University of Wollongong
Research Online

Faculty of Informatics - Papers (Archive)

Faculty of Engineering and Information Sciences

2006

Enhancing Interoperability via Generic Multimedia Syntax Translation

Joseph Thomas-Kerr
University of Wollongong, jak09@uow.edu.au

I. Burnett
University of Wollongong, ianb@uow.edu.au

C. H. Ritz
University of Wollongong, critz@uow.edu.au

Publication Details

This paper was originally published as: Thomas-Kerr, J, Burnett, I & Ritz, C, Enhancing Interoperability via Generic Multimedia Syntax Translation, Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS '06), Leeds, UK, December 2006, 85-92. Copyright IEEE 2006.

Research Online is the open access institutional repository for the University of Wollongong. For further information contact the UOW Library: research-pubs@uow.edu.au

Enhancing Interoperability via Generic Multimedia Syntax Translation

Abstract
The Bitstream Binding Language (BBL) is a new technology developed by the authors and being standardized by MPEG, which describes how multimedia content and metadata can be mapped onto streaming formats. This paper describes how BBL can be used to enhance the interoperability of multimedia content by providing a generic mechanism for the translation of content between formats. As new content formats are developed, BBL can be used to describe how to translate the content into a form that existing devices are able to render. This consequently simplifies the adoption of new multimedia content forms because existing devices are able to consume the content even though they do not understand its native format.

Disciplines
Physical Sciences and Mathematics

Publication Details
This paper was originally published as: Thomas-Kerr, J, Burnett, I & Ritz, C, Enhancing Interoperability via Generic Multimedia Syntax Translation, Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS '06), Leeds, UK, December 2006, 85-92.
Copyright IEEE 2006.

This conference paper is available at Research Online: http://ro.uow.edu.au/infopapers/485

Enhancing Interoperability via Generic Multimedia Syntax Translation

Joseph Thomas-Kerr
University of Wollongong
joetk@elec.uow.edu.au

Ian Burnett
University of Wollongong
i.burnett@elec.uow.edu.au

Christian Ritz
University of Wollongong
chritz@elec.uow.edu.au

Abstract

The Bitstream Binding Language (BBL) is a new technology developed by the authors and being standardized by MPEG, which describes how multimedia content and metadata can be mapped onto streaming formats. This paper describes how BBL can be used to enhance the interoperability of multimedia content by providing a generic mechanism for the translation of content between formats. As new content formats are developed, BBL can be used to describe how to translate the content into a form that existing devices are able to render. This consequently simplifies the adoption of new multimedia content forms because existing devices are able to consume the content even though they do not understand its native format.

1. Introduction

The ever increasing usage of multimedia content on the Internet and other digital networks has been accompanied by a proliferation of formats in which this content is stored and transmitted. Aside from the multitude of codecs and container types used for the content itself, numerous formats are being defined for associated metadata. Many of these have binary syntax, such as ID3, EXIF, and Matroska. Others use XML to structure metadata – MPEG-7, TVAnytime, and so on. The human-readable nature of XML also means that many groups define private XML grammars for communicating multimedia metadata.

Traditionally, software must be developed specifically for individual formats. However, while many operations on multimedia data are intimately tied to the particular representation – such as decoding and rendering – for other operations the format-specific aspects may be abstracted away from the program code. This results in generic software that operates on a wide range of content using data files that detail the particular requirements of individual formats. Such an approach significantly simplifies the integration of additional content representations, which is achieved via a new data file, rather than extra software modules.

This work was partially funded by the Smart Internet CRC

For example, the Formal Language for Audio Visual Object Representation (FLAVOR) [2] is a format-independent tool which describes the syntax of multimedia bitstreams in order to automatically generate bitstream parsers. Similarly, the Bitstream Syntax Description Language (BSDL) [3] is designed to generically describe the high-level syntax of a scalable bitstream for the purpose of content adaptation. BSDL is part of MPEG-21 [4] – a format-agnostic framework for multimedia transaction and delivery. Its major functions are the description and identification of collections of multimedia content and metadata, content adaptation, expression of user rights, and integration of Intellectual Property Management and Protection (DRM) tools.

The Bitstream Binding Language (BBL)[1] is also being standardized within MPEG-21 [6] following its development and proposal by the authors. BBL is a format-independent language which describes how to map collections of multimedia content and metadata into output bitstreams. It specifies how to packetize and schedule both binary and XML content, so that, for example, an MPEG-21 collection can be mapped onto a streaming multimedia format such as RTP, or an MPEG-2 Transport Stream, regardless of the format of

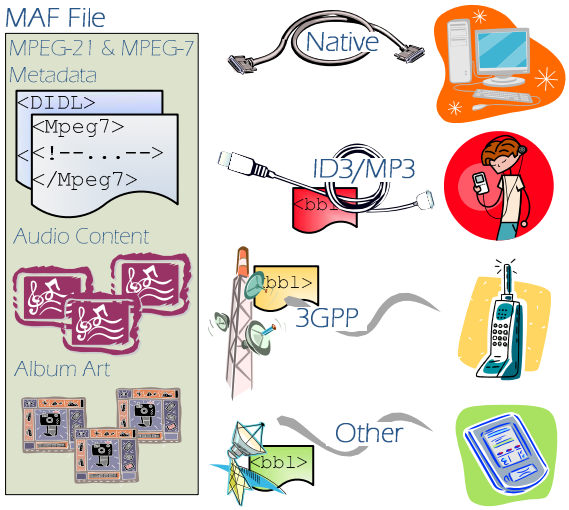

Figure 1 – Multimedia translation using BBL

[1] BBL was first proposed in [5]., but has subsequently undergone substantial revision and improvement.

Proceedings of the Second International Conference on Automated Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'06)
0-7695-2625-X/06 $20.00 © 2006 IEEE

the individual media or metadata content.

BBL, like much of the MPEG-21 framework, is designed to be generic to all multimedia content and metadata – BBL is able to process any XML and any binary content. Consequently, BBL may be applied to the more general problem of how to enable *existing* devices to consume multimedia content published in *new* formats. For example, the Music Player Multimedia Application Format (Music Player MAF) is a new content format standardized by MPEG which combines MPEG-21 and MPEG-7 metadata, JPEG images and MP3 audio content in an ISO file format to provide an augmented digital music library [1]. While existing devices – portable music players, mobile phones, PC software, and so on – may not support this format, BBL can be used to translate Music Player MAF content into representations that these devices understand. This makes it feasible for content providers to deploy the new MAF format and still allow the content to be consumed on legacy equipment.

As shown in Figure 1, a BBL description can generate an MP3 file with ID3 (version 1 or 2) metadata, streamed content and metadata for delivery to a 3GPP device, or a proprietary format for some other device. The software which performs the translation – the BBL processor – is generic with regard to multimedia formats, and all that is required to support a new output format (whether file-based or streamed) is an additional BBL description. BBL does not directly describe the transcoding of media, but instead provides a mechanism to plug in post-processing modules which can be used for this purpose.

The paper continues as follows: Section 2 considers related work, and section 3 briefly presents the MAF file format (used in Section 5 as an example application of BBL). Section 4 provides a detailed discussion of the Bitstream Binding Language. Finally, Section 6 states our conclusions.

2. Related work

Related work can be divided into languages that allow the XML description of binary bitstreams (primarily BSDL and XFlavor) and XML Transformation tools.

2.1. BSDL and XFlavor

BSDL [3] was designed for the purpose of describing the high-level structure of scalable bitstreams, to facilitate content adaptation by means of the truncation of enhancement layers. This is performed in a generic way such that a conformant BSDL Processor requires only a Bitstream Syntax Schema (BS Schema) for the content in order to be able to parse it.

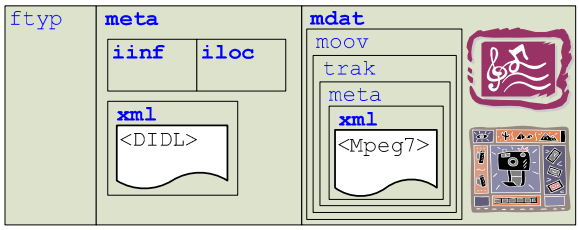

Figure 2 – Music Player MAF file structure (adapted from [1] – some boxes hidden to improve clarity)

The BS Schema language is an extension of XML Schema, which provides mechanisms to specify the bit-level equivalent for XML data types, as well as conditional structures to describe parsing of a bitstream. Because of this, a BS Schema describes both the bitstream, its XML representation, and the translation between the two.

XFlavor [2], on the other hand, uses a C++ like syntax to specify the structure of a bitstream on a field-by-field basis. if-else, switch, for and while constructs are used to describe complex bitstreams.

Both BSDL and XFlavor are able to parse a binary file into XML and vice versa. They can thus expose the structure of binary metadata – for example the boxes of a MAF file – but do not map from one structure to another. BSDL is used by BBL as an abstraction layer to provide an XML interface to binary data (see 4.1).

2.2. XML transformation

The eXtensible Stylesheet Language Transformations (XSLT) [7] was originally designed for use by XSL, which is a stylesheet language. It has, however, proven to be useful in many XML to XML transformation scenarios.

The authors of BSDL propose that XSLT, or a related XML transformation tool – STX [8] – may be used to transform the XML output of the BSDL process, in order to effect bitstream adaptation. These tools were also considered for use as the basis of the description of fragmentation within BBL (see section 3). However, the selection of fragments from one or more input documents does not require the expressivity of a full transformation, and so BBL uses XPath [9] (as does XSLT) for this purpose.

3. Music player MAF

The Music Player MAF format [1] combines audio, images and metadata within an ISO file. These files are structured as a series of nested objects known as boxes. There are numerous types of box, each designed to hold a particular type of data, identified by a four character code. Figure 2 shows one of the box structures for a MAF file. The MPEG-21 Digital Item Declaration

Proceedings of the Second International Conference on Automated
Production of Cross Media Content for Multi-Channel Distribution (AXMEDIS'06)
0-7695-2625-X/06 $20.00 © 2006 IEEE

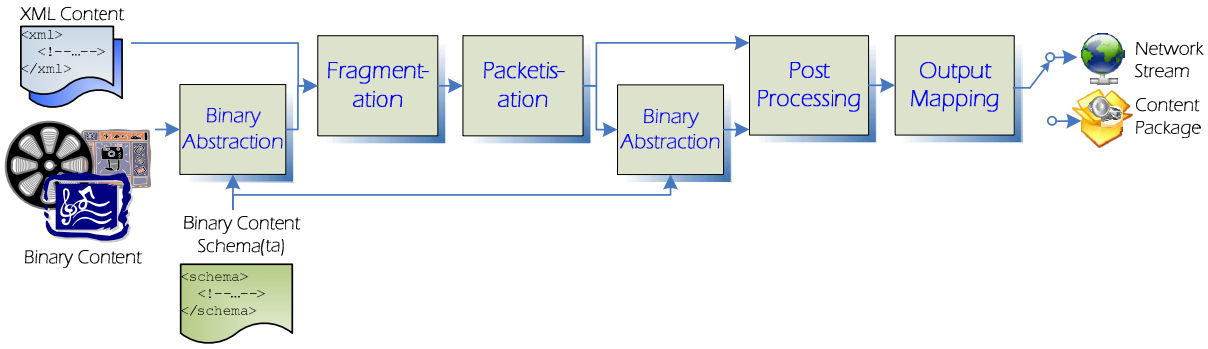

Figure 3 – BBL model

Language (DIDL) XML document in the `meta/xml` box describes the relationships between the media content included in the file, which are identified by URI. The Item Information/Item Location (`iinf/iloc`) boxes provide a link between these URIs and the data bytes representing the media content and metadata, which are stored in the media data (`mdat`) box.. The MPEG-7 metadata is stored within a nested set of boxes which have roles in other ISO files.

A final note regarding the Music Player MAF format is that the audio data is encoded as MP3onMP4 Access Units. This data must be *deformatted* in order to be rendered by existing MP3 devices, but has the advantage of being directly streamable.

4. Bitstream Binding Language

BBL is a description language for the mapping of multimedia content and metadata onto streaming and/or static formats. Figure 3 shows the model developed to describe this process. In summary, BBL describes how to extract fragments from one or more input files (XML or binary), combine these fragments into packets, and map the packets onto output formats. In addition, post-processing may be used to transcode, encrypt or otherwise modify the data.

Many recent multimedia metadata standards are expressed as XML. In fact, the hierarchical structure which XML provides is conceptually similar to the syntactical structures used in binary formats (as demonstrated by the XFlavor and BSDL mappings between binary and XML). Consequently, via a binary abstraction layer (Figure 3), XML tools may be used to manipulate binary data. This enables a single syntax for BBL regardless of the format of the input data.

4.1. Binary Abstraction

BBL uses BSDL as the basis of binary abstraction, because – unlike XFlavor – it allows bitstreams to be described in varying levels of detail (for example, header fields may be explicitly described, while payload data remains hidden). However, the paradigm used by BBL is different to that of BSDL. Whereas the latter transforms binary content into XML in its entirety, BBL uses the BS Schema as an abstraction – an XML *view* of the content. This may be implemented in a number of ways. The brute-force approach uses the BS Schema abstraction to generate the complete XML before providing it to the fragmentation engine (ie the approach taken by BSDL). However, a lighter-weight approach is to couple the binary conversion process with a streaming fragmentation processor in order to generate on-the-fly only those XML fragments which are required. Finally, a more advanced method is to use the BS Schema to compile the fragmentation expressions into a binary location expression which operates directly on the input data.

4.2. Fragmentation

The fragmentation process is required to identify each portion of the input data to be mapped into a separate output packet. This typically involves the selection of subtrees of the input data. XPath [9] is used as an addressing scheme to identify the root node of the subtree(s) which are to be retrieved, coupled with an indication of the number of levels of descendents to be included. This method is used because the transformation functionality provided by XSLT is significantly more complex than that required for fragmentation.

In order to facilitate scalability, BBL provides two modes of fragmentation. In the first each fragment is identified individually, using the above mechanism. However, a common scenario is that many groups of fragments will have similar structure but different content – such as the metadata associated with a number of tracks in a music album, or a set of frames in an encoded video. In this case, instead of identifying individual frames, the XPath expression may be composed so as to select the entire set of fragments, and a number of rules applied to determine how to divide the set. These rules may include a maximum fragment size or duration, a limit on the count of a particular

element within a single fragment, or that particular sub-structures must remain whole.

4.3. Packetization

Once fragments of input content have been identified, they are inserted into packets. This process is required to identify the temporal and other parameters associated with each fragment, and also to provide any syntactical structures not present in the input data but required by the output format. Examples of such structures are XML or binary fields to encapsulate metadata (as shown in Figure 4), or supplementary packet headers (such as that required for the carriage of MPEG-4 streams over RTP in RFC3640).

Fragmentation and Packetization are conducted as if the content were entirely XML, using the binary abstraction discussed above. Once the data has been packetized, packets may be treated as a whole by the model, whether they contain XML or binary content.

4.4. Post-processing

A post-processing stage is necessary to provide custom manipulation of the output data – such as compression of XML, transcoding of media content, or encryption. The complexity of such operations prevents their declarative specification within the BBL. Instead, BBL provides a reference to a plug-in module used to perform the processing.

4.5. Output mapping

Finally, a mechanism is required to identify the mapping of the packets of data into the output stream, in such a way that the temporal and other parameters of each packet are met. In order to provide extensibility, output mappings are specified in a similar fashion to processing modules above. MPEG is currently standardizing handlers for RTP and for MPEG-2 Transport Streams.

5. Experimental results

This section demonstrates how BBL may be applied to one of the scenarios from Figure 1 – conversion of a Music Player MAF file to ID3v2 [10] and MP3. This operation is shown in further detail in Figure 5. Inputs to the BBL processor are

a) the MAF file;
b) a BBL instruction file; and
c) BS Schemata to expose the structure of the MAF file and output data (MP3/ID3v2).

In this instance, output is mapped to a file. Changing the output handler to RTP and adding instructions to fragment the MP3 data into frames would provide a streaming output. An example demonstrating BBL in a streaming environment may be found in [11].

```
<schema>
  <complexType name="FullBoxType"
               abstract="true">
    <complexContent>
      <extension base="maf:BoxType">
        <sequence>
  <element name="version" type="unsignedByte"/>
  <element name="flags"   type="maf:hex3"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>

  <simpleType name="hex3">
    <restriction base="hexBinary">
      <length value="3"/>
    </restriction>
  </simpleType>

  <complexType name="MetadataType">
    <complexContent>
      <extension base="maf:FullBoxType">
        <choice maxOccurs="unbounded"
                bs0:layerLength="maf:size-12">
          <element type="maf:ItemLocationType"
                   name="iloc" bs0:lookAhead="4"
                     bs0:ifNextStr="iloc"/>
          <element type="maf:ItemInfoType"
                   name="iinf" bs0:lookAhead="4"
                     bs0:ifNextStr="iinf"/>
        <element name="xml" type="maf:XMLType"
                     bs0:lookAhead="4"
                     bs0:ifNextStr="xml "/>
          <!-- ... -->
        </choice>
      </extension>
    </complexContent>
  </complexType>
  <!-- ... -->
</schema>
```

Figure 4 – MAF BS Schema excerpt

5.1. MAF Bitstream Syntax Schema

Figure 4 shows a brief excerpt[2] from the BS Schema that describes the structure of a MAF file (as depicted previously in Figure 2). Each box in a MAF file is an extension of either `maf:BoxType` or `maf:FullBoxType` (which itself extends `maf:BoxType`).

Some XML Schema datatypes – `unsignedByte` for example – have a direct correspondence to a binary representation (8 bits in this case). Other datatypes may be derived from these base types – such as `maf:hex3` – via the XML Schema facets (`length`, `maxExclusive`, and so on) and also via a number of BSDL facets including `bs2:length`. The facet `bs2:length` allows the size of a datatype to be determined dynamically by the result of an XPath expression.

[2] The complete data files and a BBL processor are available at whisper.elec.uow.edu.au/people/jtkerr/BBL.html.

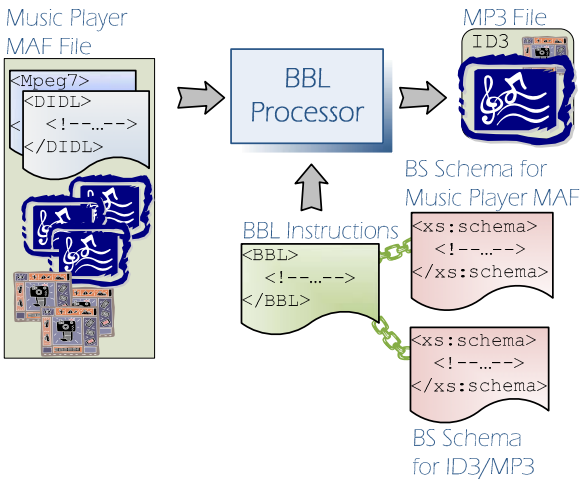

Figure 5–Example: Music Player MAF to ID3v2/MP3

The complex type definition for the metadata box is also shown (`maf:MetadataType`), to demonstrate some of the mechanisms used to specify bitstream structure. This box extends `maf:FullBoxType`, so the first items in the box are a size field, four-character id code, version number and set of flags. Following this, a metadata box may contain a number of sub-boxes, in any order. This is modeled in the schema by the `choice` particle, where the choice between the possible elements is determined by the value of the sub-box four character code (specified by `bs0:lookAhead` and `bs0:ifNextStr`). The overall size of the choice particle is specified by the XPath expression within `bs0:layerLength`, which references the box size field and subtracts the size of the box header.

5.2. MP3/ID3 Bitstream Syntax Schema

Part of the BS Schema for MP3 and ID3v2 is shown in Figure 6. It is used to specify how the structure of ID3 metadata within an MP3 file, which is the output of the BBL process in this example (Figure 5). Numerous features of the BS Schema are similar to those described above (section 5.1), and need not be described again in detail.

The macro-structure of the file is an optional ID3 tag (detected by the presence of the string `"ID3"`), followed by the MP3 data. Because the MP3 data is to be written to a file, its internal structure need not be specified explicitly, and so it is represented by a `bs1:byteRange` datatype, which provides an offset and length for the data within the referenced file (this file is specified by the `bs1:bitstreamURI` attribute).

The size of the ID3 tag payload is stored as a "synchsafe integer" [10], which is essentially four 7-bit fields interspersed by single zero bits. Consequently, computation of this value is more involved than was previously the case (the `bs0:layerLength` attribute on

```
<schema>
  <element name="mp3File">
    <complexType>
      <sequence>
        <element ref="id3:ID3" minOccurs="0"
                 bs0:ifNextStr="ID3"/>
<element name="mp3Data" type="bs1:byteRange"/>
      </sequence>
      <attribute ref="bs1:bitstreamURI"/>
    </complexType>
  </element>

  <element name="ID3">
    <complexType>
      <sequence>
<element name="header" type="id3:HeaderType"/>
        <sequence bs0:layerLength="
          id3:header/id3:size/id3:byte0 *
          2097152 +
          id3:header/id3:size/id3:byte1 *
          16384 +
          id3:header/id3:size/id3:byte2 * 128
          + id3:header/id3:size/id3:byte3">
        <element name="extendedHeader"
bs2:if="id3:header/id3:extendedHeaderPresent=1"
                 type="id3:ExtendedHeaderType"
                 minOccurs="0"/>
          <choice maxOccurs="unbounded">
    <element name="footer" bs0:ifNextStr="3DI"
                type="id3:HeaderType"/>
   <element name="textFrame" bs0:ifNextStr="T"
                type="id3:TextFrameType"/>
    <element name="urlFrame" bs0:ifNextStr="W"
                type="id3:URLFrameType"/>
       <element name="attachedPictureFrame"
                bs0:ifNextStr="APIC"
                type="id3:PictureFrameType"/>
        <!-- ... -->
          </choice>
        </sequence>
      </sequence>
    </complexType>
  </element>

  <complexType name="TextFrameType">
    <complexContent>
      <extension base="id3:AbstractFrameType">
        <sequence bs0:layerLength="id3:size">
    <element name="encoding" type="id3:hex1"/>
          <choice maxOccurs="unbounded">
   <element name="emptyString" type="id3:hex1"
            bs2:ifNext="00"/>
   <element name="value"
            type="bs0:stringUTF8NT"/>
          </choice>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!-- ... -->
</schema>
```

Figure 6 – MP3/ID3v2 BSSchema excerpt

the `sequence` particle). The ID3 payload may contain numerous frames; only `TextFrameType` is shown. This consists of a one-byte encoding indicator, and a null-terminated string which may be empty.

5.3. BBL Instructions

Figure 7 shows part of the BBL instructions for translating MAF files into a legacy format (MP3 with

ID3v2 metadata) so that the new format may be consumed by existing devices. The output of the processor is specified by the `content` element, which is converted to its binary representation (specified by the MP3/ID3v2 BS Schema – see section 5.2), and post-processed by a `deformatMP3` operation.

The majority of the BBL instructions (of which only a small subset are shown in Figure 7) perform the task of extracting the metadata from the MPEG-7 description within the MAF file, and inserting it into the appropriate ID3 frames. This is shown for the ID3 text frame with the ID `TIT2`, which corresponds to the MPEG-7 `songTitle` descriptor (`&MPEG7PATH;` is an XML entity declared elsewhere in the document). The size field of the text frame is populated via the XPath variable `$id3SongTitleLength`, which is defined in the `variables` section of the `packet` definition. ID3 text frames for other MPEG-7 descriptors are similarly specified, and the image data is inserted into an ID3 APIC frame, using a similar mechanism to that below.

The `id3:mp3Data` element contains the audio data. The location of this data within the MAF file is retrieved using the references in the DIDL and `iinf`/`iloc` boxes. This location (an offset and length) is inserted the element content, which was specified as a `bs1:byteRange` type (in the BS Schema for ID3 – Figure 6). This datatype extracts the byte array from the input file and copies it to the output.

6. Conclusion

This paper has demonstrated how the Bitstream Binding Language may be used to enhance interoperability for multimedia content, by providing a generic mechanism for translating the content from one syntax to another. Using the new Music Player MAF format as an example, we have shown how BBL can extract fragments of binary and XML data and insert them into a different output format, so that existing devices are able to render the content.

7. References

[1] K. Diepold, *et al.*, "MPEG-A: multimedia application formats," *Multimedia, IEEE*, vol. 12, pp. 34, 2005.

[2] D. Hong and A. Eleftheriadis, "XFlavor: bridging bits and objects in media representation," presented at *Multimedia & Expo, 2002, IEEE Intl. Conf. on*, 2002.

[3] M. Amielh and S. Devillers, "Bitstream Syntax Description Language: Application of XML-schema to Multimedia Content Adaptation," presented at *WWW, 11th Intl. Conf. on*, 2002.

[4] ISO/IEC, *TR 21000-1:2004 Information technology - Multimedia framework (MPEG-21) - Part 1: Vision, Technologies and Strategy*, 2004.


```
<bbl>
  <!-- ... -->
  <packet>
    <content encoding="deformatMP3">
      <id3:mp3File>
        <attribute name="bs1:bitstreamURI">
          <value-of select="$inputFile"/>
        </attribute>
        <id3:ID3>
          <id3:header>
            <id3:id>ID3</id3:id>
            <!-- ... -->
          </id3:header>

          <id3:textFrame>
            <id3:id>TIT2</id3:id>
            <id3:size>
      <value-of select="$id3SongTitleLength"/>
            </id3:size>
            <id3:flags>0000</id3:flags>
            <id3:encoding>00</id3:encoding>
            <id3:value>
              <include ref="&MPEG7PATH;
      /mp7:CreationInformation/mp7:Creation
      /mp7:Title[@type='songTitle']/text()"/>
            </id3:value>
          </id3:textFrame>

          <!-- ... -->
        <id3:mp3Data>
     <value-of select="concat($mp3Base_offset+
    $mp3Extent_offset,' ',$mp3Extent_length)"/>
        </id3:mp3Data>
      </id3:mp3File>

    </content>
    <variables>
      <define bbl:name="id3SongTitleLength"
              bbl:value="string-length(
            ./id3:mp3File/id3:ID3/id3:textFrame
              [id3:id='TIT2']/id3:value) + 2"/>
      <!-- ... -->
    </variables>
  </packet>
</bbl>
```

Figure 7 – BBL Instructions for MAF to MP3/ID3

[5] J. Thomas-Kerr, *et al.*, "Bitstream Binding Language - Mapping XML multimedia containers into streams," presented at *Multimedia & Expo, IEEE Intl. Conf. on*, 2005.

[6] ISO/IEC, *FCD 21000-18, IT - Multimedia framework (MPEG-21) -Part 18: Digital Item Streaming*, 2006.

[7] J. Clark, "XSL Transformations (XSLT)," http://www.w3.org/TR/xslt, 1999.

[8] P. Cimprich, *et al.*, "Streaming Transformations for XML (STX)," http://stx.sourceforge.net/documents/spec-stx-20040701.html, 2004.

[9] A. Berglund, *et al.*, "XML Path Language (XPath) 2.0," http://www.w3.org/TR/xpath20, 2005.

[10] M. Nilsson, "ID3 tag version 2.4.0 - Main Structure," http//www.id3.org/id3v2.4.0-structure.txt, 2000.

[11] J. Thomas-Kerr, *et al.*, "Format-Independent Multimedia Streaming," presented at *Multimedia & Expo, IEEE Intl. Conf. on*, 2006.