

Encoding Astronomical Outreach Imagery Metadata in RDF/XML

Abstract

Revisions

Initial version released. This version applies to version 1.0 of the AVM metadata specification [4].

Specify how to merge XMP documents

Fix all the examples (vr -> aoi etc.)

Abbreviations

AVM

RDF Resource Description Framework

RFC Request For Comments (Internet standard)

URL Uniform Resource Locator

URI Uniform Resource Identifier

XML Extensible Markup Language

XMP Extensible Metadata Platform

XSD XML Schema Document

Contents

[1 Introduction 1](#_Toc148695354)

[1.1 What is metadata 1](#_Toc148695355)

[2 Prerequisites 3](#_Toc148695356)

[2.1 RDF encoding 3](#_Toc148695357)

[2.1.1 RDF containers 3](#_Toc148695358)

[2.1.2 RDF lists 4](#_Toc148695359)

[3 Metadata property encoding 7](#_Toc148695360)

[3.1 XML encoding 7](#_Toc148695361)

[3.2 Lists 7](#_Toc148695362)

[3.2.1 List 7](#_Toc148695363)

[3.2.2 List-Ordered 7](#_Toc148695364)

[3.2.3 List(n) 8](#_Toc148695365)

[3.3 Data types 8](#_Toc148695366)

[3.3.1 String 9](#_Toc148695367)

[3.3.2 Integer 9](#_Toc148695368)

[3.3.3 Float 9](#_Toc148695369)

[3.3.4 Uri, Url 9](#_Toc148695370)

[3.3.5 Date 10](#_Toc148695371)

[3.4 Encoding AOI properties 10](#_Toc148695372)

[4 Merging AOI metadata 11](#_Toc148695373)

[4.1 Value types 11](#_Toc148695374)

[4.1.1 Simple values 12](#_Toc148695375)

[4.1.2 Set 12](#_Toc148695376)

[4.1.3 List 12](#_Toc148695377)

[4.2 Special considerations 12](#_Toc148695378)

[A References 13](#_Toc148695379)

[B Full example 15](#_Toc148695380)

[C Virtual Repository XSD 16](#_Toc148695381)

[D Validating property values 17](#_Toc148695382)

Tables

[Table 1 Default XML namespaces 3](#_Toc148696076)

[Table 2 Standard XML entities 7](#_Toc148696077)

[Table 3 Data types 8](#_Toc148696078)

[Table 4 Default XML namespaces 10](#_Toc148696079)

[Table 5 Keyword storage 10](#_Toc148696080)

[Table 6 Data type formats 16](#_Toc148696081)

[Table 7 Regular expression operators 16](#_Toc148696082)

Listings

[Listing 1 Full example 15](#_Toc148696085)

# Introduction

This document describes a method for encoding the Astronomical Outreach Imagery metadata in XML/RDF. The information in this document can be seen as a companion to the AOI metadata specification as it details one method of encoding the metadata properties from the specification to a format which can be stored in a computer file. The method presented is not the only method possible and this document will not describe how to embed the encoded metadata in a file. For more information about how to embed the encoded data in a file, please refer to the XMP specification.

## What is metadata

The literal meaning of the word metadata is “data about data”. Metadata is used to define or describe certain properties about some piece of data. In the case of Astronomical Outreach Imagery metadata the data is an image and the metadata should contain information about the image which is relevant to people doing astronomical outreach.

The metadata associated with an image can be described as a set of properties that apply to a particular image. Each property must have a unique name to distinguish this property from all the other properties associated with the same image. Each property has some kind of value that depends on the content of the image and the metadata standard which defines which properties are available and what their values should be.

It would be inconceivable to invent a set of properties that could fully describe any piece of data in existence (either presently or in the future). Therefore it is common to use domain specific metadata standards. A metadata standard is merely a definition for a set of properties which makes sense for the domain in which it is used. For images this could be: What is it an image of, when was it taken, and so on.

# Prerequisites

In the following a basic knowledge of the XML and RDF formats is assumed. These standards are defined in the XML standard and the RDF standard respectively. In short a RDF document can be seen as a subclass of XML in that the RDF standard further restricts the format of the document by introducing a fixed element hierarchy.

The XMP metadata format used by Adobe® is an extensible container for a multitude of metadata definitions. The XMP standard uses RDF to enclose each type of metadata associated with a document and so the techniques discussed in this document apply specifically to XMP.

In order to simplify examples the following XML prefix/namespace mappings are defined for all examples that do not explicitly define them. As such Table 1 contains all the namespaces used to encode AOI metadata in XML.

Table Default XML namespaces

|  |  |
| --- | --- |
| Prefix | Namespace |
| rdf | http://www.w3.org/1999/02/22-rdf-syntax-ns# |
| avm | <http://www.communicatingastronomy.org/avm/1.0/> |
| dc | <http://purl.org/dc/elements/1.1/> |
| Iptc4xmpCore | <http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/> |
| photoshop | <http://ns.adobe.com/photoshop/1.0/> |
| xmpRights | <http://ns.adobe.com/xap/1.0/rights/> |

## RDF encoding

The RDF specification describes in detail how to embed custom metadata in a RDF wrapper. Consequently this section will only briefly mention a few essential details regarding RDF.

### RDF containers

First of all RDF uses XML’s support for multiple namespaces in the same XML document. RDF supports multiple metadata schemas embedded in the same RDF container, so multiple namespaces is needed to be able to validate the document. The basic layout of an RDF document is:

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

<rdf:Description

rdf:about=""

xmlns:avm="<http://www.communicatingastronomy.org/avm/1.1/>"

>

<avm:Spatial.Notes>Some notes</avm:Spatial.Notes>

...

</rdf:Description>

<rdf:Description

rdf:about=""

xmlns:dc="<http://purl.org/dc/elements/1.1/>"

>

<dc:description>Image description</dc:description>

...

</rdf:Description>

</rdf:RDF>

So the outer container is a RDF element with multiple Description elements inside, each of which contains metadata properties for one specific metadata schema. In the above example both AOI and Dublin Core metadata was available as separate parallel stores of information, there need not be more than one Description element.

Note that while it is possible to store the same logical metadata in several metadata standards – that is in separate Description elements – the XMP specification specifically discourages this approach. The problem is to keep both sets of metadata synchronized at all times. This would require all tools that touch the metadata to understand and implement this synchronization – something which is clearly infeasible. Instead metadata standards should adopt existing standards for storing properties whenever possible.

### RDF lists

The AOI metadata specification contains several list type properties which may contain more than one value. Rather than encoding a list by using a specific character to separate the individual items it is more desirable to use the hierarchical structure of XML documents to separate the items in the list. To support this RDF defines special list elements which can be used for encoding such property values. An example would be:

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

<rdf:Description

rdf:about=""

xmlns:avm="<http://www.communicatingastronomy.org/avm/1.0/>*"*

>

<avm:Subject.Name>

<rdf:Bag>

<rdf:li>Sol</rdf:li>

<rdf:li>Moon</rdf:li>

</rdf:Bag>

<avm:Subject.Name>

<avm:FL.BackgroundLevel>

<rdf:Seq>

<rdf:li>0.0</rdf:li>

</rdf:Seq>

</avm:FL.BackgroundLevel>

...

</rdf:Description>

</rdf:RDF>

Each element in the list is enclosed in a li element and the entire list is wrapped in either a Seq element or a Bag element. For lists were ordering is significant a Seq element is used while unordered lists use Bag elements.

# Metadata property encoding

This section details how the individual properties should be encoded in order for other software packages to read and correctly interpret the encoded metadata.

## XML encoding

Since XML is a text format it needs to be parsed if its structure and meaning is to be extracted. This puts restrictions on the characters available to represent property values. As a solution to this problem XML introduces *entities*, simple escape sequences that are converted to the restricted characters when information is extracted. All metadata properties are subject to this conversion regardless of the property data type. In particular the characters in Table 2 should be converted to entities.

Table Standard XML entities

|  |  |
| --- | --- |
| Character | Entity |
| & | &amp; |
| < | &lt; |
| > | &gt; |
| ‘ | &apos; |
| “ | &quot; |

## Lists

The AOI metadata specification uses three distinct list types: *list*, *list-ordered*, and *list(n)***.**

### List

List is used to represent property values which have several logical values, as an example the *Subject.Name* property can describe several relevant objects in an image. List properties are stored using a RDF Bag element with each property value in a separate li element.

<avm:Subject.Name>

<rdf:Bag>

<rdf:li>Sol</rdf:li>

<rdf:li>Moon</rdf:li>

</rdf:Bag>

</avm:Subject.Name>

### List-Ordered

List-ordered is used when property values span several exposures and these values need to have a specific ordering in particular with respect to other ordered lists. List-ordered properties are stored using a RDF Seq element with each property value in a separate li element.

<avm:FL.BackgroundLevel>

<rdf:Seq>

<rdf:li>0.0</rdf:li>

<rdf:li>2.4</rdf:li>

<rdf:li>3.5</rdf:li>

</rdf:Seq>

</avm:FL.BackgroundLevel>

Any implementation of the recommendations in this document should make sure that for any property of type *list-ordered* either the property is not included or the value list contains the same number of items found in any other property of type *list-ordered*. Furthermore, these properties should not be reordered once written.

### List(n)

List(n) is used to represent vector data types and for this reason, the individual items in the list cannot be considered logically separate. Consequently a list(n) property is stored with list items separated by ‘;’.

<avm:Spatial.ReferenceValue>400;280</avm:Spatial.ReferenceValue>

## Data types

The definition for the AOI metadata contains several different data types. These data types are listed in Table 3 along with a few examples.

Table Data types

|  |  |  |
| --- | --- | --- |
| Data type | Description | Examples |
| String | Any sequence of characters. | Don’t Panic |
| Integer | A signed integer number in base 10. | 42 -4 |
| Float | A signed decimal number in base 10. This data type may use scientific notation in its representation. | 2.005e-3 -10.374 |
| Uri | A generic URI . | ivo://esa.heic/heic0503b |
| Url | A regular web address. | http://spacetelescope.org |
| Date | A date and optionally a time (UTC) in the form YYYY-MM-DD-HHMM. | 2005-07-29 2000-12-23-1205 |

It is allowed to add whitespace in the form of spaces, tabulators, line feeds, and carriage returns both before and after the values. This should, however, be avoided for strings and URIs so that there is no doubt about which characters are part of the value and which characters are whitespace. Table 4 in Appendix D contains proposed regular expressions that can be used to validate each data type.

Please note that this document only covers how to encode data types, not how to interpret the data. This means that a property value may have additional restrictions. In particular the URI style properties have very specific rules for correct values, even if they all appear to be generic URIs.

### String

The string data type requires no metadata about its value to ensure proper interpretation. For this reason a string is just stored as-is in the property element, for example:

<avm:Title>Don’t Panic</avm:Title>

It is also possible to specify a language for the string:

<avm:Title xml:lang="en">Don’t Panic</avm:Title>

With the proper entity substitutions (see Table 2) any character string is valid value subject to the restrictions imposed by the selected character encoding.

### Integer

Integers are encoded as strings using base 10. Valid values include the full integer range from -∞ to ∞. The sign is either the ASCII minus (character code 0x2D) or an optional ASCII plus with (character code 0x2B):

<avm:Example>42</avm:Example>

<avm:Example>-4</avm:Example>

<avm:Example>+10</avm:Example>

### Float

The float data type extends the integer data type to allow full decimal values. The values are encoded as strings using base scientific notation. The decimal separator is the ASCII period (character code 0x2E) and the exponent is indicated with a lower- or upper-case e:

<avm:Example>2.005e-3</avm:Example>

<avm:Example>-10.374</avm:Example>

### Uri, Url

Like strings URIs and URLs require no extra metadata and are therefore encoded as-is:

<avm:PublisherID>ivo://nasa.opo</avm:PublisherID>

<avm:PublisherURL>http://hubblesite.org</avm:PublisherURL>

The element value must be a valid URI or URL. Each property of these types may have additional restrictions on the allowed values.

### Date

AOI dates are standard Gregorian calendar dates with a time of day in 24 hour format. The time of day is always specified in the UT time zone. Dates are encoded as strings in the format specified in the AOI metadata specification [4] (YYYY-MM-DD-hhmm), where YYYY is a four digit year, MM is as two digit month (1-12), and DD is a two digit date (1-31). The remaining hhmm part is the time of day, where hh is the hours (0-23), and mm is the minutes (0-59). The separator is the ASCII minus (character code 0x2D).

<avm:Date>2005-07-29</avm:Date>

<avm:Date>2005-07-29-1246</avm:Date>

## Encoding AOI properties

Each property in the AOI metadata specification [4] is enclosed in a unique XML element. In general the property name is the name of the enclosing XML element.

Table Default XML namespaces

|  |  |
| --- | --- |
| Prefix | Namespace |
| rdf | http://www.w3.org/1999/02/22-rdf-syntax-ns# |
| avm | <http://www.communicatingastronomy.org/avm/1.0/> |
| dc | <http://purl.org/dc/elements/1.1/> |
| Iptc4xmpCore | <http://iptc.org/std/Iptc4xmpCore/1.0/xmlns/> |
| photoshop | <http://ns.adobe.com/photoshop/1.0/> |
| xmpRights | <http://ns.adobe.com/xap/1.0/rights/> |

Table Keyword storage

|  |  |  |  |
| --- | --- | --- | --- |
| Keyword | Namespace | Path | Type |
|  |  |  |  |

# Merging AOI metadata

Astronomical images received from telescopes, probes, or satellites typically come in a raw grayscale form. This grayscale image is usually captured while applying a color filter so that the image represents the intensities of light in a specific wavelength. Grayscale images like that are called exposures and to produce a color image several exposures are needed – each being captured using a different color filter.

The workflow in producing rich outreach imagery therefore revolves around combining these raw exposures into a color image while enhancing the details of the image that best conveys the new discovery which prompted the image to be created. Each exposure contains a lot of metadata and the image processing applied to the exposure would ideally also leave metadata in the exposure so that others might reproduce the final image. When combining exposures to form color images it is therefore vital to retain as much of the existing metadata from each exposure in the final image. This section describes how metadata should be merged to form the metadata for the final image and in particular how the merged data is encoded as XML.

## Value types

The AOI metadata is stored in four distinct metadata standards using several different value types. From the XML point of view, however, there are really only four different kinds of values:

Simple These include strings, numbers, vectors, and dates. The simple values contain no structural information.

Set A set of values is an unordered collection of simple values which together form the value of a specific metadata property.

List A list is an ordered collection of simple values. Lists are used when a specific value in the list relates to some subset of the described image – in AOI metadata lists are used to describe information about a single exposure.

Structured A structured value is one where the value consists of several sub-properties each having its own simple value.

For each type of metadata value there is a special way of merging it with values from other exposures.

### Simple values

While it may be possible to devise a way to merge some simple value properties merging is not performed on simple values. IIf only one exposure contains a value for a given property that value is used in the merged metadata, but if each exposure has a different value for the same property a single value is selected and put into the merged set of metadata.

### Set

For set type values the merged value can be determined by starting with an empty set and adding the value(s) from each exposure to the merged set. When performing the merge regular set operations should be employed so that the merged does not contain duplicate items.

### List

## Special considerations

Spatial.\* keywords are special

1. References
2. *XML Schema,* W3C Recommendation, 2 May 2001, <http://www.w3.org/TR/xmlschema-1/>
3. *Resource Description Framework (RDF) Model and Syntax Specification,* W3C Recommendation, 22 February 1999, <http://www.w3.org/TR/REC-rdf-syntax>
4. *XMP Specification,* Adobe Systems Incorporated, January 2004, <http://www.adobe.com/products/xmp/>
5. *Proposal for Astronomical Visual Metadata Tags for the Virtual Observatory – Version 1.0,* Robert Hurt, Lars Lindberg Christensen, October 2006, <http://www.communicatingastronomy.org/avm/1.0/>
6. *RFC 2396 - Uniform Resource Identifiers (URI): Generic Syntax,* Berners-Lee T., Fielding R., Masinter L., IETF, August 1998, <http://www.isi.edu/in-notes/rfc2396.txt>
7. Full example

The XML in Listing 1 shows what the ESA/ESO/NASA Photoshop FITS Liberator might output to Photoshop when you open a FITS file. This example contains the entire XMP envelope that is required to comply with the XMP specification, but excludes the recommended padding between </x:xmpmeta> and the <?xpacket end="w"?> processing instruction.

The example has yet to be updated; the example from the Virtual Repository specification [4] should be used instead.

Listing Full example

<?xpacket begin="ï»¿" id="W5M0MpCehiHzreSzNTczkc9d"?>

<x:xmpmeta xmlns:x="adobe:ns:meta/" x:xmptk="3.1.1-111">

<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">

<rdf:Description

rdf:about=""

xmlns:vr="<http://www.communicatingastronomy.org/aoi/1.1/>"

>

<aoi:

</rdf:Description>

</rdf:RDF>

</x:xmpmeta>

<?xpacket end="w"?>

1. Validating property values

To validate the data provided by users or other software packages a clear definition of how these data types is represented in string form is needed. Table 4 lists the regular expressions that are used internally in FITS Liberator to perform validation on the values entered by the user. The regular expressions use the operators listed in Table 5. Because of the XML format the actual data may have additional surrounding white space. In addition to validating the format of a value, the value itself may need to be validated. This is particularly true for dates and properties whose valid values are from a controlled vocabulary.

Table Data type formats[[1]](#footnote-2)

|  |  |
| --- | --- |
| Data type | Format |
| Integer | ^[-+]?[1-9]\d\*$ |
| Float | ^([-+]?[1-9]\d\*)(\.[1-9]\d\*)?([eE][-+][1-9]\d\*)?$ |
| Uri, Url | ^[0-9a-zA-Z\.]+://[0-9a-zA-Z\./]+$ |
| Date[[2]](#footnote-3) | ^(\d{4})\-(\d{2})\-(\d{2})(\-(\d{4}))?$ |

Table Regular expression operators

|  |  |
| --- | --- |
| Operator | Description |
| ^ | Match the start of a string |
| $ | Match the end of a string |
| . | Match any character |
| | | Alternation |
| {n} | Match exactly *n* times |
| [] | Character class |
| () | Grouping |
| \* | Match 0 or more times |
| + | Match 1 or more times |
| ? | Match 0 or 1 times |
| \ | Escape the following character |
| - | Range |
| \a | Letters |
| \s | Space |
| \d | Digits |

1. Note that the string data type does not need to be validated as any value is considered valid. [↑](#footnote-ref-2)
2. The listed expression only tests whether the format of the date is correct. An implementation should also check that the value represents a valid date. [↑](#footnote-ref-3)