.	Custom error page can be specified with the init parameter name: `loading-page` Default value: `/mms-loading.jsp`	This error page will always appear the first time a user tries to view an MMS message that contains audio or video elements. The responsibility of this page is to use *JavaScript* to determine which media players are available in the end user's browser and then reload itself with information about which media players were available. This is done by appending the `"players"` parameter to the location of the IFrame in which the page is loaded. Once the `"players"` parameter has been specified to the `MmsTransformationServlet` its value will be stored in the user's session, thus the detection will only be performed once[*].

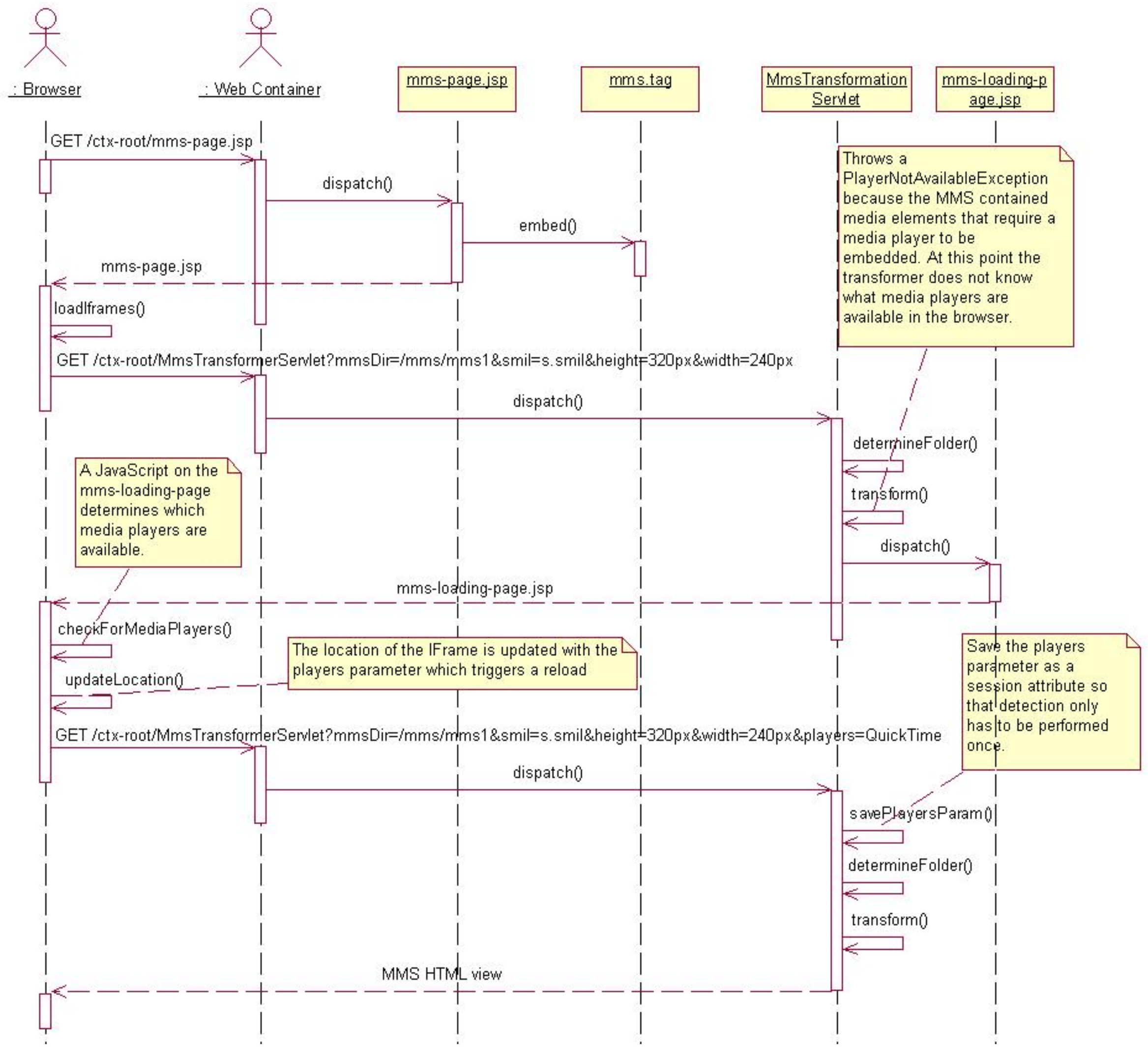
[*] If several MMS messages are loaded simultaneously, the detection may occur several times for one user.

Cause	Error page	Responsibility of the error page
Transformation failed because the MMS message that was transformed contained audio or video (requires an embedded media player) and no information about what media players were available in the end user's browser was included in the request to the `MmsTransformationServlet`.	Custom error page can be specified with the init parameter name: `loading-page` Default value: `/mms-loading.jsp`	This error page will always appear the first time a user tries to view an MMS message that contains audio or video elements. The responsibility of this page is to use *JavaScript* to determine which media players are available in the end user's browser and then reload itself with information about which media players were available. This is done by appending the `"players"` parameter to the location of the IFrame in which the page is loaded. Once the `"players"` parameter has been specified to the `MmsTransformationServlet` its value will be stored in the user's session, thus the detection will only be performed once[*].
Transformation failed because the MMS message that was transformed contained audio or video for which playback is not supported by any of the media players that were specified in the `"players"` parameter when media players were detected on the "`loading-page`". This will only occur if the `THROW_PLAYER_NOT_AVAILABLE` setting of the `TransformationConfiguration` is set to true. Otherwise the media will be excluded from the generated HTML view instead.	Custom error page can be specified with the init parameter name: `no-player-page` Default value: `/mms-no-player.jsp`	The responsibility of this page is to notify the end user that there was no available media player available in the browser that allows playback of a media element. A link to a download page for such a media player could also be included on the page. To enable the page to display a useful error message/link, a `PlayerNotAvailableBean` is added as a request-scope attribute using the key: `"playerInfo"`. The `PlayerNotAvailableBean` contains information about the file extension of the file for which no media player was found and which media players were detected on the `loading-page`.

[*] If several MMS messages are loaded simultaneously, the detection may occur several times for one user.

Cause	Error page	Responsibility of the error page
Transformation failed for any other reason.	Custom error page can be specified with the init parameter name: `error-page` Default value: `/mms-error.jsp`	The responsibility of this page is to display a message to end user explaining that there was an internal error and that the MMS message can not be displayed.

To show the quite complex flow of events involved when an MMS message contains audio or video elements, Figure 10 shows a sequence diagram that depicts the flow of events in such a scenario.



Throws a PlayerNotAvailableException because the MMS contained media elements that require a media player to be embedded. At this point the transformer does not know what media players are available in the browser.

A JavaScript on the mms-loading-page determines which media players are available.

The location of the IFrame is updated with the players parameter which triggers a reload

Save the players parameter as a session attribute so that detection only has to be performed once.

Figure 10 Viewer components sequence diagram. This sequence diagram illustrates the interaction between a browser and different parts of the MMS components that are used to transform an MMS message to HTML when the available media players must be detected.

4.2.4. MMS composer components

The MMS composer components consist of one set of components that are embedded directly on a web page. These components comprise the *MMS canvas*. The *MMS canvas* components include a snippet of HTML code and a large set of *JavaScript* methods used to “draw” the MMS message on the canvas; these *JavaScripts* are included in Appendix C.4. The HTML code acts as a placeholder for the canvas and is initially displayed to the user as an empty white area. It is upon this white area that the MMS will be displayed while it is being composed. The HTML code snippet includes a form that is used to submit the composed MMS message to the server. This form includes a number of settings that are specified as hidden HTML `input` elements.

Example 10 below shows a very simple example of how a JSP page could add the *MMS canvas* using the JSP custom tag `composer.tag`. The example assumes that the `composer.tag` tag file is located in the `WEB-INF/tags` directory of the web application.

```
<%@ taglib prefix="mms" tagdir="/WEB-INF/tags" %>
<html>
  <head>
    <title>MMS Test</title>
  </head>
  <body>
    <mms:composer height="320px"
                  width="240px"
                  jsUri="mms-resources/scripts/mms-composer.js"
                  servletPath="MmsComposerServlet"
                  forwardPath="mms-composer-ok.jsp"
                  smilFile="s.smil"
                  targetDir="/mms/mmstest" />
  </body>
</html>
```

Example 10. MMS canvas JSP tag

[illegible] to create the MMS message. The reason for this design decision is that these *JavaScript* calls would be tightly coupled to the look and feel of the page around the *MMS canvas* and the look and feel should be implemented by the developer who creates the web page.

The *MMS canvas* supports adding of audio and video clips. However, it does not play back video and audio clips as they are added to the MMS message. Instead, when a video clip is added, an image is displayed where the video clip will be placed in the resulting MMS message. Audio clips are not displayed on the *MMS canvas*.

When an MMS message is edited using *JavaScript* methods of the *MMS canvas*, two representations of the MMS message are updated. First, the actual HTML tags on the page will be dynamically updated so that a current view of the MMS message is displayed. Secondly, a *JavaScript* object representation of the MMS message is updated. When the user is finished with the MMS message and invokes the method used to submit it to the server, the *JavaScript* object representation of the MMS message is transformed to a *JavaScript* Object Notation (JSON) string before it is submitted to the server.

The second part of the MMS composer components is the `MmsComposerServlet`. This Servlet retrieves the JSON string from the request, parses it and creates an `MmsMessage` Java object based on the composed MMS message. At this point the MMS message is saved as and the request is dispatched to a specified page or Servlet. For more details, please refer to the Javadoc for the `MmsComposerServlet`.

For more information about the MMS composer components, please refer to the user guide that is included in Appendix E, the JavaScript code (including extensive usage information as comments) is included in Appendix C.4., and the Javadoc of the `MmsComposerServlet` is available in Appendix F. The MMS composer components have also been integrated into the emulator that is included in the Telecom Web Services SDK [17], which can be used as a reference when implementing a look and feel for the *MMS canvas*. An example of how the Telecom Web Services Network Emulator can be used to create an MMS message is illustrated in Figure 11.

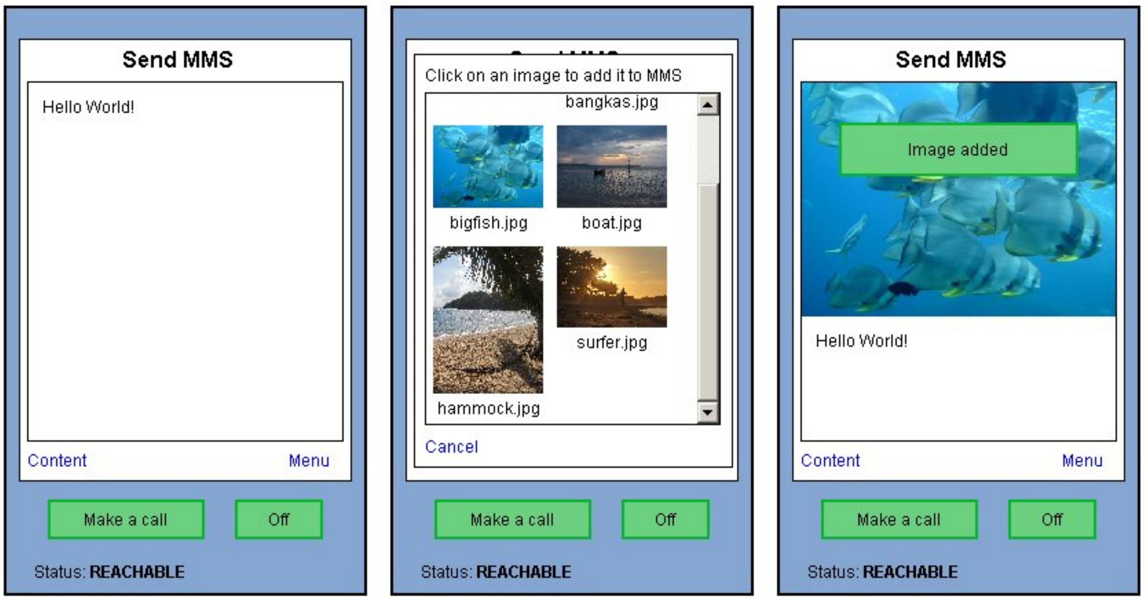


Figure 11. MMS canvas integrated with the Telecom Web Services Network Emulator

4.3. Does this result match the original goals?

The results of the implementation of the components achieve the original goals (see section 3.1) quite well. The transformer components have been implemented and support both an *HTML+TIME* view and a DHTML view. The goal of being able to detect which media player plug-ins were available was reached. Although it would also have been desirable to detect which version of the media player plug-in is available. The viewer components were implemented as JSP custom tags and allow an MMS message to be easily added to a JSP page. The goals for the composer components were all reached. The proposed solution also includes a Java object representation of an MMS message, thus also fulfilling that goal.

However, there are some goals that were not achieved, most of these were assigned low priorities from the beginning. The first of these is the support for transformation to a SMIL format that is supported by media players directly so that an MMS message may be played directly using a media player. The second goal that was not reached was the support for 3GPP SMIL as an alternative input format to the transformation components. One goal that was not fully realized and did have a high priority was the support for logging throughout the components. Another goal that was not achieved was the support for XHTML-MP versions of the components that would allow them to be used on web pages designed for mobile terminals. Furthermore, there was not time to implement JSF versions of the custom tags that would simplify embedding of MMS views and the *MMS canvas* on a web page.

The goals regarding packaging and documenting the components proposed in this thesis (as listed in section 3.1.4) for publication on *Ericsson’s Developer World* have unfortunately not been achieved at the current time. The only one of these goals that has been at least partly met is the writing of the user guide for the components (see Appendix E).

5. Conclusions

5.1. Conclusion

The implementation of the *MMS Components for Web 2.0*, as proposed by this master's thesis, has been a success. The goals with the highest priorities were all reached, thus resulting in a flexible set of components that can be used to integrate MMS messages as a part of Java web application. The proposed solution works with all tested browsers which I believe is very important for the usability of these components.

There are some limitations to the views that are generated by the transformation components (as described in sections 4.1.4 and 4.1.5). These limitations include media file clipping for the HTML+TIME views, and browser/media player incompatibilities for the DHTML views. The first of which is a big problem for MMS messages that do contain audio and/or video elements. The problem could be worked around by manually increasing the duration for slides based on how long it is likely to take a user to fully download and start playing the clip. The browser/media player compatibility issues could be considered small because the components can be configured not to allow these combinations of browsers and media players. Also, the by far most popular browsers (*Microsoft's Internet Explorer* and *Mozilla's Firefox* [55]) have no compatibility issues with the most recent versions of either *Apple's QuickTime* or *RealNetworks' RealPlayer*.

One goal that was not achieved even though it had a high priority is extensive logging of events in the components. Even though the components log at least the most critical errors, the logging capabilities of the components should have been better structured and more extensive. It would also have been useful to implement an AJAX script/Servlet combination that would report debugging information from the Java Scripts to the server.

If I would have had another chance to do this thesis project from the beginning, there are some changes I would have made to the priorities of the different tasks. First of all I would have excluded support for HTML+TIME from the components because implementing the *HTML+TIME* views was very time consuming, not easy to combine with embedded media players, and most of all, the fact that only *Microsoft's Internet Explorer* browser supports *HTML+TIME* means that it would be very restrictive to only support this view. In retrospect I should also mention that I should have had a lower ambition for the number of media players that I spent time trying to support. If I would have stuck to the two players that can play typical MMS message media types (*Apple's QuickTime* and *RealNetworks' RealPlayer*) I would have saved a lot of time.

Although the proposed solution itself is limited to the Java platform, it would be possible to use it on other types of sites using mashup*. This would require that an MMS messaging solution is hosted on a Java-based web server. Creators of other sites could then include the MMS content on their sites using any server side platform such as Microsoft's ASP.NET [56] or PHP [57], using mashup technology.

There has not been time to finish the documentation and packaging of the components proposed in this thesis at the current time. Documentation and packaging of the components will take place after this thesis project is completed and will be released on the *Ericsson Developer Program* [52] home page during quarter 1 2008.

* A mashup-site is a site that combines data from several different sources.

In conclusion I think that the components proposed in this master's thesis provide a solid solution for integrating MMS messaging capabilities on the web and I believe that it has the potential to bring content from mobile phones closer together with other multimedia as it is shared on the web today.

5.2. Future work

During the course of this master's thesis a number of ideas for additional future work were generated. These ideas are listed below in Table 13.

Table 13. Future work descriptions

Task	**Description**
Creating a demo application	After this thesis project is completed, it will complemented with a small demo application that shows how make use of the MMS components both to view and compose MMS messages. This demo application will be included as part of the release of *MMS components for Web 2.0* on the *Ericsson Developer Program* [52] home page. The demo application should also utilize the *MM7 SDK* [39] and the Telecom Web Services SDK [17] to demonstrate how to connect to service providers to and send/receive MMS messages using MM7 and Parlay X Web Services respectively.
Writing a design guide	After this thesis project is completed, it will be complemented with a design guide of the implementation of the *MMS components for Web 2.0* as proposed in this thesis project.
Test the *MMS SMIL* with mobile terminals	It would be quite useful to test whether the MMS messages created by the MMS composer components are displayed as intended on a large variety of mobile phones. Messages with many different slide compositions should also be tested across these terminals in order to ensure that *MMS SMIL* that is generated (though it already conforms to *MMS SMIL*) will be displayed correctly on most modern mobile phones.
Adding a content-serving tag	Adding a JSP tag that serves the content of an MMS message as a sorted array would be a useful addition to the components. Such a tag could be used to embed the contents of an MMS message on a web page in any way the author of a web page desires – not only as a timed slide show.

Task	**Description**
Extending view/media player selection options	An interesting feature to add to the transformation/viewer components would be an extension of the media player extension *JavaScript* so that it would also detect what version of a certain plug-in is available. The transformation components could utilize this information when making the choice of what media player to embed when generating a specific view. This could for example be used to make sure that a view generated for *Mozilla's Firefox browser* would not use an *Apple QuickTime* plug-in if the available version of *Apple's QuickTime* was 7.1.6, due to the incompatibility described in section 4.1.5.
Transformation to other SMIL formats	The transformation components could be extended with features that transform *MMS SMIL* to a version of SMIL that is displayed well in different SMIL-enabled media players. Such media players include, for example, *Apple's QuickTime* and *RealNetworks' RealPlayer*. This would enable MMS messages to be displayed without a web browser and eliminate any browser/media player plug-in incompatibilities.
Implementing new views	Another interesting feature to add to the components would be generation of more views. Such views might include: • An XHTML version of the DHTML view and the composer components. • An XHTML-MP and ECMAScript version of the DHTML view the composer components to enable mobile browsers to display the components properly. • An HTML version 5 version of the DHTML view and the composer components. This view could utilize the new `video` and `audio` tags to play media content.
Implementing JSF tags	Converting the JSP custom tags to JSF tags should be quite easy and would be a useful complement for web developers who are using JSF.

Task	**Description**
Adding transcoding support	Including transcoding support in the transformation components would be very interesting as it would eliminate the need for the user to have media player that is capable of playing all the media files of an MMS message. This would also eliminate the video/audio clipping problem with *HTML+TIME* view, as discussed in section 4.1.4.
Porting the components to other server-side platforms	To enable web developers who do not use the Java platform as the server side platform to use the transformation components, they could be ported to run work on other platforms.

6. References

[1] Open Mobile Alliance, http://www.openmobilealliance.org/ (last modified 2007-10-30)

[2] *Firefox/Feature Brainstorming:Web Standards Support*, http://wiki.mozilla.org/Firefox/Feature_Brainstorming:Web_Standards_Support (last modified 2007-08-01)

[3] Internet Explorer: Home Page, *http://www.microsoft.com/windows/products/winfamily/ie/default.mspx* (last viewed 2008-01-14)

[4] *Firefox web browser* | *Faster, more secure, & customizable*, http://en.www.mozilla.com/en/firefox/ (last viewed 2008-01-14)

[5] *Apple – Safari 3 public beta,* http://www.apple.com/safari/ (last viewed 2008-01-14)

[6] *XHTML Mobile Profile – Approved version 1.1*, http://www.openmobilealliance.org/release_program/docs/Browsing/V2_1-20061020-A/OMA-WAP-XHTMLMP-V1_1-20061020-A.pdf (last viewed 2007-11-13)

[7] *ECMAScript Languange Specification*, http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-262.pdf (last viewed 2007-11-13)

[8] OMA Multimedia Messaging Service V1.3, release date: 2005-09-27, Specification - *Multimedia Messaging Service Conformance Document,* http://www.openmobilealliance.org/Technical/release_program/mms_v1_3.aspx - OMA-TS-MMS-CONF-V1_3-20051027-C.pdf (last viewed 2008-01-28)

[9] *Jffmpeg: CODEC pack for the Java Media Framework*, http://jffmpeg.sourceforge.net/ (last modified 2005-03-26)

[10] *HTML+TIME*, http://msdn2.microsoft.com/en-us/library/ms533112.aspx (last viewed 2007-09-03)

[11] Facebook, http://www.facebook.com/ (last viewed 2007-11-15)

[12] YouTube, http://www.youtube.com/ (last viewed 2007-11-15)

[13] CNN – *I-Report Toolkit*, http://www.cnn.com/exchange/ireports/toolkit/index.html (last viewed 2008-01-30)

[14] *Wap och Surf* – Tele2 , http://www.tele2.se/tillaggstjanster-wap-och-surf.html (last viewed 2007-11-15)

[15] *Prislista Röstabonnemang Privat* – Tre, http://www.tre.se/upload/Price/PrislistaPrivat071012v10.pdf (last viewed 2007-11-15)

[16] *Telenor abonnemang* – Telenor, http://www.telenor.se/111.jsp?smid=137167 (last viewed 2007-11-15)

[17] Ericsson - *Telecom Web Services SDK v3.0, http://www.ericsson.com/mobilityworld/sub/open/technologies/parlayx/tools/tc_ws_sdk_3_0 (last viewed 2008-01-16)*

[18] *Web service* – Wikipedia, http://en.wikipedia.org/wiki/Web_Services (last modified 2007-11-13)

[19] The Parlay Group, http://www.parlay.org/en/index.asp (last viewed 2007-11-15)

[20] *Web21C SDK* – BT, http://web21c.bt.com/ (last viewed 2007-11-15)

[21] Mobilstart - Telenor, https://mobilstart.telenor.se/web/guest/home (last viewed 2007-11-15)

[22] mms2web.com, http://www.mms2web.com/ (last viewed 2007-11-15)

[23] mobilblogg.nu, http://www.mobilblogg.nu/ (last viewed 2007-11-15)

[24] JDOM, http://www.jdom.org/ (last modified 2006-05-15)

[25] *XSL Transformations (XSLT) - Version 1.0* (W3C recommendation), http://www.w3.org/TR/xslt (2007-08-28)

[26] SUN - *Java SE Technologies at a Glance*, http://java.sun.com/javase/technologies/index.jsp (last viewed 2008-02-05)

[27] Apache – *Apache Ant – Welcome*, http://ant.apache.org/ (last viewed 2008-01-16)

[28] Oracle: *Parsing XML efficiency*, http://www.oracle.com/technology/oramag/oracle/03-sep/o53devxml.html (last viewed 2007-10-01)

[29] Sun - *Serial Access with SAX*, http://java.sun.com/webservices/jaxp/dist/1.1/docs/tutorial/sax/index.html (last modified 2005-06-15)

[30] IBM: *XML and Java technologies: Data binding, Part 2: Performance*, http://www.ibm.com/developerworks/xml/library/x-databdopt2/ (last viewed 2007-10-10)

[31] *SMIL Tutorial*, http://www.w3schools.com/smil/default.asp (last viewed 2007-09-06)

[32] *QuickTime: Embedding QuickTime for web delivery*, http://docs.info.apple.com/article.html?artnum=61011 (last modified 2007-05-31)

[33] *JavaScript Support*, http://developer.apple.com/documentation/QuickTime/REF/QT41_HTML/QT41WhatsNew-72.html (last viewed 2007-10-01).

[34] *Embedded RealPlayer Extended functionality guide*, http://service.real.com/help/library/guides/extend/embed.htm (last modified 2001-10-27)

[35] *Windows Media Player 6.4 SDK*, http://msdn2.microsoft.com/en-us/library/ms984011.aspx (last viewed 2007-10-01)

[36] *Windows Media Player 11 SDK*, https://msdn2.microsoft.com/en-us/library/bb262657.aspx (last viewed 2007-10-01)

[37] W3C – World Wide Web Consortium, http://www.w3.org/

[38] Dick C. A. Bulterman & Lloyd Rutledge. *SMIL 2.0 – Interactive Multimedia for Web and Mobile Devices*. Springer-Verlag Berlin Heidelberg 2004.

[39] *MMS MM7 SDK 4.0 and 5.0* – Ericsson Developer Program, http://www.ericsson.com/mobilityworld/sub/open/technologies/mms_mm7/tools/mm7_sdk (last viewed 2007-08-15)

[40] Java Community Process – *JSR-212: Server API for Mobile Services:Messaging – SAMS: Messaging,* http://jcp.org/en/jsr/detail?id=212 (last viewed 2007-08-15)

[41] 3GPP - *Multimedia Messaging Service (MMS); Functional description; Stage 2,* http://www.3gpp.org/ftp/Specs/html-info/23140.htm (last viewed 2008-01-16)

[42] W3C - *SOAP Version 1.2 Part 1: Messaging Framework (Second Edition) - W3C Recommendation 27 April 2007*, http://www.w3.org/TR/soap12-part1/ (last modified 2007-04-27)

[43] Microsoft – Activating ActiveX Controls, http://msdn2.microsoft.com/en-us/library/ms537508.aspx (last viewed 2008-01-16)

[44] Nokia, http://www.nokia.com/, (last viewed 2007-08-16)

[45] Open mobile alliance, http://www.openmobilealliance.org/ (last viewed 2007-11-19)

[46] Timed Interactive Multimedia Extensions for HTML (HTML+TIME) - Extending SMIL into the Web Browser, http://www.w3.org/TR/1998/NOTE-HTMLplusTIME-19980918 (last modified 2000-03-13)

[47] *XHTML+SMIL Profile* – W3C Note, http://www.w3.org/TR/XHTMLplusSMIL/ (last modified 2002-01-31)

[48] Jason Hunter, William Crawford. *Java Servlet Programming 2nd Edition.* O'Reilly & Associates, Inc., Sebastopol, USA (2001)

[49] S Brown, S Dalton, D Jepp, D Johnson, S Li, M Raible. *Pro JSP 2, fourth edition*. Springer Verlag, New York, USA (2005)

[50] William Crawford, Jonathan Kaplan. *J2EE Design Patterns, first edition*. O'Reilly & Associates, Inc., Sebastopol, USA (2003)

[51] Apache Tomcat, http://tomcat.apache.org/ (last modified 2007-09-20)

[52] Ericsson – *Developer Program - Ericsson Mobility World*, http://www.ericsson.com/mobilityworld (last viewed 2008-01-16)

[53] *Ericsson IPX*, http://www.ericsson.com/solutions/ipx/ (last viewed 2007-08-04)

Appendix A - Ericsson Service Creation Study – Final Report

This Appendix A contains a copy a service creation study performed by *inCode* on behalf on of *Ericsson*. Note that slides 21, 22 and 25 have been modified because these contained information that was intended for internal use only.

Ericsson Service Creation Study

Final Report

07 September 2007

Account Manager: Bengt Nordström
Email: BNordstrom@incodewireless.com

Summary and next steps for Ericsson

Summary of inCode's study:

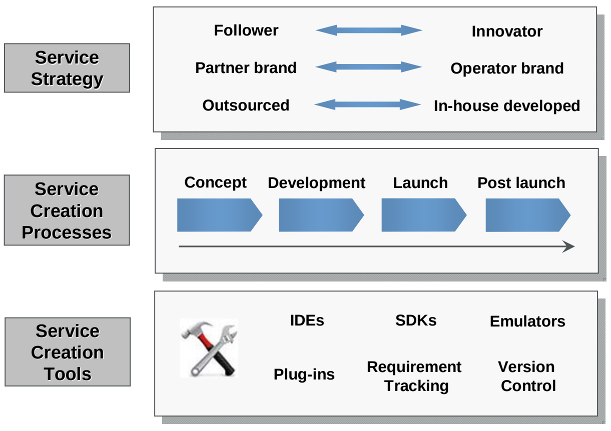
1. Operators see service innovation as key to act against falling ARPU levels. They turn to third parties for innovation and cost reduction.
2. There is a trend for operators to expose capabilities in a web-centric fashion, lowering technical and commercial hurdles for service creation.
3. Operators require help on their way towards a web-centric approach. We see potential for Ericsson in areas such as hosted IMS services, professional services and solutions for service execution (SDP) which in turn simplifies service creation.

Next steps for Ericsson:

1. Evaluate findings of this study against current product line-up and market message
2. Review inCode recommendations and identified opportunities
3. Discuss our findings with operators in order to gain further understanding of their needs

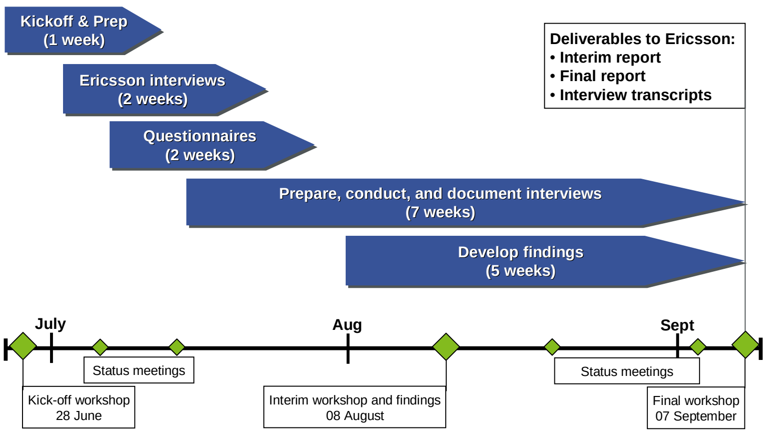
Contents

The study's aim is to provide Ericsson with a clear understanding of its clients' approaches and needs regarding service creation



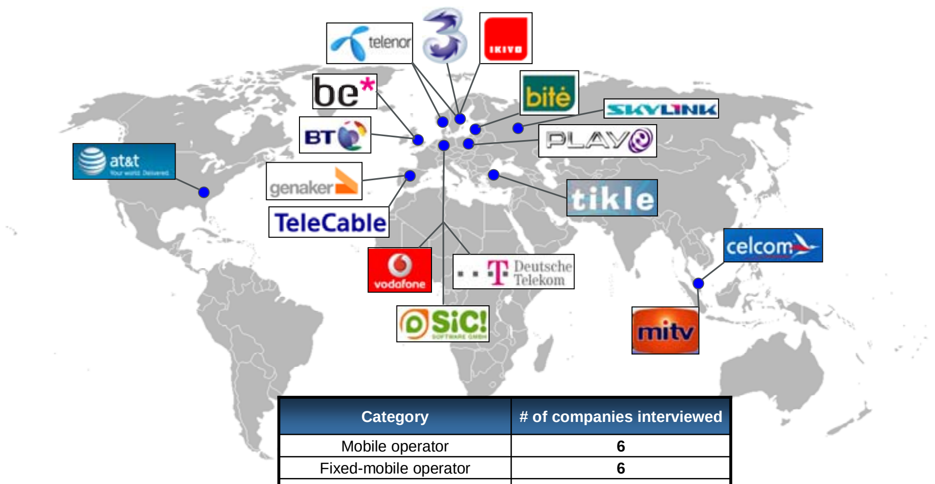
Understanding operator approaches for service creation shall help Ericsson align its product portfolio and market message.

Our project approach was mainly driven by preparing, conducting and analyzing interviews with industry representatives worldwide



Kickoff & Prep (1 week)
Ericsson interviews (2 weeks)
Questionnaires (2 weeks)
Prepare, conduct, and document interviews (7 weeks)
Develop findings (5 weeks)
Deliverables to Ericsson:
• Interim report
• Final report
• Interview transcripts
July
Aug
Sept
Status meetings
Kick-off workshop 28 June
Interim workshop and findings 08 August
Status meetings
Final workshop 07 September

The inCode team has led discussions with over twenty senior managers and experts from various types of companies and regions



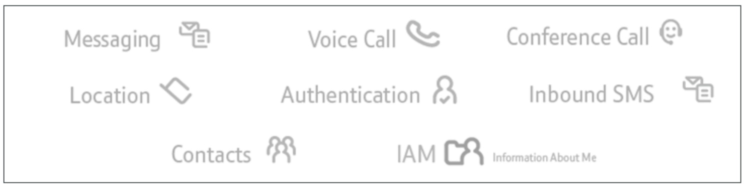
Category	# of companies interviewed
Mobile operator	6
Fixed-mobile operator	6
Broadband provider	1
Cable network operator	1
Mobile application developer	4

Contents

The project scope includes the creation of both enabling services and end user services

- **Service**
  A service can be any single part of a telecom operator's offering to end users (consumers or corporate users) or its business partners.

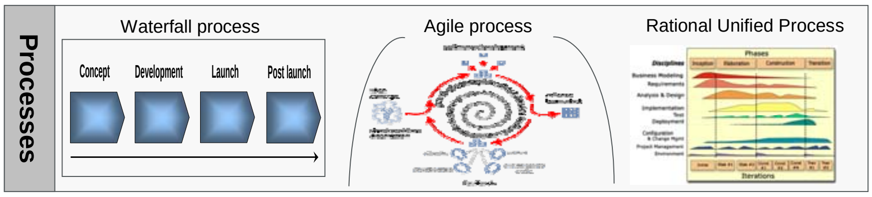
  Enabling services are capabilities that network operators offer to third parties for the creation and provisioning of end user services.

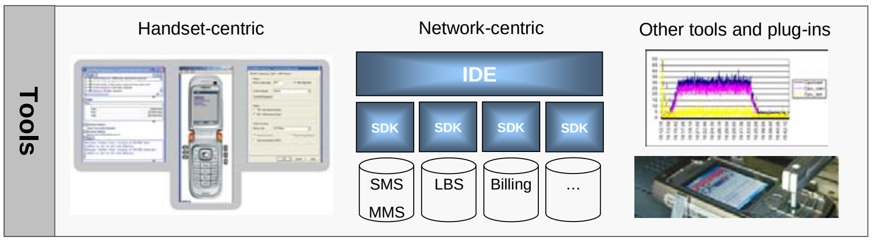


End user services are commercially offered consumer or business services, running on electronic devices such as telephones, PCs or set-top boxes.

In addition to the underlying service strategy, we wanted to find out how and with which tools services are created

- **Service creation**
  Service creation refers to the activities, processes and tools required in order to develop and launch services.

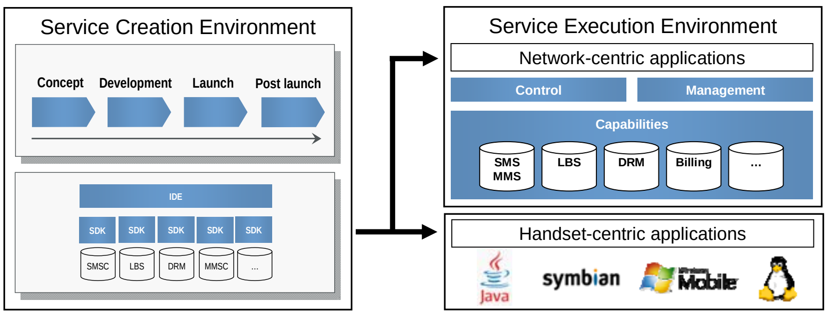




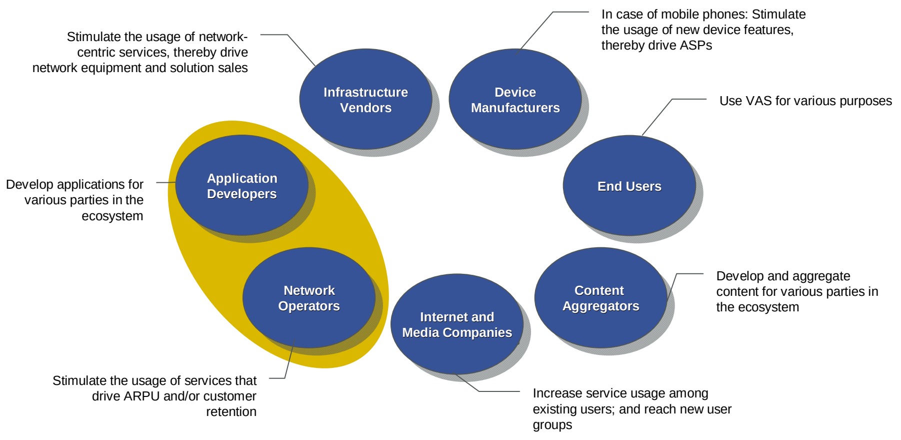
Although this project focused on service creation, the interdependencies with service execution are apparent

Service creation and service execution are closely linked:

- Before starting to work, developers need to know which systems the service shall run on
- Standards for server protocols or handset APIs exist, but are often implemented inconsistently across vendors and operators
- Handset execution environments (runtime or browser environments) have a short life-cycle and tools tend to be of limited quality



Service creation is influenced by various types of companies in the telecom, media and internet industries

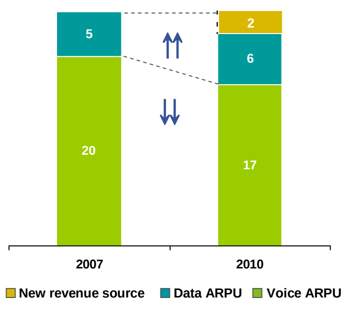


Stimulate the usage of network-centric services, thereby drive network equipment and solution sales
Infrastructure Vendors
Device Manufacturers
In case of mobile phones: Stimulate the usage of new device features, thereby drive ASPs
Use VAS for various purposes
End Users
Develop applications for various parties in the ecosystem
Application Developers
Network Operators
Internet and Media Companies
Content Aggregators
Develop and aggregate content for various parties in the ecosystem
Stimulate the usage of services that drive ARPU and/or customer retention
Increase service usage among existing users; and reach new user groups

For this first phase of the Service Creation project, Ericsson and inCode have chosen to interview network operators and application developers.

Operators see service innovation as a means to act against voice ARPU erosion and to increase customer retention

Monthly mobile ARPU worldwide (USD)



Sources: Strategy Analytics, Ericsson analysis

Example services:

- Broadband access
- Broadband services: IPTV, VoIP
- Advertising
- PC-based services
- …

- Content
- Data access
- P2P messaging
- Video telephony
- …

- Fixed-mobile substitution
- Fixed-mobile convergence
- Voice-based VAS
- Push to talk
- …

Operators need service innovation in all three areas in order to maintain ARPU, thereby making service creation a vital component of the strategy.

Contents

Service creation generally happens with high, but varying levels of third party involvement

1. Operators rely heavily on 3rd parties for service creation, in particular for the development portion.

We coordinate the SC process; our technology partners are responsible for tools and development.	*XYZ SDK was developed to take advantage of industry innovation.*	*Typically we ask what third parties can offer, see if it interests us, then we arrange an RFP.*	*Almost all services are created by partners today and we do the technical project management.*	*We have a small number of internally available developers, but we are using outsourcing.*
Operator A	Operator B	Operator C	Operator D	Operator E

Third party application developers are highly involved in service creation as a means for operators to deliver innovative services at low cost. In addition, many operators are open to fully outsource service operations as a turnkey solution.

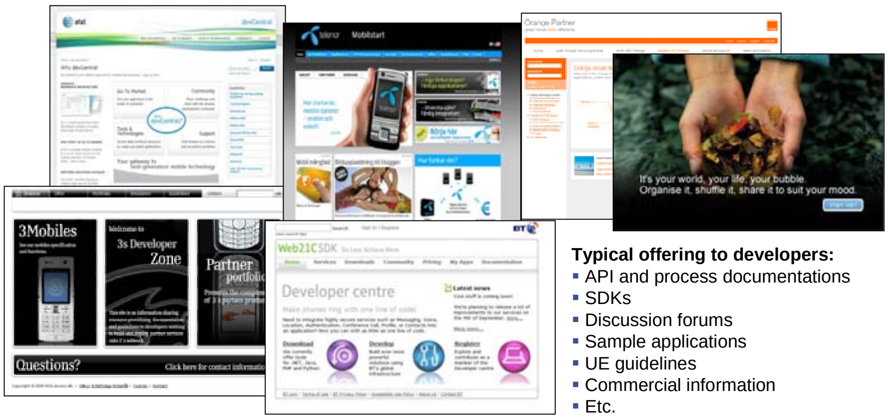
Operators have distinct approaches for service creation with third parties

2. Different strategies exist regarding third party interaction, and some major operators are moving towards a web-centric approach.

	Web centric Expose, then trial and error
▪ Operator markets itself to 3rd parties through its brand and customer base	▪ Operator markets itself towards 3rd parties as provider and broker
▪ APIs available to strategic partners	▪ APIs available online
▪ Operator-coordinated development	▪ Developers act independently
▪ Operator-branded services	▪ 3rd party branded services (initially)
▪ Specified quality levels	▪ Best effort services
▪ End users charged on operator bill	▪ Billing decided by third party

A shift towards a web-centric approach represents a major change in the way that operators attract and work with third parties, and with whom they work.

Several examples of operator web-centric approaches


devCentral
Mobilstart
Orange Partner
It's your world, your life, your bubble. Organise it, shuffle it, share it to suit your mood.
3Mobiles
Welcome to 3s Developer Zone
Partner portfolio
Questions?
Web21C SDK
Developer centre
Download
Develop
Register

Typical offering to developers:
- API and process documentations
- SDKs
- Discussion forums
- Sample applications
- UE guidelines
- Commercial information
- Etc.

A web centric approach is often accompanied by a strong marketing effort to attract third party developers. However, major differences exist regarding the technical expertise required and the ease of establishing a business relationship.

Operators' architectural preferences for exposing service capabilities are not yet decided

3. Operators are choosing to explore service innovation with third parties while concurrently evaluating what architecture should support this effort.

Today → **Tomorrow**

Walk	Jog	Run
▪ Capabilities shared on as needed basis with select strategic partners ▪ Protocols/Methods for exposing capabilities done on case-by-case basis	▪ Initial set of capabilities shared openly with 3rd parties ▪ Protocols/Methods for exposing capabilities are mixed with no standard prevailing	▪ Robust set of capabilities shared openly with 3rd parties ▪ Protocols/Methods for exposing capabilities see standards prevailing

Various operators are already taking steps to openly expose their service capabilities to third parties. The methods, standards, and business models are however not fully determined.

Operators are interested in web-based service delivery, but continue to use bespoke handset applications

4. Many interviewees see a long term trend towards web-based services, but manipulation of Internet content for mobile will still be required.

	Web-based mobile services	Proxy-based mobile browser	Mobile operator WAP portal	Bespoke handset centric solutions
Choice of content	●	◕	◔	◔
Operator SC involvement	○	○	◑	◑
Dedicated server	No	Yes	Yes	Yes
# of supporting handsets	◔	◔	●	◕
Service examples		web'n'walk Vodafone live! InternetFlat	t zones Orange World E-Plus WAP	3 msn vodafone myspace.com a place for friends

There appears to be consensus about the long-term trend towards web-based mobile services. As long as service enablers are not in place, operators continue to work with bespoke handset software irrespective of their level of web-centric positioning.

It is still early days for convergence

5. **Operators' intentions for converged services are visible, however many of their current activities are focused on setting the foundation.**

M&A & Partnerships

"TeleCable to launch MVNO service next year"
(Oct 2006)

"Italian subsidiary of BT UK secures Italian MVNO deal"
(May 2007)

"02 to launch UK broadband"
(February 2007)

Early Converged Services Examples

BT Fusion

Orange Unik

AT&T 3 Screen

IMS Trialing

"Vodafone IP Phone Professional is our first IMS-based service development"
(March 2007)

"Telenor first operator in Sweden to implement IMS"
(July 2007)

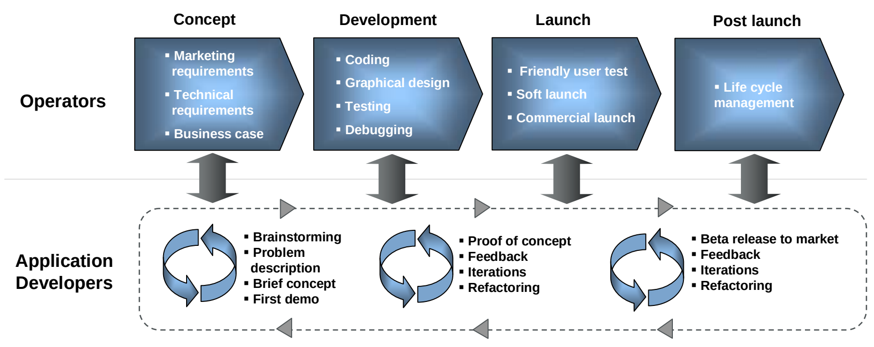
"Cingular (now AT&T) to launch IMS supported video service"
(April 2007)

Operator activities such as M&A, partnerships, IMS trialing, and service launches reveal converged service intentions.

Contents

As a result of a growing level of third party involvement, service creation is increasingly being done with "Agile" methods

6. Operators continue to use traditional "waterfall" process for service creation, while application developers embrace more "agile" methods.



When developing an operator-specific service, application developers need to accommodate operator processes requirements.

Operator complex organization and heavy processes create challenges in service creation

7. A key challenge operators face exists in organization and process execution.

The organizational structure is a key challenge to achieve business processes for introducing common services.	*… challenges related to the formalized process of XYZ with heavy governance compared to the more agile process previously used.*	*Operator C is in a transition process of adopting web-centric processes for internal service creation …this transition takes time.*	*…Lack of understanding of participants' roles, project phases and the scope are issues we face.*	*…there are no incentives or penalties for meeting or missing deadlines, and little motivation for stakeholders to deliver on time and as agreed.*
Operator A	Operator B	Operator C	Operator D	Operator E

Operators struggle with coordinating activities across their organization / company.

Integration with operators' IT systems is cited as a key challenge in service creation

8. **Service integration with an operator's existing IT generally seen as contributing to long project durations.**

Launch …and OSS/BSS integration are very lengthy. Even smaller …changes are difficult to implement. We mostly look at Internet apps that have no billing impact …	*Billing integration is a major challenge, as there are two billing systems to integrate with.*	*Generally, IT often delays service creation. For example, a service that requires extraordinary IT efforts may face lead times of 1 ½ years until those can be delivered.*	*The main problems with billing is the complexity and lack of flexibility of the system; this limits creative opportunities.*	*Billing integration requires rigid processes to work, and is not expected to become easier in the future. Interfacing with service platforms is another reason for the challenges.*
Operator A	Operator B	Operator C	Operator D	Operator E

Rigid processes, functional complexity, and the service impact on operator revenues and customer care slow the service creation process.

Contents

Developers give technical issues as major challenges in service creation

9. Application developers have strong comments about technical issues leading to high costs, especially on the device side.

Key feedback provided includes:

- Lack of interoperability in device implementations leads to high development costs
  - Operators and developers confirm that testing consumes 30 - 50% of project efforts
- Immaturity of handset-centric development tools
  - Short life cycles of handset models and even OS releases are seen as reasons
- Lack of availability of integrated tools, covering handsets and connections to operator networks
  - This becomes a problem for applications that use more sophisticated network resources

These issues lead to increased costs and longer projects for developers, which reduces or eliminates the profitability of service development projects.

Most operators are neutral about service creation tools, leaving the choice of solutions to their partners

10. Due to quality and cost reasons, most companies creating mobile services use open source development software.

Development software becomes commoditized; standardized at lower levels. We use open source as we don't want innovation at high cost.

Operator A

We use a combination of Open Source and our own developed tools.

Operator B

We use both Eclipse and NetBeans in our service creation.

Software Company A

In general, we find service creation tools are weak, they don't keep up with the pace of handset developments.

Software Company B

The operators' lack of preference is a consequence of their outsourced service creation. Developers inCode spoke to mostly use Eclipse, NetBeans and tools provided by the vendors of handsets and handset OS.

Contents

Opportunities and areas to address: Service creation strategy (I)

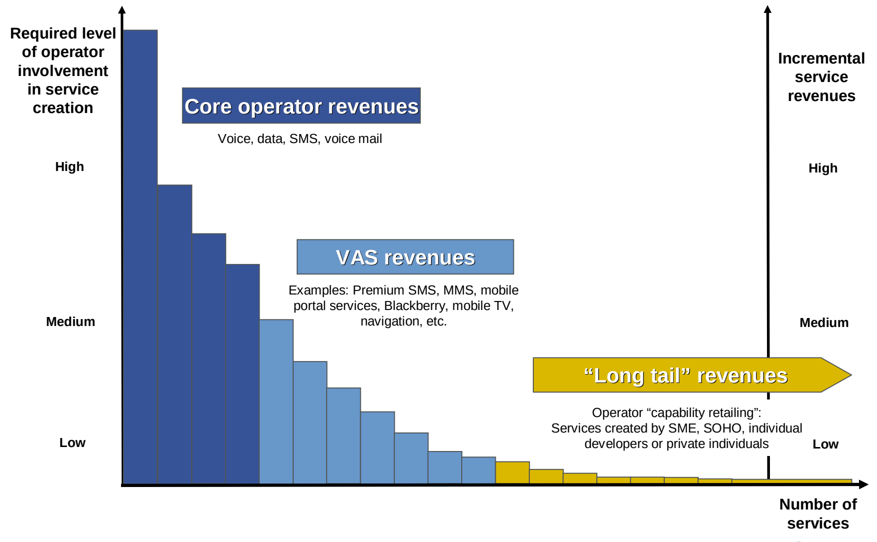
1. Operators rely heavily on 3rd parties for service creation, in particular for the development portion.

Leverage the capabilities of development partners and own solutions to provide turn-key service deployment and operation to operators.

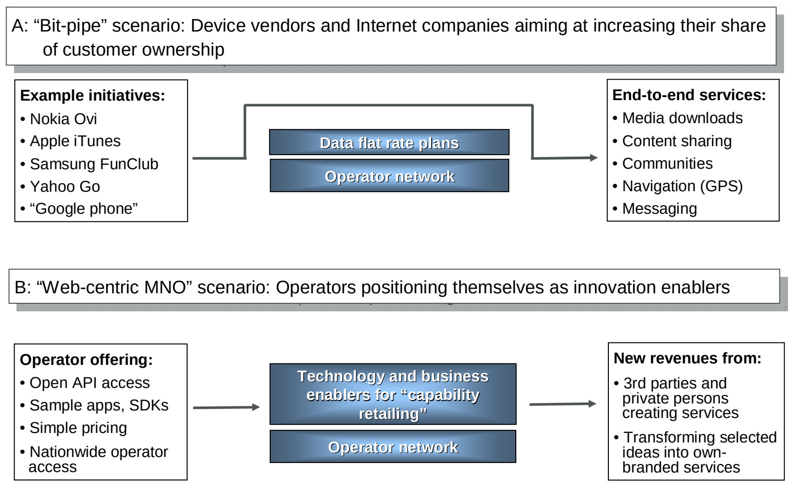
2. Different strategies exist regarding third party interaction, and some major operators are moving towards a web-centric approach.

Ericsson should facilitate this move by providing consulting services which demonstrate to operators the business rationale behind this transition. Ericsson can then drive the implementation of the web-centric approach through professional services and solutions like the SDP.

inCode believes that opening up capabilities in a web-centric manner is a logical step in the evolution of the network operator business



Ericsson should show operators that web-centric value creation is a way to avoid the bit-pipe scenario



A: "Bit-pipe" scenario: Device vendors and Internet companies aiming at increasing their share of customer ownership

Example initiatives:
- Nokia Ovi
- Apple iTunes
- Samsung FunClub
- Yahoo Go
- "Google phone"

Data flat rate plans

Operator network

End-to-end services:
- Media downloads
- Content sharing
- Communities
- Navigation (GPS)
- Messaging

B: "Web-centric MNO" scenario: Operators positioning themselves as innovation enablers

Operator offering:
- Open API access
- Sample apps, SDKs
- Simple pricing
- Nationwide operator access

Technology and business enablers for "capability retailing"

Operator network

New revenues from:
- 3rd parties and private persons creating services
- Transforming selected ideas into own-branded services

Opportunities and areas to address: Service creation strategy (II)

3. Operators are choosing to explore service innovation with third parties while concurrently evaluating what architecture should support this effort.

Ericsson should speed and facilitate the architectural decision by also offering a hosted IMS infrastructure which pools application developers and operators.

4. Many interviewees see a trend towards web-based services, but manipulation of Internet content for mobile will still be required.

For the next few years, service architectures that include web proxies or on-device portals will be preferred over pure browser-based solutions. Ericsson should monetize on this by offering end-to-end rich media solutions to operators.

5. Operators' intentions for converged services are visible, however many of their current activities are focused on setting the foundation.

Ericsson as a leading IMS provider is well positioned to serve operator needs. In addition, Ericsson can facilitate these converged services by offering a hosted IMS infrastructure which pools application developers and operators.

Opportunities and areas to address: Service creation process

6. Operators continue to use the traditional "waterfall" process for service creation, while application developers embrace more "agile" methods.

Although a shift towards "agile" is apparent, services requiring operator IT integration will continue to be created using waterfall processes. Ericsson should assess the need for tools supporting the end-to-end development cycle, particularly coordinating 3rd party interactions with operator processes.

7. A key challenge operators face exists in organization and process execution.

We expect operators to address these issues internally, or use management consultancies to improve organizational efficiency.

8. Service integration with an operator's existing IT generally seen as contributing to long project durations.

Ericsson is well positioned with an SDP proposition which helps operators to more efficiently deploy and integrate diverse services and back-office systems.

Opportunities and areas to address: Service creation tools

9. Application developers have strong comments about technical issues leading to high costs, especially on the device side.

Although no solution is expected in the short term, Ericsson should intensify discussions with developers and address issues like inconsistent API implementations and lack of tool quality.

10. Due to quality and cost reasons, most companies creating mobile services use open source development software.

Ericsson's continued support for open source tools targeted at developers can help further reduce service development costs. These tools should continue to act as sales facilitators rather than revenue generators on their own.

Contents

Overview of Application Developer Companies interviewed for this study

genaker

Core Business: Push-to-Talk-over-Cellular (PoC) applications for operators, enterprises, and institutions.

Products: PC PoC Dispatcher, Push-to-Talk PC Clients, Consumer Push-to-Talk, M-Ticketing, and Multimedia Manager.

Web address: www.genaker.net

IKIVO

Core Business: Standards based rich media solutions for mass market and Open OS mobile phones.

Products: Client software including Rich Media Client, Multimedia SVG player, Ikivo SVG player. Tools including Ikivo Animator, Ikivo IDE, and Ikivo CDK.

Web address: www.ikivo.com

tikle

Core Business: Mobile lifestyle services in Turkey creating mobile applications and services for mobile information and entertainment solutions.

Products: Content Management Platforms, VAS, Mobile Content Services, and Mobile Content Distribution.

Web address: www.tikle.com

SiC! SOFTWARE GMBH

Core Business: Service provider in the field of mobility software for Java, Symbian, Blackberry, and Windows mobile platforms.

Products: Mobile marketing, Audio Video streaming with DRM, and WAP Push Server. In addition, provide services in the areas of project management, technology consulting, feasibility studies, code reviews, and training.

Web address: www.sic-software.com

Terminology

Service
A service can be any single part of a telecom operator's offering to end users (consumers or corporate users) or its business partners. The first group of services covers end user services running on electronic devices such as mobile phones, fixed phones, PCs or set-top boxes. The second group refers to enabling services; network capabilities that the operator offers to other companies for the creation and provisioning of end user services. Although our study focuses on data-centric services with a graphical user interface (UI), voice services are also in scope.

Service creation
Service creation (SC) refers to all activities, processes and software tools required in order to develop and launch services.
Software developers creating end user services usually work with Service Creation Environments (SCE). A main element of SCEs are Integrated Development Environments (IDEs), software suites designed to facilitate service development.

What is IMS?

IMS (IP Multimedia Subsystem) allows services to share a common IP transport mechanism for any type of access. Service creation is enhanced as IMS provides new network capabilities simplifying development of peer-to-peer services; it also allows for easy combination of multiple sessions. The standardized architecture separates control from transport, aiming to improve network scalability. Such a horizontal architecture could also lead to shorter time to market for services. IMS may thus enable flexible development of strategies for different types of business models across and within networks.

What is SDP?

SDP (Service Delivery Platform) is a non-standardized evolution of the service network. It abstracts end-user services from network infrastructure, replacing vertical "silo" implementations by a common interface that exposes underlying network capabilities to all services using IT technologies (especially SOA). Many SDPs also enable high-level service composition, partner management, service discovery, and more. SDP provides an operator with better business control over its customer offerings allowing it to create, launch, sell, charge for, follow up, and retire services as needed - important to drive revenue and control costs.

IMS vs. SDP?

IMS focuses on delivering real-time, session-based, peer-to-peer IP services over any network. From an SDP's perspective, IMS is another set of network capabilities in addition to those provided by legacy systems such as IN, CS call control, content download servers, streaming servers, and more.

From an IMS perspective, the presence of an SDP is desirable as an easy way of interconnecting to supporting functionality in OSS, BSS, and the service network.

inCode®
A VeriSign Company

Appendix B - MMS SMIL from different mobile phones

B. 1. MMS SMIL from Sony Ericsson K800i

```
<smil>
  <head>
    <layout>
      <root-layout backgroundColor="#FFFFFF" background-
color="#FFFFFF" height="480px" width="640px" />
      <region id="Image" top="0" left="0" height="50%" width="100%"
fit="meet" />
      <region id="Text" top="50%" left="0" height="50%"
width="100%" fit="scroll" />
    </layout>
  </head>
  <body>
    <par dur="4000ms">
      <img src="Sharplight.jpg" region="Image"></img>
      <text src="smil.txt" region="Text">
        <param name="foreground-color" value="#000000" />
      </text>
    </par>
    <par dur="4000ms">
      <text src="smil_2.txt" region="Text">
        <param name="foreground-color" value="#000000" />
      </text>
    </par>
  </body>
</smil>
```

B. 2. MMS SMIL from Sony Ericsson P990i

```
<smil>
  <head>
    <meta name="generator" content="SEMC-UIQSMARTPHONE-P990i" />
    <layout>
      <root-layout width="200px" height="200px" background-
color="white" backgroundColor="white" />
      <region id="Image" top="0%" height="50%" fit="meet"
background-color="white" backgroundColor="white" />
      <region id="Text" top="50%" height="50%" fit="meet"
background-color="white" backgroundColor="white" />
    </layout>
  </head>
  <body>
    <par dur="5000ms">
      <img src="bangka sunset.jpg" region="Image" />
      <text src="text.txt" region="Text">
        <param name="foreground-color" value="black" />
        <param name="textsize" value="normal" />
      </text>
    </par>
    <par dur="5000ms">
      <text src="text_0.txt" region="Text">
        <param name="foreground-color" value="black" />
        <param name="textsize" value="normal" />
      </text>
    </par>
 </body>
</smil>
```

B. 3. MMS SMIL from Motorola MotoRAZR V3

```
<smil>
  <head>
    <layout>
      <root-layout width="240px" height="240px" />
      <region id="Text" left="0%" top="0%" height="33%"
width="100%" fit="meet" />
      <region id="Image" left="0%" top="33%" height="67%"
width="100%" fit="meet" />
    </layout>
  </head>
  <body>
    <par dur="8s">
      <img src="media2.jpeg" type="image/jpeg" region="Image"
alt="media2.jpeg" />
      <text src="media1.txt" type="text/plain" region="Text"
alt="media1.txt" />
    </par>
  </body>
</smil>
```

B. 4. MMS SMIL from Nokia 5140

```
<smil>
  <head>
    <layout>
      <root-layout width="122" height="96" />
      <region id="Image" width="100%" height="67%" left="0%"
top="0%" fit="meet" />
      <region id="Text" width="100%" height="33%" left="0%"
top="67%" fit="scroll" />
    </layout>
  </head>
  <body>
    <par dur="8000ms">
      <text src="cid:GxIoWErj5N" region="Text" />
      <img src="cid:EzULGNswrv" region="Image" />
    </par>
    <par dur="8000ms">
      <text src="cid:ahWY0uFfUO" region="Text" />
    </par>
  </body>
</smil>
```

B.5. MMS SMIL from Samsung SGH-Z560

```
<smil xmlns="http://www.w3.org/2000/SMIL20/CR/Language">
  <head>
    <layout>
      <root-layout width="240" height="432" />
      <region id="Image" width="100%" height="50%" left="0%"
top="0%" fit="meet" />
      <region id="Text" width="100%" height="50%" left="0%"
top="50%" fit="meet" />
    </layout>
  </head>
  <body>
    <par dur="5000ms">
      <text region="Text" src="cid:0_0.txt">
        <param name="textattribute" value="bold" />
        <param name="textsize" value="normal" />
      </text>
      <img region="Image" src="cid:0_1.jpg" />
    </par>
    <par dur="5000ms">
      <text region="Text" src="cid:1_0.txt">
        <param name="textattribute" value="bold" />
        <param name="textsize" value="normal" />
      </text>
    </par>
  </body>
</smil>
```

Appendix C – JavaScript Code

C.1. Media player detection scripts

```
var MpDetect = {

  //var that makes sure check is only performed once
  players : null,

  /*
    Get the URL and as long as it doesn't already contain the
"players"
    attribute, detect media players and reload the page by adding
the
    players attribute.
  */

  detectAndReload : function() {

    var url = location.href;

    if (url.indexOf('?') > -1) { //? indicates get request with
parameters - ok
      //ok, continue
    } else {
      alert('Error! Page was not loaded using GET and parameters.
Can\'t reload using with detected media players');
      return;
    }

    if (url.indexOf('players=') > 0) {
      //For some reason this page was loaded eventhough players
were specified, do nothing to avoid reloading forever
      alert('Players alredy detected, this page should not have
been loaded!');
      return;
    }

    //Detect, append and reload
    url = url + '&players=' + this.getAvailablePlayers();
    location.href = url;
  },

  getAvailablePlayers : function() {

    if (this.players != null)
      return this.players; //Check already performed

    this.players = '';
    var count = 0;
    var playerArray = new Array();

    if (this.detectQuickTime())
      playerArray[count++] = 'QuickTime';
    if (this.detectReal())
      playerArray[count++] = 'RealPlayer';
    if (this.detectWindowsMedia())
```

```
         playerArray[count++] = 'WindowsMediaPlayer';

      for (var i = 0; i < count; i++) {
         if (i > 0)
            this.players += ',';
         this.players += playerArray[i];
      }

      return this.players;
   },


   /************* Media player detection scripts *************/

   detectQuickTime : function () {

        var stdPlugs = new Array('QuickTime');
        var iePlugs = new Array('QuickTime.QuickTime');

        return this.detectPlugin(stdPlugs, iePlugs);
   },

   detectReal : function () {

      var stdPlugs = new Array('RealPlayer');
        var iePlugs = new Array('rmocx.RealPlayer G2 Control',
                          'rmocx.RealPlayer G2 Control.1',
                          'RealPlayer.RealPlayer(tm) ActiveX Control
(32-bit)',
                          'RealVideo.RealVideo(tm) ActiveX Control
(32-bit)',
                          'RealPlayer');

        return this.detectPlugin(stdPlugs, iePlugs);
   },

   detectWindowsMedia : function() {

        var stdPlugs = new Array('Windows Media');
        var iePlugs = new Array('WMPlayer.OCX');

        return this.detectPlugin(stdPlugs, iePlugs);

   },

   detectPlugin : function(stdPlugs, iePlugs) {

        // allow for multiple checks in a single pass
        var plugins = stdPlugs; //Check for all of

        // if plugins array is there and not fake
        if (navigator.plugins && navigator.plugins.length > 0) {

         var pluginsArrayLength = navigator.plugins.length;
         // for each plugin...
         for (pluginsArrayCounter = 0; pluginsArrayCounter <
pluginsArrayLength; pluginsArrayCounter++ ) {
```

```
           // loop through all desired names and check each against
the current plugin name
           var numFound = 0;

           for(namesCounter=0; namesCounter < plugins.length;
namesCounter++) {
            // if desired plugin name is found in either plugin name
or description

  if( (navigator.plugins[pluginsArrayCounter].name.indexOf(plugins[n
amesCounter]) >= 0) ||

(navigator.plugins[pluginsArrayCounter].description.indexOf(plugins[
namesCounter]) >= 0) ) {
                // this name was found
                numFound++;
            }
           }

           // now that we have checked all the required names
against this one plugin,
           // if the number we found matches the total number
provided then we were successful
           if(numFound == plugins.length) {
            return true;
           }
       }
      } else if (window.ActiveXObject) {

       plugins = iePlugs; //Check for any of

       for (var i = 0; i < plugins.length; i++) {
         var control;
         try {
              control = new ActiveXObject(plugins[i]);
         } catch (e) {
         }
         if (control)
            return true;
        }
      }


      return false;
  }
}
```

C.2. HTML+TIME playback scripts

```
/************ Playback methods **********/

function showSplash() {
  var splashDiv = document.getElementById('splash');
  /* Center vertically */
  splashDiv.style.top =
parseInt((splashDiv.parentNode.clientHeight/2 -
splashDiv.clientHeight/2)) + 'px';
  splashDiv.style.visibility = 'visible';
}

function hideSplash() {
  var splashDiv = document.getElementById('splash');
  splashDiv.style.visibility = 'hidden';
}



function playMms() {

  var playIt = function() {

  //Make sure all sequences are rewound
  document.getElementById('audioseq').seekTo(0,1); //Seek to
iteration 0, second 1
  document.getElementById('textseq').seekTo(0,1); //Seek to
iteration 0, second 1
  document.getElementById('visualseq').seekTo(0,1); //Seek to
iteration 0, second 1

  };

  setTimeout(playIt, 0);

}

function stopMms() {

  //Stop all sequences
  document.getElementById('audioseq').endElement();
  document.getElementById('textseq').endElement();
  document.getElementById('visualseq').endElement();

}

//Image preload method
var preloaded = new Array();

function preload(src) {

  preloaded[preloaded.length] = new Image();
  preloaded[preloaded.length - 1].src = src;

}

//Called by audio t:seq element when playback starts.
function mmsStarted(bgColor) {

  hideSplash();
  var contentDiv = document.getElementById('mms-comp-container');
  contentDiv.style.backgroundColor = bgColor;
```

```
}

//Called by audio t:seq element when playback ends.
function mmsStopped() {

  showSplash();
  var contentDiv = document.getElementById('mms-comp-container');
  contentDiv.style.backgroundColor = '';
}

/************ Playback helper methods **********/

/**
 * This method searches for a container parent (at most to document
root).
 * The container is defined by having the class name passed in the
containerClassName arg.
 * If no containerClassName is specified or if it is null, the
element will be used as container
 * Resizes all client nodes of the container to the max clientWidth
of the container, i.e. adjust content if scrollbars are present
 * Only width's in px are supported.
 */
function fitChildElements(el, containerClassName) {

  var theFunction = function() {

    //Because this is an inner function, the arguments el and
containerClassName are available here.

    var container;

    if (containerClassName == null || containerClassName == '') {
      //Class name not passed, use the element as container
      container = el;
    } else {
      //Find parent container, defined by class name matching the
containerClassName argument
      var currEl = el;

      while (true) {
        if (currEl == null) {
          alert('Could not find container with class name: ' +
containerClassName);
          return;
        } else if (currEl.className == containerClassName) {
          container = currEl;
          break;
        }
        //Get next parent node
         currEl = currEl.parentNode;
      }
    }

    //Get container's available width
    var width = container.clientWidth;
    if (width == null) {
      //TODO: Report with AJAX along with all available client data
      alert('Your browser does not support the clientWidth property
of the container, cannot resize elements when window is removed.');
      return;
    }

    //Update width of all child nodes to be max the current
available width
    var level = 0;
    var elements = {};
    var i = {};

    elements[level] = container.childNodes;
    i[0] = 0;

    //Iterate through all child nodes and make sure their width is
'width' if they have a width specified
    //TODO: Excluding option, use class name (you can have several)
    while(true) {

      if (level < 0) //End of element tree is reached
        break;

      if (i[level] >= elements[level].length) {
        //This element does not exist, go down one level and
continue
        level--;
        continue;
      }

      //Get current node
      var currNode = elements[level][i[level]];

      if (currNode.nodeName == '#text') {
        //Text node, ignore it, it doesn't have children
        i[level]++;
        continue;
      }

      //Resize this node if necessary
      var currWidth = null; //Interprets width if nor entered in
pixels, works in IE
      var widthStr = currNode.style.width;

      if (widthStr == null || widthStr == '') {
        currWidth = currNode.width;
      } else if (widthStr != null && widthStr != '') {

        //Only support size entered in pixels
        currWidth = parseInt(widthStr.substring(0, widthStr.length
- 2));
        if (isNaN(currWidth) == 1) {
          alert('Illegal width of element set, only pixels
supported. E.g. width: 100px, found: ' + widthStr);
        }
      } //else ignore node
      if (currWidth != null) {
        //Update width
        currNode.style.width = width + 'px';
      }

      //This node has been processed, update i
      i[level]++;

      //Check if this node has children
      var children = currNode.childNodes;

      if (children != null && children.length > 0) {
        //Node has children, process them
        //Set next level, add the elements and reset i for the next
level
        ++level;
        elements[level] = children;
        i[level] = 0;
      }

    }

    //Make sure the an increase didn't cause scrollbars to appear ->
clientHeight would have decreased
    setTimeout(function() {
      if (parseInt(width) > parseInt(container.clientWidth)) {
        //Do it again
        setTimeout(theFunction, 0);
      }
    }, 0);

  };

  //Do asynchronous call
  setTimeout(theFunction, 0);
}

/*********** QuickTime playback methods *******/

/**
 * Call play on QuickTime embedded element asyncronously
 */
function playQt(player) {

  //Define the function
  var theFunction = function() {
    player.Play();
  };

  //Do asynchronous call
  setTimeout(theFunction, 0);
}

/**
 * Asynchronous call to playQt() and fitChildElements() methods.
 */
```

```
function playAndResizeQt(player) {

  fitChildElements(player, 'mms-comp-container');
  playQt(player);

}

/**
 * Stop and rewind.
 */
function stopQt(player) {
  player.Stop();
  player.Rewind();
}

/*********** Real Player playback methods *******/

/**
 * Call play on Real Player embedded element asyncronously
 */
function playReal(player) {


  //Define the function
  var theFunction = function() {
    player.DoPlay();
  };

  //Do asynchronous call
  setTimeout(theFunction, 0);
}

/**
 * Asynchronous call to playReal() and fitChildElements() methods.
 */
function playAndResizeReal(player) {

  fitChildElements(player, 'mms-comp-container');
  playReal(player);

}

/**
 * Stop and rewind.
 */
function stopReal(player) {
  player.DoStop();
  player.SetPosition(0);
}

/*********** Windows Media Player playback methods *******/

/**
 * Call play on Windows Media Player embedded element asyncronously
 */
function playWmp(player) {
  //Wait for state 4
  var doOverFunc = function() {
    playWmp(player);
  }

  if (player.ReadyState < 4) {
    setTimeout(doOverFunc, 10);
    return;
  }

  //Define the function
  var theFunction = function() {
    player.Play();
  };

  //Do asynchronous call
  setTimeout(theFunction, 0);
}

/**
 * Asynchronous call to playWmp() and fitChildElements() methods.
 */
function playAndResizeWmp(player) {

  fitChildElements(player, 'mms-comp-container');
  playWmp(player);

}

/**
 * Stop and rewind.
 */
function stopWmp(player) {
  player.Stop();
}
```

C.3. DHTML playback scripts

```
/************* End user JavaScripts *****************/

/* Debug scripts, implement methods to report messages and errors */


//Example JSON object that implements debug and error methods
var MmsDebugListener = {

  debug : function(str) {
    //Implement
  },

  reportError : function(str) {
    //Implement
  }

}
```

```
/************* Playback control JavaScripts *************/

var MmsPlayer = {

  //Player variables
  height : 320,    //Integer, initialized in init().
  width : 240,     //Integer, initialized in init().
  repeatCount : 'indefinite', //Default
  isInitialized : false,
  slides : new Array(),
  preloaded : new Array(), //Preloaded images array

  //Player state variables
  currentCount : 0,  //The number of times that the MMS has been
shown
  isPlaying : false, //Boolean indicating playback state
  timerCounter : 0,  //Counter for how long to wait for media player
to start playing
  timer : null,       //Timer for next slide

  init : function(width, height, repeat) {

    MmsUtils.debug('Initializing MMS player with width: ' + width +
              ', height: ' + height);

    //Init player dimensions
    this.width = parseInt(width);
    this.height = parseInt(height);

    if (this.width.toString() == 'NaN' ||
      this.height.toString() == 'NaN') {

      MmsUtils.error('Player dimensions were not parsable
integers!');
    }

    //Set repeat count
    if (repeat != null)
      this.repeatCount = repeat;

    //Fit splash screen elements
    this.fitElements('mms-player-splash');

    this.isInitialized = true;

    MmsUtils.debug("MMS Player initialized");
  },

  /**
   * Add a slide. All slides must be added using this method
   * and they must be added in order, starting with 0 as the
   * first slide.
   *
   * The duration attribute defines the duration of the slide
   * and is given as an integer with either an 's' or 'ms'
```

```
    * suffix.
    *
    * The complexMediaString defines that the slide contains
    * a complex media type such as a video or audio clip, what
    * player the clip is embedded on the slide to play it and
    * the id of the player DOM object.
    * E.g. "video/QuickTime/video1"
    * Supported players are: QuickTime, RealPlayer and
    * WindowsMediaPlayer.
    */
   addSlide : function(slideNo, duration, complexMediaStr) {

     if (! this.isInitialized)
       return {status : 'error', message : 'Mms Player was not
initialized'};

     if (slideNo != this.slides.length)
       return {status : 'error', message : 'Illegal slide order,
slides should be registered in order starting with index 0'};

     MmsUtils.debug('Adding slide: ' + slideNo + ' with duration: ' +
duration + ' and complexMediaStr: ' + complexMediaStr);

     //Create a new slide
     var slide = new MmsSlide();

     //Set duration and complexMediaStr
     slide.duration = duration;
     slide.complexMediaStr = complexMediaStr;

     this.slides[slideNo] = slide;

     //Fit elements on slide
     this.fitElements('mms-player-slide-' + slideNo);

   },

   play : function() {

     if (! this.isInitialized)
       return {status : 'error', message : 'Mms Player was not
initialized'};

     //Make sure player is not playing
     if (this.isPlaying)
      return {status : 'error', message : 'Player already started'};

     if (this.slides.length == 0)
       return {status : 'error', message : 'MMS has no slides'};

     MmsUtils.debug('Starting MMS playback');

     //Update player state
     this.isPlaying = true;
     this.currentSlide = -1; //Indicates that no slide is shown, next
will be the first (index 0)
     this.currentCount = 1; //First time MMS is shown
```

```
      //Show first slide
      this.nextSlide();

      return {status : 'ok', message : 'Playback started'};

    },

    stop : function() {

      if (! this.isInitialized)
        return {status : 'error', message : 'Mms Player was not
initialized'};

      //Make sure player is playing
      if (!this.isPlaying)
        return {status : 'error', message : 'Player not started'};

      MmsUtils.debug('Stopping MMS playback');

      //Update player state
      this.isPlaying = false;
      clearTimeout(this.timer); //Remove any pending timer

      this.stopAndShowSplash();

      return {status : 'ok', message : 'Playback stopped'};

    },



    /*********** Internal helper methods ********************/

    //Center an HTML element
    centerElement : function(el, availableWidth) {

      MmsUtils.debug('Centering element');

      if (availableWidth == null) //Not specified, use player width
        availableWidth = this.width;

      var elWidth = parseInt(el.width);

      if (elWidth.toString() == 'NaN') {
        MmsUtils.error('Element didn\'t contain width attribute');
        return;
      }

      //Margins are mms player width - element width
      var margin = Math.floor((availableWidth - elWidth)/2);

      el.style.marginLeft = margin + 'px';
      if (margin > 0)
        //Fix for problem with recent version of FF on Win Vista
(only occurs with uneven widths)
        el.style.marginRight = margin - 1 + 'px';
      else
        el.style.marginRight = margin + 'px';

      MmsUtils.debug('Element centered');

    },

    fitElements : function(containerId) {

      MmsUtils.debug('Centering children of container: ' +
containerId);

      var slide = document.getElementById(containerId);
      var elements = slide.childNodes;

      var imageNode;
      var textNode;
      var videoNode;
      //Audio nodes do not need to fitted

      //Get elements, check for img (image), div (text), object (audio
or video)
      for (var i = 0; i < elements.length; i++) {

        var el = elements[i];

        if (el.nodeName == '#text') {
          //Do nothing, ignore text parts
        } else if (el.nodeName == 'IMG') {
          imageNode = el;
        } else if(el.nodeName == 'DIV') {
          textNode = el;
        } else if (el.nodeName == 'OBJECT') {
          //Object is video or audio

          if (el.id.substring(0, 5).toLowerCase() == 'video')
            videoNode = el; //Object was video

        } else if (el.nodeName == 'SCRIPT') {
          //Do nothing, ignore script parts
        } else if (el.nodeName == 'INPUT') {
          //Do nothing, ignore input parts
        } else {
          MmsUtils.debug('Unknown tag: ' + el.nodeName);
        }

      }

      //Text is never resized but check its height.
      var textHeight = 0;

      if (textNode != null)
        textHeight = textNode.offsetHeight;

      if (imageNode != null) {
        //Fit image node
```

```
       MmsUtils.debug('Entering image node resize');

       //Make sure image is not wider than player
       if (imageNode.width > this.width) {
         this.resizeImage(imageNode, this.width, null); //Set new
width
         this.centerElement(imageNode);
       }

       /*
         Decide whether to further downscale the image.

         Image will be downscaled if:
           - There is no text and image height exceeds player height
           - There is text and image height + text height exceeds
             player height. (This causes scrollbars)

       */

       var imageHeight = imageNode.height;
       var verticalTextConsumption = textHeight / this.height;
  //Defines how much of the height is needed for text
       var verticalImageConsumption = imageHeight / this.height;
  //Defines how much of the height is needed for image

       if (textHeight == 0 && imageNode.height > this.height) {
         //Max height will be player height

         this.resizeImage(imageNode, null, this.height); //Set new
height

         MmsUtils.debug('Image resized vertically to fit slide');

       } else if (verticalTextConsumption + verticalImageConsumption
> 1) { //Scrollbars will be added

         //Downscale image vertically

         var newImageHeight = imageHeight;

         if (verticalTextConsumption > 0.5) { // Don't downscale
image height to less than half
           //Set image height to half the
           newImageHeight = Math.floor(this.height / 2);
         } else {
           newImageHeight = this.height - textHeight - 4;
         }

         this.resizeImage(imageNode, null, newImageHeight); //Set
new height

         MmsUtils.debug('Image resized vertically to fit slide
together with text');
       }

       //Compensate width property if scrollbars were added
       if (slide.clientWidth != this.width) {
         //Scrollbars were added
         var newWidth = slide.clientWidth;

         if (imageNode.width > newWidth) {
           //Image must be downscaled
           this.resizeImage(imageNode, newWidth, null);

           MmsUtils.debug('Image resized horizontally to fit within 
scroll bars');
         }

       }

       this.centerElement(imageNode, slide.clientWidth); //Center

     }

     if (videoNode != null) {
       //Fit video node

       this.centerElement(videoNode, slide.clientWidth);

     }

  },

  /**
   * Returns the ID of the complex media based on the first
   * last part of the complexMediaStr, as specified when slide
   * as added.
   */

  getComplexMediaId : function(slideNo) {

     var cStr = this.slides[slideNo].complexMediaStr;

     if (cStr == null)
       return null;

     var pos = cStr.lastIndexOf('/');
     if (pos < cStr.length) {

       var result = cStr.substring(pos + 1, 
cStr.length).replace(/^\s+|\s+$/g, ''); //replace method trims the 
string
       return result;

     }

     return null;

  },

  /**
   * Returns "audio", "video" or null based on the first
   * part of the complexMediaStr as specified when slide
    * as added.
    */

   getComplexMediaType : function(slideNo) {

     var cStr = this.slides[slideNo].complexMediaStr;

     if (cStr == null)
       return null;

     var result = cStr.substring(0,5).toLowerCase();

     if (result == 'audio' ||
       result == 'video') {

       return result;
     } else {
       return null;
     }
   },

   /**
    * Returns "QuickTime", "RealPlayer",
    * "WindowsMediaPlayer" or null based on the first
    * middle part of the complexMediaStr, as specified when slide
    * as added.
    */

   getComplexMediaPlayer : function(slideNo) {

     var cStr = this.slides[slideNo].complexMediaStr;

     if (cStr == null)
       return null;

     var result = '';
     if (cStr.length > 6 && cStr.indexOf('/') > -1)
       result = cStr.substring(6,
cStr.lastIndexOf('/')).replace(/^\s+|\s+$/g, ''); //replace method
trims the string

     if (result == 'QuickTime' ||
       result == 'RealPlayer') {

       return result; //Only return valid Strings

     } else {
       return null;
     }

   },

   /**
    * Displays the next slide to be shown, regardless whether it
    * is the next slide or the first one (because there are no more
    * slides to show). Shows the splash screen if the number of
    * allowed repeats has already been reached.
    */

   nextSlide : function() {

     MmsUtils.debug('Showing next slide');

     var nextSlide = this.currentSlide + 1;
     var stopPlayback = false;

     if (!this.isPlaying)
       return; //Playback has been stopped, don't play next slide

     if (nextSlide >= this.slides.length) {
       //No more slides

       //Check if the slide has been played as many times as it
should.
     if (this.repeatCount.toString().toLowerCase() == 'indefinite')
{
          //The repeat is indefinite, i.e. it has not been exceeded -
do nothing
       } else if (parseInt(this.repeatCount) <= this.currentCount) {
          //There is no next slide, show splash screen
          stopPlayback = true;
       } else if (parseInt(this.repeatCount).toString() == 'NaN') {
          MmsUtils.error('Unknown value for variable repeatCount: ' +
this.repeatCount);
       }

       //Increase current repeat count
       this.currentCount++;

       //Set next slide no to 0
       nextSlide = 0;
     }

     //Hide previous slide (if it exists)
     var prev = document.getElementById('mms-player-slide-' +
this.currentSlide);
     if (prev != null) {
       prev.style.visibility = 'hidden';
     } else {
       //Assume the last "slide" was -1 i.e. the splash screen and
hide it
       var splash = document.getElementById('mms-player-splash');
       splash.style.visibility = 'hidden';
     }

     if (stopPlayback) {
       //Show splash
       var splash = document.getElementById('mms-player-splash');
       splash.style.visibility = 'visible';
       this.isPlaying = false;
       MmsUtils.debug('MMS playback finished');
     } else {
       //Show new slide
```

```
       var newSlide = document.getElementById('mms-player-slide-' +
nextSlide);
       newSlide.style.visibility = 'visible';

       //Update current slide
       this.currentSlide = nextSlide;

       //Check for media players and start playback (this means the
slide contains either audio or video)
       var mediaPlayer =
this.getComplexMediaPlayer(this.currentSlide);

       if (mediaPlayer != null) {
         //There is complex media, play it and wait for playback to
complete

         //Get id
         var mediaId = this.getComplexMediaId(this.currentSlide);

         //Play video using the specified media player
         if (mediaPlayer == 'QuickTime') {
           //Play using QuickTime
           QtUtils.play(mediaId);

         } else if (mediaPlayer == 'RealPlayer') {
           //Play using RealPlayer
           RealUtils.play(mediaId);

         } else {
           MmsUtils.error('Unknown media player: ' + mediaPlayer);
           return;//Don't continue processing
         }

         //Reset timer counter
         this.timerCounter = 0;

         //Start waiting function - starts timeout function when the
         //media element has finished loading and is playing.
         //This makes sure the media clip doesn't finish before the
         //next slide is shown.
         timer = setTimeout(this.waitForPlayer, 500); //Give time to
initialize
         return; //Don't start slide timeout

       }

       //TODO: Wait for state: playing
       this.startSlideTimeout();
     }

  },

  preloadImage : function (src) {

    this.preloaded[this.preloaded.length] = new Image();
    this.preloaded[this.preloaded.length - 1].src = src;
  },

  //Resizes image to scale. Use only height OR width and set other
to null
  resizeImage : function(imgEl, width, height) {

    var SET_HEIGHT = 1;
    var SET_WIDTH = 2;
    var SET_BOTH = 3;

    var mode = 0;

    if (height != null && width == null)
      mode = SET_HEIGHT;
    else if (height == null && width != null)
      mode = SET_WIDTH;
    else if (height != null && width != null)
      mode = SET_BOTH;

    MmsUtils.debug('Resizing image: ' + imgEl.src);

    //Initial dimensions, needed to do manual rescaling in IE
    var iWidth = imgEl.width;
    var iHeight = imgEl.height;

    if (iHeight == 0 || iWidth == 0) {
      //Image not properly loaded, report and return
      MmsUtils.error('Image dimensions were 0x0, probably image
wasn\'t loaded');
      return;
    }

    if (mode == SET_HEIGHT) {

      //Clear width style so that image will not change ratio.
      imgEl.style.width = '';

      //Set height to specified height
      imgEl.style.height = height + 'px';

      //Calculate expected value and make sure scale manually if
width wasn't updated automatically
      var scaleFactor = height / iHeight;
      var expectedWidth = Math.floor(scaleFactor * iWidth);

      if (imgEl.width > expectedWidth + 1 || imgEl.width <
expectedWidth - 1) {
        //Not resized horizontally, scale manually.
        imgEl.style.width = expectedWidth + 'px';
      }

    } else if (mode == SET_WIDTH) {

      //Clear height style so that image will not change ratio.
      imgEl.style.height = '';

      //Set width to specified width
```

```
       imgEl.style.width = width + 'px';

       //Calculate expected value and make sure scale manually if
height wasn't updated automatically
       var scaleFactor = width / iWidth;
       var expectedHeight = Math.floor(scaleFactor * iHeight);

       if (imgEl.height > expectedHeight + 1 || imgEl.height <
expectedHeight - 1) {
         //Not resized vertically, scale manually.
         imgEl.style.height = expectedHeight + 'px';
       }

     } else if (mode == SET_BOTH) {

       imgEl.style.height = height + 'px';
       imgEl.style.width = width + 'px';

     }

   },

   slideTimeoutHandler : function() {

     MmsPlayer.nextSlide(); //Show next, 'this' doesn't work because
it is used as event method

   },

   startSlideTimeout : function() {

     var dur = this.slides[this.currentSlide].duration;
     var durIntAsStr = parseInt(dur).toString();

     if (durIntAsStr == 'NaN') {

       MmsUtils.debug('Warning, illegal duration: ' + dur + '.
Falling back on default: 10s');

       dur = '10s'; //Fall back on default
     }

     var suffix = dur.substring(durIntAsStr.length, dur.length);

     var delay = 10000; //Default

     if (suffix == 'ms') {
       delay = parseInt(durIntAsStr);
     } else if (suffix == 's') {
       delay = parseInt(durIntAsStr) * 1000;
     }

     MmsUtils.debug('Slide playing, next slide shown in ' + delay +
'ms');

     this.timer = setTimeout(this.slideTimeoutHandler, delay);
```

```
  },

  stopAndShowSplash : function () {

    //Hide slide (if it exists)
    var prev = document.getElementById('mms-player-slide-' +
this.currentSlide);

    if (prev != null) {
      prev.style.visibility = 'hidden';
    }

    //Stop any media that is being played

    //Check for media player and stop playback
    var mediaPlayer = this.getComplexMediaPlayer(this.currentSlide);

    if (mediaPlayer != null) {
      //There is complex media, stop it.

      //Get id
      var mediaId = this.getComplexMediaId(this.currentSlide);

      //Play video using the specified media player
      if (mediaPlayer == 'QuickTime') {
        //Play using QuickTime
        QtUtils.stop(mediaId);

      } else if (mediaPlayer == 'RealPlayer') {
        //Play using RealPlayer
        RealUtils.stop(mediaId);

      } else {
        MmsUtils.error('Unknown media player: ' + mediaPlayer);
        return;//Don't continue processing
      }
    }

    //Show splash
    var splash = document.getElementById('mms-player-splash');
    splash.style.visibility = 'visible';

    MmsUtils.debug('Playback stopped');
  },

  /**
   * This function checks if the media player on the current slide
has finished
  * loading the media file and has started playing the file. If not,
the function
   * calls itself after 100ms. If media file has started playing,
this function
   * starts the timeout for this slide.
   */
  waitForPlayer : function() {

    //Get media player
    var mediaPlayer =
MmsPlayer.getComplexMediaPlayer(MmsPlayer.currentSlide);
    var mediaId =
MmsPlayer.getComplexMediaId(MmsPlayer.currentSlide);

    var status;

    //Play video using the specified media player
    if (mediaPlayer == 'QuickTime') {
      //Get status from QuickTime
      status = QtUtils.getPlayerStatus(mediaId);
    } else if (mediaPlayer == 'RealPlayer') {
      //Get status from RealPlayer
      status = RealUtils.getPlayerStatus(mediaId);
    } else {
      MmsUtils.error('Unknown media player: ' + mediaPlayer);
      return;//Don't continue processing
    }

    MmsUtils.debug('Media player status was: ' + status);

    if (status == 'playing') {
      MmsPlayer.startSlideTimeout();
    } else if (status == 'waiting') {
      MmsPlayer.timerCounter++; //We have waited one more time

      MmsUtils.debug(this.timerCounter);
      //wait max 10s = 100 * 100ms
      if (MmsPlayer.timerCounter <= 100) {
        MmsPlayer.timer = setTimeout(MmsPlayer.waitForPlayer, 100);
//wait for 100ms and try again
      } else {
        MmsUtils.error('Media did not start within 10 seconds!');
      }
    }
    //Else, something went wrong, stop playback - do nothing

  }

}



/* Utilities for the MMS components */
var MmsUtils = {

  debug : function (str) {
    try {
      MmsDebugListener.debug(str);
    } catch (e) {
      //Do nothing
    }
  },

  error : function(str) {
    try {
      MmsDebugListener.reportError(str);
```

```
     } catch (e) {
       //Do nothing
     }
   }
}

//MMS objects

function MmsSmil() {

   this.layout = 1; //1 -> Image/video on top, 2 -> text on top
   this.slides = new Array();

}

function MmsSlide() {

   //MMS info fields
   this.duration; //Duration for the slide
   this.complexMediaStr; //Complex med info. E.g.
audio/RealPlayer/audio1

}

//Media player utilties

//RealPlayer helper
var RealUtils = {
   play : function(mediaId) {

     //Define the function
     var theFunction = function() {
       try {
         //RealPlayer plugin consists of 1 object tag and one nested
embed
         //tag with ids: <id>-ie and <id>-em. The former plays in
Internet Explorer
         //and the latter in at least Firefox (2.0)

         //Try to get the IE tag and play it
         var player = document.getElementById(mediaId + '-ie');

         try {
            player.DoPlay();
         } catch (e) {

           //Use the embed tag to try to play the file
           player = document.getElementById(mediaId + '-em');
           player.DoPlay();
         }

         MmsUtils.debug('RealPlayer playback started');

       } catch (ex) {
         MmsUtils.error(ex);
       }
     };
```

```
      //Do asynchronous call
      setTimeout(theFunction, 0);
   },

   /**
    * Get the current status of the media player.
    * Returns one of the values: 'waiting', 'playing' or 'error'
    */
   getPlayerStatus : function(mediaId) {

      //RealPlayer plugin consists of 1 object tag and one nested
embed
      //tag with ids: <id>-ie and <id>-em. The former plays in
Internet Explorer
      //and the latter in at least Firefox (2.0)

      var realStatus;

      //Try to get the IE tag and play it
      var player = document.getElementById(mediaId + '-ie');
      try {
         realStatus = player.GetPlayState();
      } catch (e) {
         //Use the embed tag to try to get the status
         player = document.getElementById(mediaId + '-em');

         try {
            realStatus = player.GetPlayState();
         } catch (ex) {
            MmsUtils.error(ex);
         }
      }

      if (realStatus < 2) {
         return 'waiting';
      } else if (realStatus > 1 && realStatus < 5) {
         return 'playing';
      }

      //Status did not match any of the specified values
      MmsUtils.error('Unexpected play state for RealPlayer: ' +
realStatus);
      return 'error';

   },

   stop : function(mediaId) {

      //Define the function
      var theFunction = function() {

         try {
            //RealPlayer plugin consists of 1 object tag and one nested
embed
            //tag with ids: <id>-ie and <id>-em. The former plays in
Internet Explorer
```

```
          //and the latter in at least Firefox (2.0)

          //Try to get the IE tag and stop it
          var player = document.getElementById(mediaId + '-ie');

          try {
             player.DoStop();
          } catch (e) {
            //Use the embed tag to try to stop the file
            player = document.getElementById(mediaId + '-em');
            player.DoStop();
          }

          MmsUtils.debug('RealPlayer playback stopped');
        } catch (e) {
          MmsUtils.error(e);
        }

      };

      //Do asynchronous call
      setTimeout(theFunction, 0);
    }
  }

  //QuickTime helper
  var QtUtils = {

    lastPlayId : '',
    playErrCount : 0,

    /**
     * This function starts playback of the QuickTime media player
with the
     * specified id. An embedded QuickTime media player consists of 2
tags,
     * One for IE and one for standards compliant browsers. The ids of
the
     * actual tags should have "-ie" and "-std" suffixes respectively.
     *
     * If the mediaId is not provided, the previously used ID will be
used.
     *
     * If playback fails, this function will call itself after a
timeout
     * of 100ms. After a maximum of 5 consecutive fails the error will
be
     * reported. This is required because it may take the plugin time
to
     * initialize and before it is initialized it will result in an
error.
     */

    play : function(mediaId) {

      if (mediaId == null) {
        mediaId = this.lastPlayId; //Media ID not specified, use
previous
      } else {
        this.lastPlayId = mediaId; //Media ID specified, update as
last used
        this.playErrCount = 0; //Reset error count
      }


      //Define the function
      var theFunction = function() {

        try {
          //QT plugin consists of 2 nested object tags with ids:
          //<id>-ie and <id>-std. The former plays in Internet
Explorer
          //and the latter in standard compatible browsers

          //Try to get the IE tag and play it
          var player = document.getElementById(mediaId + '-ie');

          try {
            player.Play();
            QtUtils.playErrCount = 0; //Reset error count
          } catch (e) {

            //Assume the player is standard compliant and play
            player = document.getElementById(mediaId + '-std');

            player.Play();
            QtUtils.playErrCount = 0; //Reset error count

          }

          MmsUtils.debug('QuickTime playback started');
        } catch (ex) {
          QtUtils.playErrCount++;
          if (QtUtils.playErrCount > 5)
            MmsUtils.error(ex); // 5 Consecutive errors, stop trying
and report.
          else
            setTimeout(QtUtils.retryPlay, 100); //Try again after
100ms
        }

      };

      //Do asynchronous call
      setTimeout(theFunction, 0);
    },

    retryPlay : function() {

      MmsUtils.debug('Retrying playback start after error ' +
QtUtils.playErrCount);
      QtUtils.play(); //Retry
    },
    /**
     * Get the current status of the media player.
     * Returns one of the values: 'waiting', 'playing' or 'error'
     */
    getPlayerStatus : function(mediaId) {

      //QT plugin consists of 2 nested object tags with ids:
      //<id>-ie and <id>-std. The former plays in Internet Explorer
      //and the latter in standard compatible browsers

      var qtStatus;

      //Try to get the IE tag and play it
      var player = document.getElementById(mediaId + '-ie');

      try {
        qtStatus = player.GetPluginStatus();
      } catch (e) {

        //Assume the player is standard compliant and play
        player = document.getElementById(mediaId + '-std');

        try {
          qtStatus = player.GetPluginStatus();
        } catch (ex) {
          MmsUtils.error(ex);
        }
      }

      if (qtStatus == 'Waiting') {
        return 'waiting';
      } else if (qtStatus == 'Loading') {
        return 'waiting';
      } else if (qtStatus == 'Playable') {
        return 'playing';
      } else if (qtStatus == 'Complete') {
        return 'playing';
      } else if (qtStatus != null && qtStatus.substring(0, 5) ==
  'Error') {
        //Report error and return
        MmsUtils.error('QuickTime playback error: ' + qtStatus);
        return 'error';
      }

      //Status did not match any of the specified values
      MmsUtils.error('Unknown plug-in status for QuickTime: ' +
  qtStatus);
      return 'error';

    },

    stop : function(mediaId) {

      //Define the function
      var theFunction = function() {
        try {
          //QT plugin consists of 2 nested object tags with ids:
          //<id>-ie and <id>-std. The former plays in Internet
Explorer
          //and the latter in standard compatible browsers

          //Try to get the IE tag and play it
          var player = document.getElementById(mediaId + '-ie');
          try {
             player.Stop();
             player.Rewind();
          } catch (e) {
            //Assume the player is standard compliant and play
            player = document.getElementById(mediaId + '-std');
            player.Stop();
            player.Rewind();
          }

          MmsUtils.debug('QuickTime playback stopped');
        } catch (e) {
          MmsUtils.error(e);
        }
      };

      //Do asynchronous call
      setTimeout(theFunction, 0);
    }
  }
```

C.4. MMS canvas scripts

```
/********* End user scripts - implement to debug *************/
var MmsDebugListener = {

  debug : function(str) {
    //Do nothing
  },

  reportError : function(str) {
    //Do nothing
  }

}

/********************** INTRODUCTION *********************

  Use JSON singleton "MmsCanvas" to initialize the canvas,
  manipulate MMS and get information about the current
  state of the canvas.

  Initialization method

  init(width, height,                Initialize the canvas
     defaultDuration                 with height and width
                                     and default duration
                                     attributes. defaultDuration
```

```
                              is optional.

     Optional parameter description

     defaultDuration                       The default duration to
use
                              for new slides.

                              DEFAULT: '10s'



   Manipulation methods

   Note that manipulation methods always apply to the
   current slide and that removing a slide will decrease
   the indices of succeeding slides.

   addAudio(src)          Current slide must not already
               contain a audio or video element.
               src points to image that is
               from the current page

   addImage(src)          Current slide must not already
               contain an image or video element.
               src points to the image to add
               (relative on server or whole URL)

   addSlide(duration)  No conditions, duration optional

   addText(text)          Current slide must not already
               contain a text element.

   addVideo(src, imgSrc)Current slide must not already
               contain an image or video element.
               src points to the video to add
               (relative on server or whole URL).
               imgSrc points to image to display
               on canvas (videos will not be
               embedded on canvas)

   nextSlide()            Sets the slide to next slide of
               the MMS unless there are no more
               slides.

   post()           Posts the form of the MMS canvas
               to the specified URL. The MMS
               is converted to JSON and specified
               in the paramter "jsonMms". Also
               adds all externally specified
               parameters to the post.

   previousSlide()        Sets the slide to previous slide of
               the MMS unless there are no prevoius
               slides.

   removeAudio()          Removes audio from current slide.
```

removeImage()	Removes image from current slide.
removeSlide()	Removes current slide.
removeText()	Removes text from current slide.
removeVideo()	Removes video from current slide.
setBackgroundColor(color)	Sets the background color of the MMS.
setDuration(newDur)	Sets the duration for the current slide.
setLayout(type)	type must be one of: 1 || 2
setRequestParamter(name, value)	Sets the request paramter of the specified name to the specified value. This can be used to add surrounding info about how the MMS should be handled after it is posted and parsed by the MmsComposerServlet. Such info is for example address of whom to send the MMS to.
setSlide(slideNo)	slideNo must be numeric
setText(text)	Set the text for this slide.
setTextSize(size)	Sets the size of the text on this slide.

Manipulation methods are used to add, manipulate, remove slides, media elements and attributes from the MMS. All manipulation methods return a JSON object with two members: status and message.

```
status:  'ok'|'error'
message: A text string explaining the result
```

Example use of a manipulation method:

```
var res = MmsCanvas.addSlide();
if(res.result != 'ok')
  alert(res.message);
```

State information methods

State information methods are used to get information about the MMS that is being created on the MMS canvas.

getCurrentSlide()	Get the numeric value of the slide currently being displayed and edited.
getDuration()	Get the duration of the current slide.

```
   getLayout()           Get the layout of the current slide.

   getNumSlides()        Get the number of slides of the MMS.

   getRequestParamter(name)
              Get the value of the parameter with
              the specified name.

   getText()             Get the text of the current slide.

   hasAudio()            Returns true if the current slide
              contains an audio file.

   hasImage()            Returns true if the current slide
              contains an image.

   hasNextSlide()        Returns true if there is a next slide.

   hasPreviousSlide() Returns true of there is a previous slide.

   hasText()             Returns true if the current slide
              contains text.

   hasVideo()            Returns true if the current slide
              contains a video file.

***********************************************************/

/* JSON implementation of the MMS Canvas*/

var MmsCanvas = {

  isInited : false,

  //Initialization variables
  height : 320,    //Integer, initialized in init().
  width : 240,     //Integer, initialized in init().
  defaultDuration : '10s', //Default duration for slides
  mms : null,      //Really an object, initialized in init().

  //Dynamic variables (changed during MMS composing phase)
  bgColor : '',   //Background color, default is white
  reqParams : new Array(),

  //HTML element variables
  canvasDiv : '', //Really a div element, initialized in init().
  currentSlide : 0, //Index of the current slide

  //Exposed methods
  init : function(width, height, defaultDuration) {

    this.mms = new MmsSmil();

    if (defaultDuration != null)
      this.defaultDuration = defaultDuration;
```

```
     MmsUtils.debug('Initializing MMS canvas with width: ' + width +
               ', height: ' + height +
               ' and default duration: ' + this.defaultDuration);

     //Init canvas dimensions
     this.width = parseInt(width);
     this.height = parseInt(height);

     if (this.width.toString() == 'NaN' ||
       this.height.toString() == 'NaN') {

       MmsUtils.error('Canvas dimensions were not parsable
integers!');
     }

     //Initialize object representation
     MmsUtils.debug('Initializing MMS object');

     //Initialize HTML elements
    this.canvasDiv = document.getElementById('mms-canvas-container');

     this.isInited = true;

     //Add first slide automatically
     var res = this.addSlide(this.defaultDuration);

     if (res.status != 'ok') {
       MmsUtils.error('Could not add slide');
     }

     MmsUtils.debug('MMS canvas initialized');

   },

   addAudio : function(src) {

     if (! this.isInited)
       return {status : 'error', message : 'MMS canvas was not
initialized!'};

     //Check that current slide doesn't have video or audio already
     var slide = this.mms.slides[this.currentSlide];

     if (slide.video != null)
       return {status : 'error', message : 'Slide already contains a
video'};
     if (slide.audio != null)
       return {status : 'error', message : 'Slide already contains
audio'};

     //Update object representation
     var audioObj = new MmsAudio();

     audioObj.src = src;

     slide.audio = audioObj;
     //Audio has no HTML node, presentation is up to surrounding page

     MmsUtils.debug('Audio added to slide');

     return {status : 'ok', message : 'Audio added'};

   },

   addImage : function(src) {

     if (! this.isInited)
       return {status : 'error', message : 'MMS canvas was not
initialized!'};

     //Check that current slide doesn't have video or image already
     var slide = this.mms.slides[this.currentSlide];

     if (slide.video != null)
       return {status : 'error', message : 'Slide already contains a
video'};
     if (slide.image != null)
       return {status : 'error', message : 'Slide already contains
an image'};

     //Update object representation
     var imageObj = new MmsImage();

     imageObj.src = src;

     slide.image = imageObj;

     //Create HTML node and add it to document
     var imageEl = document.createElement('img');



     var slideEl = document.getElementById('mms-canvas-slide-' +
this.currentSlide);

     if (slideEl == null)
       MmsUtils.error('Slide element not found for slide: ' +
this.currentSlide);

     //Insert element according to layout
     this.insertVisual(imageEl, slideEl);

     //Add event listener that will fit the image when it has loaded
     imageEl.onload = this.resizeImageOnLoad;
     imageEl.onerror = this.onImageError; //Reports error

     imageEl.className = 'mms-canvas-image';
     imageEl.id = 'mms-canvas-image-' + this.currentSlide;
     imageEl.src = src; //Set src last because it may fire onload
right away
     imageEl.style.display = 'block'; //Make sure no space is added
after image
```

```
    MmsUtils.debug('Image added to slide');

    return {status : 'ok', message : 'Image added'};

  },

  //Add a slide to current MMS
  addSlide : function(duration) {

    if (! this.isInited)
      return {status : 'error', message : 'MMS canvas was not
initialized!'};

    //No restrictions, any number of slides may be added to an MMS.

    //Update MMS object representation
    var slide = new MmsSlide();

    if (duration != null) {
      slide.duration = duration;
    }

    var index = this.mms.slides.length;

    this.mms.slides[index] = slide; //Add last

    //Update HTML and set focus to new slide
    if (document.getElementById('mms-canvas-slide-' +
this.currentSlide))
      document.getElementById('mms-canvas-slide-' +
this.currentSlide).style.visibility = 'hidden';

    var htmlSlide = document.createElement('div');

    htmlSlide.className = 'mms-canvas-slide';
    htmlSlide.id = 'mms-canvas-slide-' + index;
    htmlSlide.style.width = this.width + 'px';
    htmlSlide.style.height = this.height + 'px';
    htmlSlide.style.overflow = 'auto';
    htmlSlide.style.position = 'absolute';

    this.canvasDiv.appendChild(htmlSlide);

    this.currentSlide = index;

    MmsUtils.debug('MMS slide added');

    //Return obj with status ('ok'|'error') and message (text
message)
    return {status : 'ok', message : 'Slide added'};

  },

  addText : function(text) {

    if (! this.isInited)
```

```
       return {status : 'error', message : 'MMS canvas was not
initialized!'};

     //Check that current slide doesn't have a text already
     var slide = this.mms.slides[this.currentSlide];

     if (slide.text != null)
       return {status : 'error', message : 'Slide already contains
text'};

     //Update object representation
     var textObj = new MmsText();

     textObj.text = text;

     slide.text = textObj;

     //Create HTML node and add it to document
     var textEl = document.createElement('div');

     textEl.className = 'mms-canvas-text';
     textEl.id = 'mms-canvas-text-' + this.currentSlide;
     textEl.innerHTML = this.plainToHtml(text); //Reformat to HTML
markup

     var slideEl = document.getElementById('mms-canvas-slide-' +
this.currentSlide);

     if (slideEl == null)
       MmsUtils.error('Slide element not found for slide: ' +
this.currentSlide);

     //Insert element according to layout
     this.insertText(textEl, slideEl);

     //Fit elements as good as possible within current boundaries if
the canvas
     this.fitElements();

     MmsUtils.debug('Text added to slide');

     return {status : 'ok', message : 'Text added'};

   },

   addVideo : function(src, imageSrc) {

     if (! this.isInited)
       return {status : 'error', message : 'MMS canvas was not
initialized!'};

     //Check that current slide doesn't have video, audio or image
element already
     var slide = this.mms.slides[this.currentSlide];

     if (slide.video != null)
```

```
       return {status : 'error', message : 'Slide already contains a
video'};
     if (slide.image != null)
       return {status : 'error', message : 'Slide already contains
an image'};
     if (slide.audio != null)
       return {status : 'error', message : 'Slide already contains
audio'};

     //Update object representation
     var videoObj = new MmsVideo();

     videoObj.imageSrc = imageSrc;
     videoObj.src = src;

     slide.video = videoObj;

     //Create HTML node and add it to document
     var imageEl = document.createElement('img');

     var slideEl = document.getElementById('mms-canvas-slide-' +
this.currentSlide);

     if (slideEl == null)
       MmsUtils.error('Slide element not found for slide: ' +
this.currentSlide);

     //Insert element according to layout
     this.insertVisual(imageEl, slideEl);

     //Add event listener that will fit the image when it has loaded
     imageEl.onload = this.resizeImageOnLoad;
     imageEl.onerror = this.onImageError; //Reports error

     imageEl.className = 'mms-canvas-video';
     imageEl.id = 'mms-canvas-video-' + this.currentSlide;
     imageEl.src = imageSrc; //Set src last because it may fire
onload right away
     imageEl.style.display = 'block'; //Make sure no space is added
after image

     MmsUtils.debug('Video added to slide');

     return {status : 'ok', message : 'Video added'};

   },

   nextSlide : function() {

     if (this.hasNextSlide()) {
       return this.setSlide(this.currentSlide + 1);
     }

    return {status : 'error', message : 'There were no more slides'};
   },

   post : function() {
     var form = document.forms['mms-canvas-form'];

     form.jsonMms.value = this.toJsonString();

     //Add hidden inputs for all request parameters
     for (var i = 0; i < this.reqParams.length; i++) {

       var inputEl = document.createElement('input');
       inputEl.type = 'hidden';
       inputEl.name = this.reqParams[i].name;
       inputEl.value = this.reqParams[i].value;
       form.appendChild(inputEl);

     }

     form.submit();
   },

   previousSlide : function() {

     if (this.hasPreviousSlide()) {
       return this.setSlide(this.currentSlide - 1);
     }

     return {status : 'error', message : 'There was no previous
slide'};
   },

   //Remove audio from current slide
   removeAudio : function() {

     if (! this.isInited)
       return {status : 'error', message : 'MMS canvas was not
initialized!'};

     //Check existence
     var slide = this.mms.slides[this.currentSlide];

     if (slide.audio == null)
       return {status : 'error', message : 'No audio to remove'};

     //Remove object representation
     slide.audio = null;

     //Audio has not HTML elements to remove

     MmsUtils.debug('Audio removed from slide');

     return {status : 'ok', message : 'Audio removed'};
   },

   //Remove image from current slide
   removeImage : function() {

     if (! this.isInited)
```

```
      return {status : 'error', message : 'MMS canvas was not
initialized!'};

    //Check existence
    var slide = this.mms.slides[this.currentSlide];

    if (slide.image == null)
      return {status : 'error', message : 'No image to remove'};

    //Remove object representation
    slide.image = null;

    //Remove HTML element
    var htmlImage = document.getElementById('mms-canvas-image-' +
this.currentSlide);

    var parent = htmlImage.parentNode;
    parent.removeChild(htmlImage);

    MmsUtils.debug('Image removed from slide');

    return {status : 'ok', message : 'Image removed'};

  },

  //Remove slide from MMS. Decrease indices for succeeding slides.
  removeSlide : function() {

    if (! this.isInited)
      return {status : 'error', message : 'MMS canvas was not
initialized!'};

    //Copy slide array but exclude the slide to be removed
    var newArr = new Array();

    for (var i = 0; i < this.mms.slides.length; i++) {

      if (i < this.currentSlide) {
        newArr[i] = this.mms.slides[i];
      } else if (i == this.currentSlide) {
        //Don't copy object representation

        //Remove HTML tag
        var htmlSlide = document.getElementById('mms-canvas-slide-'
+ i);
        var parent = htmlSlide.parentNode;
        parent.removeChild(htmlSlide);

      } else if (i > this.currentSlide) {
        newArr[i - 1] = this.mms.slides[i];

        //Change IDs for HTML elements
        var idBasesArr = ['mms-canvas-slide-',
                          'mms-canvas-text-',
                          'mms-canvas-image-',
                          'mms-canvas-audio-',
                          'mms-canvas-video-'];
```

```
          //Change index from i to i -1
          this.changeIdIndices(idBasesArr, i, i - 1);
        }

      }

      this.mms.slides = newArr;

      //Add slide if the last one was removed
      if (this.mms.slides.length == 0)
        this.addSlide();

      //Update current slide
      if (this.currentSlide > 0)
        this.currentSlide--; //Show previous slide

      //Make sure slide is visible
      var newHtmlSlide = document.getElementById('mms-canvas-slide-' +
this.currentSlide);
      newHtmlSlide.style.visibility = 'visible';

      MmsUtils.debug('Slide removed from MMS');

      return {status : 'ok', message : 'Slide removed'};

    },

    //Remove text from current slide
    removeText : function() {

      if (! this.isInited)
        return {status : 'error', message : 'MMS canvas was not
initialized!'};

      //Check existence
      var slide = this.mms.slides[this.currentSlide];

      if (slide.text == null)
        return {status : 'error', message : 'No text to remove'};

      //Remove object representation
      slide.text = null;

      //Remove HTML element
      var htmlText = document.getElementById('mms-canvas-text-' +
this.currentSlide);

      var parent = htmlText.parentNode;
      parent.removeChild(htmlText);

      this.fitElements();

      MmsUtils.debug('Text removed from slide');

      return {status : 'ok', message : 'Text removed'};
```

```
  },

  //Remove video from current slide
  removeVideo : function() {

    if (! this.isInited)
      return {status : 'error', message : 'MMS canvas was not
initialized!'};

    //Check existence
    var slide = this.mms.slides[this.currentSlide];

    if (slide.video == null)
      return {status : 'error', message : 'No video to remove'};

    //Remove object representation
    slide.video = null;

    //Remove HTML element
    var htmlVideo = document.getElementById('mms-canvas-video-' +
this.currentSlide);

    var parent = htmlVideo.parentNode;
    parent.removeChild(htmlVideo);

    MmsUtils.debug('Video removed from slide');

    return {status : 'ok', message : 'Video removed'};

  },

  setBackgroundColor : function(bgColor) {

    if (! this.isInited)
      return {status : 'error', message : 'MMS canvas was not
initialized!'};

    //Set Background color in object representation
    this.bgColor = bgColor;

    //Set background color of HTML element
    var el = document.getElementById('mms-canvas-container');

    el.style.backgroundColor = bgColor;

    MmsUtils.debug('Background color updated to: ' + bgColor);

    return {status : 'ok', message : 'Background color set'};

  },

  /**
   * Set duration for the current slide.
   */
  setDuration : function(newDur) {

    if (! this.isInited)
```

```
        return {status : 'error', message : 'MMS canvas was not 
initialized!'};

      this.mms.slides[this.currentSlide].duration = newDur;

      MmsUtils.debug('Duration set to: ' + newDur);

      return {status : 'ok', message : 'Duration set'};
    },


    //Set layout type. type is on of:
    //1 - Visual media on top
    //2 - Text on top
    setLayout : function(type) {

      if (! this.isInited)
        return {status : 'error', message : 'MMS canvas was not 
initialized!'};

      //type must be 1 or 2
      type = parseInt(type);

      if (type.toString() == 'NaN'){
        return {status : 'error', message : 'Layout type was not 
numeric'};
      } else if (type < 1 || type >  2) {
        return {status : 'error', message : 'Layout type was not 
valid: ' + type};
      }

      if (this.mms.layout == type)
        return {status : 'ok', message : 'Layout type already: ' + 
type};

      this.mms.layout = type;

      return this.changeLayout(type);

    },

    setRequestParameter : function(name, value) {

      //Check for existing parameter
      var existingParam = this.getRequestParameter(name);

      if (existingParam != null) {
        //Parameter exists, update content
        MmsUtils.debug('Updating: ' + value);
        existingParam.value = value;

      } else {
        //Parameter does not exist, add it
        MmsUtils.debug('Adding: ' + value);
        this.reqParams[this.reqParams.length] = {name : name, value : 
value};
      }
```

```
  },

  //Set current slide no
  setSlide : function(slideNo) {

    if (! this.isInited)
      return {status : 'error', message : 'MMS canvas was not
initialized!'};

    slideNo = parseInt(slideNo);

    //slideNo must be numeric and valid index.
    if (slideNo.toString() == 'NaN'){
      return {status : 'error', message : 'Slide number was not
numeric'};
    } else if (slideNo < 0 || slideNo >= this.mms.slides.length) {
      return {status : 'error', message : 'Slide number was not
valid: ' + slideNo};
    } else if (slideNo == this.currentSlide) {
      return {status : 'ok', message : 'Slide already set'};
    }

    //MMS object representation needs no update

    //Update HTML and set focus to new slide
    var htmlSlide = document.getElementById('mms-canvas-slide-' +
slideNo);

    if (htmlSlide == null)
      return {status : 'error', message : 'Could not find slide: '
+ slideNo};

    //Hide previous slide
    document.getElementById('mms-canvas-slide-' +
this.currentSlide).style.visibility = 'hidden';

    this.currentSlide = slideNo;

    //Show new slide
    htmlSlide.style.visibility = 'visible';

    MmsUtils.debug('MMS slide set, current slide is: ' +
this.currentSlide);

    //Return obj with status ('ok'|'error') and message (text
message)
    return {status : 'ok', message : 'Slide set'};

  },

  setTextColor : function(color) {

    if (! this.isInited)
      return {status : 'error', message : 'MMS canvas was not
initialized!'};
```

```
     //Get slide
     var slide = this.mms.slides[this.currentSlide];

     if (slide.text == null)
       return {status : 'error', message : 'Slide has no text, text
color cannot be set'};

     //Update object representation
     slide.text.color = color;

     //Update HTML element
     var htmlText = document.getElementById('mms-canvas-text-' +
this.currentSlide);

     htmlText.style.color = color;

     MmsUtils.debug('Text color updated to: ' + color);

     return {status : 'ok', message : 'Text color updated'};
   },

   /**
    * Set a new text size.
    */
   setTextSize : function(size) {

     if (! this.isInited)
       return {status : 'error', message : 'MMS canvas was not
initialized!'};

     //Get slide
     var slide = this.mms.slides[this.currentSlide];

     if (slide.text == null)
       return {status : 'error', message : 'Slide has no text, text
size cannot be set'};

     //Update object representation
     slide.text.size = size;

     //Update HTML element
     var htmlText = document.getElementById('mms-canvas-text-' +
this.currentSlide);

     htmlText.style.fontSize = size;

     MmsUtils.debug('Text size updated to: ' + size);

     return {status : 'ok', message : 'Text size updated'};
   },

   /*********** Info methods *********************************/

   getCurrentSlide : function() {

     if (! this.isInited)
```

```
      return {status : 'error', message : 'MMS canvas was not
initialized!'};

    return this.currentSlide;

  },

  getDuration : function() {

    if (! this.isInited)
      return {status : 'error', message : 'MMS canvas was not
initialized!'};

    return this.mms.slides[this.currentSlide].duration;
  },

  getLayout : function() {

    return this.mms.layout;

  },

  getNumSlides : function() {

    if (! this.isInited)
      return {status : 'error', message : 'MMS canvas was not
initialized!'};

    return this.mms.slides.length;

  },

  getRequestParameter : function(name) {

    this.reqParams;

    for (var i = 0; i < this.reqParams.length; i++) {

      var param = this.reqParams[i];

      if (param.name == name)
        return param;

    }

    return null;

  },

  //Returns the current text
  getText : function() {

    var slide = this.mms.slides[this.currentSlide];

    if (slide.text == null)
      return '';
    return slide.text.text;
  },

  //Returns a boolean indicating whether the current slide has audio
or not
  hasAudio : function() {

    var slide = this.mms.slides[this.currentSlide];

    if (slide.audio == null)
      return false;

    return true;
  },

  //Returns a boolean indicating whether the current slide has an
image or not
  hasImage : function() {

    var slide = this.mms.slides[this.currentSlide];

    if (slide.image == null)
      return false;

    return true;

  },

  /**
   * Boolean indicating whether there is a next slide after current
one.
   */
  hasNextSlide : function() {

    if (this.currentSlide < this.mms.slides.length -1)
      return true;

    return false;
  },

  /**
   * Boolean indicating whether there is a previous slide before
current one.
   */
  hasPreviousSlide : function() {

    if (this.currentSlide > 0)
      return true;

    return false;
  },

  //Returns a boolean indicating whether the current slide has text
or not
  hasText : function() {

    var slide = this.mms.slides[this.currentSlide];
    if (slide.text == null)
      return false;

    return true;

  },

  //Returns a boolean indicating whether the current slide has video
or not
  hasVideo : function() {

    var slide = this.mms.slides[this.currentSlide];

    if (slide.video == null)
      return false;

    return true;

  },

  /*********** Internal helper methods
*********************************/

  //Center an HTML element
  centerElement : function(el, availableWidth) {

    if (availableWidth == null) //Not specified, use canvas width
      availableWidth = this.width;

    var elWidth = parseInt(el.width);

    if (elWidth.toString() == 'NaN') {
      MmsUtils.error('Element didn\'t contain width attribute');
      return;
    }

    //Margins are mms canvas width - element width
    var margin = Math.floor((availableWidth - elWidth)/2);

    el.style.marginLeft = margin + 'px';
    el.style.marginRight = margin + 'px';

    MmsUtils.debug('Element centered');

  },

  changeIdIndices : function(idBases, oldIndex, newIndex) {

    for (var i = 0; i < idBases.length; i++) {
      var el = document.getElementById(idBases[i] + oldIndex);

      if (el != null)
        el.id = idBases[i] + newIndex;
    }
  },
   changeLayout : function(newType) {

      //Go through all slides and all slides that have both text and
      //visual media will be rearranged.

      var slides = this.mms.slides;

      for (var i = 0; i < slides.length; i++) {

         if ((slides[i].text != null && slides[i].image != null) ||
             (slides[i].text != null && slides[i].video != null)) {

            //Slide contained both text and visual media. Move text
            //to match new layout.
            var text = document.getElementById('mms-canvas-text-' + i);
            var parent = text.parentNode;

            //Remove from previous location - Not required?
            //parent.removeChild(text);

            //MmsUtils.debug('Text node removed');

            this.mms.layout = newType;

            //Move node
            this.insertText(text, parent);
         }
      }

      return {status : 'ok', message : 'Layout type changed'};
   },

   fitElements : function() {

      var parent = document.getElementById('mms-canvas-slide-' +
this.currentSlide);
      var elements = parent.childNodes;

      var imageNode; //Video nodes are also represented as images
      var textNode;

      //Get elements, one img (image or video) and one div (text)
      for (var i = 0; i < elements.length; i++) {

         var el = elements[i];

         if (el.nodeName == 'IMG') {
            imageNode = el;
         } else if(el.nodeName == 'DIV') {
            textNode = el;
         }

      }

      //Text is never resized but check its height.
      var textHeight = 0;
```

```
     if (textNode != null)
       textHeight = textNode.offsetHeight;

     if (imageNode != null) {
       //Fit image node

       MmsUtils.debug('Entering image node resize');

       //Make sure image is not wider than canvas
       if (imageNode.width > this.width) {
         this.resizeImage(imageNode, this.width, null); //Set new
width
         this.centerElement(imageNode);
       }

       //If image size was reduced earlier dur to a text that has
been removed,
       //increase its size again
       if(imageNode.width < this.width &&
          textNode == null) {

         this.resizeImage(imageNode, this.width, null);

       }

       /*
         Decide whether to further downscale the image.

         Image will be downscaled if:
           - There is no text and image height exceeds canvas height
           - There is text and image height + text height exceeds
             canvas height. (This causes scrollbars)

       */

       var imageHeight = imageNode.height;
       var verticalTextConsumption = textHeight / this.height;
  //Defines how much of the height is needed for text
       var verticalImageConsumption = imageHeight / this.height;
  //Defines how much of the height is needed for image

       if (textHeight == 0 && imageNode.height > this.height) {
         //Max height will be canvas height

         this.resizeImage(imageNode, null, this.height); //Set new
height

         MmsUtils.debug('Image resized vertically to fit slide');

       } else if (verticalTextConsumption + verticalImageConsumption
> 1) { //Scrollbars will be added

         //Downscale image vertically

         var newImageHeight = imageHeight;
```

```
        if (verticalTextConsumption > 0.5) { // Don't downscale
image height to less than half
          //Set image height to half the
          newImageHeight = Math.floor(this.height / 2);
        } else {
          newImageHeight = this.height - textHeight - 4;
        }

        MmsUtils.debug(newImageHeight);

        this.resizeImage(imageNode, null, newImageHeight); //Set
new height

        MmsUtils.debug('Image resized vertically to fit slide
together with text');
      }

      //Compensate width property if scrollbars were added
      if (parent.clientWidth != this.width) {

        //Scroll bars were added
        var newWidth = parent.clientWidth;

        if (imageNode.width > newWidth) {
          //Image must be downscaled
          this.resizeImage(imageNode, newWidth, null);

          MmsUtils.debug('Image resized horizontally to fit within
scroll bars');
        }

      }

      this.centerElement(imageNode, parent.clientWidth); //Center

    }

  },

  //Inserts text according to current layout
  insertText : function(el, parent) {

    var layout = this.mms.layout;

    if (layout == 1) {
      //Text at the bottom, do an append
      parent.appendChild(el);
    } else if (layout == 2) {
      //Locate video/image node
      var children = parent.childNodes;
      var otherNode = null;

      for (var i = 0; i < children.length; i++) {
        if (children[i].className == 'mms-canvas-image' ||
          children[i].className == 'mms-canvas-video') {

          otherNode = children[i];
```

```
              break;

            }
          }

          parent.insertBefore(el, otherNode); //If otherNode is still
null the text is put last

        }

      },

      //Inserts visable media according to current layout
      insertVisual : function(el, parent) {

        var layout = this.mms.layout;

        if (layout == 2) {
          //Visual media at the bottom, do an append
          parent.appendChild(el);
        } else if (layout == 1) {
          //Locate text node
          var children = parent.childNodes;
          var textNode = null;

          for (var i = 0; i < children.length; i++) {
            if (children[i].className == 'mms-canvas-text') {

              textNode = children[i];
              break;

            }
          }

          parent.insertBefore(el, textNode); //If textNode is null, the
visual media element is put last

        }

      },

      onImageError : function() {
        MmsUtils.error('Could not load image');
      },

      //Resizes image to scale. Use only height OR width and set other
to null
      resizeImage : function(imgEl, width, height) {

        var SET_HEIGHT = 1;
        var SET_WIDTH = 2;
        var SET_BOTH = 3;

        var mode = 0;

        if (height != null && width == null)
          mode = SET_HEIGHT;
```

```
     else if (height == null && width != null)
       mode = SET_WIDTH;
     else if (height != null && width != null)
       mode = SET_BOTH;

     MmsUtils.debug('Resizing image element with scaling setting: ' +
mode);

     //Initial dimensions, needed to do manual rescaling in IE
     var iWidth = imgEl.width;
     var iHeight = imgEl.height;

     if (iHeight == 0 || iWidth == 0) {
       //Image not properly loaded, report and return
       MmsUtils.error('Image dimensions were 0x0, probably image
wasn\'t loaded');
       return;
     }

     if (mode == SET_HEIGHT) {

       //Clear width style so that image will not change ratio.
       imgEl.style.width = '';

       //Set height to specified height
       imgEl.style.height = height + 'px';

       //Calculate expected value and make sure scale manually if
width wasn't updated automatically
       var scaleFactor = height / iHeight;
       var expectedWidth = Math.floor(scaleFactor * iWidth);

       if (imgEl.width > expectedWidth + 1 || imgEl.width <
expectedWidth - 1) {
         //Not resized horizontally, scale manually.
         imgEl.style.width = expectedWidth + 'px';
       }

     } else if (mode == SET_WIDTH) {

       //Clear height style so that image will not change ratio.
       imgEl.style.height = '';

       //Set width to specified width
       imgEl.style.width = width + 'px';

       //Calculate expected value and make sure scale manually if
height wasn't updated automatically
       var scaleFactor = width / iWidth;
       var expectedHeight = Math.floor(scaleFactor * iHeight);

       if (imgEl.height > expectedHeight + 1 || imgEl.height <
expectedHeight - 1) {
         //Not resized vertically, scale manually.
         imgEl.style.height = expectedHeight + 'px';
       }
```

```
    } else if (mode == SET_BOTH) {

      imgEl.style.height = height + 'px';
      imgEl.style.width = width + 'px';

    }

  },

  /**
   * This function converts plain text to HTML text, replacing
special
   * characters and line breaks
   */
  plainToHtml : function(str) {

    var resultStr = '';

    for (var i = 0; i < str.length; i++) {

      var char = str.charAt(i);

      if (char == '\n')
        resultStr += '<br>';
      else if (char == '<')
        resultStr += '&lt;';
      else if (char == '>')
        resultStr += '&gt;';
      else if (char == '"')
        resultStr += '&quot;';
      else if (char == '&')
        resultStr += '&amp;'
      else if (char == 'å')
        resultStr += '&aring;'
      else if (char == 'Å')
        resultStr += '&Aring;'
      else if (char == 'ä')
        resultStr += '&auml;'
      else if (char == 'Ä')
        resultStr += '&Auml;';
      else if (char == 'ö')
        resultStr += '&ouml;';
      else if (char == 'Ö')
        resultStr += '&Ouml;';
      else
        resultStr += char; //Default

    }

    return resultStr;

  },

  //This method handles resizing of an image after it has finished
loading
  //Note that it seemingly does nothing but the fitElements method
cannot be
```

```
  //set directly as event listener or its variables will not be
available.
  resizeImageOnLoad : function(e) {

    MmsCanvas.fitElements();

  },

  toJsonString : function() {

    //Initialize and add MMS properties
    var jsonStr =  '{\r\n' +
              '\t\"width\" : \"' + this.width + '\",\r\n'+
              '\t\"height\" : \"' + this.height + '\",\r\n' +
              '\t\"layout\" : \"' + this.mms.layout + '\",\r\n' +
              '\t\"backgroundColor\" : \"' + this.bgColor +
'\",\r\n' +
              '\t\"slides\" : [\r\n';

    //Add slides
    for (var i = 0; i < this.mms.slides.length; i++) {

      var hasPrevEl = false; //Indicated if an element has been
added to this slide (requires a comma)

      if (i > 0) //There were slides before, add comma
        jsonStr +=      '\t\t,\r\n';

      jsonStr +=  '\t\t{\r\n'; //Start JSON el

      jsonStr += '\t\t\t\"duration\" : \"' +
this.mms.slides[i].duration + '\"\r\n';

      if (this.mms.slides[i].audio != null) {

        jsonStr += '\t\t\t\,\r\n'; //add comma

        jsonStr += '\t\t\t\"audio\" : \"' +
this.mms.slides[i].audio.src + '\"\r\n';

      }

      if (this.mms.slides[i].image != null) {

        jsonStr += '\t\t\t\,\r\n'; //add comma

        jsonStr += '\t\t\t\"image\" : \"' +
this.mms.slides[i].image.src + '\"\r\n';

      }

      if (this.mms.slides[i].text != null) {

        jsonStr += '\t\t\t\,\r\n'; //add comma

        jsonStr += '\t\t\t\"text\" : {\r\n'; //Start text el
```

```
          jsonStr += '\t\t\t\t\"text\" : \"' +
this.mms.slides[i].text.text + '\",\r\n';
          jsonStr += '\t\t\t\t\"color\" : \"' +
this.mms.slides[i].text.color + '\",\r\n';
          jsonStr += '\t\t\t\t\"size\" : \"' +
this.mms.slides[i].text.size + '\"\r\n';

          jsonStr += '\t\t\t}\r\n'; //End text el

        }

        if (this.mms.slides[i].video != null) {

          jsonStr += '\t\t\t\,\r\n'; //add comma

          jsonStr += '\t\t\t\"video\" : \"' +
this.mms.slides[i].video.src + '\"\r\n';

        }

        jsonStr +=  '\t\t}\r\n'; //End JSON el

      }



      jsonStr +=     '\t]\r\n' +
              '}';

      return jsonStr;
  }

}

/* Utilities for the MMS components */
var MmsUtils = {

  debug : function (str) {
    try {
      MmsDebugListener.debug(str);
    } catch (e) {
      //Do nothing
    }
  },

  error : function(str) {
    try {
      MmsDebugListener.reportError(str);
    } catch (e) {
      //Do nothing
    }
  }
}

//MMS objects

function MmsSmil() {
  this.layout = 1; //1 -> Image/video on top, 2 -> text on top
  this.slides = new Array();

}

function MmsSlide() {

  //MMS info fields
  this.duration = "10s"; //Duration for the slide
  this.text;  //Reference to an MmsText JS obj
  this.image; //Reference to an MmsImage JS obj
  this.video; //Reference to an MmsVideo JS obj
  this.audio; //Reference to an MmsAudio JS obj

  //JavaScript-only fields
  this.id;

}

function MmsText() {

  this.text;
  this.color;
  this.size;

}

function MmsImage() {

  this.src;

}

function MmsVideo() {

  this.src; //Src attribute for the video file on the server, not
displayed
  this.imageSrc; //The src attribute to the image that is displayed
for this video

}

function MmsAudio() {

  this.src;

}
```

Appendix D – E-mail correspondence with Eric Hyche

Appendix F contains e-mail correspondence between the author of this master’s thesis (Kristofer Borgström) and Real Networks employee Eric Hyche.

E-mail 1, sent by Kristofer Borgström 2007-08-17

Hi

I noticed something that seems like a bug while trying to embed a SMIL presentation using RealPlayer (Helix Powered).

If you try to open a SMIL file in an embedded RealPlayer Window, relative URLs within the SMIL are not interpreted properly and you get an error saying something like:

"A general error has occurred"
mem://03D4FAD0/localhost:8080/embedded/smil/msg2/bigfish.jpg

I.e. the player tries to read this file rather than the file relative to the original document which would be:
http://localhost:8080/embedded/smil/msg2/bigfish.jpg.

Is this a bug in the Helix renderer or in RealPlayer? Is there a known solution short of changing all the src attributes to absolute URLs? If this is not the place to report this, where should I?

========= Reproduction ==========
The problem can not be reproduced by opening the SMIL in RealPlayer directly because that actually works. Therefore I added a small demo to one of my sites so that you can see what happens.

The following link has a basic SMIL presentation embedded in it and does not play...
http://www.asia06.com/smil/real.jsp

The HTML on that page references the SMIL file below, which when opened in RealPlayer directly plays fine.
http://www.asia06.com/smil/smil/msg2/hello.smil
============================

Best regards
Kristofer

E-mail 2, sent by Eric Hyche 2007-08-23

What happens when you indirectly reference the SMIL file
through a .ram file? For instance, try having your
.jsp script output the following:

```
-   <embed src="smil/msg2/hello.smil" type="audio/x-pn-realaudio-plugin" ...
+    <embed src="smil/msg2/hello.ram" type="audio/x-pn-realaudio-plugin" ...
```

and then hello.ram would just have one line:

http://www.asia06.com/smil/msg2/hello.smil

Does this work?

Eric

==
Eric Hyche (ehyche@real.com)
Technical Lead
RealNetworks, Inc.

E-mail 3, sent by Kristofer Borgström 2007-09-20

That may or may not work, but it doesn't solve my problem. When this embed tag is generated (using JSP), the absolute URL of the SMIL file is not know so I couldn't generate this .ram file even if I wanted to. Also, I am skeptical to implement proprietary solutions because it leads to too much work when supporting many different players.

In conclusion I suppose you are aware of this weakness although you don't necessarily see it as a bug. Well, due partly to this limitation but mostly due to the very strange behavior of scroll bars when text is displayed, support for RealPlayer has been put at the bottom of my priority list and I am forced to implement a simple SMIL player in HTML/JavaScript. This was a real shame because apart from scroll bars and text (which is central to my solution) the I found the SMIL rendering in RealPlayer to be the best I have found.

/Kristofer

Appendix E – User guide

Appendix E contains the user guides written to describe setup and usage of the MMS components proposed in this master's thesis. This user guide is directed at the Java developer community and puts emphasis is on readability and ease of use. As such, this guide should not be considered a technical report.

MMS Components for Web 2.0

User Guide

ERICSSON

TAKING YOU FORWARD

Copyright

Disclaimer

Trademark List

JAVA™	is a trademark of SUN Microsystems inc.
Windows®	is a registered trademark of the Microsoft Corporation
Apache™	is a trademark of The Apache Software Foundation

Abstract

This document gives an introduction to MMS messaging and describes how to install and use the MMS Components for Web 2.0 in a web application.

Contents

1 Introduction

1.1 Scope

MMS Components for Web 2.0 is used to add MMS capabilities to web sites. The components support both displaying MMS messages as well as creating MMS messages. The components also include support for MMS SMIL, as defined by Open Mobile Alliance[1], which allows MMS messages to be displayed as they would be on a mobile terminal and also ensures that MMS messages that are composed on the web are displayed properly when sent to a mobile terminal.

The components are designed for Java web applications. As such, it contains Servlets and JSP tags that simplify the integration of the components on the web.

The components also include a Java object representation of MMS which can be used to author MMS messages in any context, not just the web. This object representation can then be transformed to an MMS SMIL, thus providing any Java application with an easy way of composing MMS messages.

This document gives a brief introduction to MMS messages in general, thus helping the reader understand what an MMS message is, how it is structured and how it is handled by mobile networks. It also gives a brief introduction to the different components, describes their features and provides examples of how to integrate them on a web page.

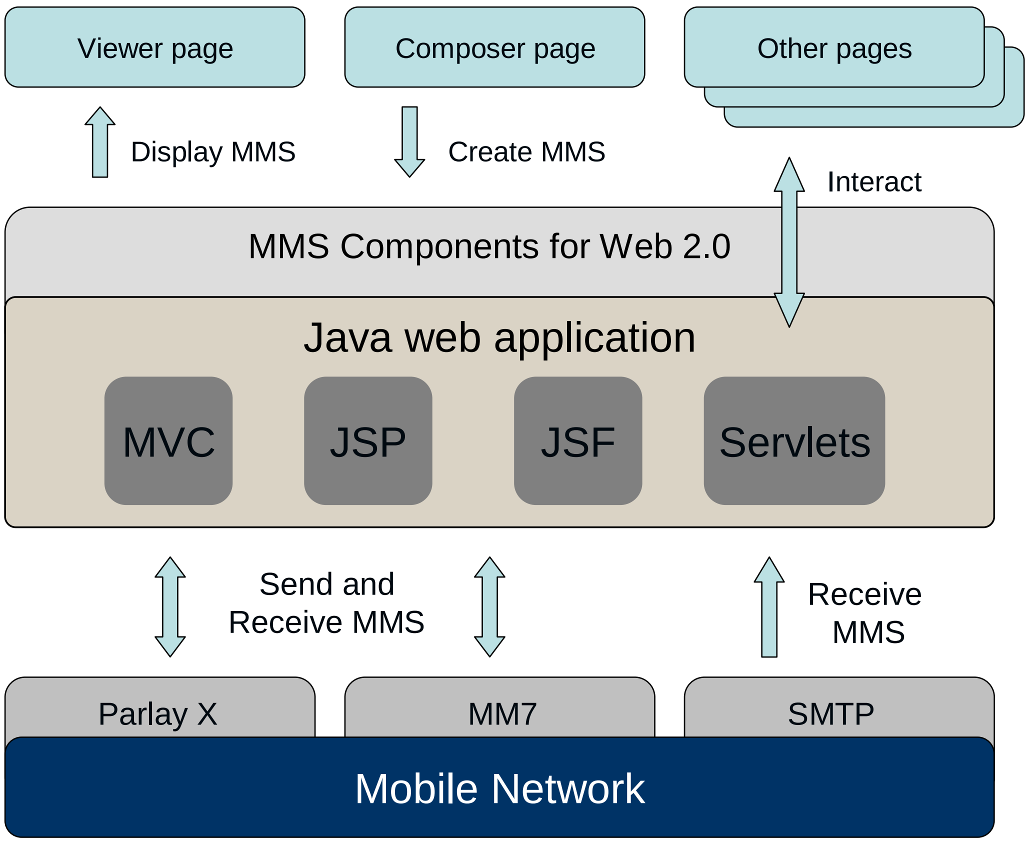


Figure 1 Java web application architecture.

1.2 Purpose

The purpose of the components is to simplify integration of MMS messages on the web. The components are a contribution to Web 2.0 that aims to help bridging the telecom world and the web.

1.3 Target group

This document is for developers who want to add MMS functionality to a web site. It can also benefit people who want a general understanding of MMS messages.

1.4 Document disposition

This document contains two major sections:

- Introduction to MMS messaging
- A description of the components and how to use them.

  This section should be read selectively unless a more complete understanding of the components is desired.
- Compatibility issues

The following layout conventions are used:

- *Italics*
  Sections marked in italics contain background and troubleshooting information – not required reading.
- `Courier New, font size 8`
  This typeface is used for code-related text, parameters, catalog names and file names.

1.5 Compliancy

The MMS components are compliant with all major modern browsers (Internet Explorer, Firefox, Opera and Safari). The components rely on external media players (QuickTime and/or RealPlayer) to play video and audio files. There are some minor known issues regarding browser/media player compatibility which are discussed in section 4.

1.6 Features

The features of the components are listed below.

Table 1 Features of the emulator.

Feature	**Description**
MMS transformation components	Allows the following transformations: • MMS SMIL to Java object representation • Java object representation to MMS SMIL • Java object representation to DHTML • Java object representation to HTML+TIME (Internet

	Explorer only)
MMS transformation Servlet	A servlet that uses the transformation components to create an appropriate HTML view of an MMS.
MMS viewer JSP tag	A JSP custom tag that enables MMS messages to be added to a JSP page using a single line of code.
MMS composer Servlet	A servlet that creates a Java object representation of an MMS sent as a JSON string. The object representation is then saved to disk (or other persistent storage) as MMS SMIL. Finally the MMS composer servlet dispatches the HTTP request to any page or Servlet.
MMS canvas	The MMS canvas consists of a container element of variable size and a set of JavaScript methods used to manipulate the MMS that is “drawn” on the canvas. The resulting MMS message is shown in real time on the canvas as it is composed. Further more, the canvas has JavaScript methods that can be used to post the MMS (preferably to the MMS composer Servlet) and also to add additional parameters to the request.
MMS canvas JSP tag	Allows the MMS canvas to be added to a JSP page using a single line of code. Note that the canvas does not include any graphical interface that calls the JavaScript methods which are used to compose the MMS. Thus allowing the developer to implement their own look and feel.
Open implementation	The source code of the components is included.

2 A brief introduction to MMS messages

2.1 What is an MMS?

Multimedia Messaging Service (MMS) is multimedia messaging format designed for mobile terminals. An MMS message can contain text, images, audio clips and video clips. An MMS message consists of the media files of the message and an MMS SMIL file that defines how the message should be displayed.

MMS SMIL is based on the XML based Synchronized Multimedia Integration Language (SMIL) as defined by the World Wide Web Consortium (W3C) [2]. There are several profiles (subsets) of the SMIL specification. However, none of them were considered suited to multimedia messaging between mobile terminals. Thus, the Open Mobile Alliance (OMA) [1] defined a new subset of the SMIL specification known as MMS SMIL and described it in the MMS Conformance Document [3].

The most notable characteristics of an MMS message are:

- It consists of a series of slides
- Each slide is displayed for a set period of time (its duration)
- A slide may contain only the following visible media elements: one text and one image or one video clip (images and video clips are not allowed on the same slide).
- The same “layout” applies for all slides. This basically means that there are two layout modes that apply to the entire MMS messages: 1 – text on top, 2 – images and video clips on top.
- The slide may also contain background audio if it does not also contain a video clip.

2.2 How is an MMS structured?

2.2.1 MMS SMIL

As mentioned in section 2.1, an MMS message is defined by its media files and an MMS SMIL template that defines how the media elements should be presented. The following example shows an MMS SMIL file that defines layout and timing for a presentation including all allowed types of media elements (text, images, audio and video):

```
<smil>
  <head>
    <layout>
      <root-layout width="240" height="432" />
      <region id="Image" width="100%" height="50%" left="0%" top="0%" fit="meet" />
      <region id="Text" width="100%" height="50%" left="0%" top="50%" fit="meet" />
    </layout>
  </head>
  <body>
    <par dur="5240ms">
      <text region="Text" src="0_0.txt" />
      <img region="Image" src="0_1.jpg" />
      <audio src="0_2.amr" dur="5240ms" />
    </par>
    <par dur="7320ms">
      <text region="Text" src="1_0.txt" />
      <video region="Image" src="1_1.3gp" dur="7320ms" />
    </par>
  </body>
</smil>
```

As seen in the example, an MMS SMIL file contains a `head` section where the layout of the MMS is defined by specifying areas for the regions `Image` and `Text` (the region ids may not assume any other values). The head section is followed by a `body` section that defines timing for each slide (one slide is defined within one `par` tag) and region assignment for the media elements. Each slide is defined by a `par` tag and the media elements within it will be presented simultaneously within each element's specified region.

Note that according to the MMS Conformance Document, an MMS SMIL player is not required to follow the layout strictly. This is because the layout of an MMS may not match the physical limitations (e.g. resolution) of the device that it is displayed on.

2.2.2 Media file references

The different media element tags of MMS SMIL all have a `src` attribute. This attribute defines which media file should be presented by the tag. The media files themselves are also included when an MMS message is sent and each of the media files has a set of headers that describes that file. The `src` attribute may reference the media file in one of two ways as shown in the following code snippets:

```
<text region="Text" src="hello.txt" />

<text region="Text" src="cid:hello.txt" />
```

The first snippet contains the location (URI) of the media file. Specifying only a URI in the MMS SMIL file means that the file referenced by that element should be identified by the HTTP header `Content-Location`. The second tag has a `src` attribute that contains a `cid:` prefix before what appears to be a file name. This prefix specifies that the media is referenced by the `Content-ID` MIME header. It should also be noted that the `Content-ID` header may not contain a valid file name, in which case a file name will have to be generated based on the `Content-Type` header of the file before it can be saved to disk. An example MMS message sent using MM7 is given in section 2.3.

2.3 How are MMS messages sent?

There are several protocols used to transfer MMS messages. They include but are not limited to: MM7, Parlay X Web Services and SMTP. What they share in common is that they all send MMS messages as MIME encoded multipart messages. The multipart message contains one part that is specific to the protocol and another part which contains the MMS message including the MMS SMIL file.

This is illustrated in the following example which contains a trace of an MMS sent using MM7:

```
POST /mms/mm7 HTTP/1.1

Accept: text/xml, text/html, image/gif, image/jpeg, *; q=.2, */*; q=.2

Content-Type: multipart/related; type="text/xml"; boundary="----
=_Part_0_25199001.1192611773065"

Authorization: Basic bW9iaWxpdHlfd2hiuGQ6bGQ1eG05SjVPNg==

Cache-Control: no-cache

Pragma: no-cache

User-Agent: Java/1.6.0

Host: mm7.provider.com

Connection: keep-alive

Content-Length: 2011

------=_Part_0_25199001.1192611773065

Content-Type: text/xml; charset=utf-8

<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"><SOAP-
ENV:Header><mm7:TransactionID
xmlns:mm7="http://www.3gpp.org/ftp/Specs/archive/23_series/23.140/schema/REL-6-MM7-1-
2">TransactionID-1</mm7:TransactionID></SOAP-ENV:Header><SOAP-ENV:Body><mm7:SubmitReq
xmlns:mm7="http://www.3gpp.org/ftp/Specs/archive/23_series/23.140/schema/REL-6-MM7-1-
2"><mm7:MM7Version>6.5.0</mm7:MM7Version><mm7:SenderIdentification><mm7:VASPID>vasp-
```

```
id</mm7:VASPID><mm7:SenderAddress><mm7:ShortCode>71160</mm7:ShortCode></mm7:SenderAddr
ess></mm7:SenderIdentification><mm7:Recipients><mm7:To><mm7:Number>123</mm7:Number></m
m7:To></mm7:Recipients><mm7:ServiceCode>tariffClass=SEK0</mm7:ServiceCode><mm7:Deliver
yReport>true</mm7:DeliveryReport><mm7:Subject>Ericsson IPX</mm7:Subject><mm7:Content
href="cid:attachement_1"/></mm7:SubmitReq></SOAP-ENV:Body></SOAP-ENV:Envelope>

------=_Part_0_25199001.1192611773065

Content-Type:
 multipart/mixed; boundary="----=_Part_0_1484678.1192611773018"

Content-ID: <attachement_1>

------=_Part_0_1484678.1192611773018

Content-Type: application/smil

Content-ID: <smil_1.smil>

<smil>
  <head>
    <layout>
      <root-layout height="240px"width="160px"/>
      <region id="Image" top="0" left="0" height="50%"width="100%" fit="hidden"/>
      <region id="Text" top="50%" left="0" height="50%"width="100%" fit="hidden"/>
    </layout>
  </head>
  <body>
    <par dur="3s">
      <text src="cid:text_1.txt"region="Text"></text>
    </par>
    <par dur="3s">
      <text src="cid:text_2.txt" region="Text"></text>
    </par>
  </body>
</smil>

------=_Part_0_1484678.1192611773018

Content-Type: text/plain

Content-ID: <text_1.txt>

MM sent using Ericsson IPX

------=_Part_0_1484678.1192611773018

Content-Type: text/plain

Content-ID: <text_2.txt>

All for now

------=_Part_0_1484678.1192611773018--

------=_Part_0_25199001.1192611773065--
```

As mentioned above, the message contains an MM7-specific multipart request which contains MM7 data such as who the MMS was sent from, to etc.. The MM7 multipart request in turn, contains another nested multipart encoded part which is the actual MMS message. As seen in the example, each media element in the MMS has the `Content-Type` and the `Content-ID` headers.

As described in section 2.2.2, when the `src` attribute of a media element in an MMS SMIL file has a `cid:` prefix, as is the case in the previous example, a media file is referenced by its `Content-ID` header (not necessarily a valid file name). It should be noted that the `Content-ID` header has the ID enclosed within brackets which need to be stripped away before it is matched against the `src` attribute.

If the `src` attributes would not have contained `cid:` prefixes, the `Content-Location` header would be present for all files. The `Content-Location` does not have any surrounding brackets as the `Content-ID` does. The following code snippet shows what the first text might have looked like if the `src` attribute in the MMS SMIL file would not have had the `cid:` prefix:

```
------=_Part_0_1484678.1192611773018

Content-Type: text/plain

Content-Location: text_1.txt

MM sent using Ericsson IPX

------=_Part_0_1484678.1192611773018
```

2.4 What MMS SDKs are available?

2.4.1 Ericsson MM7 SDK

The Ericsson MM7 SDK is a Java SDK that simplifies sending and receiving MMS messages using the MM7 protocol. It also includes an emulator that enables testing without a live network. The SDK can be downloaded for free from Ericsson Developer Program: http://www.ericsson.com/mobilityworld/sub/open/technologies/mms_mm7/tools/mm7_sdk.

2.4.2 Telecom Web Services SDK (Parlay X)

The Telecom Web Services SDK is a Java SDK that simplifies access to telecom network capabilities using Parlay X Web Services. The SDK includes support for sending and receiving MMS messages and it also includes an emulator which enables testing without a live network. It can be downloaded from Ericsson Developer Program http://www.ericsson.com/mobilityworld/sub/open/technologies/parlayx/tools/tc_ws_sdk_3_0 (released Q1 2008). This is SDK contains two kits that were previously released separately: Java SE Components for Telecom Web Services and Telecom Web Services Network Emulator.

The emulator included in this SDK also utilizes the MMS Components for Web 2.0 and can be used as a reference for how to use the components.

3 How to use the MMS Components

3.1 Getting started

3.1.1 Adding the components to a web application

The following section describes how to add the components to a Java web application. The folder of the web application will be referred to as `<WEB_APP_HOME>`.

- Extract the components to a folder of your choice, this folder will be referred to as `<COMP_HOME>`.
- Add the `<COMP_HOME>/dist/mms-components.jar` to the classpath of the web application. One way of doing this is by copying it to the lib folder of the web application. For example: `<WEB_APP_HOME>/web/WEB-INF/lib/`.
- Copy the contents of the `<COMP_HOME>/web/mms-resources/` folder to a folder within the web content directory of the web application. For example: `<WEB_APP_HOME>/web/mms-resources/`.

  This is the resource directory for the MMS components and it contains JavaScripts, stylesheets and images for the playback control. Make sure to maintain the folder structure within the resource directory.
- To enable the use of the JSP custom tags, copy the `viewer.tag` and `composer.tag` files from the `<COMP_HOME>/web/WEB-INF/tags/` folder to a folder within the WEB-INF folder of the web application. For example: `<WEB_APP_HOME>/web/WEB-INF/tags`.

  Note that the tags require that there is an implementation of the Java Standard Tag Library (JSTL) available in the web container. If there is no such implementation currently installed in the web container that will be used, please refer to section 3.1.2 for details on how to install it.
- To enable the default configuration for the components, copy the `<COMP_HOME>/web/WEB-INF/mms-media-mappings.xml` to the `WEB-INF` folder of the web application. For example: `<WEB_APP_HOME>/web/WEB-INF/mms-media-mappings.xml`.
- To use the default alternate views (see section 3.2.3.2 for more information) copy `mms-loading.jsp`, `mms-error.jsp` and mms-`no-player.jsp` from `<COMP_HOME>/web/` to the web root of the application, for example: `<WEB_APP_HOME>/web/`.

3.1.2 Installing JSTL

If there is no JSTL implementation installed in the web container that the web application will be deployed to, one needs to be installed. The following steps describe how to install Apache's JSTL implementation in a Java web container:

1 Download the Standard Taglib 1.1.2 libraries from http://jakarta.apache.org/site/downloads/downloads_taglibs-standard.cgi.

2 Extract the contents to a directory to a directory of your own choice. This directory will from here on be referred to as `<JSTL_HOME>`, for example, `c:\java\jstl`.

3 Copy `<JSTL_HOME>/lib/jstl.jar` and `<JSTL_HOME>/lib/standard.jar` to the lib folder of the web container. For example: `<TOMCAT6_HOME>/lib`.

3.2 Using the viewer components

3.2.1 Introduction to the viewer components

The MMS viewer components are used to display MMS messages on a web page by transforming an MMS message from MMS SMIL to an HTML view. The transformation of the MMS to an HTML view is performed in real time.

If an MMS contains either audio or video clips, the client's browser is checked for available media player plug-ins and the result of this check is used to decide which media player will be embedded on the page. Possible media players include QuickTime and RealPlayer.

The area within which an MMS message is displayed using the viewer components has a fixed dimension. Playback of an MMS follows the following sequence if playback controls are displayed:

1 The splash screen and playback controls are displayed.

2 When the play button is pushed the playback of the MMS starts and continues as long as its number of set iterations is not exceeded.

3 When the number of iterations of on an MMS is exceeded or when the stop button is pushed, the splash screen is displayed again.

This sequence is depicted in Figure 2.

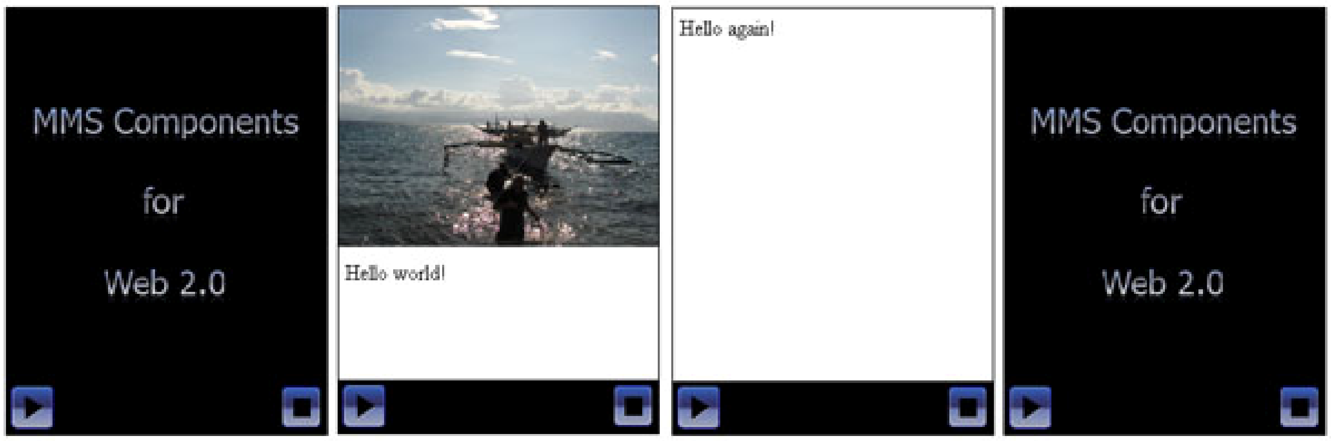


Figure 2 Playback sequence of an MMS with playback controls.

If playback controls are disabled, the MMS will start as soon as it is loaded and when its set number of iterations is exceeded, the splash screen will be displayed. An example of the playback sequence of the same MMS message without playback controls is shown in Figure 3.

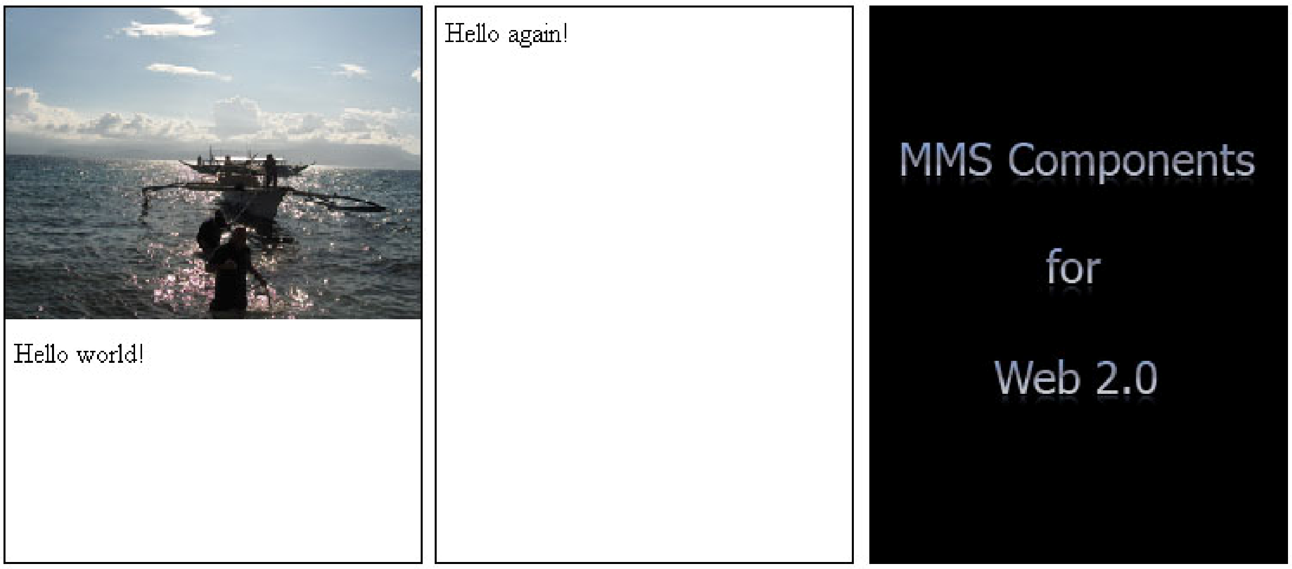


Figure 3 Playback sequence of an MMS without playback controls

3.2.2 Using the MMS viewer custom JSP tag

The following section describes how to add an MMS message to a JSP page. The MMS viewer tag uses the `mms.components.presentation.web.MmsTransformationServlet` class to transform an MMS SMIL file to HTML format and display it on a web page.

At the top of the JSP page, the folder that contains the `viewer.tag` file needs to be declared as a tag file directory and assigned a prefix. Assuming that the `viewer.tag` file has been copied to the `WEB-INF/tags` folder as described in section 3.1.1 this can be done using the following line of JSP code:

```
<%@ taglib prefix="mms" tagdir="/WEB-INF/tags" %>
```

This line of code enables the viewer tag to be added using the `mms` prefix. At this point, an MMS can be added anywhere on the JSP page by adding an mms viewer tag. This is depicted in the following code snippet:

```
<mms:viewer height="320px"
            width="240px"
            mmsDir="/mms/mms1"
            servletPath="MmsTransformationServlet"
            smil="s.smil"
            resourceDir="/mms-resources">
</mms:viewer>
```

This example uses the `mms.components.presentation.web.MmsTransformationServlet` to locate the MMS SMIL file available at `/mms/mms1/s.smil` (relative to context root). The MMS SMIL file will then be transformed to an HTML view that is 240x320 pixels and embedded on the page using an IFrame. The result will be similar to what is displayed in Figure 4.

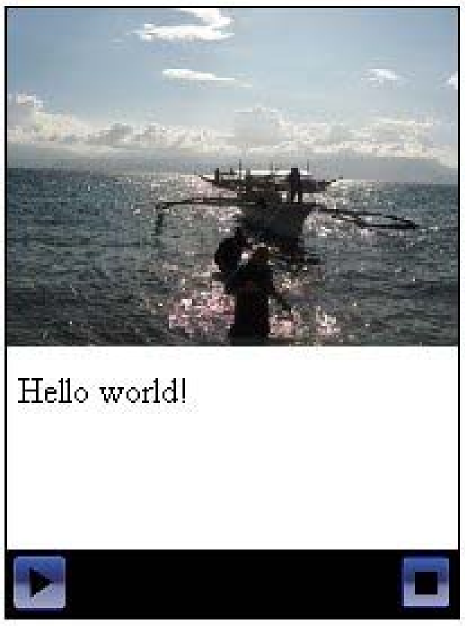


Figure 4 Playback of an MMS using a JSP custom tag.

The MMS viewer tag has a number of attributes that allows the web developer customize which MMS should be displayed and how it should be displayed. All attributes of the MMS viewer tag are described in Table 2. Note that attributes that are not mandatory can be set for all MMS by configuring the `mms.components.presentation.web.MmsTransformationServlet`, see section3.2.3.

Table 2 Attributes of the MMS viewer custom tag

Attribute	Description
height	The height of the MMS that will be added to the page. Should always be specified in pixels and with a "px" suffix. Mandatory: YES Example: `320px`
width	The width of the MMS that will be added to the page. Should always be specified in pixels and with a "px" suffix. Mandatory: YES Example: `240px`
mmsDir	Specifies the directory of the MMS SMIL file relative to the context root of the web application. Must start with a slash. Mandatory: YES Example: `/mms/mms1`
smilFile	Specifies the name of the MMS SMIL file in the `mmsDir` directory. Mandatory: YES Example: `s.smil`
servletPath	Specifies the path to Servlet that is used to generate the HTML view, i.e. the path to `mms.components.presentation.web.MmsTransformationServlet`. Mandatory: YES Example: `MmsTransFormationServlet`

resourceDir	Specifies the directory where the external resources (JavaScripts, images and stylesheets) for the MMS viewer components are located. The directory must start with a slash and is relative to the context root. Note: To avoid setting this property to a custom value for each MMS, this property could be set for all MMS using the properties file for the `mms.components.presentation.web.MmsTransformationServlet`, see section 3.2.3. Mandatory: NO Default: `/mms-resources` Example: `/data/mms-resources`
repeat	Specifies the number of times to play the MMS before the splash screen is displayed. Possible values include positive integers and the keyword `indefinite` to specify that the MMS should play forever. Mandatory: NO Default: `indefinite` Example: `2`
videoScaling	Specifies a custom value for the video scaling property. Mandatory: NO Possible values: `original` - Use original video resolution. `stretch` - Stretch to media player area. `stretch-keep-aspect` - Stretch to media player area but retain aspect ratio. (Default)
controls	A Boolean value specifying whether to show the playback controls below the MMS. If set to `false`, the MMS will start playing as soon as it loads. Mandatory: NO

	Default: `true`
controlsHeight	Specifies the height of the playback controls. Height is specified in pixels and should always have a “px” suffix. Mandatory: NO Default: `30px`

3.2.3 Setting up the MmsTransformationServlet

3.2.3.1 General setup

The `MmsTransformationServlet` is basically a wrapper for the transformation components that allows it to be called directly from a web browser.

To be able to use the Servlet, first make sure that the `mms-components.jar` file is in the class path as described in section 3.1.1. Then, it must be declared and mapped in the deployment descriptor of the web application. The following code snippet shows how the `mms.components.presentation.web.MmsTransformationServlet` could be declared and mapped in the `WEB-INF/web.xml` deployment descriptor of the web application:

```
...
<servlet>
   <servlet-name>MmsTransformationServlet</servlet-name>
   <servlet-class>mms.components.presentation.web.MmsTransformationServlet</servlet-
class>
</servlet>
<servlet-mapping>
   <servlet-name>MmsTransformationServlet</servlet-name>
   <url-pattern>/MmsTransformationServlet</url-pattern>
</servlet-mapping>
...
```

This is the only required configuration for the `MmsTransformationServlet` although there are a lot of customization possibilities explained in sections 3.2.3.2 and3.2.3.3.

3.2.3.2 Setting up optional init parameters

The `mms.components.presentation.web.MmsTransformationServlet` has a number of init parameters that can be specified to customize its behavior. This section describes how to set init parameters for a Servlet and what init parameters are supported by the `mms.components.presentation.web.MmsTransformationServlet`.

To add an init parameter to a Servlet, modify the declaration of the Servlet in the deployment descriptor (`WEB-INF/web.xml`) by adding nested `init-param` elements as depicted in the following example:

```
<servlet>
   <servlet-name>MmsTransformationServlet</servlet-name>
   <servlet-class>mms.components.presentation.web.MmsTransformationServlet</servlet-
class>
   <init-param>
      <param-name>data-source-folder</param-name>
      <param-value>package.DataSourceFolderImpl</param-value>
   </init-param>
</servlet>
```

The above example shows how to specify a custom data source folder class. For a list of all init parameters and descriptions thereof, please refer to Table 3. The data source folder class is used to perform file serving to the transformation components. This enables the developer to use any type of persistence strategy for the MMS messages. By default the `mms.components.FileDataSourceFolder` class is used. This class serves files directly from the file system of the web container and can be used when no advanced persistence-strategy is employed. For information about how to implement a custom data source folder, please refer to the design guide of the MMS Components for Web 2.0.

The result of a request to the `mms.components.presentation.web.MmsTransformationServlet` is not always going to result in a successfully generated MMS view. To handle these cases, there are several alternative pages that will be used to display messages to the user and also to detect which media players are available in the current browser configuration. These pages are listed and described in Table 3.

Table 3 MmsTransformationServlet init parameters

Init parameter	**Description**
data-source-folder	This init parameter is used to specify a class name of a custom data source folder implementation. The data source folder is responsible for serving files to the transformation components from any source such as a database or directly from the file system. The data source folder must implement the interface `mms.components.DataSourceFolder`. If left empty or not specified, the `mms.components.FileDataSourceFolder` is used to serve files directly from the file system which may be sufficient for basic web applications. Example: `package.DbDataSourceFolderImpl`
loading-page	This init parameter is used to specify what page is responsible for detecting which media players are available and report this to the `mms.components.presentation.web.MmsTransformationServlet`. While doing so, it should also display a message to the end user saying that the MMS is loading.

	The value of this parameter is a URI relative to the current context root. It must start with a slash. The default page (/`mms-loading.jsp`, see section 3.1.1) will probably be sufficient in most scenarios and it can easily be modified with a custom look and feel. Example: `/data/mms/mms-loading.jsp`
error-page	This init parameter is used to specify the page that is shown if the transformation fails for some reason. The value of this parameter is a URI relative to the current context root. It must start with a slash. The default page (/`mms-error.jsp`, see section 3.1.1) will probably be sufficient in most scenarios and it can easily be modified with a custom look and feel. Example: `/data/mms/mms-error.jsp`
no-player-page	This init parameter is used to specify the page that is displayed if no media player is available to play a certain media file that is part of an MMS. Note that the no-player-page will **only** be displayed if the transformation configuration parameter that specifies that the transformer throws a `mms.components.transformation.PlayerNotAvailableException` is set to `true` (this is the default value). If this parameter is set to false, the media file will instead be excluded from the MMS and an MMS view with the rest of the content will be displayed. The value of this parameter is a URI relative to the current context root. It must start with a slash. The default page (/`mms-no-player.jsp`, see section 3.1.1) will probably be sufficient in most scenarios and it can easily be modified with a custom look and feel. Example: `/data/mms/mms-no-player.jsp`

3.2.3.3 Specifying custom transformation properties

There is a large set of configuration properties that can be specified in order to customize the output of the MMS transformations (some of these properties, for example repeat count and resource directory, can be specified directly using the JSP custom tag as described in section 3.2.2.).

The custom transformation settings can be customized by adding the property file `WEB-INF/mms-transformation.properties` to the web application. The properties that are specified in this property file correspond to the settings that are defined in the `mms.components.transformation.TransformationConfiguration` class. The following code snippet shows an example extract from this properties file:

```
mms.components.players.order=RealPlayer,QuickTime
mms.components.view.order=HtmlTime,Dhtml
```

All the possible transformation parameters that can be customized using the `WEB-INF/mms-transformation.properties` file are listed in Table 4 along with a description of each property.

Table 4 Transfomation properies listing

Property name	Property description
mms.components.players.order	This property defines which media players may be embedded on a generated HTML page. The order of the media players defined by this property will affect which media player is embedded if several are available and able to play a certain media file. Example: `RealPlayer,QuickTime` (Default)
mms.components.controls.height	This property defines the height in pixels reserved for playback controls (must have a `px` suffix). Example: `30px` (Default)
mms.components.background.color	This property defines the background color used for the generated HTML view. The background color only applies when the splash screen is displayed. During MMS playback, the background color defined in the MMS (or white by default) will be used. Example: `black` (Default)
mms.components.repeat	This property defines the number of times to play the MMS before the splash screen is displayed. Possible values include positive integers and the keyword `indefinite` to specify that the MMS

	should play forever (or until manually stopped). Example: `indefinite` (Default)
mms.components.resource.directory	This property defines the resource directory for the MMS components. This directory should contain images, scripts and style sheets for the MMS components. The value should be relative to the current context root and must start with a slash. Example: `/mms-resources` (default)
mms.components.controls	This property defines whether or not to display playback controls for MMS messages. Possible values include `true` and `false`. Example: `true` (Default)
mms.components.player.not.available .exception	This property defines whether or not to throw an exception if no media player was found that could play a specific media file. Possible values include `true` and `false`. If set to `true`, then the `mms.components.presentation.web.MmsTransformationServlet` will dispatch the request to the no-player-page (see section 3.2.3.2). If set to `false` then any media file for which no media player was found will be excluded from the generated HTML version of the MMS. Example: `true` (Default)
mms.components.video.scaling	This property defines the video scaling setting. Possible values include: • `original` - Use original video resolution. • `stretch` - Stretch to media player area. • `stretch-keep-aspect` - Stretch to media player area but retain aspect ratio. (Default)
mms.components.view.order	This property defines the order of priority for HTML views. Values are comma-separated. Possible values include: • `Dhtml` • `HtmlTime`

Example: `Dhtml,HtmlTime` (Default)

3.2.3.4 Editing the configuration file

The `mms.components.transformation.MmsTransformationServlet` uses an external configuration file to determine what media players can play what types of content, what media players are compatible with what browsers etc. This configuration file should be located in the `WEB-INF` directory of the web application and should be named `mms-components-conf.xml`. The default configuration, which was copied in section 3.1.1, should be sufficient in most cases. However, to simplify custom configuration, this section describes the different parts of this configuration file and how to customize them.

The User-Agent HTTP header is used to determine which browser was used to request an HTML view. Any number of browsers can be defined in the configuration file. Each defined browser has a name, one or more strings to match against the User-Agent header (if several, all must match) as well as elements to define which media players it is compatible with. If none of the defined browsers match the current User-Agent header, the default one will be used. This is depicted in the following extract from the default configuration file:

```
<default-browser>Firefox</default-browser>

<browser name="Firefox">
   <user-agent-match-string>Firefox</user-agent-match-string>
   <compatible-view>Dhtml</compatible-view>
   <compatible-player>QuickTime</compatible-player>
   <compatible-player>RealPlayer</compatible-player>
</browser>

<browser name="InternetExplorer">
   <user-agent-match-string>MSIE</user-agent-match-string>
   <compatible-view>HtmlTime</compatible-view>
   <compatible-view>Dhtml</compatible-view>
   <compatible-player>QuickTime</compatible-player>
   <compatible-player>RealPlayer</compatible-player>
</browser>
```

Furthermore, each media player is defined in the configuration. Included in such a definition is which media formats (identified by the file extension) are supported by the media player and which level of support the media player has for each media type. The different support levels include: `direct`, `auto-download` and `none`. Direct support indicates that the media player has full support for the specified file type. Auto-download support indicates that the media player supports the media file but may require an automatic codec download which may interrupt the user experience. Support level `none` indicates that the media type is not supported by the media player. The following code snippet shows the default configuration for RealPlayer:

```
<player name="RealPlayer">
```

```
   <media-support>
      <media-type ext="amr" support="auto-download"/>
      <media-type ext="aac" support="auto-download"/>
      <media-type ext="au" support="direct"/>
      <media-type ext="wav" support="direct"/>
      <media-type ext="mp3" support="direct"/>
      <media-type ext="mp4" support="direct"/>
      <media-type ext="mid" support="auto-download"/>
      <media-type ext="3gp" support="auto-download"/>
      <media-type ext="3g2" support="auto-download"/>
      <media-type ext="avi" support="auto-download"/>
      <media-type ext="m4a" support="none"/>
   </media-support>
</player>
```

3.2.4 Customizing look and feel

There are several ways of customizing the look and feel of generated MMS messages. This section describes how to do the following customizations:

- Setting CSS style for text in MMS messages
- Customizing the splash screen
- Customizing the playback control bar

3.2.4.1 Customizing CSS attributes for text

To customize set the CSS style for text in MMS messages, locate the file `css/mms-style.css` in the resource directory for the MMS components and open it. To customize text style, modify the style definition for the class `mms-comp-text`. For example, to increase the size of the text and set the color to blue, change the style definition as depicted in the following code snippet:

```
.mms-comp-text {
   padding: 5px;
   font-size: large;
   color: blue;
}
```

3.2.4.2 Customizing the splash screen

Customizing the splash screen for the MMS components can be done in two ways: changing the image that is displayed (and centered) as the splash screen and changing the background color of the splash screen. The image is specified by replacing the `images/splash.jpg` file in the resource directory of the MMS components. The background color of the splash screen can be specified using the `mms.components.background.color` property of the `mms.components.transformation.MmsTransformationServlet`, see section 3.2.3.3 for details on how to specify this property.

3.2.4.3 Customizing the playback control bar

The look and feel of playback control bar can be customized by replacing the images that are used for the bar. The bar contains one play button, one stop button and a spacer between them. The play and stop buttons have three representations. One that is displayed by default, one that is displayed when the mouse hovers over the button and one that is displayed when the button is clicked. These buttons should have the same height and width because the height and width of them will both be specified to match the height of the specified height for the playback control bar. The images used by the playback control bar include (within the resource directory of the MMS components):

- images/play.jpg
- images/play-over.jpg
- images/play-click.jpg
- images/space.jpg
- images/stop.jpg
- images/stop-over.jpg
- images/stop-click.jpg

3.3 Using the composer components

3.3.1 Introduction to the composer components

The composer components are used to embed an MMS canvas on a web page. On the MMS canvas an MMS message can be “drawn” in real-time using a set of JavaScript methods. When the MMS is finished it is converted to a JSON string and posted to the `mms.components.composer.web.MmsComposerServlet` class. This class creates a Java object representation (a `mms.components.mms.MmsMessage` object) of the MMS and saves it as an MMS SMIL file. The HTTP request is then dispatched to a page or a Servlet that is responsible for displaying a message to the user.

Figure 5 below shows an example of an MMS message being “drawn” on the MMS canvas. This example is taken from the Telecom Web Services Emulator which is part of the Telecom Web Services SDK which is available through Ericsson’s Developer Program [4].

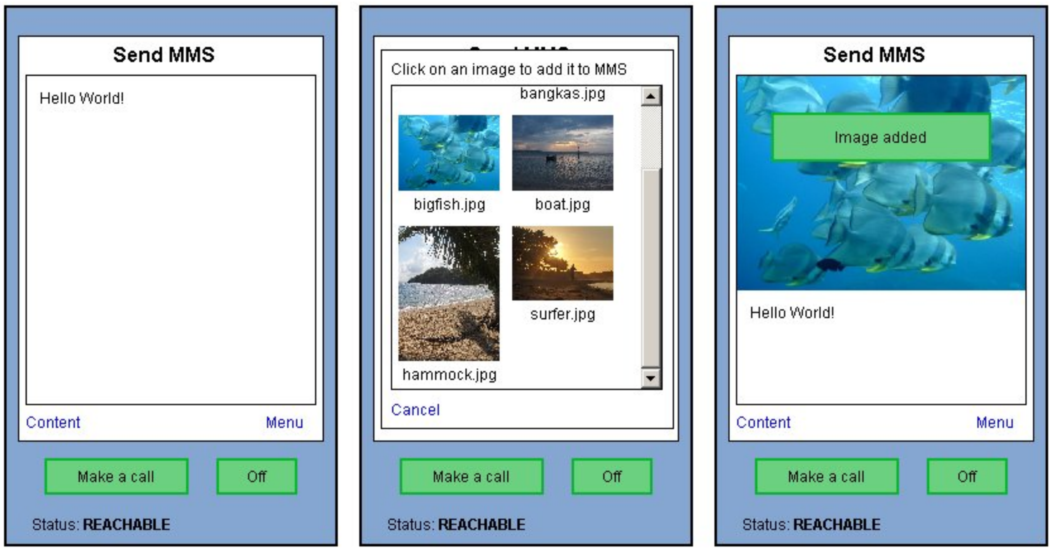


Figure 5 MMS canvas example.

3.3.2 Embedding the MMS canvas

The MMS canvas consists of small snippet of HTML code that is used as a container when "drawing" the MMS message on it and also of a large set of JavaScript methods that can be used to add or remove elements from the MMS, navigate between MMS slides, as well as to get the state of the MMS. There are also methods to add custom request parameters to the request and post the MMS to the `mms.components.composer.web.MmsComposerServlet`.

To simplify the process of embedding the HTML code of the MMS canvas, there is a JSP custom tag that can be used to automate this process. Unlike the MMS viewer HTML code, the MMS canvas is not embedded as an IFrame, but instead as a seamless part of the HTML page. The following section describes how to embed the MMS canvas using the JSP custom tag.

At the top of the JSP page, the folder that contains the `composer.tag` file needs to be declared as a tag file directory and assigned a prefix. Assuming that the `composer.tag` file has been copied to the `WEB-INF/tags` folder as described in section 3.1.1 this can be done using the following line of JSP code:

```
<%@ taglib prefix="mms" tagdir="/WEB-INF/tags" %>
```

When the custom tag has been defined, the MMS canvas can be embedded on a page as depicted in the following code snippet:

```
<mms:composer height="320px"
              width="240px"
              servletPath="MmsComposerServlet"
              forwardPath="mms-composer-ok.jsp"
```

```
                    smilFile="s.smil"
                    targetDir="/mms/mmstest"
                    resourceDir="/mms-resources" />
```

This example will embed the MMS canvas on the web page on area 240 by 320 pixels. The path to the `mms.components.composer.web.MmsComposerServlet` is specified as well as the forward path that specifies to what page the request should be forwarded when the MMS has been composed and saved. The `targetDir` attribute specifies which directory the MMS should be saved to (relative to context root and starting with a slash) and the `smilFile` attribute specifies the name of the saved MMS SMIL file.

The MMS composer tag has a number of attributes that allows the web developer customize how the MMS message is composed. All such attributes are described in. Note that attributes that are not mandatory can be set globally by configuring the `mms.components.composer.web.MmsComposerServlet`, see section 3.3.3.

Table 5 Attributes for the MMS composer custom tag

Attribute	Description
`height`	The height of the MMS canvas that will be added to the page. Should always be specified in pixels and with a "px" suffix. Mandatory: YES Example: `320px`
`width`	The width of the MMS canvas that will be added to the page. Should always be specified in pixels and with a "px" suffix. Mandatory: YES Example: `240px`
`targetDir`	Specifies the directory that the resulting MMS SMIL file will be saved to relative to the context root of the web application. Must start with a slash. Mandatory: YES Example: `/mms/mms1`
`smilFile`	Specifies the name of the MMS SMIL file in the `targetDir` directory.

	Mandatory: YES Example: `s.smil`
`forwardPath`	Specifies the path, relative to the `MmsComposerServlet`, to the page that is responsible for displaying the result of the composed MMS. Mandatory: YES Example: `mms-composer-ok.jsp`
`servletPath`	Specifies the path to Servlet that is used to compose the MMS SMIL file, i.e. the path to the `mms.components.composer.web.MmsComposerServlet`. Mandatory: YES Example: `MmsComposerServlet`
`resourceDir`	Specifies the directory where the external resources (JavaScripts) for the MMS composer components are located. The directory must start with a slash and is relative to the context root. Mandatory: NO Default: `/mms-resources` Example: `/data/mms-resources`
`baseDir`	Specifies the logical directory from where the MMS was composed (src attributes are resolved relative to this directory). This directory must be relative to the current context root and start with a slash. Mandatory: NO Default: `/`
`defaultDur`	Specifies the default duration for new slides. Mandatory: NO Default: `10s`

3.3.3 Configuring the MmsComposerServlet

3.3.3.1 General setup

The `mms.components.composer.web.MmsComposerServlet` is basically a wrapper for the transformation components that allows it to be called directly from a web browser.

To be able to use the Servlet, first make sure that the `mms-components.jar` file is in the class path as described in section 3.1.1. Then, it must be declared and mapped in the deployment descriptor of the web application. The following code snippet shows how the `mms.components.composer.web.MmsComposerServlet` could be declared and mapped in the `WEB-INF/web.xml` deployment descriptor of the web application:

```
...
<servlet>
   <servlet-name>MmsComposerServlet</servlet-name>
   <servlet-class>mms.components.composer.web.MmsComposerServlet</servlet-class>
</servlet>
<servlet-mapping>
   <servlet-name>MmsComposerServlet</servlet-name>
   <url-pattern>/MmsComposerServlet</url-pattern>
</servlet-mapping>
...
```

This is the only required configuration for the `MmsComposerServlet` although there are customization possibilities that are explained in section 3.3.3.2.

3.3.3.2 Setting up optional init parameters

The `mms.components.composer.web.MmsComposerServlet` has a number of init parameters that can be specified to customize its behavior. This section describes how to set init parameters for a Servlet and what init parameters are supported by the `mms.components.composer.web.MmsComposerServlet`.

To add an init parameter to a Servlet, modify the declaration of the Servlet in the deployment descriptor (`WEB-INF/web.xml`) by adding nested `init-param` elements as depicted in the following example:

```
<servlet>
   <servlet-name>MmsComposerServlet</servlet-name>
   <servlet-class>mms.components.composer.web.MmsComposerServlet</servlet-class>
   <init-param>
      <param-name>data-source-folder</param-name>
      <param-value>package.DataSourceFolderImpl</param-value>
   </init-param>
</servlet>
```

The above example shows how to specify a custom data source folder class. For a list of all init parameters and descriptions thereof, please refer to Table 6. The data source folder class is used to perform file serving to the composer components (it is also used to save files). This enables the developer to use any type of persistence strategy for the MMS messages. By default the `mms.components.FileDataSourceFolder` class is used. This class serves files directly from the file system of the web container and can be used when no advanced persistence-strategy is employed. For information about how to implement a custom data source folder, please refer to the design guide of the MMS Components for Web 2.0.

Table 6 Init parameters for MmsComposerServlet

Init parameter	Description
`data-source-folder`	This init parameter is used to specify a class name of a custom data source folder implementation. The data source folder is responsible for serving files to the MMS components from any source such as a database or directly from the file system. The data source folder must implement the interface `mms.components.DataSourceFolder`. If left empty or not specified, the `mms.components.FileDataSourceFolder` is used to serve files directly from the file system which may be sufficient for basic web applications. Example: `package.DbDataSourceFolderImpl`
`copy`	This init parameter specifies whether to copy media files to the target directory along with the resulting MMS SMIL and text files. This enables the `src` attributes of the generated MMS SMIL file to only contain the file name (not the entire URL) and it is strongly recommended to set it to `true`. Default: `true`

3.3.4 Using the MMS canvas

In order to be able to make use of the MMS canvas, a graphical interface needs to be implemented. This graphical interface will be used to add and remove content from the MMS canvas and to post the MMS when it is finished. The JavaScript methods used to compose an MMS are available after the `onload` event is triggered on the page.

All JavaScript methods that are used to compose an MMS message are children of the `MmsCanvas` JavaScript object. Thus “`MmsCanvas.`“ precedes all method calls. For example, the following JavaScript code snippet shows how to add a new slide and add an image and a text to it:

```
MmsCanvas.addSlide();
MmsCanvas.addImage(‘images/myimage.jpg’);
MmsCanvas.addText(‘Hello World!’);
```

There is a large set of methods that can be used to edit the MMS, navigate within it, get the current state of it, set custom request parameters and post the MMS to the `mms.composer.composer.web.MmsComposerServlet`. All the methods of the MMS canvas are listed in Table 7 along with a description of each method.

The methods that affect the state of the MMS all return a JSON object with two members: `status` and `message`. This object can be used to determine whether the method call was successful or not and to display a message to the end user describing the performed action. The `status` member will have one of the values: `ok` or `error`. The message member contains the end user message. The following code snippet depicts how this JSON object could be used to display the result of an action to the end user:

```
var result = MmsCanvas.addImage(‘images/myimage.jpg’);
if (result.status == ‘ok’) {
   notifyUser(result.message); //Method used to notify the end user
} else if (result.status == ‘error’) {
   alert(result.message); //Show a dialog with the error message
}
```

Table 7 MMS canvas JavaScript methods.

Method	Description
`init(width, height, defaultDuration)`	This method should not be called as it is called directly by the HTML code that is used to embed the MMS canvas. Initializes the MMS canvas with height and width and default duration attributes. `defaultDuration` is optional.
`addAudio(src)`	Add an audio clip to the current slide of the MMS. Current slide must not already contain a audio or video element. `src` attribute points to an audio clip (relative to the current page on server or absolute URL). Audio clips will not be visible or audible on the

	MMS canvas. Returns a JSON object.
`addImage(src)`	Add an image to the current slide of the MMS. Current slide must not already contain an image or video element. `src` attribute points to the image to add (relative to the current page on server or absolute URL). Returns a JSON object.
`addSlide(duration)`	Add a slide to the MMS. `duration` attribute is optional. Returns a JSON object.
`addText(text)`	Add a text to the current slide of the MMS. Current slide must not already contain a text element. Returns a JSON object.
`addVideo(src, imgSrc)`	Add a video clip to the current slide of the MMS. Current slide must not already contain an image or video element. `src` points to the video to add (relative to the current page on server or absolute URL). `imgSrc` points to an image to display on the MMS canvas where the video is placed in the resulting MMS (the video files themselves will not be displayed on the canvas). Returns a JSON object.
`getCurrentSlide()`	Gets the index of the current slide. Returns an integer.
`getDuration()`	Gets the duration of the current slide. Returns a string.

`getLayout()`	Gets the current layout of the MMS. 1 – Visual media on top (image/video) 2 – Text on top Returns an integer.
`getNumSlides()`	Gets the total number of slides. Returns an integer.
`getRequestParamter(name)`	Gets the current value of a request parameter. Returns a string
`getText()`	Gets the text on the current slide. Returns a string
`hasAudio()`	Returns true if the current slide contains an audio clip. Returns a boolean.
`hasImage()`	Returns true if the current slide contains an image. Returns a boolean.
`hasNextSlide()`	Returns true if there is a next slide. Returns a boolean.
`hasPreviousSlide()`	Returns true of there is a previous slide. Returns a boolean.
`hasText()`	Returns true if the current slide contains text. Returns a boolean.
`hasVideo()`	Returns true if the current slide contains a video clip. Returns a boolean.

`nextSlide()`	Navigates to the next slide unless there are no more slides. Returns a boolean.
`post()`	Posts the form of the MMS canvas to the specified URL.
`previousSlide()`	Navigates to the previous slide unless there are no previous slides. Returns a JSON object.
`removeAudio()`	Removes audio from current slide. Returns a JSON object.
`removeImage()`	Removes image from current slide. Returns a JSON object.
`removeSlide()`	Removes current slide. Returns a JSON object.
`removeText()`	Removes text from current slide. Returns a JSON object.
`removeVideo()`	Removes video from current slide. Returns a JSON object.
`setBackgroundColor(color)`	Sets the background color of the MMS. Returns a JSON object.
`setDuration(newDur)`	Sets the duration of the current slide. Returns a JSON object.
`setLayout(type)`	Set the layout type of the MMS. 1 – Visual media on top (image/video) 2 – Text on top

	Returns a JSON object.
`setRequestParameter(name, value)`	Sets the request paramter of the specified name to the specified value. This can be used to add surrounding info about how the MMS should be handled after it is posted and parsed by the `mms.components.composer.web.MmsComposerServlet`. Such info is for example the number of whom to send the MMS to. Returns a JSON object.
`setSlide(slideNo)`	Sets the current slide. slideNo must be numeric. Returns a JSON object.
`setTextColor(text)`	Set the text color the text on this slide. Returns a JSON object.
`setTextSize(size)`	Sets the size of the text on this slide. Returns a JSON object.

3.4 Authoring MMS messages from Java code

The components include Java class representations of MMS messages. This class representation enables Java developers to create MMS SMIL in a simple manner using a few lines of Java code. The following code snippet shows how to create an MMS SMIL and save it to disk:

```
mms.components.mms.MmsMessage mms = new mms.components.mms.MmsMessage();

//Initialize the MMS
mms.createRegions(); //Automatically create MMS regions

//Create a slide with duration 15 seconds and add it to the MMS
mms.components.mms.MmsSlide slide1 = new mms.components.mms.MmsSlide("15s", mms);
mms.getSlides().add(slide1);

//Add an image to the slide
mms.components.mms.MmsImage image1 = new mms.components.mms.MmsImage();
image1.setSrc("image.jpg");
slide1.setVisualMedia(image1);

//Add a text file to the slide
mms.components.mms.MmsText text1 = new mms.components.mms.MmsText();
text1.setSrc("text.txt");
slide1.setText(text1);

//Save the file to s.smil
FileOutputStream out = new FileOutputStream("s.smil");
mms.marshallTo(out);
```

4 Known issues

This section contains descriptions of the known issues of the MMS components.

4.1 RealPlayer plug-in together with Opera and Safari browsers

The Opera and Safari (at least on Windows) browsers do not support JavaScript access to the RealPlayer plug-in. Therefore, in the original configuration, a QuickTime plug-in will be required to play video and audio content of MMS messages in these browsers.

4.2 QuickTime 7.1.6 and Firefox

There are some compatibility issues when running QuickTime 7.1.6 plug-in together with Mozilla Firefox. The reason is some known, but yet unfixed bugs in QuickTime 7.1.6. These bugs occasionally result in error messages and even browser crashes.

This is mostly an issue for people running Windows 2000 because on Windows XP and Windows Vista, QuickTime can be updated to a more recent version.

5 Glossary

DHTML	Dynamic HyperText Markup Language
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
JSP	Java Server Pages
JSTL	Java Standard Tag Library
MIME	Multipurpose Internet Mail Extensions
MMS	Multimedia Messages Service
SMIL	Synchronized Multimedia Integration Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
W3C	World Wide Web Consortium
XML	eXtensible Markup Lanugage

6 References

[1] Open Mobile Alliance, http://www.openmobilealliance.org.

[2] World Wide Web Consortium, http://www.w3.org/

[3] OMA Multimedia Messaging Service V1.3, release date: 2005-09-27, Specification - *Multimedia Messaging Service Conformance Document,* http://www.openmobilealliance.org/release_program/mms_archive.html - OMA-TS-MMS-CONF-V1_3-20050927-C.pdf

[4] Ericsson Mobility World Developer Program, http://www.ericsson.com/mobilityworld.

[5] Mozilla Firefox, http://www.mozilla.com/firefox/.

[6] Ericsson IPX, http://www.ericsson.com/solutions/ipx/.

Appendix F – Javadoc

mms.components.mms

Class AbstractMmsContent

```
java.lang.Object
  └mms.components.mms.AbstractMmsContent
```

Direct Known Subclasses:
MmsAudio, MmsText, VisualMmsMedia

```
public abstract class AbstractMmsContent
extends java.lang.Object
```

This class is an abstract implementation for media content. It defines the fields listed below as well as getters and setters for these.

Field	**Description**
`alt`	Alternative text for the content.
`begin`	Defines the start time for this content.
`contentType`	Defines the Content-Type for this content.
`dur`	Defines the duration for this content.
`end`	Defines the end time for this content.
`region`	Defines which region this content should be displayed within. Not applicable for audio.
`sizeInKb`	Defines the size of this content in kilobytes.
`src`	Defines the file name of the content. Sometimes with the prefix "cid:"

Version:
1.0
Author:
Kristofer Borgstrom

Field Summary

`protected java.lang.String`	**`alt`** Alternative text for the content.
`protected java.lang.String`	**`begin`** Defines the start time for this content.
`protected java.lang.String`	**`contentType`** Defines the Content-Type for this content.
`protected java.lang.String`	**`dur`** Defines the duration for this content.

protected java.lang.String	**end** Defines the end time for this content.
protected java.lang.String	**region** Defines which region this content should be displayed within.
protected java.lang.String	**src** Defines the file name of the content.

Constructor Summary

AbstractMmsContent()

Method Summary

java.lang.String	**getAlt**() Get the alt attribute value.
java.lang.String	**getBegin**() Get the begin attribute value.
java.lang.String	**getContentType**() Get the Content-Type .
java.lang.String	**getDur**() Get the dur attribute value.
java.lang.String	**getEnd**() Get the end attribute value.
java.lang.String	**getRegion**() Get the region attribute value.
java.lang.String	**getSrc**() Get the src attribute value.
void	**setAlt**(java.lang.String alt) Set the alt attribute.
void	**setBegin**(java.lang.String begin) Set the begin attribute.
void	**setContentType**(java.lang.String contentType) Set the Content-Type
void	**setDur**(java.lang.String dur) Set the dur attribute.
void	**setEnd**(java.lang.String end) Set the end attribute.
void	**setSrc**(java.lang.String src) Set the src attribute.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

alt

```
protected java.lang.String alt
```

Alternative text for the content.

---

begin

```
protected java.lang.String begin
```

Defines the start time for this content.

---

contentType

```
protected java.lang.String contentType
```

Defines the Content-Type for this content.

---

dur

```
protected java.lang.String dur
```

Defines the duration for this content.

---

end

```
protected java.lang.String end
```

Defines the end time for this content.

---

region

```
protected java.lang.String region
```

Defines which region this content should be displayed within. Not applicable for audio.

---

src

```
protected java.lang.String src
```

Defines the file name of the content. Sometimes with the prefix "cid:"

Constructor Detail

AbstractMmsContent

```
public AbstractMmsContent()
```

Method Detail

getAlt

```
public java.lang.String getAlt()
```

Get the alt attribute value.

Returns:
The alt attribute value.

---

getBegin

```
public java.lang.String getBegin()
```

Get the begin attribute value.

Returns:
The begin attribute value.

---

getContentType

```
public java.lang.String getContentType()
```

Get the Content-Type .

Returns:
The Content-Type

---

getDur

```
public java.lang.String getDur()
```

Get the dur attribute value.

Returns:
The dur attribute value.

---

getEnd

```
public java.lang.String getEnd()
```

Get the end attribute value.

Returns:
The end attribute value.

---

getRegion

```
public java.lang.String getRegion()
```

Get the region attribute value.

Returns:
The region attribute value.

---

getSrc

```
public java.lang.String getSrc()
```

Get the src attribute value.

Returns:
The src attribute value.

---

setAlt

```
public void setAlt(java.lang.String alt)
```

Set the alt attribute.

Parameters:
`alt` - The new value of the alt attribute.

---

setBegin

```
public void setBegin(java.lang.String begin)
```

Set the begin attribute.

Parameters:
`begin` - The new value of the begin attribute.

---

setContentType

```
public void setContentType(java.lang.String contentType)
```

Set the Content-Type

Parameters:
`contentType` - The new Content-Type.

---

setDur

```
public void setDur(java.lang.String dur)
```

Set the dur attribute.

Parameters:
`dur` - The new value of the dur attribute.

---

setEnd

```
public void setEnd(java.lang.String end)
```

Set the end attribute.

Parameters:
`end` - The new value of the end attribute.

---

setSrc

```
public void setSrc(java.lang.String src)
```

Set the src attribute.

Parameters:
`src` - The new value of the src attribute.

---

mms.components

Interface DataSourceFolder

All Known Implementing Classes:
FileDataSourceFolder

`public interface` **`DataSourceFolder`**

A `DataSourceFolder` is linked to a folder. The folder may simply be a folder on the local hard drive, it may also not be an actual folder but instead the data may come from another type of data source such as a database or directly from memory. A `DataSourceFolder` has a method that allows a `javax.activation.DataSource` to be retrieved relative to the folder the `DataSourceFolder` represents.

Version:
1.0

Author:
Kristofer Borgstrom

Method Summary

`javax.activation.DataSource`	**`getDataSource`**`(java.net.URI relativeUri)` Returns the `javax.activation.DataSource` for the specified file relative to directory of this `DataSourceFolder` or null if no such file was found.
`java.net.URL`	**`getFolderUrl`**`()` Gets the complete URL for this `DataSourceFolder`.
`void`	**`init`**`(java.lang.String relativeDir, javax.servlet.ServletContext ctx)` Initiates the DataSourceFolder with context information.

Method Detail

init

```
void init(java.lang.String relativeDir,
          javax.servlet.ServletContext ctx)
          throws java.io.IOException
```

Initiates the DataSourceFolder with context information. The base folder will be the folder referenced by the specified relative directory. The directory will be relative to context path of the specified `ServletContext` and must start with a forward slash.

Parameters:
`relativeDir` -
`ctx` -

Throws:

java.io.IOException

---

getDataSource

```
javax.activation.DataSource getDataSource(java.net.URI relativeUri)
                                   throws java.io.IOException
```

Returns the `javax.activation.DataSource` for the specified file relative to directory of this `DataSourceFolder` or null if no such file was found.

Parameters:
`relativeUri` - The name of the file. For example: "image.jpg".

Returns:
The `javax.activation.DataSource` of the specified resource or null if not found.

Throws:
`java.io.IOException` - For I/O errors.

---

getFolderUrl

```
java.net.URL getFolderUrl()
                   throws java.net.MalformedURLException
```

Gets the complete URL for this `DataSourceFolder`.

Returns:
the URL

Throws:
`java.io.MalformedURLException`

---

mms.components.transformation

Class DhtmlGenerator

```
java.lang.Object
  └mms.components.transformation.DhtmlGenerator
```

All Implemented Interfaces:

HtmlViewGenerator

```
public class DhtmlGenerator
extends java.lang.Object
implements HtmlViewGenerator
```

This class is used to generate DHTML web pages that represent MMS messages.

Version:

1.0

Author:

Kristofer Borgstrom

Constructor Summary

DhtmlGenerator()

Create a new `DhtmlGenerator` instance.

Method Summary

void	**generate**(MmsMessage message, java.io.OutputStream output, TransformationConfiguration config) Generate the DHTML output and write it to the specified `OutputStream`.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

DhtmlGenerator

`public DhtmlGenerator()`

Create a new `DhtmlGenerator` instance.

Method Detail

generate

```
public void generate(MmsMessage message,
                     java.io.OutputStream output,
                     TransformationConfiguration config)
              throws MmsTransformationException,
                     PlayerNotAvailableException,
                     java.io.IOException
```

Generate the DHTML output and write it to the specified `OutputStream`.

Specified by:

`generate` in interface `HtmlViewGenerator`

Parameters:

`message` - The `MmsMessage` for which to generate a new view.
`output` - The `OutputStream` to write the result to.
`config` - The `TransformationConfiguration` to use.

Throws:

`java.lang.IllegalStateException` - If the `MediaUtils` class has not been initialized.
`MmsTransformationException` - If there was an error during transformation.
`PlayerNotAvailableException`
- If no media player was available to play a certain media file and the `THROW_PLAYER_NOT_AVAILABLE_EXCEPTION` setting has been set to "`true`".
`java.io.IOException` - If there is an error while writing to the `OutputStream`.

mms.components

Class FileDataSourceFolder

```
java.lang.Object
  └mms.components.FileDataSourceFolder
```

All Implemented Interfaces:

DataSourceFolder

```
public class FileDataSourceFolder
extends java.lang.Object
implements DataSourceFolder
```

This class is an implementation of the `DataSourceFolder` interface which can be used for all resources that can be referenced by a `java.io.File` object.

Version:
1.0
Author:
Kristofer Borgstrom

Constructor Summary

`FileDataSourceFolder``()`

Method Summary

`javax.activation.DataSource`	**`getDataSource`**`(java.net.URI relativeUri)` Returns the `javax.activation.DataSource` for the specified file relative to directory of this `DataSourceFolder` or null if no such file was found.
`java.net.URL`	**`getFolderUrl`**`()` Gets the complete URL for this `DataSourceFolder`.
`void`	**`init`**`(java.io.File dir)`
`void`	**`init`**`(java.lang.String relativeDir, javax.servlet.ServletContext ctx)` Initiates the DataSourceFolder with context information.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

FileDataSourceFolder

```
public FileDataSourceFolder()
```

Method Detail

getDataSource

```
public javax.activation.DataSource getDataSource(java.net.URI relativeUri)
                                          throws java.io.IOException
```

Description copied from interface: `DataSourceFolder`
Returns the `javax.activation.DataSource` for the specified file relative to directory of this `DataSourceFolder` or null if no such file was found.

Specified by:
`getDataSource` in interface `DataSourceFolder`

Parameters:
`relativeUri` - The name of the file. For example: "image.jpg".

Returns:
The `javax.activation.DataSource` of the specified resource or null if not found.

Throws:
`java.io.IOException` - For I/O errors.

---

init

```
public void init(java.lang.String relativeDir,
                 javax.servlet.ServletContext ctx)
          throws java.io.IOException
```

Description copied from interface: `DataSourceFolder`

Initiates the DataSourceFolder with context information. The base folder will be the folder referenced by the specified relative directory. The directory will be relative to context path of the specified `ServletContext` and must start with a forward slash.

Specified by:
`init` in interface `DataSourceFolder`

Throws:
`java.io.IOException`

---

init

```
public void init(java.io.File dir)
          throws java.io.IOException
```

Throws:
`java.io.IOException`

---

getFolderUrl

```
public java.net.URL getFolderUrl()
                          throws java.net.MalformedURLException
```

Description copied from interface: `DataSourceFolder`
Gets the complete URL for this `DataSourceFolder`.

Specified by:
`getFolderUrl` in interface `DataSourceFolder`

Returns:
the URL

Throws:
`java.net.MalformedURLException`

mms.components.transformation

Class HtmlTimeGenerator

```
java.lang.Object
  └mms.components.transformation.HtmlTimeGenerator
```

All Implemented Interfaces:

HtmlViewGenerator

```
public class HtmlTimeGenerator
extends java.lang.Object
implements HtmlViewGenerator
```

This class is used to generate HTML+TIME web pages that represent MMS messages.

Version:

1.0

Author:

Kristofer Borgstrom

Constructor Summary

`HtmlTimeGenerator``()` Create a new `HtmlTimeGenerator` instance.

Method Summary

`void`	**`generate`**`(MmsMessage message, java.io.OutputStream output, TransformationConfiguration config)` Generate the HTML+TIME output and write it to the specified `OutputStream`.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

HtmlTimeGenerator

```
public HtmlTimeGenerator()
```

Create a new `HtmlTimeGenerator` instance.

Method Detail

generate

```
public void generate(MmsMessage message,
                     java.io.OutputStream output,
                     TransformationConfiguration config)
              throws MmsTransformationException,
                     PlayerNotAvailableException,
                     java.io.IOException
```

Generate the HTML+TIME output and write it to the specified `OutputStream`.

Specified by:

`generate` in interface `HtmlViewGenerator`

Parameters:

`message` - The `MmsMessage` for which to generate a new view.
`output` - The `OutputStream` to write the result to.
`config` - The `TransformationConfiguration` to use.

Throws:

`java.lang.IllegalStateException` - If the `MediaUtils` class has not been initialized.
`MmsTransformationException` - If there was an error during transformation.
`PlayerNotAvailableException`
- If no media player was available to play a certain media file and the `THROW_PLAYER_NOT_AVAILABLE_EXCEPTION` setting has been set to "`true`".
`java.io.IOException` - If there is an error while writing to the `OutputStream`.

mms.components.transformation

Interface HtmlViewGenerator

All Known Implementing Classes:
DhtmlGenerator, HtmlTimeGenerator

`public interface **HtmlViewGenerator**`

This interface defines a method that is used to generate different MMS views such as HTML+TIME for example.

Version:
1.0

Author:
Kristofer Borgstrom

Method Summary

`void`	`generate(MmsMessage message, java.io.OutputStream output, TransformationConfiguration config)` Generate an MMS HTML view from the specified `MmsMessage` using the specified `TransformationConfiguration`.

Method Detail

generate

```
void generate(MmsMessage message,
              java.io.OutputStream output,
              TransformationConfiguration config)
              throws MmsTransformationException,
                     PlayerNotAvailableException,
                     java.io.IOException
```

Generate an MMS HTML view from the specified `MmsMessage` using the specified `TransformationConfiguration`. The result will be written to the specified `OutputStream`.

Parameters:
`message` - The `MmsMessage` for which to generate a new view.
`output` - The `OutputStream` to write the result to.
`config` - The `TransformationConfiguration` to use.

Throws:
`MmsTransformationException` - If there was an error during transformation.
`PlayerNotAvailableException`
- If no media player was available to play a certain media file and the `THROW_PLAYER_NOT_AVAILABLE_EXCEPTION` setting has been set to "`true`".
`java.io.IOException` - If there is an error while writing to the `OutputStream`.

mms.components

Class IoUtils

```
java.lang.Object
  └mms.components.IoUtils
```

```
public class IoUtils
extends java.lang.Object
```

This is a utility class with methods to do basic IO functions such as read/write to files and streams.

Version:
1.0

Author:
Kristofer Borgstrom

Constructor Summary

`IoUtils()`

Method Summary

static java.lang.String	`readInputStream(java.io.InputStream in)` Reads the specified `InputStream` using a 10KB buffer and then converts the read byte data to a String using the platform's default charset.
static java.lang.String	`readTextFile(java.io.File file)` Read a text file to a String using the platforms default charset.
static java.lang.String	`toHtmlText(java.lang.String plainText)` This method converts a plain text string to one that can be displayed in a web browser with text formatting intact.
static void	`transferStream(java.io.InputStream in, java.io.OutputStream out)` Reads the specified `InputStream` and writes to the specified `OutputStream` using a 10KB buffer.

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

IoUtils

```
public IoUtils()
```

Method Detail

transferStream

```
public static void transferStream(java.io.InputStream in,
                                  java.io.OutputStream out)
                           throws java.io.IOException
```

Reads the specified `InputStream` and writes to the specified `OutputStream` using a 10KB buffer. Both `InputStream` and `OutputStream` will be closed by this method.

Parameters:
`in` - The `InputStream` from which to read data.
`out` - The `OutputStream` to which data is written.
Throws:
`java.io.IOException`

readInputStream

```
public static java.lang.String readInputStream(java.io.InputStream in)
                                        throws java.io.IOException
```

Reads the specified `InputStream`
using a 10KB buffer and then converts the read byte data to a String using the platform's default charset.

Parameters:
`in` - The `InputStream` to read.
Returns:
The String data of the `InputStream`.
Throws:
`java.io.IOException`

readTextFile

```
public static java.lang.String readTextFile(java.io.File file)
                                     throws java.io.IOException
```

Read a text file to a String using the platforms default charset.

Parameters:
`file` - The file to read.
Returns:
The String data of the file.
Throws:
`java.io.IOException`

toHtmlText

```
public static java.lang.String toHtmlText(java.lang.String plainText)
```

This method converts a plain text string to one that can be displayed in a web browser with text formatting intact. This implementation only changes line breaks from "\n" or "\r\n" to "
\r\n".

Parameters:

`plainText` - The plain text string.

Returns:

The text as HTML.

mms.components

Class MediaUtils

```
java.lang.Object
  └mms.components.MediaUtils
```

```
public class MediaUtils
extends java.lang.Object
```

This class is a utility class used to get information about browsers, media types and media players. Included methods provide features that allow Content-Type lookup based on file extension and vice versa as well as methods that determine what type of support a given media player has for a certain file extension and what browsers are compatible with what media players and views.

The results from method calls on this class are all based on an XML configuration file which is specified in the constructor. An XML schema for for the XML configuration file is available here An example of such a file is given below:

```
 <media-conf>
   <default-browser>Firefox</default-browser>

         <browser name="Firefox">
                <user-agent-match-string>Firefox</user-agent-match-string>
                <compatible-view>Dhtml</compatible-view>
                <compatible-player>QuicktTime</compatible-player>
                <compatible-player>RealPlayer</compatible-player>
         </browser>

   <player name="QuickTime">
     <media-support>
       <media-type ext="amr" support="direct" />
     </media-support>
   </player>
   <content-type-mappings>
     <type ext="sms" description="SMS over MMS">
       <content-type>application/x-sms</content-type>
       <content-type>application/vnd.3gpp.sms</content-type>
     </type>
   </content-type-mappings>
 </media-conf>
```

Version:
    1.0
Author:
    Kristofer Borgstrom

Method Summary

`static java.lang.String`	`getBrowserByUserAgent(java.lang.String userAgent)` Returns the browser associated with the specified user agent string or the default browser user agent string does not match any of the defined browsers.
`static java.util.List<java.lang.String>`	`getCompatiblePlayersByBrowser(java.lang.String browser)` Returns a list of the media players that are compatible with this browser or an empty list if the browser was not found.
`static java.util.List<java.lang.String>`	`getCompatibleViewsByBrowser(java.lang.String browser)` Returns a list of the views that are compatible with this browser or an empty list if the browser was not found.

`static java.lang.String`	`getContentTypeByExtension(java.lang.String extension)` Get the first Content-Type type matching a given extension.
`static java.lang.String`	`getContentTypeByExtension(java.lang.String extension, java.lang.String prefix)` Get the first Content-Type matching a given extension.
`static java.lang.String`	`getExtensionByContentType(java.lang.String contentType)` Get the extension for a given Content-Type.
`static void`	`init(java.io.File file)` Initialize MediaUtils with the specified file as configuration file.
`static boolean`	`supportsExtDirectly(java.lang.String playerName, java.lang.String extension)` Returns true if the player supports the given extension directly.
`static boolean`	`supportsExtThroughAutoDownload(java.lang.String playerName, java.lang.String extension)` Returns true if the player supports the given extension through automatic download or directly.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Method Detail

init

`public static void **init**(java.io.File file)`

Initialize MediaUtils with the specified file as configuration file.

Parameters:
`file` - The configuration file, a basic example is given in the class summary.

---

getBrowserByUserAgent

`public static java.lang.String **getBrowserByUserAgent**(java.lang.String userAgent)`

Returns the browser associated with the specified user agent string or the default browser user agent string does not match any of the defined browsers.

Parameters:
`userAgent` - User-Agent string
Returns:
Browser name. For example: "Firefox"

---

getCompatiblePlayersByBrowser

`public static java.util.List<java.lang.String> **getCompatiblePlayersByBrowser**(java.lang.String browser)`

Returns a list of the media players that are compatible with this browser or an empty list if the browser was not found.

Parameters:
`browser` - For example: "Firefox"
Returns:

---

getCompatibleViewsByBrowser

```
public static java.util.List<java.lang.String> getCompatibleViewsByBrowser(java.lang.String browser)
```

Returns a list of the views that are compatible with this browser or an empty list if the browser was not found.

Parameters:
`browser` - For example: "Firefox"
Returns:

---

getContentTypeByExtension

```
public static java.lang.String getContentTypeByExtension(java.lang.String extension)
                                                  throws ContentTypeNotFoundException,
                                                         java.lang.IllegalStateException
```

Get the first Content-Type type matching a given extension.

Parameters:
`extension` - For example: "jpg"
Returns:
Content-Type - For example: "image/jpeg"
Throws:
`ContentTypeNotFoundException` - If the content type was not found.
`java.lang.IllegalStateException` - If this class has not been initialized.

---

getContentTypeByExtension

```
public static java.lang.String getContentTypeByExtension(java.lang.String extension,
                                                         java.lang.String prefix)
                                                  throws ContentTypeNotFoundException,
                                                         java.lang.IllegalStateException
```

Get the first Content-Type matching a given extension.

Parameters:
`extension` - For example: "3gp"
`prefix` - The prefix is must match the start of the Content-Type. For example: "video"
Returns:
Content-Type - For example: "video/3gpp".
Throws:
`ContentTypeNotFoundException` - If the content type was not found.
`java.lang.IllegalStateException` - If this class has not been initialized.

---

getExtensionByContentType

```
public static java.lang.String getExtensionByContentType(java.lang.String contentType)
                                                  throws ExtensionNotFoundException,
                                                         java.lang.IllegalStateException
```

Get the extension for a given Content-Type.

Parameters:
`contentType` - For example: "image/jpeg"
Returns:
The extension, for example: "jpg"
Throws:
`ExtensionNotFoundException` - If no extension was found for the specified Content-Type.
`java.lang.IllegalStateException` - If this class has not been initialized.

---

supportsExtDirectly

```
public static boolean supportsExtDirectly(java.lang.String playerName,
                                          java.lang.String extension)
                                   throws PlayerNotFoundException,
                                                       java.lang.IllegalStateException
```

Returns true if the player supports the given extension directly.

Parameters:
`playerName` - For example: "QuickTime"
`extension` - For example: "3gp"

Returns:
true if the player supports the given extension directly.

Throws:
`PlayerNotFoundException` - If no media player with the specified name was found.
`java.lang.IllegalStateException` - If this class has not been initialized.

---

supportsExtThroughAutoDownload

```
public static boolean supportsExtThroughAutoDownload(java.lang.String playerName,
                                                     java.lang.String extension)
                                              throws PlayerNotFoundException,
                                                     java.lang.IllegalStateException
```

Returns true if the player supports the given extension through automatic download or directly.

Parameters:
`playerName` - For example: "QuickTime"
`extension` - For example: "3gp"

Returns:
true if the player supports the given extension through automatic download or directly.

Throws:
`PlayerNotFoundException` - If no media player with the specified name was found
`java.lang.IllegalStateException` - If this class has not been initialized.

---

mms.components.mms

Class MmsAudio

```
java.lang.Object
  └mms.components.mms.AbstractMmsContent
      └mms.components.mms.MmsAudio
```

```
public class MmsAudio
extends AbstractMmsContent
```

This class represents an audio file that can be added to an `MmsSlide`.

Version:
1.0
Author:
Kristofer Borgstrom

Field Summary

Fields inherited from class mms.components.mms.AbstractMmsContent

`alt, begin, contentType, dur, end, region, src`

Constructor Summary

`MmsAudio()` Creates a new `MmsAudio` instance.

Method Summary

`java.lang.String`	`getId()` Get the ID.
`void`	`setId(java.lang.String id)` Set the ID.

Methods inherited from class mms.components.mms.AbstractMmsContent

`getAlt, getBegin, getContentType, getDur, getEnd, getRegion, getSrc, setAlt, setBegin, setContentType, setDur, setEnd, setSrc`

Methods inherited from class java.lang.Object

`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

MmsAudio

```
public MmsAudio()
```

Creates a new `MmsAudio` instance.

Method Detail

getId

```
public java.lang.String getId()
```

Get the ID.

Note that this ID is only applicable when HTML is generated from the MMS. and that when HTML is being generated this the ID must exist and be unique within the current MMS.

Returns:
The id.

---

setId

```
public void setId(java.lang.String id)
```

Set the ID.

Note that this ID is only applicable when HTML is generated from the MMS. and that when HTML is being generated this the ID must exist and be unique within the current MMS.

Parameters:
`id` - The new ID.

---

mms.components.composer.web

Class MmsComposerServlet

```
java.lang.Object
  └javax.servlet.GenericServlet
      └javax.servlet.http.HttpServlet
          └mms.components.composer.web.MmsComposerServlet
```

All Implemented Interfaces:
java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

```
public class MmsComposerServlet
extends javax.servlet.http.HttpServlet
implements javax.servlet.Servlet
```

This servlet handles incoming MMS composer requests and will write an MMS SMIL file to the specified target directory.

All file I/O operations will be performed using the specified `DataSourceFolder` implementation (specified as init parameter). If not specified, the default `FileDataSourceFolder` will be used.

NOTE The `DataSourceFolder`
must have a constructor that takes a String and a ServletContext in that order. The String defines the directory relative to the context root and the ServletContext may used to determine context information.

If the `copy`
property for the MMS is set to true, all media sources will be resolved and copied to the target directory.

Version:
1.0
Author:
Kristofer Borgstrom
See Also:
Serialized Form

Field Summary

`static java.lang.String`	**`INIT_PARAM_COPY_CONTENT`** This static field defines whether to copy the contents (media files) of composed MMS messages to the target directory.
`static java.lang.String`	**`INIT_PARAM_CUSTOM_DATA_SOURCE_FOLDER`** This static field contains the class name of the servlet init param that can be used to set a custom `DataSourceFolder` from which is used as an I/O base for reading and writing to files.
`static java.lang.String`	**`INIT_PARAM_ERROR_PAGE`** This static field defines the name of the servlet init param that can be used to set a custom error page that will be shown when error occurs while

	composing the MMS.
static java.lang.String	**REQUEST_ATTRIBUTE_MMS** This static field defines the name of the request attribute that is set by this servlet to temporarily store an `MmsMessage` instance that represents the generated MMS message.
static java.lang.String	**REQUEST_ATTRIBUTE_SMIL_FILE** This static field defines the name of the request attribute that is set by this servlet to indicate the resulting name of the MMS SMIL that was created.
static java.lang.String	**REQUEST_PARAM_BASE_DIR** This static field contains the parameter name of the URI of the base directory for this MMS (This is the URI of the directory of the composer page.
static java.lang.String	**REQUEST_PARAM_FORWARD_PATH** This static field contains the parameter name of parameter that defines what page the request will be forwarded or redirected to if the MMS is successfully composed.
static java.lang.String	**REQUEST_PARAM_MMS_AS_JSON** This static field contains the parameter name of the parameter that contains the MMS in JSON format.
static java.lang.String	**REQUEST_PARAM_TARGET_DIR** This static field contains the parameter name of the URI of the target directory for this MMS (This is the URI of the directory where the the SMIL and text files are saved, if init parameter `INIT_PARAM_COPY_CONTENT` is set to true then media files are also copied to this directory.)
static java.lang.String	**SMIL_FILE_PARAM_NAME** This static field defines the parameter name ("smilFile") to use for requests to this servlet.

Constructor Summary

MmsComposerServlet()

Method Summary

protected void	**dispatch**(java.lang.String page, javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse res) Dispatch to the specified page.
protected void	**doGet**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response)
protected void	**doPost**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response)
protected void	**handleRequest**(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response)
void	**init**()

Methods inherited from class javax.servlet.http.HttpServlet
`doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service`

Methods inherited from class javax.servlet.GenericServlet
`destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, log, log`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Methods inherited from interface javax.servlet.Servlet
`destroy, getServletConfig, getServletInfo, init, service`

Field Detail

INIT_PARAM_CUSTOM_DATA_SOURCE_FOLDER

`public static final java.lang.String` **`INIT_PARAM_CUSTOM_DATA_SOURCE_FOLDER`**

This static field contains the class name of the servlet init param that can be used to set a custom `DataSourceFolder`
from which is used as an I/O base for reading and writing to files. If not set, the files will be read from disk on the web server using the `FileDataSourceFolder` class.

Init param name: "`data-source-folder`"

Use: **OPTIONAL**

Default: **FileDataSourceFolder**

See Also:
Constant Field Values

INIT_PARAM_COPY_CONTENT

`public static final java.lang.String` **`INIT_PARAM_COPY_CONTENT`**

This static field defines whether to copy the contents (media files) of composed MMS messages to the target directory. Src attributes in the resulting MMS SMIL will also be affected by this option. If true then src attributes will only be the file names of the media files. If false then the src attribute will contains relative to the server root and start with a '/' (if relative e.g. "../a.jpg") or it will contain the absolute path to the media file (if absolute e.g. "http://a.com/b.jpg").

Init param name: "`copy`"

Use: **OPTIONAL**

Default: **true**

See Also:

Constant Field Values

REQUEST_ATTRIBUTE_MMS

`public static final java.lang.String` **`REQUEST_ATTRIBUTE_MMS`**

This static field defines the name of the request attribute that is set by this servlet to temporarily store an `MmsMessage`
instance that represents the generated MMS message. This attribute can be used on the page that the request is dispatched to after an MMS SMIL was successfully composed.

See Also:
Constant Field Values

REQUEST_ATTRIBUTE_SMIL_FILE

`public static final java.lang.String` **`REQUEST_ATTRIBUTE_SMIL_FILE`**

This static field defines the name of the request attribute that is set by this servlet to indicate the resulting name of the MMS SMIL that was created. This attribute can then be used on the page that the request is dispatched to after an MMS SMIL was successfully composed.

Note that request parameters such as `baseDir` and `targetDir` will still be available and can be used to determine more exact path info of the MMS SMIL file.

See Also:
Constant Field Values

INIT_PARAM_ERROR_PAGE

`public static final java.lang.String` **`INIT_PARAM_ERROR_PAGE`**

This static field defines the name of the servlet init param that can be used to set a custom error page that will be shown when error occurs while composing the MMS.

Init param name: "`error-page`"

Use: **OPTIONAL**

NOTE! The pathname specified may be relative, although it cannot extend outside the current servlet context. If the path begins with a "/" it is interpreted as relative to the current context root.

Default: `/mms-composer-error.jsp`

See Also:
Constant Field Values

REQUEST_PARAM_MMS_AS_JSON

`public static final java.lang.String` **`REQUEST_PARAM_MMS_AS_JSON`**

This static field contains the parameter name of the parameter that contains the MMS in JSON format.

Request param name: "`jsonMms`"

Use: **MANDATORY**

Default: -

See Also:
Constant Field Values

REQUEST_PARAM_FORWARD_PATH

`public static final java.lang.String` **`REQUEST_PARAM_FORWARD_PATH`**

This static field contains the parameter name of parameter that defines what page the request will be forwarded or redirected to if the MMS is successfully composed.

Request param name: "`okPage`"

Use: **OPTIONAL**

NOTE! The pathname specified may be relative, although it cannot extend outside the current servlet context. If the path begins with a "/" it is interpreted as relative to the current context root.

Default: `/mms-composer-ok.jsp`

See Also:
Constant Field Values

REQUEST_PARAM_BASE_DIR

`public static final java.lang.String` **`REQUEST_PARAM_BASE_DIR`**

This static field contains the parameter name of the URI of the base directory for this MMS (This is the URI of the directory of the composer page. I.e. the page that sent the request to this servlet).

Request param name: "`baseDir`"

Use: **OPTIONAL**

NOTE! The URI must be relative to context root and start with a '/'.

Default: `/`

See Also:
Constant Field Values

REQUEST_PARAM_TARGET_DIR

`public static final java.lang.String` **`REQUEST_PARAM_TARGET_DIR`**

This static field contains the parameter name of the URI of the target directory for this MMS (This is the URI of the directory where the the SMIL and text files are saved, if init parameter `INIT_PARAM_COPY_CONTENT` is set to true then media files are also copied to this directory.)

Request param name: "`targetDir`"

Use: **OPTIONAL**

NOTE! The URI must be relative to context root and start with a '/'.

Default: `/`

See Also:
Constant Field Values

SMIL_FILE_PARAM_NAME

```
public static final java.lang.String SMIL_FILE_PARAM_NAME
```

This static field defines the parameter name ("smilFile") to use for requests to this servlet.

See Also:
Constant Field Values

Constructor Detail

MmsComposerServlet

```
public MmsComposerServlet()
```

Method Detail

init

```
public void init()
          throws javax.servlet.ServletException
```

Overrides:
`init` in class `javax.servlet.GenericServlet`

Throws:
`javax.servlet.ServletException`

doGet

```
protected void doGet(javax.servlet.http.HttpServletRequest request,
                     javax.servlet.http.HttpServletResponse response)
              throws javax.servlet.ServletException,
                     java.io.IOException
```

Overrides:
`doGet` in class `javax.servlet.http.HttpServlet`

Throws:
`javax.servlet.ServletException`
`java.io.IOException`

doPost

```
protected void doPost(javax.servlet.http.HttpServletRequest request,
                      javax.servlet.http.HttpServletResponse response)
```

```
             throws javax.servlet.ServletException,
                    java.io.IOException
```

Overrides:

`doPost` in class `javax.servlet.http.HttpServlet`

Throws:

`javax.servlet.ServletException`

`java.io.IOException`

---

handleRequest

```
protected void handleRequest(javax.servlet.http.HttpServletRequest request,
                             javax.servlet.http.HttpServletResponse response)
                      throws javax.servlet.ServletException,
                             java.io.IOException
```

Throws:

`javax.servlet.ServletException`

`java.io.IOException`

---

dispatch

```
protected void dispatch(java.lang.String page,
                        javax.servlet.http.HttpServletRequest req,
                        javax.servlet.http.HttpServletResponse res)
                 throws javax.servlet.ServletException,
                        java.io.IOException
```

Dispatch to the specified page.

Parameters:

`page` - The page to which to dispatch.

`req` - The `HttpServletRequest`.

`res` - The `HttpServletResponse`.

Throws:

`javax.servlet.ServletException`

`java.io.IOException`

---

mms.components.mms

Class MmsImage

```
java.lang.Object
  └mms.components.mms.AbstractMmsContent
      └mms.components.mms.VisualMmsMedia
          └mms.components.mms.MmsImage
```

```
public class MmsImage
extends VisualMmsMedia
```

This class represents an image file that can be added to an `MmsSlide`.

Version:
1.0

Author:
Kristofer Borgstrom

Field Summary

Fields inherited from class mms.components.mms.AbstractMmsContent
`alt, begin, contentType, dur, end, region, src`

Constructor Summary

`MmsImage``()` Creates a new `MmsImage` instance.

Method Summary

`int`	**`getHeight`**`()` Get the height of the image in pixels.
`int`	**`getWidth`**`()` Get the width of the image in pixels.
`void`	**`setHeight`**`(int height)` Set the height of the image in pixels.
`void`	**`setWidth`**`(int width)` Set the width of the image in pixels.

Methods inherited from class mms.components.mms.AbstractMmsContent
`getAlt, getBegin, getContentType, getDur, getEnd, getRegion, getSrc, setAlt, setBegin, setContentType, setDur, setEnd, setSrc`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

MmsImage

`public MmsImage()`

Creates a new `MmsImage` instance.

Method Detail

getHeight

`public int getHeight()`

Get the height of the image in pixels.

Returns:
The height of the image in pixels.

getWidth

`public int getWidth()`

Get the width of the image in pixels.

Returns:
The width of the image in pixels.

setHeight

`public void setHeight(int height)`

Set the height of the image in pixels.

Parameters:
`height` - The height of the image in pixels.

setWidth

`public void setWidth(int width)`

Set the width of the image in pixels.

Parameters:
`width` - The width of the image in pixels.

mms.components.mms

Enum MmsLayout

```
java.lang.Object
  └java.lang.Enum<MmsLayout>
      └mms.components.mms.MmsLayout
```

All Implemented Interfaces:

java.io.Serializable, java.lang.Comparable<MmsLayout>

```
public enum MmsLayout
extends java.lang.Enum<MmsLayout>
```

This enum represents different layouts for MMS messaages.

Version:

1.0

Author:

Kristofer Borgstrom

Enum Constant Summary

TEXT_ON_TOP
VISUAL_MEDIA_ON_TOP

Method Summary

static MmsLayout	**valueOf**(java.lang.String name) Returns the enum constant of this type with the specified name.
static MmsLayout[]	**values**() Returns an array containing the constants of this enum type, in the order they are declared.

Methods inherited from class java.lang.Enum

clone, compareTo, equals, finalize, getDeclaringClass, hashCode, name, ordinal, toString, valueOf

Methods inherited from class java.lang.Object

getClass, notify, notifyAll, wait, wait, wait

Enum Constant Detail

VISUAL_MEDIA_ON_TOP

```
public static final MmsLayout VISUAL_MEDIA_ON_TOP
```

TEXT_ON_TOP

`public static final MmsLayout` **`TEXT_ON_TOP`**

Method Detail

values

`public static MmsLayout[]` **`values`**`()`

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

```
for (MmsLayout c : MmsLayout.values())
    System.out.println(c);
```

Returns:
an array containing the constants of this enum type, in the order they are declared

valueOf

`public static MmsLayout` **`valueOf`**`(java.lang.String name)`

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

Parameters:
`name` - the name of the enum constant to be returned.
Returns:
the enum constant with the specified name
Throws:
`java.lang.IllegalArgumentException` - if this enum type has no constant with the specified name
`java.lang.NullPointerException` - if the argument is null

mms.components.mms

Class MmsMessage

```
java.lang.Object
  └mms.components.mms.MmsMessage
```

```
public class MmsMessage
extends java.lang.Object
```

This class represents an MMS message.

Version:
1.0

Author:
Kristofer Borgstrom

Constructor Summary

`MmsMessage()` Create a new `MmsMessage`.

Method Summary

`void`	`createRegions()` This method automatically creates region elements according to height and width attributes and current layout, effectively replacing any existing regions.
`java.util.List<AbstractMmsContent>`	`getAllMediaContent()` Returns a List of all the media content elements of this MMS including text, audio, image and video elements.
`java.lang.String`	`getBackgroundColor()` Get the background color of this MMS.
`java.lang.String`	`getHeight()` Get the height of the MMS.
`MmsLayout`	`getLayout()` Get the layout for this MMS.
`DataSourceFolder`	`getMmsFolder()` Get the `DataSourceFolder` for this MMS.
`java.util.List<MmsRegion>`	`getRegions()` Get a `List` of `MmsRegion` objects for this MMS.
`java.util.List<MmsSlide>`	`getSlides()` Get a `List` of `MmsSlide` objects for this MMS.
`java.lang.String`	`getWidth()` Get the width of the MMS.
`boolean`	`hasComplexContent()` Returns true if the MMS contains any audio or video.

void	**marshallTo**(java.io.OutputStream out) Marshall this MMS as MMS SMIL to the specified `OutputStream` using compiled JAXB stubs.
void	**setBackgroundColor**(java.lang.String backgroundColor) Set the background color of this MMS.
void	**setHeight**(java.lang.String height) Set the height of the MMS.
void	**setLayout**(MmsLayout layout) Set the layout of this MMS.
void	**setMmsFolder**(DataSourceFolder mmsFolder) Set the `DataSourceFolder` for this MMS.
void	**setWidth**(java.lang.String width) Set the width of the MMS.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

MmsMessage

`public **MmsMessage**()`

Create a new `MmsMessage`.

Method Detail

createRegions

`public void **createRegions**()`

This method automatically creates region elements according to height and width attributes and current layout, effectively replacing any existing regions.

---

getMmsFolder

`public DataSourceFolder **getMmsFolder**()`

Get the `DataSourceFolder` for this MMS.

The folder is required to read files referenced by the SMIL.

Returns:
The `DataSourceFolder` for this MMS.

---

setMmsFolder

`public void **setMmsFolder**(DataSourceFolder mmsFolder)`

Set the `DataSourceFolder` for this MMS.

The folder is required to read files referenced by the SMIL.

Parameters:

`mmsFolder` - The new `DataSourceFolder` for this MMS.

---

getRegions

```
public java.util.List<MmsRegion> getRegions()
```

Get a `List` of `MmsRegion` objects for this MMS.

Returns:

The regions for this MMS.

---

getAllMediaContent

```
public java.util.List<AbstractMmsContent> getAllMediaContent()
```

Returns a List of all the media content elements of this MMS including text, audio, image and video elements.

Returns:

A List of media content elements.

---

getSlides

```
public java.util.List<MmsSlide> getSlides()
```

Get a `List` of `MmsSlide` objects for this MMS.

Returns:

the slides

---

getBackgroundColor

```
public java.lang.String getBackgroundColor()
```

Get the background color of this MMS.

Returns:

The background color

---

getLayout

```
public MmsLayout getLayout()
```

Get the layout for this MMS.

Returns:

The layout of the MMS.

---

getHeight

```
public java.lang.String getHeight()
```

Get the height of the MMS. Must have a "px" suffix.

Note that this is the height of the MMS as added in the SMIL file, it will not affect how the MMS is presented when transformed to HTML.

Returns:
The height of the MMS.

---

getWidth

```
public java.lang.String getWidth()
```

Get the width of the MMS. Must have a "px" suffix.

Note that this is the width of the MMS as added in the SMIL file, it will not affect how the MMS is presented when transformed to HTML.

Returns:
The width of the MMS.

---

hasComplexContent

```
public boolean hasComplexContent()
```

Returns true if the MMS contains any audio or video.

Returns:
true if the MMS contains any audio or video.

---

marshallTo

```
public void marshallTo(java.io.OutputStream out)
                throws javax.xml.bind.JAXBException
```

Marshall this MMS as MMS SMIL to the specified `OutputStream` using compiled JAXB stubs.

Parameters:
`out` - the `OutputStream` to marshall to.

Throws:
`javax.xml.bind.JAXBException` - for JAXB related errors.

---

setBackgroundColor

```
public void setBackgroundColor(java.lang.String backgroundColor)
```

Set the background color of this MMS.

Parameters:
`backgroundColor` - The new background color.

---

setLayout

```
public void setLayout(MmsLayout layout)
```

Set the layout of this MMS.

Parameters:
`layout` - The new layout.

---

setHeight

```
public void setHeight(java.lang.String height)
```

Set the height of the MMS. Must have a "px" suffix

Note that this is the height of the MMS as added in the SMIL file, it will not affect how the MMS is presented when transformed to HTML.

Parameters:
`height` - The new height with "px" suffix.

---

setWidth

```
public void setWidth(java.lang.String width)
```

Set the width of the MMS. Must have a "px" suffix.

Parameters:
`width` - The new width with "px" suffix.

---

mms.components.mms

Class MmsRegion

```
java.lang.Object
  └mms.components.mms.MmsRegion
```

```
public class MmsRegion
extends java.lang.Object
```

This class represents a region in an MMS SMIL.

Note that an `MmsRegion` is only applicable in MMS SMIL and not for the HTML version of the MMS.

Author:

Kristofer Borgstrom

Constructor Summary

`MmsRegion()`

Method Summary

java.lang.String	`getFit()` Get the fit attribute for this region.
java.lang.String	`getHeight()` Get the height of the region.
java.lang.String	`getId()` Get the ID of the region.
java.lang.String	`getLeft()` Get the offset to the left.
java.lang.String	`getTop()` Get the offset to the top.
java.lang.String	`getWidth()` Get the width of the region.
void	`setFit(java.lang.String fit)` Set the fit attribute.
void	`setHeight(java.lang.String height)` Set the height of the region.
void	`setId(java.lang.String id)` Set the ID of the region.

`void`	`setLeft(java.lang.String left)` Set the offset to the left.
`void`	`setTop(java.lang.String top)` Set the offset to the top.
`void`	`setWidth(java.lang.String width)` Set the width of the region.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

MmsRegion

`public MmsRegion()`

Method Detail

getLeft

`public java.lang.String getLeft()`

Get the offset to the left.

Returns:
The offset to the left.

---

getTop

`public java.lang.String getTop()`

Get the offset to the top.

Returns:
The offset to the top.

---

getHeight

`public java.lang.String getHeight()`

Get the height of the region.

Returns:
The height of the region.

---

getWidth

```
public java.lang.String getWidth()
```

Get the width of the region.

Returns:
The width of the region.

---

getFit

```
public java.lang.String getFit()
```

Get the fit attribute for this region.

Returns:
The fit attribute.

---

getId

```
public java.lang.String getId()
```

Get the ID of the region.

Must be one of:

- `"Image"`
- `"Text"`

Returns:
The ID of the region.

---

setLeft

```
public void setLeft(java.lang.String left)
```

Set the offset to the left.

Parameters:
`left` - The new offset to the left.

---

setTop

```
public void setTop(java.lang.String top)
```

Set the offset to the top.

Parameters:
`top` - The new offset to the top.

---

setHeight

```
public void setHeight(java.lang.String height)
```

Set the height of the region.

Parameters:
`height` - The new height.

---

setWidth

```
public void setWidth(java.lang.String width)
```

Set the width of the region.

Parameters:
`width` - The new width.

---

setFit

```
public void setFit(java.lang.String fit)
```

Set the fit attribute.

Parameters:
`fit` - The new fit attribute.

---

setId

```
public void setId(java.lang.String id)
```

Set the ID of the region.

Must be one of:

- `"Image"`
- `"Text"`

Parameters:
`id` - The new ID of the region.

Overview **Package** **Class** **Tree** **Deprecated** **Index** **Help**

PREV CLASS **NEXT CLASS** **FRAMES** **NO FRAMES** **All Classes**

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

mms.components.mms

Class MmsSlide

```
java.lang.Object
  └mms.components.mms.MmsSlide
```

```
public class MmsSlide
extends java.lang.Object
```

This class represents an MMS slide that can be added to an `MmsMessage`.

Version:
1.0

Author:
Kristofer Borgstrom

Constructor Summary

`MmsSlide``(MmsMessage mms)` Creates a new `MmsSlide` instance.
`MmsSlide``(java.lang.String duration, MmsMessage mms)` Creates a new `MmsSlide` instance with the specified duration.

Method Summary

`MmsAudio`	**`getAudio`**`()`
`java.lang.String`	**`getDuration`**`()`
`MmsMessage`	**`getMms`**`()` Get the `MmsMessage` of this slide.
`MmsText`	**`getText`**`()`
`VisualMmsMedia`	**`getVisualMedia`**`()`
`void`	**`setAudio`**`(MmsAudio audio)`
`void`	**`setDuration`**`(java.lang.String duration)`
`void`	**`setText`**`(MmsText text)`

`void`	`setVisualMedia(VisualMmsMedia visualMedia)`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

MmsSlide

`public MmsSlide(MmsMessage mms)`

Creates a new `MmsSlide` instance.

Parameters:
`mms` - The `MmsMessage` of this slide.

MmsSlide

```
public MmsSlide(java.lang.String duration,
                MmsMessage mms)
```

Creates a new `MmsSlide` instance with the specified duration.

Parameters:
`duration` -
`mms` - The `MmsMessage` of this slide.

Method Detail

getMms

`public MmsMessage getMms()`

Get the `MmsMessage` of this slide.

Returns:
the `MmsMessage`

---

getText

`public MmsText getText()`

Returns:
the text

---

getVisualMedia

```
public VisualMmsMedia getVisualMedia()
```

Returns:
the visualMedia

---

getAudio

```
public MmsAudio getAudio()
```

Returns:
the audio

---

getDuration

```
public java.lang.String getDuration()
```

Returns:
the duration

---

setText

```
public void setText(MmsText text)
```

Parameters:
`text` - the text to set

---

setVisualMedia

```
public void setVisualMedia(VisualMmsMedia visualMedia)
```

Parameters:
`visualMedia` - the visualMedia to set

---

setAudio

```
public void setAudio(MmsAudio audio)
```

Parameters:
`audio` - the audio to set

---

setDuration

```
public void setDuration(java.lang.String duration)
```

Parameters:
`duration` - the duration to set

---

mms.components.mms

Class MmsText

```
java.lang.Object
  └mms.components.mms.AbstractMmsContent
      └mms.components.mms.MmsText
```

```
public class MmsText
extends AbstractMmsContent
```

This class represents a text file that can be added to an `MmsSlide`.

Version:
1.0

Author:
Kristofer Borgstrom

Field Summary

`protected java.lang.String`	**`foregroundColor`** Defines an optional text color parameter for text.
`protected java.lang.String`	**`textSize`** Defines an optional text size parameter for text.

Fields inherited from class mms.components.mms.AbstractMmsContent
`alt, begin, contentType, dur, end, region, src`

Constructor Summary

`MmsText``()` Creates a new `MmsText` instance.

Method Summary

`java.lang.String`	**`getForegroundColor`**`()` Get text color.
`java.lang.String`	**`getTextSize`**`()` Gets the text size.
`void`	**`setForegroundColor`**`(java.lang.String foregroundColor)` Set text color.
`void`	**`setTextSize`**`(java.lang.String textSize)` Sets the text size.

Methods inherited from class mms.components.mms.AbstractMmsContent
`getAlt, getBegin, getContentType, getDur, getEnd, getRegion, getSrc, setAlt, setBegin, setContentType, setDur, setEnd, setSrc`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

textSize

`protected java.lang.String` **`textSize`**

Defines an optional text size parameter for text.

NOTE! Not part of the OMA MMS Conformance Document.

foregroundColor

`protected java.lang.String` **`foregroundColor`**

Defines an optional text color parameter for text.

NOTE! Not part of the OMA MMS Conformance Document.

Constructor Detail

MmsText

`public` **`MmsText`**`()`

Creates a new `MmsText` instance.

Method Detail

getTextSize

`public java.lang.String` **`getTextSize`**`()`

Gets the text size.

Returns:
text size

setTextSize

`public void` **`setTextSize`**`(java.lang.String textSize)`

Sets the text size.

Parameters:
`textSize` - text size

---

getForegroundColor

```
public java.lang.String getForegroundColor()
```

Get text color.

Returns:
text color

---

setForegroundColor

```
public void setForegroundColor(java.lang.String foregroundColor)
```

Set text color.

Parameters:
`foreGroundColor` - text color

---

mms.components.presentation.web

Class MmsTransformationServlet

```
java.lang.Object
  └javax.servlet.GenericServlet
      └javax.servlet.http.HttpServlet
          └mms.components.presentation.web.MmsTransformationServlet
```

All Implemented Interfaces:
java.io.Serializable, javax.servlet.Servlet, javax.servlet.ServletConfig

```
public class MmsTransformationServlet
extends javax.servlet.http.HttpServlet
```

This class handles incoming requests for MMS messages using HTTP GET.

An HTML version of the MMS will be returned if one can be generated. Otherwise the request will be dispatch ed to another page that is responsible for displaying a message to the client.

This servlet should be called in the same folder as the SMIL file that defines the MMS. To avoid setting up several servlet mappings it is suggested that this servlet is mapped to *.mms. An MMS could then be transformed with the following GET request:

`http://localhost:8080/mms/user/mms23/servlet.mms?smil=s.smil`

If a request does not contain the `players`
parameter which defines what media players are available and the MMS contains audio or video media, the req uest will be dispatched to the `loading-page`. The default `loading-page` is `/mms-loading.jsp` (must start with "/" and is relative to context root) but this can be customized by specifying a different page with the init parameter `loading-page`. The loading MMS page is responsible for detecting which media players are available and then automaticall y calling this servlet again with the `players` parameter specified (even if it is empty to indicate that no players were available).

If the MMS contains complex content, i.e. audio or video that requires embedding of media players, and the available players have been defined with the `players` parameter but no media player is capable of playing one or more elements of the MMS a check is performed. More specifically, the current `TransformationConfiguration` (discussed below) is checked for the `THROW_PLAYER_NOT_AVAILABLE_EXCEPTION` setting. If it is set to "true" the `MmsTransformer` will throw an exception and this servlet will handle the exception by dispatching to the `no-player-page` ("`/mms-no-player.jsp`" by default but can be set with the `no-player-page` init parameter). Before the dispatch is performed, and in intance of the `PlayerNotAvailableBean` will be added as a request attribute with the key `"playerInfo"`. This bean contains info about media type extension and which media players were available so that the `no-player-page` can display a valid message to the end user with information about how to download a media player. If on t he other hand the `THROW_PLAYER_NOT_AVAILABLE_EXCEPTION` setting is set to `"false"` any unsupported media will not be added to the HTML version of the SMIL.

If something else goes wrong with the transformation the request is dispatched to the `error-page`. The default `error-page` is "`/mms-error.jsp`" but this can be changed by specifying the `error-page` init parameter.

The `TransformationConfiguration` that will be used for the transformation will be read from a properties file if one is found (the properti es file should be located at: `WEB-INF/mms-transformation.properties`). This properties file takes precedence over the default values.

In addition to specifying custom `TransformationConfiguration`
settings in a properties file, some settings can and some must be set as parameters to this servlet. These parameters are listed and explained in the `handleRequest` method.

Author:
Kristofer Borgstrom
See Also:
Serialized Form

Field Summary

`static java.lang.String`	**`CUSTOM_DATA_SOURCE_FOLDER_INIT_PARAM`** This is static field contains the class name of the servlet init param that can be used to set a custom `DataSourceFolder` from which is used as an I/O base for reading and writing to files.
`static java.lang.String`	**`DETECTION_SCRIPT_SRC_INFO_KEY`** The key: "`detectScriptSrc`" is used to store a string with the source of the media player detection s script in the request scope if a the MMS contained complex media and the media player configuration has not been detected in this session.
`static java.lang.String`	**`ERROR_PAGE_INIT_PARAM`** This is static field contains the name of the servlet init param that can be used to set a custo m error page that will be shown when transformation fails.
`static java.lang.String`	**`LOADING_PAGE_INIT_PARAM`** This is static field contains the name of the servlet init param that can be used to check for a loading page that is shown while available media players are checked.
`static java.lang.String`	**`NO_PLAYER_FOUND_INFO_KEY`** The key: "`playerInfo`" is used to store a `PlayerNotAvailableBean` in the request scope if a `PlayerNotFoundException` was thrown.
`static java.lang.String`	**`NO_PLAYER_PAGE_INIT_PARAM`** This is static field contains the name of the servlet init param that can be used to set a custo m no-player-available page that will be shown when transformation fails with a PlayerNotAvailableException.

Constructor Summary

`MmsTransformationServlet``()` Creates a new `MmsTransformationServlet`.

Method Summary

`protected void`	**`dispatch`**`(java.lang.String page, javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse res)` Dispatch to the specified page.

protected void	**doGet**(javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse resp) See `handleRequest(HttpServletRequest, HttpServletResponse)`.
protected void	**doNoPlayerPage**(javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse res, PlayerNotAvailableException ex) Creates a new `PlayerNotAvailableBean` using data from the specified `PlayerNotAvailableException` and sets it as a request scope attribute using the key "`playerInfo`".
protected void	**doPost**(javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse resp) Not supported.
protected void	**doSendMms**(byte[] mmsData, javax.servlet.http.HttpServletResponse res) Writes the byte data to the specified `HttpServletResponse`'s OutputStream.
protected void	**handleRequest**(javax.servlet.http.HttpServletRequest req, javax.servlet.http.HttpServletResponse res) Handles both GET and POST Parameters: (See `TransformationConfiguration` for more info and default values for optional parameters).
void	**init**() Performs the following init actions: Loads the init paramaters (see class description above) Read s the transformation properties file at /WEB-INF/mms-transformation.properties Determines which `DataSourceFolder` to use based on whether the data-source-folder init parameter has been set.

Methods inherited from class javax.servlet.http.HttpServlet
doDelete, doHead, doOptions, doPut, doTrace, getLastModified, service, service

Methods inherited from class javax.servlet.GenericServlet
destroy, getInitParameter, getInitParameterNames, getServletConfig, getServletContext, getServletInfo, getServletName, init, log, log

Methods inherited from class java.lang.Object
clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Field Detail

CUSTOM_DATA_SOURCE_FOLDER_INIT_PARAM

`public static final java.lang.String` **`CUSTOM_DATA_SOURCE_FOLDER_INIT_PARAM`**

This is static field contains the class name of the servlet init param that can be used to set a custom `DataSourceFolder` from which is used as an I/O base for reading and writing to files. If not set, the files will be read from disk on the web server using the `FileDataSourceFolder` class.

Init param name: "`data-source-folder`"

Use: **OPTIONAL**

See Also:
Constant Field Values

LOADING_PAGE_INIT_PARAM

`public static final java.lang.String` **`LOADING_PAGE_INIT_PARAM`**

This is static field contains the name of the servlet init param that can be used to check for a loading pa ge that is shown while available media players are checked.

Init param name: "`loading-page`"

Use: **OPTIONAL**

NOTE! The path must begin with a "/" and is interpreted as relative to the current context root.

If not set the following page will be used: `/mms-loading.jsp`

See Also:
Constant Field Values

ERROR_PAGE_INIT_PARAM

`public static final java.lang.String` **`ERROR_PAGE_INIT_PARAM`**

This is static field contains the name of the servlet init param that can be used to set a custom error pag e that will be shown when transformation fails.

Init param name: "`error-page`"

Use: **OPTIONAL**

NOTE! The path must begin with a "/" and is interpreted as relative to the current context root.

If not set the following page will be used: `/mms-error.jsp`

See Also:
Constant Field Values

NO_PLAYER_PAGE_INIT_PARAM

`public static final java.lang.String` **`NO_PLAYER_PAGE_INIT_PARAM`**

This is static field contains the name of the servlet init param that can be used to set a custom no-player-available page that will be shown when transformation fails with a PlayerNotAvailableException.

Init param name: "`no-player-page`"

Use: **OPTIONAL**

NOTE! The path must begin with a "/" and is interpreted as relative to the current context root.

If not set the following page will be used: `/mms-no-player.jsp`

See Also:
Constant Field Values

NO_PLAYER_FOUND_INFO_KEY

```
public static final java.lang.String NO_PLAYER_FOUND_INFO_KEY
```

The key: "`playerInfo`" is used to store a `PlayerNotAvailableBean` in the request scope if a `PlayerNotFoundException` was thrown. Thus, this bean will be available on the defined `no-player-page` (default: `/mms-no-player.jsp`).

See Also:
Constant Field Values

DETECTION_SCRIPT_SRC_INFO_KEY

```
public static final java.lang.String DETECTION_SCRIPT_SRC_INFO_KEY
```

The key: "`detectScriptSrc`" is used to store a string with the source of the media player detection s script in the request scope if a the MMS contained complex media and the media player configuration has not been detected in this session. This script is used on the `no-player-page` (default: `/mms-no-player.jsp`).

See Also:
Constant Field Values

Constructor Detail

MmsTransformationServlet

```
public MmsTransformationServlet()
```

Creates a new `MmsTransformationServlet`.

Method Detail

init

```
public void init()
          throws javax.servlet.ServletException
```

Performs the following init actions:

- Loads the init paramaters (see class description above)
- Reads the transformation properties file at /WEB-INF/mms-transformation.properties
- Determines which `DataSourceFolder` to use based on whether the data-source-folder init parameter has been set.
- Creates a new `TransformerPool`.

Overrides:
`init` in class `javax.servlet.GenericServlet`

Throws:
`javax.servlet.ServletException`

doGet

```
protected void doGet(javax.servlet.http.HttpServletRequest req,
                     javax.servlet.http.HttpServletResponse resp)
              throws javax.servlet.ServletException,
                     java.io.IOException
```

See `handleRequest(HttpServletRequest, HttpServletResponse)`.

Overrides:
`doGet` in class `javax.servlet.http.HttpServlet`

Throws:
`javax.servlet.ServletException`
`java.io.IOException`

doPost

```
protected void doPost(javax.servlet.http.HttpServletRequest req,
                      javax.servlet.http.HttpServletResponse resp)
               throws javax.servlet.ServletException,
                      java.io.IOException
```

Not supported.

Overrides:
`doPost` in class `javax.servlet.http.HttpServlet`

Throws:
`javax.servlet.ServletException` - Always.
`java.io.IOException`

handleRequest

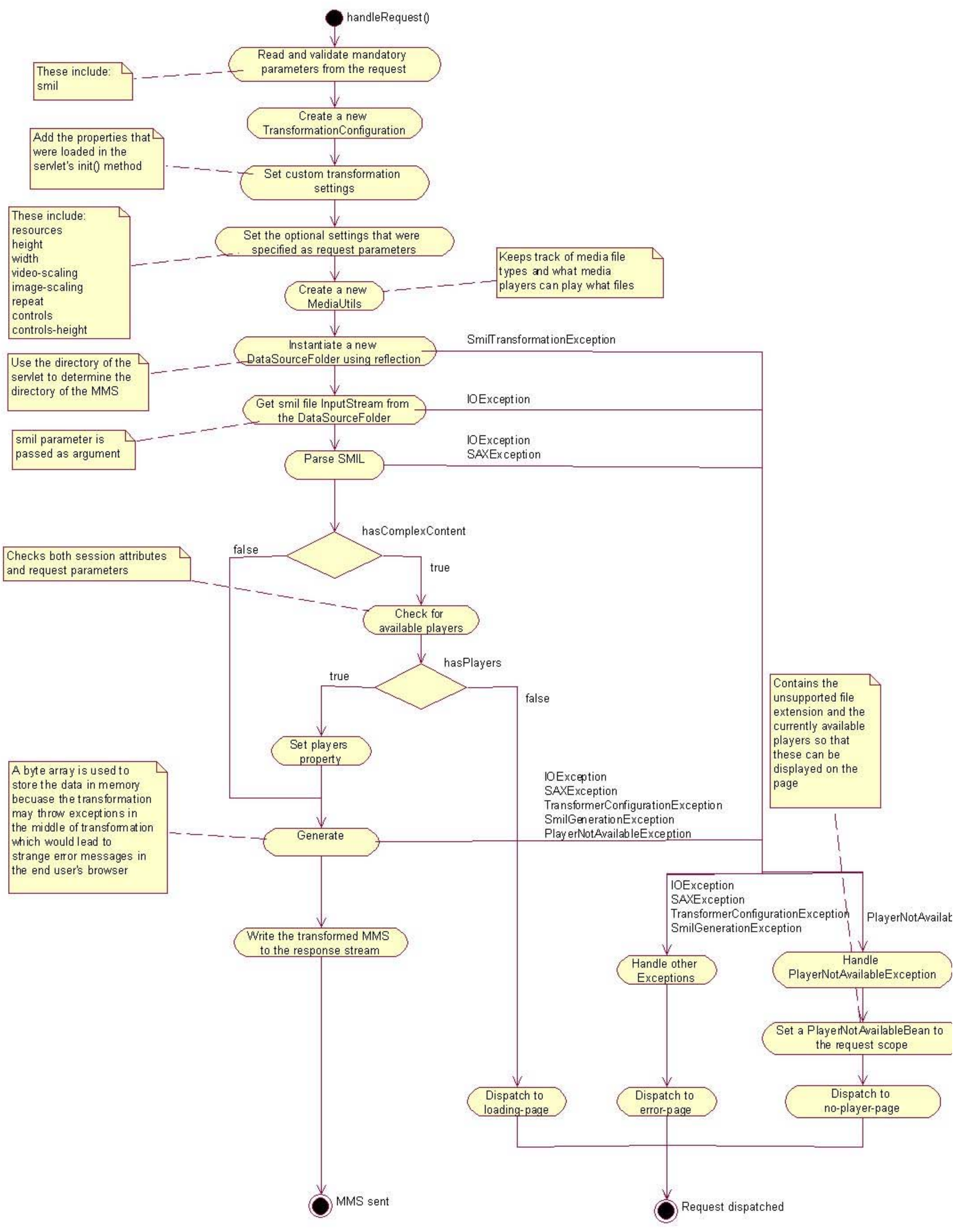
```
protected void handleRequest(javax.servlet.http.HttpServletRequest req,
                             javax.servlet.http.HttpServletResponse res)
                      throws javax.servlet.ServletException,
                             java.io.IOException
```

Handles both GET and POST

Parameters: (See `TransformationConfiguration` for more info and default values for optional parameters).

Parameter	**Mandatory**	**Description**
`smil`	YES	The name of the smil file in the current directory.
`mms-dir`	YES	The directory of the MMS relative to context root. Must start with a forward slash ('/')
`players`	CONDITIONALLY	Mandatory if MMS contains complex content (audio or video). A string defining which media players are avail able. If not specified, MMS may not be generated depending on MMS content and current settings. Corresponds to the `AVAILABLE_PLAYERS` setting.
`resources`	NO	A string defining where MMS resources are located relative to current dir or to server root if it starts wi th a "/". Corresponds to the `RESOURCE_DIR` setting.
`height`	NO	A string defining the height of the MMS display area, must have a "`px`" suffix. Corresponds to the `OUTPUT_HEIGHT` setting.
`width`	NO	A string defining the width of the MMS display area, must have a "`px`" suffix. Corresponds to the `OUTPUT_WIDTH` setting.
`video-scaling`	NO	A string defining the video scaling setting to use. Corresponds to the `VIDEO_SCALING` setting.
`image-scaling`	NO	A string defining the image scaling setting to use. Corresponds to the `IMAGE_SCALING` setting.
`repeat`	NO	A string defining the number of times to iterate the MMS or "indefinite" to play forever. Corresponds to th e `REPEAT_COUNT` setting.
`controls`	NO	A boolean string defining whether to add playback controls to MMS. Corresponds to the `SHOW_PLAYBACK_CONTROLS` setting.
`controls-height`	NO	A string defining the height of the playback controls, must have a "px" suffix. Corresponds to the `PLAYBACK_CONTROLS_HEIGHT` setting.

The following activity diagram shows the steps performed by the `handleRequest` method


handleRequest()
Read and validate mandatory parameters from the request
These include: smil
Create a new TransformationConfiguration
Add the properties that were loaded in the servlet's init() method
Set custom transformation settings
These include: resources height width video-scaling image-scaling repeat controls controls-height
Set the optional settings that were specified as request parameters
Keeps track of media file types and what media players can play what files
Create a new MediaUtils
Instantiate a new DataSourceFolder using reflection
SmilTransformationException
Use the directory of the servlet to determine the directory of the MMS
Get smil file InputStream from the DataSourceFolder
IOException
smil parameter is passed as argument
Parse SMIL
IOException SAXException
hasComplexContent
false
true
Checks both session attributes and request parameters
Check for available players
hasPlayers
true
false
Set players property
Contains the unsupported file extension and the currently available players so that these can be displayed on the page
A byte array is used to store the data in memory because the transformation may throw exceptions in the middle of transformation which would lead to strange error messages in the end user's browser
Generate
IOException SAXException TransformerConfigurationException SmilGenerationException PlayerNotAvailableException
IOException SAXException TransformerConfigurationException SmilGenerationException
PlayerNotAvailab
Write the transformed MMS to the response stream
Handle other Exceptions
Handle PlayerNotAvailableException
Set a PlayerNotAvailableBean to the request scope
Dispatch to loading-page
Dispatch to error-page
Dispatch to no-player-page
MMS sent
Request dispatched

Parameters:

`req` - An `HttpServletRequest` object that contains the request the client has made of the servlet.

`res` - An `HttpServletResponse` object that contains the response the servlet sends to the client.

Throws:

`javax.servlet.ServletException`

`java.io.IOException`

doNoPlayerPage

```
protected void doNoPlayerPage(javax.servlet.http.HttpServletRequest req,
                              javax.servlet.http.HttpServletResponse res,
                              PlayerNotAvailableException ex)
                       throws javax.servlet.ServletException,
                              java.io.IOException
```

Creates a new `PlayerNotAvailableBean` using data from the specified `PlayerNotAvailableException` and sets it as a request scope attribute using the key "`playerInfo`". The request is then dispatched to the `no-player-page`.

Parameters:
`req` - The `HttpServletRequest`
`res` - The `HttpServletResponse`
`ex` - The `PlayerNotAvailableException` to use to create the `PlayerNotAvailableBean`.

Throws:
`javax.servlet.ServletException`
`java.io.IOException`

---

dispatch

```
protected void dispatch(java.lang.String page,
                        javax.servlet.http.HttpServletRequest req,
                        javax.servlet.http.HttpServletResponse res)
                 throws javax.servlet.ServletException,
                        java.io.IOException
```

Dispatch to the specified page.

Parameters:
`page` - The page to which to dispatch.
`req` - The `HttpServletRequest`.
`res` - The `HttpServletResponse`.

Throws:
`javax.servlet.ServletException`
`java.io.IOException`

---

doSendMms

```
protected void doSendMms(byte[] mmsData,
                         javax.servlet.http.HttpServletResponse res)
                  throws java.io.IOException
```

Writes the byte data to the specified `HttpServletResponse`'s OutputStream.

Parameters:
`mmsData` - The data.
`res` - The `HttpServletResponse`.

Throws:
`java.io.IOException`

---

mms.components.transformation

Class MmsTransformer

```
java.lang.Object
  └mms.components.transformation.MmsTransformer
```

```
public class MmsTransformer
extends java.lang.Object
```

This class provides methods to perform transformation of MMS messages between different markup languages such as MMS SMIL and HTML+TIME and also to and from a Java object representation (`MmsMessage`).

Version:
1.0

Author:
Kristofer Borgstrom

Method Summary

`static MmsTransformer`	`newInstance()` Create a new `MmsTransformer` instance.
`MmsMessage`	`parseMmsSmil(java.io.InputStream smilStream)` This method parses an MMS SMIL file and creates an `MmsMessage` representation of the MMS.
`void`	`transform(java.io.InputStream smilStream, DataSourceFolder mmsFolder, java.io.OutputStream output, TransformationConfiguration config)` This method parses an MMS SMIL file and then transforms it to a HTML view of the MMS and writes it to the specified `OutputStream`.
`void`	`transform(MmsMessage mms, java.io.OutputStream output, TransformationConfiguration config)` This method transforms the `MmsMessage` to a HTML version of the MMS and writes it to the specified `OutputStream`.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Method Detail

newInstance

```
public static MmsTransformer newInstance()
                                  throws javax.xml.parsers.ParserConfigurationException,
                                         org.xml.sax.SAXException
```

Create a new `MmsTransformer` instance. This operation is expensive and `MmsTransformer` instances should be reused. It is recommended to use the `TransformerPool` class to handler instances.

Returns:

A new `MmsTransformer` instance.

Throws:

`javax.xml.parsers.ParserConfigurationException` - If the `SAXParserFactory` has not been properly configured.

`org.xml.sax.SAXException` - For SAX errors.

---

parseMmsSmil

```
public MmsMessage parseMmsSmil(java.io.InputStream smilStream)
                        throws java.io.IOException,
                               org.xml.sax.SAXException
```

This method parses an MMS SMIL file and creates an `MmsMessage` representation of the MMS. The `DataSourceFolder`
is required to extract data from the different media files that are referenced by the SMIL (mainly used to read text files and get dimension of images).

Parameters:

`smilStream` - An `InputStream` containing the MMS SMIL to be parsed.

Returns:

A new `MmsMessage`

Throws:

`java.io.IOException` - If a read error occurs while reading the input stream.

`org.xml.sax.SAXException` - For SAX errors.

---

transform

```
public void transform(java.io.InputStream smilStream,
                      DataSourceFolder mmsFolder,
                      java.io.OutputStream output,
                      TransformationConfiguration config)
               throws org.xml.sax.SAXException,
                      java.io.IOException,
                      MmsTransformationException,
                      PlayerNotAvailableException,
                      java.lang.IllegalStateException
```

This method parses an MMS SMIL file and then transforms it to a HTML view of the MMS and writes it to the specified `OutputStream`. The settings from the specified `TransformationConfiguration` is used. The `MediaUtils`
is used to determine which media players can be used to play which files based on an XML configuration file.

Parameters:

`smilStream` - An input stream containing the MMS SMIL to be parsed.

`mmsFolder` - A `DataSourceFolder`
representation of the folder where the content of the MMS is located.

`output` - The `OutputStream` to write the result to.

`config` - The `TransformationConfiguration` to use.

`mediaUtil` - The `MediaUtils` instance to use.

Throws:

`org.xml.sax.SAXException` - For SAX errors.

`java.io.IOException` - For I/O errors.

`MmsTransformationException` - If the transformation could not be performed.

`PlayerNotAvailableException`
- If no media player was available to play a certain media file and the `THROW_PLAYER_NOT_AVAILABLE_EXCEPTION` setting has been set to "`true`".
`java.lang.IllegalStateException` - If the `MediaUtils` class has not been initialized.

---

transform

```
public void transform(MmsMessage mms,
                      java.io.OutputStream output,
                      TransformationConfiguration config)
               throws org.xml.sax.SAXException,
                      java.io.IOException,
                      MmsTransformationException,
                      PlayerNotAvailableException,
                      java.lang.IllegalStateException
```

This method transforms the `MmsMessage` to a HTML version of the MMS and writes it to the specified `OutputStream`. The settings from the specified `TransformationConfiguration` is used. The `MediaUtils`
is used to determine which media players can be used to play which files based on an XML configuration file.

Note that this method requires that a `DataSourceFolder` has been set on the `MmsMessage`.

Parameters:
`mms` - The MMS to transform, must have a valid `DataSourceFolder` set.
`output` - The `OutputStream` to write the result to.
`config` - The `TransformationConfiguration` to use.

Throws:
`org.xml.sax.SAXException` - For SAX errors.
`java.io.IOException` - For I/O errors.
`MmsTransformationException` - If the transformation could not be performed.
`PlayerNotAvailableException`
- If no media player was available to play a certain media file and the `THROW_PLAYER_NOT_AVAILABLE_EXCEPTION` setting has been set to "`true`".
`java.lang.IllegalStateException` - If the `MediaUtils` class has not been initialized.

---

mms.components.mms

Class MmsUtils

```
java.lang.Object
  └mms.components.mms.MmsUtils
```

```
public class MmsUtils
extends java.lang.Object
```

This class is a utility class for MMS messages.

Version:
1.0

Author:
Kristofer Borgstrom

Constructor Summary

`MmsUtils()`

Method Summary

static `MmsMessage`	`createMmsFromJson(JsonObject jsonMms, DataSourceFolder dataFolder)` This method is used to create a new `MmsMessage` from a `JsonObject`.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

MmsUtils

```
public MmsUtils()
```

Method Detail

createMmsFromJson

```
public static MmsMessage createMmsFromJson(JsonObject jsonMms,
                                           DataSourceFolder dataFolder)
                                    throws java.io.IOException
```

This method is used to create a new `MmsMessage` from a `JsonObject`.

Parameters:

`jsonMms` - The `JsonObject` representation of the MMS
`dataFolder` - The `DataSourceFolder` that text will be saved to.

Returns:

The `MmsMessage`

Throws:

`java.io.IOException` - If an error occurred while writing text to the `DataSourceFolder`

mms.components.mms

Class MmsVideo

```
java.lang.Object
  └mms.components.mms.AbstractMmsContent
      └mms.components.mms.VisualMmsMedia
          └mms.components.mms.MmsVideo
```

```
public class MmsVideo
extends VisualMmsMedia
```

This class represents a video file that can be added to an `MmsSlide`.

Version:
1.0

Author:
Kristofer Borgstrom

Field Summary

Fields inherited from class mms.components.mms.AbstractMmsContent
`alt, begin, contentType, dur, end, region, src`

Constructor Summary

`MmsVideo``()` Creates a new `MmsVideo` instance.

Method Summary

`java.lang.String`	**`getId`**`()` Get the ID.
`void`	**`setId`**`(java.lang.String id)` Set the ID.

Methods inherited from class mms.components.mms.AbstractMmsContent
`getAlt, getBegin, getContentType, getDur, getEnd, getRegion, getSrc, setAlt, setBegin, setContentType, setDur, setEnd, setSrc`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

MmsVideo

```
public MmsVideo()
```

Creates a new `MmsVideo` instance.

Method Detail

getId

```
public java.lang.String getId()
```

Get the ID.

Note that this ID is only applicable when HTML is generated from the MMS. and that when HTML is being generated this the ID must exist and be unique within the current MMS.

Returns:
The ID.

---

setId

```
public void setId(java.lang.String id)
```

Set the ID.

Note that this ID is only applicable when HTML is generated from the MMS. and that when HTML is being generated this the ID must exist and be unique within the current MMS.

Parameters:
`id` - The new ID.

---

mms.components.transformation

Class TransformationConfiguration

```
java.lang.Object
  └java.util.Dictionary<K,V>
      └java.util.Hashtable<java.lang.Object,java.lang.Object>
          └java.util.Properties
              └mms.components.transformation.TransformationConfiguration
```

All Implemented Interfaces:

java.io.Serializable, java.lang.Cloneable, java.util.Map<java.lang.Object,java.lang.Object>

```
public class TransformationConfiguration
extends java.util.Properties
```

This class defines a `TransformationConfiguration` which must be supplied every time an MMS is transformed. This class extends `java.util.Properties`
and has static fields specifying which settings can be used during transformation as well as what their default values are.

Version:
1.0

Author:
Kristofer Borgstrom

See Also:
Serialized Form

Field Summary

`static java.lang.String`	**`ACTION_SERVLET`** Standard property: URL to action servlet.
`static java.lang.String`	**`AVAILABLE_PLAYERS`** Standard property: Comma separated values containing available media players.
`static java.lang.String`	**`BROWSER`** Standard property: Browser name.
`static java.lang.String`	**`IMAGE_SCALING`** Standard property: Image scaling.
`static java.lang.String`	**`MEDIA_PLAYER_PRIORTY_ORDER`** Standard property: A comma separated list of allowed media players and their order of priority.
`static java.lang.String`	**`MMS_DIRECTORY`** Standard property: Path to MMS directory.
`static java.lang.String`	**`OUTPUT_HEIGHT`** Standard property: The height property in pixels (must have the "px" suffix).
`static java.lang.String`	**`OUTPUT_WIDTH`** Standard property: The width property in pixels (must have the "px" suffix).

`static java.lang.String`	**`PLAYBACK_CONTROLS_HEIGHT`** Standard property: The height in pixels reserved for playback controls (must have the "px" suffix).
`static java.lang.String`	**`PLAYER_BACKGROUND_COLOR`** Standard property: The background color for the playback area when splash screen is shown.
`static java.lang.String`	**`REPEAT_COUNT`** Standard property: The exact string use as `repeatCount` property.
`static java.lang.String`	**`REQUIRE_AUTO_CODEC_DOWNLOAD_MEDIA_SUPPORT`** Standard property: Require the player to media support through an automatic codec download.
`static java.lang.String`	**`REQUIRE_DIRECT_MEDIA_SUPPORT`** Standard property: Require direct support for a media types by to use a player.
`static java.lang.String`	**`RESOURCE_DIR`** Standard property: Resource directory, relative to the context root, that contains scripts, stylesheets, and images that are used for playback control.
`static java.lang.String`	**`SHOW_PLAYBACK_CONTROLS`** Standard property: Show playback controls for the MMS presentation.
`static java.lang.String`	**`THROW_PLAYER_NOT_AVAILABLE_EXCEPTION`** Standard property: Defines what the generator should do when an MMS contains media elements for which there is no media player available of the required level of support.
`static java.lang.String`	**`VIDEO_SCALING`** Standard property: Video scaling.
`static java.lang.String`	**`VIEW_PRIORITY_ORDER`** Standard property: The order of priority for HTML views.

Fields inherited from class java.util.Properties
`defaults`

Constructor Summary

`TransformationConfiguration``()`
Create a TransformationConfiguration with default settings.

Method Summary

Methods inherited from class java.util.Properties
`getProperty, getProperty, list, list, load, load, loadFromXML, propertyNames, save, setProperty, store, store, storeToXML, storeToXML, stringPropertyNames`

Methods inherited from class java.util.Hashtable
`clear, clone, contains, containsKey, containsValue, elements, entrySet, equals, get, hashCode, isEmpty, keys, keySet, put, putAll, rehash, remove, size, toString, values`

Methods inherited from class java.lang.Object
`finalize, getClass, notify, notifyAll, wait, wait, wait`

Field Detail

ACTION_SERVLET

```
public static final java.lang.String ACTION_SERVLET
```

Standard property: URL to action servlet. The action servlet is responsible for serving the viewer components with content in the form of generated JavaScript snippets and .ram files.

The URL is embedded into the MMS and it should therefore be relative to the Servlet that was used to generate the MMS view.

Typically, the `MmsTransformationServlet`
is used as action servlet but any other servlet could be used as well.

Example value: "MmsTransformationServlet" (DEFAULT)

See Also:
Constant Field Values

AVAILABLE_PLAYERS

```
public static final java.lang.String AVAILABLE_PLAYERS
```

Standard property: Comma separated values containing available media players. This should match the players available in the user's browser.

This property must be set because it has no default value.

Possible values include:

- "**QuickTime**"
- "**RealPlayer**"
- "**WindowsMediaPlayer**"

See Also:
`MEDIA_PLAYER_PRIORTY_ORDER`, Constant Field Values

BROWSER

```
public static final java.lang.String BROWSER
```

Standard property: Browser name. Possible values include but are not limited to:

- "**InternetExplorer**"
- "**Firefox**"
- "**Safari**"
- "**Opera**"

See Also:
Constant Field Values

IMAGE_SCALING

```
public static final java.lang.String IMAGE_SCALING
```

Standard property: Image scaling. Possible values:

- "**fit**" - Stretch to presentation area but retain aspect ratio. (DEFAULT)

See Also:
Constant Field Values

MEDIA_PLAYER_PRIORTY_ORDER

`public static final java.lang.String` **`MEDIA_PLAYER_PRIORTY_ORDER`**

Standard property: A comma separated list of allowed media players and their order of priority. Possible values include:

- "**InternetExplorer**"
- "**QuickTime**"
- "**RealPlayer**"
- "**WindowsMediaPlayer**"

DEFAULT: "`InternetExplorer,QuickTime,RealPlayer,WindowsMediaPlayer`"

See Also:
Constant Field Values

MMS_DIRECTORY

`public static final java.lang.String` **`MMS_DIRECTORY`**

Standard property: Path to MMS directory.

This property is mandatory and has no default value.

See Also:
Constant Field Values

OUTPUT_HEIGHT

`public static final java.lang.String` **`OUTPUT_HEIGHT`**

Standard property: The height property in pixels (must have the "px" suffix).

For example: "`320px`" (DEFAULT)

See Also:
`OUTPUT_WIDTH`, Constant Field Values

OUTPUT_WIDTH

`public static final java.lang.String` **`OUTPUT_WIDTH`**

Standard property: The width property in pixels (must have the "px" suffix).

For example: "`240px`" (DEFAULT)

See Also:

`OUTPUT_HEIGHT`, Constant Field Values

PLAYBACK_CONTROLS_HEIGHT

`public static final java.lang.String` **`PLAYBACK_CONTROLS_HEIGHT`**

Standard property: The height in pixels reserved for playback controls (must have the "px" suffix).

For example: "`30px`" (DEFAULT)

See Also:
Constant Field Values

PLAYER_BACKGROUND_COLOR

`public static final java.lang.String` **`PLAYER_BACKGROUND_COLOR`**

Standard property: The background color for the playback area when splash screen is shown.

For example: "`black`" (DEFAULT)

See Also:
Constant Field Values

REPEAT_COUNT

`public static final java.lang.String` **`REPEAT_COUNT`**

Standard property: The exact string use as `repeatCount` property.

For example: "`indefinite`" (DEFAULT)
For example: "`1`"

See Also:
Constant Field Values

REQUIRE_DIRECT_MEDIA_SUPPORT

`public static final java.lang.String` **`REQUIRE_DIRECT_MEDIA_SUPPORT`**

Standard property: Require direct support for a media types by to use a player. If such a player does not exist, either a `PlayerNotAvailableException` will be thrown or the media will be left out (depending on the `THROW_PLAYER_NOT_AVAILABLE_EXCEPTION` setting).

Possible values: "`true`"|"`false`" (DEFAULT: "`false`")

See Also:
`REQUIRE_AUTO_CODEC_DOWNLOAD_MEDIA_SUPPORT`, `THROW_PLAYER_NOT_AVAILABLE_EXCEPTION`, Constant Field Values

REQUIRE_AUTO_CODEC_DOWNLOAD_MEDIA_SUPPORT

`public static final java.lang.String` **`REQUIRE_AUTO_CODEC_DOWNLOAD_MEDIA_SUPPORT`**

Standard property: Require the player to media support through an automatic codec download. If such a player does not exist, either a `PlayerNotAvailableException` will be thrown or the media will be left out (depending on the `THROW_PLAYER_NOT_AVAILABLE_EXCEPTION` setting).

Possible values: "`true`"|"`false`" (DEFAULT: "`true`")

See Also:
`REQUIRE_DIRECT_MEDIA_SUPPORT`, `THROW_PLAYER_NOT_AVAILABLE_EXCEPTION`, Constant Field Values

RESOURCE_DIR

```
public static final java.lang.String RESOURCE_DIR
```

Standard property: Resource directory, relative to the context root, that contains scripts, stylesheets, and images that are used for playback control. Must start with a slash.

The following resources are required within the resource directory:

- `css/html-time.css`
- `scripts/html-time.js`
- `scripts/show-mms.js`
- `images/play.jpg`
- `images/play-over.jpg`
- `images/play-click.jpg`
- `images/space.jpg`
- `images/stop.jpg`
- `images/stop-over.jpg`
- `images/stop-click.jpg`

Example value: "/mms-resources/" - (DEFAULT)

See Also:
Constant Field Values

SHOW_PLAYBACK_CONTROLS

```
public static final java.lang.String SHOW_PLAYBACK_CONTROLS
```

Standard property: Show playback controls for the MMS presentation.

Possible values: "`true`"|"`false`" (DEFAULT: "`true`")

See Also:
Constant Field Values

THROW_PLAYER_NOT_AVAILABLE_EXCEPTION

```
public static final java.lang.String THROW_PLAYER_NOT_AVAILABLE_EXCEPTION
```

Standard property: Defines what the generator should do when an MMS contains media elements for which there is no media player available of the required level of support.

- "**`true`**" - Throw `PlayerNotAvailableException` if MMS contains media elements that are not supported by the current media player configuration.
- "**`false`**" - Perform generation anyway but leave out unsupported content. (DEFAULT)

See Also:
`REQUIRE_DIRECT_MEDIA_SUPPORT`, `REQUIRE_AUTO_CODEC_DOWNLOAD_MEDIA_SUPPORT`, Constant Field Values

VIDEO_SCALING

```
public static final java.lang.String VIDEO_SCALING
```

Standard property: Video scaling. Note that this setting will not effect the amount of space the embedded media player will occupy on the page (It will always occupy half of the height of the presentation area or the whole area if there is no text on the current slide). Rather, this setting defines the scaling within the area of the embedded media player.

Possible values:

- "**original**" - Use original video resolution.
- "**stretch**" - Stretch to media player area.
- "**stretch-keep-aspect**" - Stretch to media player area but retain aspect ratio. (DEFAULT)

See Also:
Constant Field Values

VIEW_PRIORITY_ORDER

```
public static final java.lang.String VIEW_PRIORITY_ORDER
```

Standard property: The order of priority for HTML views. Possible values (one or several, separated by commas):

- "**HtmlTime**"
- "**Dhtml**"

Example value: "Dhtml,HtmlTime" - (DEFAULT)

See Also:
Constant Field Values

Constructor Detail

TransformationConfiguration

```
public TransformationConfiguration()
```

Create a TransformationConfiguration with default settings.

mms.components.transformation

Class TransformerPool

```
java.lang.Object
  └mms.components.transformation.TransformerPool
```

```
public class TransformerPool
extends java.lang.Object
```

This pool implementation is allowed to vary in size between its set `MAX_POOL_SIZE` and `MIN_POOL_SIZE`.

Transformers that have not been used within the `REMOVE_UNUSED_DELAY` will be removed from the queue if the pool size is larger than the minimum pool size.

Transformers that have not been returned within the `RECLAIM_DELAY` will automatically be removed from the pool. This is done regardless of minimum pool size.

Version:
1.0
Author:
Kristofer Borgstrom

Field Summary

`static java.lang.String`	**`CLEANUP_CHECK_INTERVAL`** This is the properties key that is used to define the pool cleanup interval in milliseconds.
`static java.lang.String`	**`MAX_POOL_SIZE`** This is the properties key that is used to define maximum pool size.
`static java.lang.String`	**`MIN_POOL_SIZE`** This is the properties key that is used to define minimum pool size.
`static java.lang.String`	**`RECLAIM_DELAY`** This is the properties key that is used to define the RECLAIM_DELAY in milliseconds.
`static java.lang.String`	**`REMOVE_UNUSED_DELAY`** This is the properties key that is used to define the REMOVE_UNUSED_DELAY in milliseconds.

Constructor Summary

`TransformerPool``(int initialPoolSize, java.util.Properties props)` Creates a new `TransformerPool` instance with the specified initial size and properties.

Method Summary

`MmsTransformer`	`getTransformer()` Gets an `MmsTransformer` from the pool.
`void`	`returnTransformer(MmsTransformer transformer)` Return an `MmsTransformer` to the pool.

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Field Detail

MAX_POOL_SIZE

```
public static final java.lang.String MAX_POOL_SIZE
```

This is the properties key that is used to define maximum pool size.

Default: `100`

Example use:

```
properties.put(TransformerPool.MAX_POOL_SIZE, "100");
```

See Also:
Constant Field Values

MIN_POOL_SIZE

```
public static final java.lang.String MIN_POOL_SIZE
```

This is the properties key that is used to define minimum pool size.

Default: `"1"`

Example use:

```
properties.put(TransformerPool.MIN_POOL_SIZE, "1");
```

See Also:
Constant Field Values

CLEANUP_CHECK_INTERVAL

```
public static final java.lang.String CLEANUP_CHECK_INTERVAL
```

This is the properties key that is used to define the pool cleanup interval in milliseconds. This is the interval that defines how often to check for unused and unreturned transformers.

Default: `"2000"` (2s)

Example use:

```
properties.put(TransformerPool.CLEANUP_CHECK_INTERVAL, "2000");
```

See Also:
Constant Field Values

REMOVE_UNUSED_DELAY

```
public static final java.lang.String REMOVE_UNUSED_DELAY
```

This is the properties key that is used to define the REMOVE_UNUSED_DELAY in milliseconds. This delay is the maximum amount of time that a transformer is kept in pool if it is unused.

Default: `"10000"` (10s)

Example use:

```
properties.put(TransformerPool.REMOVE_UNUSED_DELAY, "10000");
```

See Also:
Constant Field Values

RECLAIM_DELAY

```
public static final java.lang.String RECLAIM_DELAY
```

This is the properties key that is used to define the RECLAIM_DELAY in milliseconds. This delay is the maximum amount of time that a transformer can be used before it is returned, after this delay has passed the transformer is removed from the pool.

Default: `"120000"` (2min)

Example use:

```
properties.put(TransformerPool.RECLAIM_DELAY, "120000");
```

See Also:
Constant Field Values

Constructor Detail

TransformerPool

```
public TransformerPool(int initialPoolSize,
                       java.util.Properties props)
                throws javax.xml.parsers.ParserConfigurationException,
                       org.xml.sax.SAXException
```

Creates a new `TransformerPool` instance with the specified initial size and properties.

Throws:
`org.xml.sax.SAXException`
`javax.xml.parsers.ParserConfigurationException`

Method Detail

getTransformer

```
public MmsTransformer getTransformer()
                              throws javax.xml.parsers.ParserConfigurationException,
                                     org.xml.sax.SAXException
```

Gets an `MmsTransformer` from the pool.

Returns:
An available `MmsTransformer`.

Throws:
`javax.xml.parsers.ParserConfigurationException`
`org.xml.sax.SAXException`

---

returnTransformer

```
public void returnTransformer(MmsTransformer transformer)
```

Return an `MmsTransformer` to the pool.

Parameters:
`transformer` - The `MmsTransformer` to return to the pool.

---

mms.components.mms

Class VisualMmsMedia

```
java.lang.Object
  └mms.components.mms.AbstractMmsContent
      └mms.components.mms.VisualMmsMedia
```

Direct Known Subclasses:

MmsImage, MmsVideo

```
public class VisualMmsMedia
extends AbstractMmsContent
```

This interface defines a visual media object.

Author:

Kristofer Borgstrom

Field Summary

Fields inherited from class mms.components.mms.AbstractMmsContent
`alt, begin, contentType, dur, end, region, src`

Constructor Summary

`VisualMmsMedia()`

Method Summary

Methods inherited from class mms.components.mms.AbstractMmsContent
`getAlt, getBegin, getContentType, getDur, getEnd, getRegion, getSrc, setAlt, setBegin, setContentType, setDur, setEnd, setSrc`

Methods inherited from class java.lang.Object
`clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

Constructor Detail

VisualMmsMedia

```
public VisualMmsMedia()
```

[54] *J2EE Web Application Template* – Ericsson Developer Program, http://www.ericsson.com/mobilityworld/sub/open/technologies/open_development_tips/tools/j2ee_web_app (last viewed 2007-09-04)

[55] W3Schools – *Browser statistics*, http://www.w3schools.com/browsers/browsers_stats.asp (last viewed 2008-01-16)

[56] Microsoft – *ASP.NET Developer Center*, http://msdn2.microsoft.com/en-us/netframework/aa336522.aspx (last viewed 2008-01-31)

[57] PHP: Hypertext Preprocessor, http://www.php.net/ (last viewed 2008-01-31)

COS/CCS 2008-01