INTERNATIONAL STANDARD

ISO/ASTM 52915

Second edition
2016-02-15

Specification for Additive Manufacturing File Format (AMF) Version 1.2

Spécification normalisée pour le format de fichier pour la fabrication additive (AMF) Version 1.2

INTERNATIONAL

Reference number
ISO/ASTM 52915:2016(E)

Contents

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO, also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electrotechnical standardization.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/TC 261, *Additive manufacturing*, in cooperation with ASTM F 42.91, *Terminology*, on the basis of a partnership agreement between ISO and ASTM International with the aim to create a common set of ISO/ASTM standards on Additive Manufacturing.

This second edition cancels and replaces the first edition (ISO/ASTM 52915:2013), which has been technically revised. This revision contains changes to normative language and details of a minimum implementation, as well as corrections and clarifications.

Introduction

This International Standard describes an interchange format to address the current and future needs of additive manufacturing technology. For the last three decades, the stereolithography (STL) file format has been the industry standard for transferring information between design programs and additive manufacturing equipment. An STL file defines only a surface mesh and has no provisions for representing colour, texture, material, substructure and other properties of the fabricated object. As additive manufacturing technology is evolving quickly from producing primarily single-material, homogeneous objects to producing geometries in full colour with functionally-defined gradations of materials and microstructures, there is a growing need for a standard interchange file format that can support these features.

The Additive Manufacturing File Format (AMF) has many benefits. It describes an object in such a general way that any machine can build it to the best of its ability, and as such is technology independent. It is easy to implement and understand, scalable and has good performance. Crucially, it is both backwards compatible, allowing any existing STL file to be converted, and future compatible, allowing new features to be added as advances in technology warrant.

Specification for Additive Manufacturing File Format (AMF) Version 1.2

1 Scope

This International Standard provides the specification for the Additive Manufacturing File Format (AMF), an interchange format to address the current and future needs of additive manufacturing technology.

The AMF may be prepared, displayed and transmitted provided the requirements of this specification are met. When prepared in a structured electronic format, strict adherence to an extensible markup language (XML)[1] schema is required to support standards-compliant interoperability.

A W3C XML schema definition (XSD) for the AMF is available from ISO from http://standards.iso.org/iso/52915 and from ASTM from www.astm.org/MEETINGS/images/amf.xsd. An implementation guide for such an XML schema is provided in Annex A.

It is recognized that there is additional information relevant to the final part that is not covered by the current version of this International Standard. Suggested future features are listed in Annex B.

This International Standard does not specify any explicit mechanisms for ensuring data integrity, electronic signatures and encryptions.

2 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

2.1
AMF consumer
software reading (parsing) the Additive Manufacturing File Format (AMF) file for fabrication, visualization or analysis

Note 1 to entry: AMF files are typically imported by additive manufacturing equipment, as well as viewing, analysis and verification software.

2.2
AMF editor
software reading and rewriting the Additive Manufacturing File Format (AMF) file for conversion

Note 1 to entry: AMF editor applications are used to convert an AMF from one form to another, for example, convert all curved triangles to flat triangles or convert porous material specification into an explicit mesh surface.

2.3
AMF producer
software writing (generating) the Additive Manufacturing File Format (AMF) file from original geometric data

Note 1 to entry: AMF files are typically exported by computer-aided design (CAD) software, scanning software or directly from computational geometry algorithms.

2.4
attribute
characteristic of data, representing one or more aspects or descriptors of the data in an element

Note 1 to entry: In the XML framework, attributes are characteristics of elements.

2.5
comments
all text elements associated with any data within the Additive Manufacturing File Format (AMF) to be ignored by import software

Note 1 to entry: Comments are used for enhancing human readability of the file and for debugging purposes.

2.6
element
information unit within an XML document consisting of a start tag, an end tag, the content between the tags and any attributes

Note 1 to entry: In the XML framework, an element can contain data, attributes and other elements.

2.7
extensible markup language
XML
standard from the WorldWideWeb Consortium (W3C) that provides for tagging of information content within documents offering a means for representation of content in a format that is both human and machine readable

Note 1 to entry: Through the use of customizable style sheets and schemas, information can be represented in a uniform way, allowing for interchange of both content (data) and format (metadata).

[SOURCE: ISO/ASTM 52900:2015, 2.4.7]

2.8
STL
stereolithography
file format for model data describing the surface geometry of an object as a tessellation of triangles used to communicate 3D geometries to machines in order to build physical parts

Note 1 to entry: The STL file format was originally developed as part of the CAD package for the early stereolithography apparatus, thus referring to that process. It is sometimes also described as "Standard Triangulation Language" or "Standard Tessalation Language", though it has never been recognized as an official standard by any standardization organization.

[SOURCE: ISO/ASTM 52900:2015, 2.4.16]

3 Key considerations

3.1 General

3.1.1 There is a natural trade-off between the generality of a file format and its usefulness for a specific purpose. Thus, features designed to meet the needs of one community may hinder the usefulness of a file format for other uses. To be successful across the field of additive manufacturing, the file format described in this International Standard, the AMF, is designed to address the concerns listed in 3.1.2 to 3.1.7.

3.1.2 Technology independence. The AMF describes an object in such a general way that any machine can build it to the best of its ability. It is resolution and layer-thickness independent and does not contain information specific to any one manufacturing process or technique. This does not negate the inclusion of features that describe capabilities that only certain advanced machines support (for example, colour, multiple materials), but these are defined in such a way as to avoid exclusivity.

3.1.3 Simplicity. The AMF is easy to implement and understand. The format can be read and debugged in a simple text viewer to encourage comprehension and adoption. Identical information is not stored in multiple places.

3.1.4 Scalability. The file size and processing time scales well with the increase in part complexity and with the improving resolution and accuracy of manufacturing equipment. This includes being able to handle large arrays of identical objects, complex periodic internal features (for example, meshes and lattices) and smooth curved surfaces when fabricated with very high resolution.

3.1.5 Performance. The AMF enables reasonable duration (interactive time) for read-and-write operations and reasonable file sizes for a typical large object. Detailed performance data are provided in Annex B.

3.1.6 Backwards compatibility. Any existing STL file can be converted directly into a valid AMF file without any loss of information and without requiring any additional information. AMF files are also easily converted back to STL for use on legacy systems, although advanced features will be lost. This format maintains the triangle-mesh geometry representation to take advantage of existing optimized slicing algorithms and code infrastructure already in existence.

3.1.7 Future compatibility. To remain useful in a rapidly changing industry, this file format is easily extensible while remaining compatible with earlier versions and technologies. This allows new features to be added as advances in technology warrant, while still working flawlessly for simple homogeneous geometries on the oldest hardware.

3.2 Guidelines for the inclusion of future new elements

3.2.1 Any new element proposed shall be applicable across all hardware platforms and technologies that could conceivably be used to generate the desired outcome.

3.2.2 In support of the consideration above, new elements proposed for this International Standard shall describe the final object, not how to build it. For instance, a hypothetical future element `<hollow>` might be allowed to tell an additive manufacturing system to leave the volume empty if possible. However, an element `<objectLayerFillPath>` that describes how to build a hollow volume shall not be included since it assumes a particular fabrication process.

4 Structure of this specification

4.1 Format. Information specified throughout this specification is stored in XML 1.0 format. XML is a text file comprising a list of elements and attributes. Using this widely accepted data format allows for the use of many tools for creating, viewing, manipulating, parsing and storing AMF files. XML is human-readable, which makes debugging errors in the file possible. XML can be compressed or encrypted or both if desired in a post-processing step using highly optimized standardized routines.

4.2 Flexibility. Another significant advantage of XML is its inherent flexibility. Missing or additional parameters do not present a problem for a parser as long as the document conforms to the XML standard. Practically, the use of XML namespaces allows new features to be added without breaking old versions of the parser, such as in legacy software.

4.3 Precision. This file format is agnostic as to the precision of the representation of numeric values. It is the responsibility of the generating program to write as many or as few digits as are necessary for proper representation of the target object. However, an AMF consumer should read and process real numbers in double precision (64 bits).

4.4 Future amendments and additions. While additional XML elements can be added provisionally to any AMF file for internal purpose, such additions shall not be considered part of this specification. An unofficial AMF element may be ignored by any AMF consumer and may not be stored or reproduced by an editor application. An element becomes official only when it is formally accepted into this specification.

5 General structure

5.1 The AMF file shall begin with the XML declaration line specifying the XML version and encoding, for example:

<?xml version="1.0" encoding="UTF-8"?>

The XML version shall be 1.0. Only UTF-8 and UTF-16 should be specified. Unrecognized encodings should cause the file to fail to load.

5.2 Whitespace characters and standard XML comments may be interspersed in the file and shall be ignored by any interpreter, for example:

<!-- ignore this comment -->

5.3 The remainder of the file shall be enclosed between start `</amf>` and end `</amf>` element tags. This element denotes the file type and fulfils the requirement that all XML files have a single root element. A version attribute denoting the version of the AMF standard the file is compliant with should be used. Standard XML namespace attributes may also be used, such as the lang attribute designed to identify the natural human language used. The unit system may also be specified (millimetre, inch, foot, metre or micron). In the absence of a unit specification, the attribute value millimetres is assumed, for example:

<amf unit="millimeter" version="1.0" xml:lang="en"

xmins:amf="www.astm.org/Standards/F2915-14">

5.4 Enclosed within the `<amf/>` element start- and end-tags, there are five top level elements, as described in 5.4.1 to 5.4.5.

5.4.1 `<object>` The object element defines a volume or volumes of material, each of which might also reference a material identifier (ID) for AM processing. The object element shall also declare an object ID, which shall be unique. At least one object element shall be present in the file. Additional objects are optional.

5.4.2 `<material>` The optional material element defines one material for fabrication, each of which declares an associated material ID. The material ID declared shall be unique and shall not be 0. If no material element is included, a single default material is assumed.

5.4.3 `<texture>` The optional texture element defines one image or texture for colour or texture mapping, each of which declares an associated texture ID. The texture ID declared shall be unique.

5.4.4 `<constellation>` The optional constellation element hierarchically combines objects and other constellations into a relative pattern for printing. The constellation element may also declare an object ID, which shall be unique. If no constellation elements are specified, each object element shall be imported with no relative position data. The consumer software may determine the relative positioning of the objects if more than one object is specified in the file.

5.4.5 `<metadata>` The optional metadata element specifies additional information about the object(s) and elements contained in the file.

5.5 Only a single object element is required for a fully functional AMF file.

6 Geometry specification

6.1 General

6.1.1 The top level `<object>` element declares a unique ID and shall contain one child `<mesh>` element. The `<mesh>` element shall contain two child elements: `<vertices>` and `<volume>`. The `<object>` element may optionally reference a material.

6.1.2 The required `<vertices>` element shall contain all vertices that are used in this object. Each vertex is implicitly assigned an identifying integer in the order in which it is declared, starting at zero and increasing monotonically. The required child element `<coordinates>` gives the position of the vertex in three-dimensional (3D) space using the `<x>`, `<y>` and `<z>` child elements.

6.1.3 After the vertex information, at least one `<volume>` element shall be included. Each volume encapsulates a closed volume of the object. Multiple volumes may be included in a single object. Volumes may share vertices at interfaces but shall not have any overlapping volume.

6.1.4 Within each volume, multiple child `<triangle>` elements shall be used to define the triangles that tessellate the surface of the volume. Each `<triangle>` element shall reference three vertices from the set of indices of the previously defined vertices. The indices of the three vertices of the triangles shall be specified using the `<v1>`, `<v2>` and `<v3>` child elements. The vertices shall be ordered according to the right-hand rule such that vertices are listed in counter-clockwise order as viewed from the outside of the volume. Each triangle is implicitly assigned an identifying integer in the order in which it was declared starting at zero and increasing monotonically (see Figure 1).

6.1.5 The geometry shall not be used to describe support structure. Only the final target structure shall be described.

```
<?xml version="1.0" encoding="UTF-8"?>
<amf unit="millimeter">
  <object id="0">
    <mesh>
      <vertices>
        <vertex>
          <coordinates>
            <x>0</x>
            <y>1.32</y>
            <z>3.715</z>
          </coordinates>
        </vertex>
        <vertex>
          <coordinates>
            <x>0</x>
            <y>1.269</y>
            <z>2.45354</z>
          </coordinates>
        </vertex>
        ...
      </vertices>

      <volume>
        <triangle>
          <v1>0</v1>
          <v2>1</v2>
          <v3>3</v3>
        </triangle>
        <triangle>
          <v1>1</v1>
          <v2>0</v2>
          <v3>4</v3>
        </triangle>
        ...
      </volume>
    </mesh>
  </object>
</amf>
```

NOTE The figure shows a basic AMF file containing only a list of vertices and triangles. This structure is compatible with the STL standard and can be readable by a minimal implementation of an AMF consumer.

Figure 1 — Basic AMF file

6.2 Smooth geometry

6.2.1 By default, all triangles shall be assumed to be flat and all triangle edges shall be assumed to be straight lines connecting their two vertices. However, curved triangles and curved edges may optionally be specified to reduce the number of mesh elements required to describe a curved surface. Minimal AMF consumer software (see Clause 13) may ignore curvature information associated with triangles.

6.2.2 During import, a curved triangle patch shall be recursively subdivided into four triangles to generate a final temporary set of flat triangles. The depth of recursion shall be exactly five (that is, a single curved triangle will be converted into 1 024 flat triangles).

6.2.3 During production, the producing software that generates curved triangles shall determine automatically the number of curved triangles required to specify the target geometry to the desired tolerance, knowing that the consuming software will perform five levels of subdivision for any curved triangle.

6.2.4 To specify curvature, a vertex may contain a child element `<normal>` to specify the desired surface normal at the vertex. The normal should be unit length and pointing outwards. If this normal is specified, all triangle edges meeting at that vertex shall be curved so that they are perpendicular to that normal and in the plane defined by the normal and the original straight edge.

6.2.5 If a vertex is referenced by two volumes, the normal is considered identically for each volume, but its direction should be interpreted as consistent with the volume in consideration (so that it is pointing outwards). Vertices that have an ambiguous normal because they are common to multiple volumes should not specify a normal.

6.2.6 A curved triangle shall not be more than 25 % out of plane and shall not include inflections.

6.2.7 When the curvature of the volume's surface at a vertex is undefined (for example, at a cusp, corner or edge), an `<edge>` element may be used to specify the curvature of a single nonlinear edge joining two vertices. The curvature is specified using the tangent direction vectors at the beginning and end of that edge. The `<edge>` element shall take precedence in case of a conflict with the curvature implied by a `<normal>` element.

6.2.8 Normals should not be specified for vertices referenced only by planar triangles. Edge elements should not be specified for linear edges in flat triangles.

6.2.9 When interpreting normal and tangents, second degree Hermite interpolation shall be used. See A.3 for formulae for carrying out this interpolation.

6.3 Restrictions on geometry

All geometry shall comply with the following restrictions.

— Every triangle shall have exactly three different non-co-linear vertices.

— Triangles shall not intersect or overlap except at their common edges or common vertices.

— Volumes shall enclose a closed contiguous space with non-zero volume.

— Volumes shall not overlap.

— Every vertex shall be referenced by at least three triangles.

— Every pair of vertices shall be referenced by exactly zero or two triangles per volume.

— Any two vertices shall not have identical coordinates. The tolerance used to define equality is 10^{-8} units.

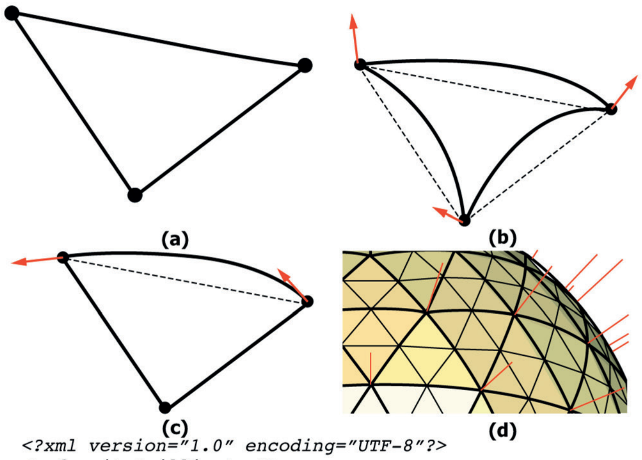
— The outward direction of triangles that share an edge in the same volume shall be consistent. The outward direction is defined by the order of vertices.

7 Material specification

7.1 General

7.1.1 Materials are introduced using the optional `<material>` element. Any number of materials may be defined using one `<material>` element for each. Each material is assigned a unique ID. Geometric volumes may specify a material ID attribute value on the `<volume>` element that references a material. The material ID "0" is reserved to represent no selected material (void), see Figure 2.

7.1.2 Material characteristics are contained within each `<material>` element. The child element `<colour>` is used to specify the red/green/blue/alpha (RGBA) appearance of the material (see Clause 8 on colour). Additional material properties may be specified using the `<metadata>` element, such as the material name for operational purposes or elastic properties for equipment that can control such properties, see Figure 3. See A.1 for a description of the AMF elements.



```
<?xml version="1.0" encoding="UTF-8"?>
<amf unit="millimeter">
  <object id="0">
    <mesh>
      <vertices>
        <vertex>
          <coordinates >
            ...
          </coordinates >
          <normal>
            <nx>0</nx>
            <ny>0.707</ny>
            <nz>0.707</nz>
          </normal>
        </vertex>
        ...
        <edge>
          <v1>0</v1>
          <dx1>0.577</dx1>
          <dy1>0.577</dy1>
          <dz1>0.577</dz1>
          <v2>1</v2>
          <dx2>0.707</dx2>
          <dy2>0</dy2>
          <dz2>0.707</dz2>
        </edge>
        ...
      </vertices>

      <volume materialid="0">
        <triangle>
          ...
        </triangle>
        ...
      </volume>
    </mesh>
  </object>
</amf>
```

(e)

Key

(a) default (flat) triangle patch

(b) triangle curved using vertex normals

(c) triangle curved using edge tangents

(d) subdivision of a curved triangle patch into four curved subpatches

(e) AMF file containing curved geometry

Figure 2 — Types of triangles used in a mesh

```
<?xml version="1.0" encoding="UTF-8"?>
<amf unit="millimeter">
  <material id="1">
    <metadata type="Name">StiffMaterial</metadata>
  </material>
  <material id="2">
    <metadata type="Name">FlexibleMaterial</metadata>
  </material>
  <material id="3">
    <metadata type="Name">MediumMaterial</metadata>
    <composite materialid="1">0.4</composite>
    <composite materialid="2">0.6</composite>
  </material>
  <material id="4">
    <metadata type="Name">VerticallyGraded</metadata>
    <composite materialid="1">z</composite>
    <composite materialid="2">10-z</composite>
  </material>
  <material id="5">
    <metadata type="Name">Checkerboard</metadata>
    <composite materialid="1">
          floor(mod(x+y+z,1))+0.5) </composite>
    <composite materialid="2">
          1-floor(mod(x+y+z,1))+0.5) </composite>
  </material>

  <object id="0">
    <mesh>
      <vertices>
        ...
      </vertices>
      <volume materialid="1">
        ...
      </volume>
      <volume materialid="2">
        ...
      </volume>
    </mesh>
  </object>
</amf>
```

NOTE The figure shows an AMF file containing five materials. Material 3 is a 40/60 % homogeneous mixture of the first two materials. Material 4 is a vertically graded material and Material 5 has a periodic checkerboard substructure.

Figure 3 — Homogeneous and composite materials

7.2 Mixed and graded materials and substructures

7.2.1 New materials can be defined as compositions of other materials. The element `<composite>` is used to specify the proportions of the composition as a constant or a formula dependent of the *x*, *y* and *z* coordinates. A constant mixing proportion will lead to a homogeneous material. A coordinate dependent composition can lead to a graded material. More complex coordinate-dependent proportions can lead to nonlinear material gradients as well as periodic and non-periodic substructure. The proportion formula can also refer to a texture map using the tex (`textureid,x,y,z`) function (see A.1).

7.2.2 Any number of materials may be used within a composite.

7.2.3 Any negative material proportion value shall be interpreted as a zero proportion. Material proportions shall be normalized to a sum of 1.0 to determine actual ratios.

7.3 Porous materials

7.3.1 Reference to `materialid` "0" (void) may be used to specify porous structures. The proportion of void shall be either 0 or 1. Any fractional shall be interpreted as 1 (that is, any fractional void shall be treated as 0 or fully void).

7.3.2 Although the `<composite>` element could theoretically be used to describe the complete geometry of an object as a single function or texture with reference to void, producers shall not do this (see B.2.5 and B.2.6). The intended use of the `<composite>` element with reference to void is to describe cellular mesostructures.

7.4 Stochastic materials

Reference to the `rand(x,y,z)` function may be used to specify pseudo-random materials. For example, a composite material could combine two base materials in random proportions in which the exact proportion might depend on the coordinates in various ways. The `rand(x,y,z)` function produces a random floating point scalar in the range [0,1] that is persistent across function calls (see A.4).

8 Colour specification

8.1 General

8.1.1 Colours may be introduced using the `<colour>` element by specifying the RGBA (transparency) values in a specified colour space. By default, the colour space shall be sRGB[2] but alternative profiles may be specified using the metadata tag in the root `<amf>` element (see A.1). The `<colour>` element may be a `child` of the `<material>` element to associate a colour with a material, the `<object>` element to colour an entire object, the `<volume>` element to colour an entire volume, the `<triangle>` element to colour a triangle or the `<vertex>` element to associate a colour with a particular vertex (see Figure 4).

8.1.2 If no colour is specified, the default colour is white.

8.1.3 Object colour overrides material colour specification; a volume colour overrides an object and material colours; vertex colours override volume, object and material colours; and triangle colouring overrides vertex, object and material colours.

```
<?xml version="1.0" encoding="UTF-8"?>
<amf unit="millimeter">

  <material id="1">
    <metadata type="Name">StiffMaterial</metadata>
      <color>
        <r>0</r>
        <g>z</g>
        <b>1-z</b>
      </color>
  </material>

  <texture id="1" width="10" height="26" type="grayscale">
    TWFuIGlzIGRpc3RpbmdlaXNoZWQsIG5vdCB
    vbmx5IGJ5IGhpcyByZWFzb24sIGJ1dCBieS
    B0aGlzIHNpbmdlbGFyIHBhc3Npb24gZnJvb
    SBvdGhlcIBhbmltYWxzLCB3aGljaCBpcyBh
    ...
  </texture>

  <object id="0">
    <mesh>
      <vertices>
        ...
      </vertices>
      <volume materialid="1">
        <color>
          <r>0.9</r>
          <g>0.9</g>
          <b>0.2</b>
          <a>0.8</a>
        </color>
        ...
        <triangle>
          <v1>0</v1>
          <v2>1</v2>
          <v3>3</v3>
          <texmap rtexid="1" gtexid="2" btexid="3">
            <utex1>0.1</utex1>
            <utex2>0.21</utex2>
            <utex3>0.15</utex3>
            <vtex1>0.65</vtex1>
            <vtex2>0.72</vtex2>
            <vtex3>0.91<1vtex3>
          </texmap>
        </triangle>
      </volume>
    </mesh>
  </object>
</amf>
```

NOTE A solid colour may be associated with a material, a volume or a vertex. A vertex may also be associated with a coordinate in a colour texture file.

Figure 4 — Colour specification

8.2 Colour gradations and texture mapping

8.2.1 A colour may also be specified by referring to a formula that might use a variety of functions, including a texture map function.

8.2.2 When referring to a formula, the `<colour>` element may specify a colour that depends on the coordinates such as a colour gradation. Any mathematical expression that combines the functions described in A.2 may be used. For example, use of the rand function would allow for pseudorandom colour patterns. The tex function would allow the colour to depend on a texture map or image. To specify a full-colour graphic, typically three textures would be needed, one for each colour channel. To create a monochrome graphic, one texture is typically sufficient.

8.2.3 When the vertices of a single triangle have different colours, the interior colour of the triangle shall be linearly interpolated between those colours, unless a triangle colour has been explicitly specified (because a triangle colour takes precedence over a vertex colour). If all three vertices of a triangle contain a mapping to the same texture ID for any channel (*r*, *g*, *b* or *a*), the colour of this channel of the triangle shall be extracted from the texture map, overriding the triangle colour.

8.3 Transparency

The transparency channel `<a>` determines alpha compositing for combining the specified foreground colour with a background colour to create the appearance of partial transparency. A value of 0 specifies zero transparency, that is, only the foreground colour is used. A value of 1 specifies full transparency, that is, only the background colour is used. Intermediate values shall be linearly interpolated between the background and foreground colours. Negative values are rounded to 0 and values greater than one are truncated to 1. The background colour of a triangle shall be the vertex colour, then volume colour, then object colour, then material colour in decreasing precedence. The background colour of a vertex shall be the volume colour, then object colour, then material colour in decreasing precedence. The background colour of a volume shall be the object colour, then material colour in decreasing precedence. The background colour of an object shall be the material colour.

9 Texture specification

9.1 The `<texture>` element is used to associate a textureid with particular texture data. The texture map size shall be specified and both two-dimensional (2D) and three-dimensional (3D) textures are supported. The data shall be represented as a series of grayscale values in the range [0,255]. Each value is stored in one byte and encoded in Base64. The ordering of pixel data shall be consistent with the texture map reference coordinate.

9.2 The producer shall ensure that the amount of data matches the specified size of the texture. If the amount of data is excessive, the consumer shall truncate it. If the amount of data is too low, the consumer shall append the data with 0 values as needed to meet the specified texture size.

9.3 In order to map a texture onto a triangle, the `<texmap>` element may be used to define *u*, *v* and (optionally) *w* coordinates for each vertex of this triangle. If the texture's "tiled" property is "true", any *u,v,w* mappings outside of the [0,1] range shall be determined according to the coordinate modulo 1. If the texture's tiled property is not "true", any mappings that fall outside of the [0,1] range shall return zero.

9.4 Textures shall be linearly interpolated for each triangle. A triangle shall include only a single `<texmap>` element. Overlapping textures shall be combined by the producer into a single texture before being mapped onto a mesh.

10 Constellations

10.1 Multiple objects may be arranged together using the `<constellation>` element (see Figure 5). A constellation may specify the position and orientation of objects to increase packing efficiency and describe large arrays of identical objects. The `<instance>` element specifies the displacement and rotation that an existing object shall be transformed to position it in the constellation. The displacement and rotation shall be defined relative to the original position and orientation of the object when it was originally defined. Rotation angles shall be specified in degrees and shall first apply rotations about the *x* axis, then the *y* axis and then the *z* axis.

10.2 A constellation may include another constellation, with multiple levels of hierarchy. However, cyclic definitions of constellations shall not be used.

10.3 When multiple objects and constellations are defined in a single file, the position and relative orientation of only the direct children objects and constellations of the `<amf>` element may be optimized by the consuming software.

10.4 When interpreting orientation, the *z* axis shall be assumed to be the vertical axis, with the positive direction pointing upwards and zero referring to the base surface. The *x* and *y* directions shall correspond to the main build stage axes, consistent with the right-hand rule.

```
<?xml version="1.0" encoding="UTF-8"?>
<amf unit="millimeter">
  <object id="1">
        ...
  </object>
  <constellation id="2">
    <instance objectid="1">
      <deltay>5</deltay>
      <rz>90</rz>
    </instance>
    <instance objectid="1">
      <deltax>-10</deltay>
      <deltay>10</deltay>
      <rz>180</rz>
    </instance>
    ...
  </constellation>

</amf>
```

NOTE A constellation may assemble multiple objects together.

Figure 5 — Constellations

11 Metadata

The `<metadata>` element may optionally be used to specify additional information about the objects, geometries and materials being defined (see Figure 6). For example, this information might specify a name, textual description, authorship, copyright information and special instructions. The `<metadata>` element may be included as a child of the `<amf>` element to specify attributes of the entire file or as a child element of objects, volumes and materials to specify metadata local to that entity. Reserved metadata types and their meaning are listed in A.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<amf unit="millimeter">
  <metadata type="description">Product 123</metadata>
  <metadata type="author">John Smith</metadata>
  <metadata type="cad">SolidX 2.2</metadata>
  <metadata type="name">Part 1</metadata>
  <metadata type="revision">1.3A</metadata>
      ...
  <object ObjectID="0">
    <metadata type="name">Component 1</metadata>
      ...
  </object>
</amf>
```

NOTE Additional information can be stored about the object using the metadata element.

Figure 6 — Metadata

Custom metadata types should use a different namespace, for example:

<xamf:metadata type=“myattribute”>mydata</xamf:metadata>

12 Compression and distribution

12.1 An AMF shall be stored either in plain text or be compressed. If compressed, the compression shall be in a ZIP archive format[3] and can be done by a user or at write time by an application, using any one of several open compression libraries, such as Reference [4].

12.2 Both the compressed and uncompressed version of the file shall have the AMF extension and the consuming software shall determine whether or not the file is compressed and, if so, to perform decompression during read. Any file that does not begin with an `<?xml>` tag shall be interpreted as a compressed xml file.

12.3 Additionally, other files may be included in the ZIP archive, such as manifest files or electronic signatures. However, only the AMF file with the same name as the archive file will be parsed. Absence of a file with that name shall constitute an error.

13 Minimal implementation

13.1 A minimal implementation of an AMF producer shall be able to generate a compressed file with a single object and no colour, material, texture, constellations or metadata. The single object will comprise a mesh containing only one volume.

13.2 A minimal implementation of an AMF consumer shall be able to parse a compressed file with one object and ignore any colour, material, texture, constellations or metadata information.

Annex A
(informative)

AMF XML schema implementation guide

A.1 AMF elements

Table A.1 lists the AMF elements.

Table A.1 — AMF elements

Element	Parent element(s)	Attributes	Multiple elements allowed?	Description
<amf>			No	Root XML element.
		Unit		The units to be used may be “inch”, “millimetre,” “metre”, “foot” or “micron”.
		version		The amf specification version of this file in the format of X.XXX
<object>	<amf>		Yes	An object definition.
		id		A unique integer identifying this object.
<colour>	<material> <object> <volume> <vertex> <triangle>		No	The colour to display the object in and to fabricate it in if supported.
				When conflicting colour specifications exist, precedence is given according to the order listed in the specification.
<r>, <g>, <b>, <a>	<colour>		No	Red, green, blue and alpha (transparency) component of a colour in sRGB space, floating point values ranging from 0 to 1. The values may be specified as constants or as a formula depending on the coordinates.
<mesh>	<object>		No	A 3D triangular mesh definition.
<vertices>	<mesh>		No	The list of vertices to be used in defining triangles in the mesh.
<vertex>	<vertices>		Yes	A vertex to be referenced by triangles.
<coordinates>	<vertex>		No	Specifies the 3D location of this vertex.
<x>, <y>, <z>	<coordinates>		No	*x*, *y* or z coordinate, respectively, of a vertex position in space.
<normal>	<vertex>		No	Specifies the 3D normal of the object surface at this vertex.
<edge>	<vertices>		Yes	Specifies the 3D tangent of an object edge between two vertices.
<dx1>, <dy1>, <dz1> <dx2>, <dy2>, <dz2>	<edge>		No	The normalized *x*, *y* or z component (respectively) of the first or second edge direction vector.

Table A.1 *(continued)*

Element	Parent element(s)	Attributes	Multiple elements allowed?	Description
<nx>, <ny>, <nz>	<normal>		No	The normalized *x*, *y* or z component (respectively) of a surface normal at a vertex.
<volume>	<mesh>		Yes	Defines a volume from the established vertex list.
		materialid		The ID of the material to apply to this volume.
<triangle>	<volume>		Yes	Defines a 3D triangle from three vertices, according to the right-hand rule (counter-clockwise when looking from the outside).
<v1>, <v2>, <v3>	<triangle> <edge>			Index of the vertices defining the triangle or edge.
<texture>			Yes	Specifies texture data to be used by a colour or by a tex function in a formula. Contains a sequence of Base64 encoded values specifying values for pixels of a texture.
		id		Assigns a unique texture ID for the new texture.
		width		Width (horizontal size, *x*) of the texture, in pixels.
		height		Height (lateral size, *y*) of the texture, in pixels.
		depth		Depth (vertical size, *z*) of the texture, in pixels.
		tiled		Defines if a texture should be tiled. A value of “true” implies tiling is enabled. Any other value implies no tiling whereby any texture mapping outside of the defined image shall return zero.
		type		Encoding of the data in the texture. Currently shall be “grayscale”. In grayscale mode, each pixel is represented by one byte in the range [0,255]. When the texture is referenced using the tex function, these values are converted into a single floating point number in the range [0,1] (see A.2). Full-colour graphics might typically require three textures, one for each of the colour channels. A graphic involving transparency might require a fourth channel.
<texmapl>	<triangle>		No	Maps the vertices of this triangle to (*u*,*v*,*w*) coordinates of the specified textures. If unspecified, the *w* coordinate shall be assumed to be zero.
		rtexid		The texture ID of the red channel.
		gtexid		The texture ID of the green channel.
		btexid		The texture ID of the blue channel.
		atexid		The texture ID of the alpha channel.
<utext>, <utex2>, <utex3>	<texmap>		No	Description: *u*, *v* and *w* (optional) coordinates for triangle vertices 1, 2 and 3.
<ytext1>, <ytex2>, <ytex3>				
<wtext1>, <wtex2>,				

Table A.1 *(continued)*

Element	Parent element(s)	Attributes	Multiple elements allowed?	Description
<wtex3>				
<material>	<amf>		Yes	An available material for reference by volumes or composites.
		id		A unique material ID. Material ID “0” is reserved to denote no material (void) or sacrificial material.
<composite>	<material>		Yes	Blends existing materials. Value provides a numeric constant or mathematical function of coordinates *x*, *y*, *z* specifying the proportion of material of type materialid. If the value is negative, it shall be treated as zero. The proportions shall be normalized to add up to 1, unless they are all zero, in which case no material is specified (void). A materialid value of “0” shall be treated as a reference to no material (void). The void material shall not be mixed. See A.2 for a list of allowable mathematical functions.
		materialid		Reference to an existing material. Reference shall not be recursive or cyclic.
<constellation>	<amf>		Yes	A grouping of objects or constellations with specific relative locations.
		id		The Object ID of the new constellation being defined.
<instance>	<constellation>		Yes	An instance of an object or constellation to print.
		objectid		A reference to the ID of the existing object or constellation being instantiated. Recursive or cyclic references shall not be used.
<deltax>, <deltay>, <deltaz>	<instance>		No	The distance of translation in the *x*, *y* or *z* direction, respectively, in the referenced object’s coordinate system, to create an instance of the object in the current constellation.
<rx>, <ry>, <rz>	<instance>		No	The rotation, in degrees, to rotate the referenced object about its *x*, *y* and *z* axes, respectively, to create an instance of the object in the current constellation. Rotations shall be executed in order of *x* first, then *y*, then *z*.
<metadata>	<amf> , <object>, <volume>, <material>, <vertex>		Yes	Specify additional optional information about an entity.

Table A.1 *(continued)*

Element	Parent element(s)	Attributes	Multiple elements allowed?	Description
		type		The type of the attribute. Values shall be one of the following: “name”. The alphanumeric label of the entity may be used by the interpreter when interacting with the user. “description” – A description of the content of the entity. “url” – A link to an external resource relating to the entity. “author” – Specifies the name(s) of the author(s) of the entity. “company” – Specifies the company generating the entity. “producer” – Specifies the name of the originating software and version. “revision” – Specifies the revision of the entity. “tolerance” – Specifies the desired manufacturing tolerance of the entity in entity’s unit system. “volume” – Specifies the total volume of the entity, in the entity’s unit system, to be used for verification (applies to object elements and volume elements only). “elasticmodulus” – Specifies the elastic modulus of the entity, in SI units (material only). “poissonratio” – Specifies the Poisson Ratio of the material, in SI units (material only). “colourprofile” – The ICC colour space used to interpret the three colour channels <r>, <g> and <b> . May be one of 9sRGB9, 9AdobeRGB9, 9Wide-Gamut-RGB9, 9CIERGB9, 9CIELAB9 or 9CIEXYZ9 (top level <aml> only).

A.2 Mathematical operations and functions

Table A.2 lists the mathematical operations and functions.

Formulae containing characters that are restricted in XML, such as “<” and “>”, shall be contained within a CDATA clause (i.e. start with “<![CDATA]” and end with “]]>”).

Table A.2 — Mathematical operations and functions

Precedence	Operator	Description
1	()	Parentheses block
2	^	Power
3	*	Multiply
3	/	Divide

[a] Logical operators return a Boolean value of either 1 or 0 representing TRUE and FALSE, respectively. When processing non-Boolean numbers as Boolean values, a zero value represents FALSE and a non-zero value represents TRUE.

Table A.2 *(continued)*

Precedence	Operator	Description
4	+	Add
4	–	Subtract
5	=	Equal[a]
5	<, < =	Less than (or equal to)[a]
5	>, > =	Greater than (or equal to)[a]
6	and	Intersection (Logical AND)[a]
6	or	Union (Logical OR)[a]
6	xor	Difference (Logical XOR)[a]
6	1	Negation (Logical NOT)[a]
6	mod(*a*,*b*)	Modulus, including fractional. Returns remainder after dividing *a* by *b*.
6	sin(*x*)	Sine, radians
6	cos(*x*)	Cosine, radians
6	tan(*x*)	Tangent, radians
6	asin(*x*)	Arc sine, radians
6	acos(*x*)	Arc cosine, radians
6	atan(*x*)	Arc tangent, radians
6	floor(*x*)	Round down to nearest integer
6	ceil(*x*)	Round up to nearest integer
6	sqrt(*x*)	Square root
6	ln(*x*)	Natural logarithm
6	log10(*x*)	Base 10 logarithm
6	exp(*x*)	Natural exponent
6	abs(*x*)	Absolute value
6	max(*x*,*y*)	Maximum value
6	min(*x*,*y*)	Minimum value
6	rand(*x*,*y*)	Maps the 2D or 3D coordinate onto a real (fractional) pseudo-random number uniformly distributed in the range [0,1] (exclusive of 1). The number returned shall be persistent across multiple calls (that is, the same number shall always be returned for the same coordinate). If $k = 1$, a second (probably different) number shall be returned for that coordinate. See sample implementation in A.4.

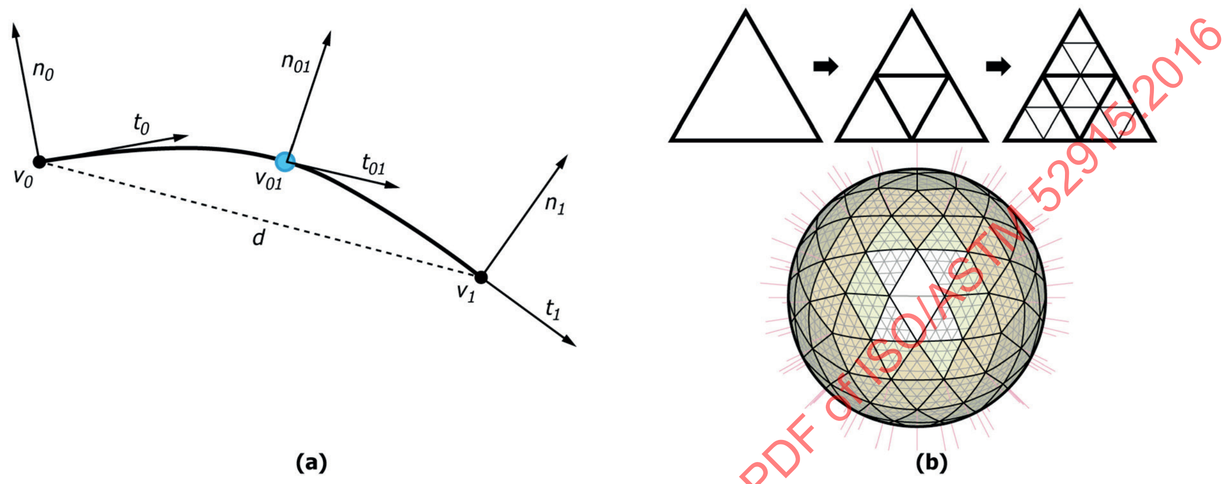
6	tex(textureid,*u*,*v*,*w*)
	tex(textureid,*u*,*v*)
		Returns a floating point scalar value in the range [0,1] that interpolates the texture with the textureid at the coordinate (*u*,*v*,*w*) for 3D textures and (*u*,*v*) for 2D textures. if the texture is of type “grayscale”, the range [0,1] corresponds to the range [0,255] in the texture data. Whole coordinate values shall refer to the centre of the texture map pixels with the first pixel having an index of 1. If the values are fractional, linear interpolation shall be used. If the coordinates fall outside a non-tiled texture, a zero value shall be returned. If the texture is 2D and a *z* coordinate is specified, the *z* coordinate shall be ignored.

[a] Logical operators return a Boolean value of either 1 or 0 representing TRUE and FALSE, respectively. When processing non-Boolean numbers as Boolean values, a zero value represents FALSE and a non-zero value represents TRUE.

A.3 Formulae for interpolating a curved triangular patch

A.3.1 Nonplanar triangular patches with specified surface normals or edge tangents shall be interpolated from their three vertices and six tangent vectors and/or three surface normals using interpolated second order Hermite curves, as described in A.3.2 to A.3.4.

A.3.2 For each of the three edges of the triangle [refer to Figure A.1(a)] follow the steps described in A.3.2.1 to A.3.2.6.



Key

(a) notation used for the subdivision of a curve

(b) triangles are divided recursively to a depth of five

NOTE The figure shows an example of a spherical surface represented with 320 triangles, each subdivided into 16 sub-triangles by using the procedure described in A.3 recursively.

Figure A.1 — Interpolating a curved triangle edge

A.3.2.1 If the normal, n_0, at point, v_0, is not specified explicitly using the `<normal>` element, compute the normal, n_0, by computing the cross product between the two edge tangents meeting at that point. For this calculation, use the edge tangents specified by the `<edge>` element, if available, from A.3.2.6 executed in a prior recursion or, if neither are available, use a straight line connecting the edge end points.

The resulting normal, n_0, should be normalized to unit length and have its sign set so that it is pointing outwards.

A.3.2.2 Repeat A.3.2.1 for the normal, n_i, at point, v_i.

A.3.2.3 If the tangent, t_0, is not specified explicitly in an `<edge>` element or a previous recursion, compute the tangent vector, t_0, such that it is perpendicular to the normal at n_0 and resides in the plane defined by that normal and the vector connecting the two vertices, v_0 and v_1. Formula (A.1) for t_0 can be used. Given v_0, n_0 and v_1, define $d = v_1 - v_0$:

$$t_0 = |d| \frac{-(n_0 \times d) \times n_0}{\left\| (n_0 \times d) \times n_0 \right\|} \tag{A.1}$$

A.3.2.4 Repeat A.3.2.3 for the tangent, t_1, at point, v_1.

A.3.2.5 Compute the centre point, $v_{01} = h_{(0.5)}$, using second order Hermite curve interpolation, as shown in Formula (A.2):

$$h\left(s\right)=\left(2\,s^3-3\,s^2+1\right)v_0+\left(s^3-2\,s^2+s\right)t_0+\left(-2\,s^3+3\,s^2\right)v_1+\left(s^3-s^2\right)t_1 \tag{A.2}$$

A.3.2.6 Compute the centre tangent, $t_{01} = t_{(0.5)}$, using the derivative of the second order Hermite curve interpolation, as shown in Formula (A.3):

$$t\left(s\right)=\left(6\,s^2-6\,s\right)v_0+\left(3\,s^2-4\,s+1\right)t_0+\left(-6\,s^2+6\,s\right)v_1+\left(3\,s^2-2\,s\right)t_1 \tag{A.3}$$

A.3.3 Using the three new vertices and normals, split the triangle into four sub-triangles.

A.3.4 Repeat A.3.3 recursively for each triangle to a depth of five [refer to Figure A.1(b)].

A.4 Code for pseudo-random spatial map (PRSM)

The goal of the rand function is to allow for pseudo-random textures to be used in material and colour definitions. The rand(x,y), rand(x,y,z) and rand(x,y,z,k) return a persistent random number as function of a given coordinate. These functions map the 2D or 3D coordinate onto a real (fractional) pseudo-random number uniformly distributed in the range [0,1] (exclusive of 1). The number returned will be persistent across multiple calls (that is, the same number shall always be returned for the same coordinate). If k = 1, a second (probably different) number may be returned for that coordinate. A third (probably different) number may be returned for that coordinate with k = 2 and so on. A sample C++ implementation of rsm(x,y,z,k) is provided in Figure A.2.