EMu Documentation

Multimedia Web Service

Document Version 1.1

Version 1.1

Contents

Introduction

This document provides a detailed description the facilities available in the EMu Multimedia Web Service. The service accepts HTTP GET or PUT requests and responds with standard HTTP responses. The following types requests are accepted:

- Resource Request
- Metadata Request
- Resource Discovery
- Resource Listing
- Resource Insertion
- Metadata Update
- Resource Replacement
- Resource Deletion

Each of these request types is described in the following sections.

Resource Request

A resource is requested by submitting an HTTP POST or GET request to the Multimedia Web Service with a request type of "resource".

Parameters for all Resource requests

Parameter Name	Mandatory	Default Value	Description
request	Yes	n/a	To request a multimedia resource the value of the request parameter must be "resource".
irn	No	n/a	Integer containing the EMu internal record number of the multimedia resource. A request must contain an **irn** or **identifier**, or both.
identifier	No	n/a	Text containing a 3rd Party unique identifier for the resource. A request must contain an **irn** or **identifier**, or both.
encoding	No	raw	The encoding of the resource to be returned. Acceptable values are "raw" (the default) or "base64".
checksum	No	none	If set to a value other than "none" (which is the default) a checksum of the resource will be added to the response generated. Acceptable values are "none" or "crc32".

Additional parameters for image-specific Resource requests

Parameter	Mandatory	Default Value	Description
width	No	Width of the master image.	Pixel width of generated image. Note that if width is not specified but height is, then width will scale to maintain aspect ratio.
height	No	Height of the master image.	Pixel height of generated image. Note that if height is not specified but width is, then height will scale to maintain aspect ratio.
format	No	n/a	The image format. See Appendix 2 for a list of supported formats. If no format is specified the resource will be in the same format as the resource it is generated from.
aspect	No	yes	If set to "yes" (or "y"), aspect ratio will be conserved in the generated image. If set to "no" (or "n") aspect ratio will not necessarily be preserved.

rendermaster	No	no (or n)	If set to “yes” (or “y”) the generated image will be rendered from the master image in EMu. If set to “no” (or “n”) the generated image will be rendered from smallest generated resolution in EMu that is larger than this request’s resolution. For lossy formats it is better to be generating an image from the master image.

Example 1:

http://server/optionalpath/multimedia/entry.php?request=resource&irn=1234

This is the simplest form of this request. It retrieves the multimedia resource associated with the MMR record with an IRN of 1234. The resource is returned in raw (binary) format.

Example 2:

http://server/optionalpath/multimedia/entry.php?request=resource&identifier=abcd&encoding=base64

This request retrieves the multimedia resource associated with the MMR record which has “abcd” in its identifier (MulIdentifier) field. This request fails if more than one record matches this identifier. The resource returned is base64 encoded.

Example 3:

http://server/optionalpath/multimedia/entry.php?request=resource&irn=1234&width=30

This request retrieves the multimedia resource associated with the MMR record with an IRN of 1234. An image with a width of 30 pixels is generated. Aspect ratio of the image is preserved. The returned image is generated from the smallest resolution of the master image that is larger than the image being generated.

Example 4:

http://server/optionalpath/multimedia/entry.php?request=resource&irn=1234&width=30&height=30&format=tif&aspect=no

This request retrieves the multimedia resource associated with the MMR record with an IRN of 1234. An image with a width and height of 30 pixels is generated. Aspect ratio of the image is not preserved. The generated image is a TIFF, and is generated from the smallest resolution of the master image in EMu that is larger than the image being generated.

Example 5:

http://server/optionalpath/multimedia/entry.php?request=resource&irn=1234&width=30&height=30&format=tif&rendermaster=yes

This request retrieves the multimedia resource associated with the Multimedia record with an IRN of 1234. An image with a width and height of 30 pixels is generated. Aspect ratio of the image is preserved. The generated image is a TIFF, and is generated from the master image in EMu.

Response

Success

An HTTP 200 OK response containing the requested multimedia. The content is encoded using the requested encoding, if any. Otherwise the content will be in raw (binary) form. The details of the HTTP response headers will depend on the resource being delivered. As an example the headers for a response containing a JPEG image include:

```
Content-type: image/jpeg
Content-Length: xxx
Content-Transfer-Encoding: base64
Content-Crc32: deadbeef

...contents of image...
```

Failure

HTTP 403 Forbidden if the access is not granted to the resource.

HTTP 404 Not Found if the resource cannot be located.

Metadata Request

Metadata for a resource is requested by submitting an HTTP POST or GET request to the Multimedia Web Service with a request type of “metadata”.

Parameters for all Metadata requests

Parameter Name	Mandatory	Default Value	Description
request	Yes	n/a	To request the metadata of a multimedia resource the value of the request parameter must be “metadata”.
irn	No	n/a	Integer containing the EMu internal record number of the multimedia resource. A request must contain an **irn** or **identifier**, or both.
identifier	No	n/a	Text containing a 3rd Party unique identifier for the resource. A request must contain an **irn** or **identifier**, or both.

Additional parameters for image-specific Metadata requests

Parameter	Mandatory	Default Value	Description
format	No	all	The **format** parameter specifies what metadata to return. See below for supported metadata formats.

The requested metadata is returned in Adobe's Extensible Metadata Platform (XMP) format (http://www.adobe.com/products/xmp/).

The “format” parameter will accept the following values:

- **EXIF**

  Requests with this value will return the full set of EXIF 2.2 elements embedded in the XMP document. See http://www.exif.org/Exif2-2.PDF for more details. If the resource does not contain EXIF metadata an error response will be returned.

- **IPTC**

  Requests with this value will return the full set of IPTC 4.1 elements embedded in the XML document. See http://www.iptc.org/IPTC4XMP/ for more details. If the resource does not contain IPTC metadata an error

response will be returned.

- **DC**

  Requests with this value will return the full set of Dublin Core fields recorded in the MMR record embedded in the XMP document.

- **native**

  Requests with this value will include a complete description of the data in the MMR record embedded in the XML document. The values returned are not limited to metadata embedded within the digital resource itself. The format of the XML is based on the standard RDF document structure.

- **all**

  Requests with this value will return all known metadata formats: EXIF, IPTC, DC and native in the one XMP document. This is the default.

Example 1:

http://server/optionalpath/multimedia/entry.php?request=metadata&irn=1234

This is the simplest form of this request. It retrieves all known metadata formats for the multimedia resource associated with IRN 1234 in the MMR.

Example 2:

http://server/optionalpath/multimedia/entry.php?request=metadata&irn=1234&format=EXIF

This request retrieves the EXIF metadata for the multimedia resource associated with IRN 1234 in the MMR. If the resource does not contain EXIF metadata the server will respond with an HTTP 400 Bad Request error

Response

Success

An HTTP 200 OK response containing the requested metadata. For example:

```
Content-type: text/xml
Content-Length: xxx

<?xml version="1.0"?>
<?xpacket begin="" id="W5M0MpCehiHzreSzNTczkc9d"?>
<x:xmpmeta xmlns:x="adobe:ns:meta/">
  <rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <rdf:Description
      rdf:about=""
      xmlns:dc="http://purl.org/dc/elements/1.1"
      xmlns:tiff="http://ns.adobe.com/tiff/1.0/"
      xmlns:exif="http://ns.adobe.com/exif/1.0/"
      xmlns:photoshop="http://ns.adobe.com/photoshop/1.0/"
      xmlns:Iptc4xmpCore="http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/"
      xmlns:native="http://kesoftware.com/emu/xmlns/emultimedia/"
    >

      <!-- Dublin Core -->
```

```
      <dc:format>image/jpeg</dc:format>
      <dc:identifier>picture.jpg</dc:identifier>
      …

      <!-- EXIF -->
      <exif:ThumbnailOffset>4852</exif:ThumbnailOffset>
      <exif:XResolution>72</exif:XResolution>
      <exif:ExposureTime>1/400</exif:ExposureTime>
      <exif:Model>E-20,E-20N,E-20P</exif:Model>
      …

      <!-- IPTC -->
      <Iptc4xmpCore:DateCreated>2006:12:18</Iptc4xmpCore:DateCreated>
      <Iptc4xmpCore:City>Broome</Iptc4xmpCore:City>
      <Iptc4xmpCore:Province-State>WA</Iptc4xmpCore:Province-State>
      …

      <!-- Native -->
      <native:AdmDateInserted>19/08/2009</native:AdmDateInserted>
      <native:AdmDateModified>19/08/2009</native:AdmDateModified>
      <native:AdmTimeInserted>10:44</native:AdmTimeInserted>
      <native:AdmTimeModified>10:44</native:AdmTimeModified>
      <native:ChaFileSize>1331398</native:ChaFileSize>
      <native:ChaImageColorDepth>8</native:ChaImageColorDepth>
      <native:ChaImageHeight>1920</native:ChaImageHeight>
      <native:ChaImageResolution>144</native:ChaImageResolution>
      <native:ChaImageWidth>2560</native:ChaImageWidth>
      <native:ChaMd5Sum>7f648024f39541e439a666ccd3de4493</native:ChaMd5Sum>
      <native:DocBitsPerPixel_tab>
        <rdf:seq>
          <rdf:li>8</rdf:li>
          <rdf:li>16</rdf:li>
        </rdf:seq>
      </native:DocBitsPerPixel_tab>
      <native:DocColourSpace_tab>
        <rdf:seq>
          <rdf:li>RGB</rdf:li>
          <rdf:li>RGB</rdf:li>
        </rdf:seq>
      </native:DocColourSpace_tab>
      <native:DocCompression_tab>
        <rdf:seq>
          <rdf:li>JPEG</rdf:li>
          <rdf:li>None</rdf:li>
        </rdf:seq>
      </native:DocCompression_tab>
      …

    </rdf:Description>
  </rdf:RDF>
</x:xmpmeta>
<?xpacket end="w"?>
```

Failure

HTTP 404 Not Found if the record cannot be located.

HTTP 403 Forbidden if the access is not granted to the record.

HTTP 400 Bad Request if the resource does not include the requested metadata

Resource Discovery

Resources can be discovered by submitting an HTTP POST or GET request to the Multimedia Web Service with a request type of “search”.

Parameters for Resource Discovery requests

Parameter Name	Mandatory	Default Value	Description
request	Yes	n/a	To request a list of matching resources the value of the request parameter must be “search”.
filename	No	n/a	Filename to match. This may include “right-truncation” wildcards such as “123-456*”.

Example 1:

http://server/optionalpath/multimedia/entry.php?request=search&filename=123-456*

This request searches for resources which have a filename that starts with “123-456”.

Response

Success

An HTTP 200 OK response containing an XML document with a list of the EMu IRNs of the records that match. For example:

```
Content-type: text/xml
Content-Length: xxx

<?xml version="1.0" ?>
<response matches=”6”>
   <irn>1235</irn>
   <irn>4567</irn>
   …
</response>
```

The IRNs returned can be used to identify the resources in subsequent requests.

If no matches are found a success response will still be sent but the *matches* attribute in the *response* tag will contain the value 0.

Resource Listing

A list of the entire set of resources in the MMR in the repository can be retrieved by submitting an HTTP POST or GET request to the Multimedia Web Service with a request type of “list”.

Parameters for Resource Listing requests

Parameter Name	Mandatory	Default Value	Description
request	Yes	n/a	To request a list of matching resources the value of the request parameter must be “list”.
type	No	“all”	The type of resources to list. Acceptable values are “external”, “shared”, or “all” (the default). Requesting “all” returns the full list of all “external” and “shared” resources. No “internal” resources are listed.

Example 1:

http://server/optionalpath/multimedia/entry.php?request=list

This is the simplest form of this request. It lists all resources in the MMR which are flagged as “external” or “shared”.

Example 2:

http://server/optionalpath/multimedia/entry.php?request=list&type=external

This request lists all resources in the MMR which are flagged as “external”.

Response

Success

Asn HTTP 200 OK response containing an XML document with a list of the EMu IRNs of the records that match. For example:

```
Content-type: text/xml
Content-Length: xxx

<?xml version="1.0"?>
<resources matches=”2”>
     <resource type=”shared”>
          <irn>42</irn>
          <identifier>picture.tif</identifier>
      </resource>
      <resource type=”external”>
```

```
                <irn>86</irn>
            <identifier>football.jpg</identifier>
        </resource>        …
    </resources>
```

The IRNs returned can be used to identify the resources in subsequent requests.

If no resources can be listed a success response will still be sent but the *matches* attribute in the *resources* tag will contain the value 0.

Resource Insertion

New resources can be added to the MMR. A single multimedia resource is uploaded using an HTTP POST request with a `Content-type: multipart/form-data` header.

The first part of multi-part request specifies the request type. It must contain the value “insert”.

The second part of the request contains the asset type. This can be “external” or “shared”. This part is optional and the asset type will default to “external” if not supplied.

The third part contains a 3rd party identifier. This part is also optional.

The fourth part contains a checksum. The data supplied is formatted as

```
<checksum-type>;<checksum-value>
```

The `<checksum-type>` identifies the checksum algorithm used. The only acceptable value is “crc32”. The `<checksum-value>` is the actual value of the checksum, represented as a hexadecimal string. This part is optional. If this part is supplied the file contents will be checked against the checksum. If the two do not agree an error response will be generated.

The final part of the request contains the multimedia resource itself. The content must be submitted in either raw (binary) or base64 encoded format.

Example 1.

A request to insert an “external” asset comprising an image file named “image.jpg” with a 3rd party identifier of “abcd” would contain:

```
Content-type: multipart/form-data, boundary=AaB03x
Content-Length: xxx

--AaB03x
content-disposition: form-data; name="request"

insert
--AaB03x
content-disposition: form-data; name="type"

external
--AaB03x
content-disposition: form-data; name="identifier"

abcd
--AaB03x
content-disposition: form-data; name="checksum"

crc32;deadbeef
--AaB03x
Content-disposition: form-data; filename="image.jpg"
Content-type: image/jpg
Content-Transfer-Encoding: base64
```

```
...contents of image.jpg...
--AaB03x--
```

Response

Success

The response is an HTTP 200 OK response containing an XML document with the EMu IRN of the record inserted. For example:

```
Content-type: text/xml
Content-Length: xxx

<?xml version="1.0"?>
<response>
  <irn>1235</irn>
</response>
```

The IRN returned can be used to identify the resources in all subsequent requests. If a 3rd party identifier is supplied it can also be used to identify the resource.

Failure

HTTP 400 Bad request if the checksum fails

HTTP 403 Forbidden response if access is not granted to insert the resource, or if the format of the resource is excluded within the EMu Registry explicitly.

Metadata Update

The metadata associated with a resource can be updated. The metadata for a single multimedia resource is updated using an HTTP POST request with a `Content-type: multipart/form-data` header.

The first part of multi-part request specifies the request type. It must contain the value “update”.

The second part of multi-part request identifies the resource to be updated. This can be either an MMR IRN or a 3rd party identifier. This part is mandatory.

The third part contains the native-format XML specifying which fields in the MMR should be modified. Note that this updates the fields in the MMR record. It does not alter any technical metadata stored in the physical resource file. Details of the native format are given in Appendix 3.

Example 1.

To add a Title and list of Creators to the MMR record with identifier “abcd” the following request would be transmitted:

```
Content-type: multipart/form-data, boundary=AaB03x
Content-Length: xxx

--AaB03x
content-disposition: form-data; name="request"

update
--AaB03x
content-disposition: form-data; name="identifier"

abcd
--AaB03x
Content-disposition: form-data; name="metadata"
Content-type: text/xml
<?xml version="1.0" encoding="ISO-8859-1"?>
<table>
      <tuple>
          <atom name="MulTitle">An image of a house.</atom>
          <table name="MulCreator_tab">
              <tuple>
                  <atom>John Smith</atom>
              </tuple>
              <tuple>
                  <atom>Bill Wilson</atom>
              </tuple>
          </table>
      </tuple>
</table>
--AaB03x--
```

Response

Success

An HTTP 200 OK message.

Failure

An HTTP 403 Forbidden response if the access is not granted to update the resource, or the format of the metadata supplied is invalid.

Resource Replacement

The file associated with a record in the MMR can be replaced by submitting an HTTP POST request with a `Content-type: multipart/form-data` header.

The first part of multi-part request specifies the request type. It must contain the value “replace”.

The second part of multi-part request identifies the resource to be modified. This must contain the IRN of the resource. This part is mandatory.

The remaining parts are identical to those for the Resource Insertion request.

The third part contains a 3rd party identifier. This part is optional.

The fourth part contains a checksum. The data supplied is formatted as

```
<checksum-type>;<checksum-value>
```

The `<checksum-type>` identifies the checksum algorithm used. The only acceptable value is “crc32”. The `<checksum-value>` is the actual value of the checksum, represented as a hexadecimal string. This part is optional. If this part is supplied the file contents will be checked against the checksum. If the two do not agree an error response will be generated.

The final part of the request contains the multimedia resource itself. The content must be submitted in either raw (binary) or base64 encoded format.

When the resource is replaced all the technical metadata in the MMR record is updated. This includes any EXIT, IPTC or XMP data extracted from the new resource file. New resolutions are also generated and the resolutions table in the MMR record is updated. No other information in the MMR record (with the exception of record modification date and time stamps) is updated.

Example 1.

A request to replace the resource file associated with the MMR record with an IRN of 1234 with an asset comprising an image file named “football.jpg” with a 3rd party identifier of “abcd”:

```
Content-type: multipart/form-data, boundary=AaB03x
Content-Length: xxx

--AaB03x
content-disposition: form-data; name="request"

replace
--AaB03x
content-disposition: form-data; name="irn"

1234
--AaB03x
content-disposition: form-data; name="identifier"

abcd
--AaB03x
content-disposition: form-data; name="checksum"
```

```
crc32;deadbeef
--AaB03x
Content-disposition: form-data; filename="football.jpg"
Content-type: image/jpg
Content-Transfer-Encoding: base64

...contents of image.jpg...
--AaB03x--
```

Response

Success

An HTTP 200 OK message.

Failure

An HTTP 403 Forbidden response if the access is not granted to update the resource, or the format of the metadata supplied is invalid.

Resource Deletion

Resources can be removed from the MMR by sending an HTTP POST or GET request to the Multimedia Web Service with a request type of “delete”.

Parameters for Resource Deletion requests

Parameter Name	Mandatory	Default Value	Description
request	Yes	n/a	To remove a spcific multimedia resource from the repository the parameter must be “delete”.
irn	No	n/a	Integer containing the EMu internal record number of the multimedia resource. A request must contain an **irn** or **identifier**, or both.
identifier	No	n/a	Text containing a 3rd Party unique identifier for the resource. A request must contain an **irn** or **identifier**, or both.

The requested resource will be removed from the MMR if permitted. Note that the physical resource may not be removed at the time of deletion. Instead the record in the MMR may be flagged for deletion and removed at a later time.

As of version 1.1 a request to delete a “shared” resource will fail. This is to avoid problems where other EMu modules have attached to this resource.

Example 1:

http://server/optionalpath/multimedia/entry.php?request=delete&irn=1234

The request deletes the resources associated with IRN 1234.

Success

An HTTP 200 OK message.

Failure

An HTTP 403 Forbidden response if the access is not granted to update the resource, or the format of the metadata supplied is invalid.

Appendix 1: Authentication/Security

Access to the Multimedia Web Service is controlled by the web server. If the web server used is Apache this would be handled using standard Basic Authentication configured by an .htaccess file.

For more secure transmission of HTTP requests the web server can be configured to accept HTTP requests across an SSL connection only.

Within the Multimedia Web Service, processes are run as a non-privileged EMu user, and access to records is controlled internally using Texpress's record level security.

Appendix 2: Supported Image Formats

Image formats supported for requests.

ART	PFS: 1st Publisher
AVI	Microsoft Audio/Visual Interleaved
AVS	AVS X image
BMP	Microsoft Windows bitmap
CGM	Computer Graphics Metafile
CIN	Kodak Cineon Image Format
CMYK	Raw cyan, magenta, yellow, and black samples
CMYKA	Raw cyan, magenta, yellow, black, and alpha samples
CR2	Canon Digital Camera Raw Image Format
CRW	Canon Digital Camera Raw Image Format
CUR	Microsoft Cursor Icon
CUT	DR Halo
DCM	Digital Imaging and Communications in Medicine (DICOM) image
DCR	Kodak Digital Camera Raw Image File
DCX	ZSoft IBM PC multi-page Paintbrush image
DIB	Microsoft Windows Device Independent Bitmap
DJVU	
DNG	Digital Negative
DOT	Graph Visualization
DPX	SMPTE Digital Moving Picture Exchange 2.0 (SMPTE 268M-2003)
EMF	Microsoft Enhanced Metafile (32-bit)
EPDF	Encapsulated Portable Document Format

EPI	Adobe Encapsulated PostScript Interchange format
EPS	Adobe Encapsulated PostScript
EPS2	Adobe Level II Encapsulated PostScript
EPS3	Adobe Level III Encapsulated PostScript
EPSF	Adobe Encapsulated PostScript
EPSI	Adobe Encapsulated PostScript Interchange format
EPT	TIFF preview
EXR	High dynamic-range (HDR) file format developed by Industrial Light & Magic
FAX	Group 3 TIFF
FIG	FIG graphics format
FITS	Flexible Image Transport System
FPX	FlashPix Format
GIF	CompuServe Graphics Interchange Format
GPLT	Gnuplot plot files
GRAY	Raw gray samples
HPGL	HP-GL plotter language
HTML	Hypertext Markup Language with a client-side image map
ICO	Microsoft icon
INFO	Format and characteristics of the image
JBIG	Joint Bi-level Image experts Group file interchange format
JNG	Multiple-image Network Graphics
JP2	JPEG-2000 JP2 File Format Syntax
JPC	JPEG-2000 Code Stream Syntax
JPEG	Joint Photographic Experts Group JFIF format
MAN	Unix reference manual pages

MAT	MATLAB image format
MIFF	Magick image file format
MONO	Bi-level bitmap in least-significant-byte first order
MNG	Multiple-image Network Graphics
MPEG	Motion Picture Experts Group file interchange format (version 1)
M2V	Motion Picture Experts Group file interchange format (version 2)
MPC	Magick Persistent Cache image file format
MRW	Sony (Minolta) Raw Image File
MSL	Magick Scripting Language
MTV	MTV Raytracing image format
MVG	Magick Vector Graphics.
NEF	Nikon Digital SLR Camera Raw Image File
ORF	Olympus Digital Camera Raw Image File
OTB	On-the-air Bitmap
P7	Xv's Visual Schnauzer thumbnail format
PALM	Palm pixmap
PAM	Common 2-dimensional bitmap format
PBM	Portable bitmap format (black and white)
PCD	Photo CD
PCDS	Photo CD
PCL	HP Page Control Language
PCX	ZSoft IBM PC Paintbrush file
PDB	Palm Database ImageViewer Format
PDF	Portable Document Format
PEF	Pentax Electronic File

PFA	Postscript Type 1 font (ASCII)
PFB	Postscript Type 1 font (binary)
PFM	Portable float map format
PGM	Portable graymap format (gray scale)
PICON	Personal Icon
PICT	Apple Macintosh QuickDraw/PICT file
PIX	Alias/Wavefront RLE image format
PNG	Portable Network Graphics
PNM	Portable anymap
PPM	Portable pixmap format (color)
PS	Adobe PostScript file
PS2	Adobe Level II PostScript file
PS3	Adobe Level III PostScript file
PSD	Adobe Photoshop bitmap file
PTIF	TIFF
PWP	Seattle File Works multi-image file
RAD	Radiance image file
RAF	Fuji CCD-RAW Graphic File
RGB	Raw red, green, and blue samples
RGBA	Raw red, green, blue, and alpha samples
RLA	Alias/Wavefront image file
RLE	Utah Run length encoded image file
SCT	Scitex Continuous Tone Picture
SFW	Seattle File Works image
SGI	Irix RGB image
SHTML	Hypertext Markup Language client-side image map

SUN	SUN Rasterfile
SVG	Scalable Vector Graphics
TGA	Truevision Targa image
TIFF	Tagged Image File Format
TIM	PSX TIM file
TTF	TrueType font file
TXT	Raw text file
UIL	X-Motif UIL table
UYVY	Interleaved YUV raw image
VICAR	VICAR rasterfile format
VIFF	Khoros Visualization Image File Format
WBMP	Wireless bitmap
WMF	Windows Metafile
WPG	Word Perfect Graphics File
X	display or import an image to or from an X11 server
XBM	X Windows system bitmap, black and white only
XCF	GIMP image
XPM	X Windows system pixmap
XWD	X Windows system window dump
X3F	Sigma Camera RAW Picture File
YCbCr	Raw Y, Cb, and Cr samples
YCbCrA	Raw Y, Cb, Cr, and alpha samples
YUV	CCIR 601 4:1:1

Appendix 3: Native Format

The “native” request format can be used to update all or part of the database record associated with the multimedia asset.

The format used is the native export format used by EMu’s underlying database system Texpress for data export.

Documents submitted in “native” format are structured as follows:

1. A standard XML header
2. A top-level “table” tag.
3. An inner-level “tuple” tag.
4. Repeated “column-level” tags within the “tuple” tag. The column-level tags have the following format:
    a. Single-valued columns are represented by simple text enclosed in an “atom” tag.
    b. Multi-valued columns are enclosed in a “table” tag. The “table” tag itself includes multiple “tuple” tags, just as the outer-level data does. This is a recursive structure.

    Each column-level tag includes the name of the column as an attribute.

An example of the “native” format is shown below.

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<table>
  <tuple>
    <atom name="MulTitle">Footballers I Have Known</atom>
    <atom name="DetCoverage">AFL</atom>
    <atom name="DetRights">Museum Victoria (2009)</atom>
    <atom name="SecAssetType">shared</atom>
  </tuple>
</table>
```

The following table describes each of the columns in the MMR database:

Column	Description
ChaAudience_tab	The intended audience for the multimedia resource.
ChaAudioBitsPerSample	The precision of the digital audio sample is determined by the number of bits per sample, typically 8 or 16 bits.
ChaAudioNumberOfChannels	The number of simultaneous sounds possible in the audio file.
ChaAudioSamplesPerSec	The number of samples of a sound that are taken per second to

	represent the event digitally.
ChaFileSize	The size of the multimedia resource as a stored file, in bytes.
ChaImageColorDepth	The number of colours in the image, e.g. 2 (B&W).
ChaImageHeight	The height of the image in mm, inches, etc.
ChaImageResolution	The number of pixels in the image, e.g. 640 x 480.
ChaImageWidth	The width of the image in mm, inches, etc.
ChaMediaForm	The format of the multimedia resource, e.g. JPG, MPG, etc.
ChaRepository_tab	The storage facility that houses the multimedia resource.
ChaVideoFilmLength	The length of the video in minutes and seconds.
DetContributor_tab	Lists other parties who have contributed to the creation of the multimedia resource.
DetCoverage	The extent or scope of the content of the resource.
DetDate0	
DetLanguage_tab	The primary language used in the title or content of the multimedia resource.
DetPublisher	The publisher of the multimedia resource, i.e. the party responsible for making the resource publicly available.
DetRelation_tab	A reference to a related resource.
DetResourceType	The type of multimedia resource, e.g. image, sound, etc.
DetRights	Rights associated with the multimedia resource, e.g. copyright, access privileges, etc.

DetSource	The person or organisation who provided the multimedia resource.
DetSubject_tab	The subject matter of the multimedia resource, as chosen from prescribed standards.
MulCreator_tab	The party responsible for creating the multimedia item.
MulDescription	An account of the content of the resource, for instance a description of an image, an abstract or table of contents of a text document. (Dublin Core Metadata standard)
MulDocumentType	
MulIdentifier	The file name and extension for the multimedia resource.
MulMimeFormat	The format of the multimedia resource, e.g. JPG, MPG, MS Word document, etc. (MIME is an abbreviation for Multipurpose Internet Mail Extensions)
MulMimeType	The type of multimedia resource, e.g. image, video, document, etc. (MIME is an abbreviation for Multipurpose Internet Mail Extensions)
MulTitle	The name by which the multimedia item is known.
NotNotes	Text notes that are not already contained in a defined field.
SecAssetType	The type of asset. This will contain the value “external”, “shared” or “internal”.
SecRecordStatus	Status of the record. The default value is blank, which indicates an active record. A common alternative is 'Retired'.