ISO/ASTM 52915:2020(E)

Specification for additive manufacturing file format (AMF) Version 1.2[1]

This standard is issued under the fixed designation ISO/ASTM 52915; the number immediately following the designation indicates the year of original adoption or, in the case of revision, the year of last revision.

INTRODUCTION

This document 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 color, 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 color 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.

1. Scope

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

1.2 This document specifies the requirements for the preparation, display and transmission for the AMF. When prepared in a structured electronic format, strict adherence to an extensible markup language (XML)(**1**)[2] schema supports standards-compliant interoperability.

NOTE 1—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 A1.

1.3 It is recognized that there is additional information relevant to the final part that is not covered by the current version of this document. Suggested future features are listed in Annex A2.

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

2. Normative references

2.1 There are no normative references in this document.

3. Terms and definitions

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

– ISO Online browsing platform: available at http://www.iso.org/obp

– IEC Electropedia: available at http://www.electropedia.org/

3.1 **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.

3.2 **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

[1] This International Standard is under the jurisdiction of ASTM Committee F42 on Additive Manufacturing Technologies and is the direct responsibility of Subcommittee F42.04 on Design, and is also under the jurisdiction of ISO/TC 261, Additive manufacturing, 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.

Current edition approved May 1, 2020. Published August 2020. Originally published as ASTM F2915-11. Last previous edition ISO/ASTM 52915-16.

[2] The boldface numbers in parentheses refer to the Bibliography at the end of this standard.

curved triangles to flat triangles or convert porous material specification into an explicit mesh surface.

3.3 **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.

3.4 **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.

3.5 **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.

3.6 **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.

3.7 **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]

3.8 **STL**

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]

4. Key considerations

4.1 **General**

4.1.1 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 document, the AMF, is designed to address the concerns listed in 4.1.2 to 4.1.7.

4.1.2 *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 only certain advanced machines support (for example, color, multiple materials), but these are defined in such a way as to avoid exclusivity.

4.1.3 *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.

4.1.4 *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.

4.1.5 *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 A2.

4.1.6 *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.

4.1.7 *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.

4.2 *Guidelines for the inclusion of future new elements:*

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

4.2.2 In support of the consideration above, new elements proposed for this document 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.

5. Structure of this specification

5.1 *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.

5.2 *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.

5.3 *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).

5.4 *Future amendments and additions*—While additional XML elements can be added provisionally to any AMF file for internal purpose, such additions will 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.

6. General structure

6.1 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.

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

```
<!-- ignore this comment -->
```

6.3 The remainder of the file shall be enclosed between start `</amf>` and end `</amf>` element tags. This element denotes the file type and fulfills 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 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">
```

6.4 Enclosed within the `<amf/>`element start- and end-tags, there are five top level elements, as described in 6.4.1 to 6.4.5.

6.4.1 `<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.

6.4.2 `<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.

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

6.4.4 `<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.

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

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

7. Geometry specification

7.1 *General:*

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

7.1.2 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.

7.1.3 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.

7.1.4 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 Fig. 1).

```
<?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 1—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.

FIG. 1 Basic AMF file

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

7.2 *Smooth geometry:*

7.2.1 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 Section 13) may ignore curvature information associated with triangles.

7.2.2 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 1024 flat triangles).

7.2.3 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.

7.2.4 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.

7.2.5 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.

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

7.2.7 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.

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

7.2.9 When interpreting normal and tangents, second degree Hermite interpolation shall be used. See A1.3 for formulae for carrying out this interpolation.

7.3 *Restrictions on geometry*—All geometry shall comply with the following restrictions:

7.3.1 Every triangle shall have exactly three different non-colinear vertices.

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

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

7.3.4 Volumes shall not overlap.

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

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

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

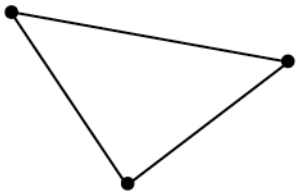
7.3.8 The outward direction of triangles that share an edge in the same volume shall be consistent. The outward direction is defined by the order of vertices.

8. Material specification

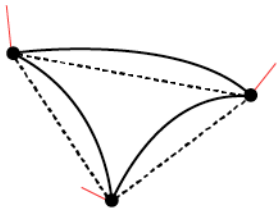
8.1 *General:*

8.1.1 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 Fig. 2.

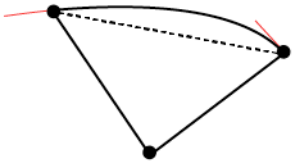
8.1.2 Material characteristics are contained within each <material> element. The child element <color> is used to specify the red/green/blue/alpha (RGBA) appearance of the material (see Section 9 on color). Additional material properties may be specified using the <metadata> element, such as the material name for operational purposes or elastic



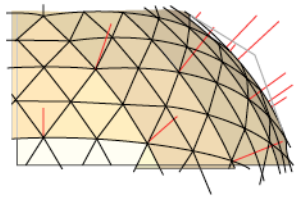
a) Default (flat) triangle patch



b) Triangle curved using vertex normals



c) Triangle curved using edge tangents



d) Subdivision of a curved triangle patch into four curved subpatches

```
<?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) AMF file containing curved geometry

FIG. 2 Types of triangles used in a mesh

properties for equipment that can control such properties, see Fig. 3. See Annex A1 for a description of the AMF elements.

8.2 *Mixed and graded materials and substructures:*

8.2.1 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

```
<?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 1—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.

FIG. 3 Homogeneous and composite materials

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 Annex A1).

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

8.2.3 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.

8.3 *Porous materials:*

8.3.1 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).

8.3.2 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 A2.2.5 and A2.2.6). The intended use of the `<composite>` element with reference to void is to describe cellular mesostructures.

8.4 *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 A1.4).

9. Color specification

9.1 *General:*

9.1.1 Colors may be introduced using the `<color>` element by specifying the RGBA (transparency) values in a specified color space. By default, the color space shall be sRGB (**2**) but alternative profiles may be specified using the metadata tag in the root `<amf>` element (see Annex A1). The `<color>` element may be a `child` of the `<material>` element to associate a color with a material, the `<object>` element to color an entire object, the `<volume>` element to color an entire volume, the `<triangle>` element to color a triangle, or the `<vertex>` element to associate a color with a particular vertex (see Fig. 4).

9.1.2 If no color is specified, the default color is white.

9.1.3 Object color overrides material color specification; a volume color overrides an object and material colors; vertex

```
<?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
    vbmx5IGJ5IGhpcyByZWFzb24sIGJldCBieS
    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<lvtex3>
          </texmap>
        </triangle>
      </volume>
    </mesh>
  </object>
</amf>
```

NOTE 1—A solid color may be associated with a material, a volume or a vertex. A vertex may also be associated with a coordinate in a color texture file.

FIG. 4 Color specification

colors override volume, object, and material colors; and triangle coloring overrides vertex, object, and material colors.

9.2 *Coulor gradations and texture mapping:*

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

9.2.2 When referring to a formula, the `<color>` element may specify a color that depends on the coordinates such as a color gradation. Any mathematical expression that combines the functions described in A1.2 may be used. For example, use of the rand function would allow for pseudo-random color patterns. The tex function would allow the color to depend on a texture map or image. To specify a full-color graphic, typically three textures would be needed, one for each color channel. To create a monochrome graphic, one texture is typically sufficient.

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

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

10. Texture specification

10.1 The `<texture>` element is used to associate a `texture ID` or `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.

10.2 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.

10.3 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.

10.4 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.

11. Constellations

11.1 Multiple objects may be arranged together using the `<constellation>` element (see Fig. 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 by means of which 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.

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

11.3 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.

11.4 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.

12. Metadata

12.1 The `<metadata>` element may optionally be used to specify additional information about the objects, geometries and materials being defined (see Fig. 6). For example, this

```
<?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 1—A constellation can assemble multiple objects together.

FIG. 5 Constellations