~~ISO/ASTM 52915:2013(E)~~

ISO/ASTM 52915 – 2016(E)

Standard Specification for Additive Manufacturing File Format (AMF) Version ~~1.1~~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 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.

1. Scope

1.1 This ~~specification describes a framework for~~ 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. ~~For the last three decades, the STL file format has been the industry standard for transferring information between design programs and additive manufacturing equipment. An STL file contains information only about a surface mesh and has no provisions for representing color, texture, material, substructure, and other properties of the fabricated target object. As additive manufacturing technology is quickly evolving from producing primarily single-material, homogenous shapes to producing multimaterial geometries in full color with functionally graded materials and microstructures, there is a growing need for a standard interchange file format that can support these features.~~

1.2 The ~~additive manufacturing file (AMF)~~ AMF may be prepared, ~~displayed,~~displayed and transmitted ~~on paper or electronically,~~ provided the ~~information required by~~ requirements of this specification ~~is included.~~are met. When prepared in a structured electronic format, strict adherence to an extensible markup language (XML)(**1**)[2] schema is required to support standards-compliant interoperability. ~~The adjunct to this specification contains a W3C XML schema and Annex A1 contains an implementation guide for such representation.~~

~~1.3 *This standard does not purport to address all of the safety concerns, if any, associated with its use. It is the responsibility of the user of this standard to establish appropriate safety and health practices and determine the applicability of regulatory limitations prior to use.*~~

1.3 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] This ~~specification~~ 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.

Current edition approved ~~April 9, 2013~~Oct. 19, 2015. Published ~~May 2013.~~ April 2016. Originally published as ASTM F2915-11. Last previous edition ~~ASTM~~ISO/ASTM ~~F2915-12.~~52915-13.

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

~~1.4 *This standard also does not purport to address any copyright and intellectual property concerns, if any, associated with its use. It is the responsibility of the user of this standard to meet any intellectual property regulations on the use of information encoded in this file format.*~~

1.4 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 A2.

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

2. Terminology

2.1 ~~*Definitions of Terms Specific to This Standard:*~~*Definitions:*

~~2.1.1 This section provides definitions of terms specific to this standard—these terms also include the common terms seen in many documents related to extensible markup language (XML) and additive manufacturing. See also Annex A1 for definitions of additional terms specific to this specification.~~ For the purposes of this document, the following terms and definitions apply.

2.1.1 *AMF consumer*—software reading (parsing) the Additive Manufacturing File Format (AMF) file for fabrication, visualization or analysis.

2.1.1.1 *Discussion*—

AMF files are typically imported by additive manufacturing equipment, as well as viewing, analysis and verification software.

2.1.2 *AMF editor*—software reading and rewriting the Additive Manufacturing File Format (AMF) file for conversion.

2.1.2.1 *Discussion*—

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.

2.1.3 *AMF producer*—software writing (generating) the Additive Manufacturing Format (AMF) file from original geometric data.

2.1.3.1 *Discussion*—

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

2.1.4 ~~*attribute, attribute—n*—~~characteristic of data, representing one or more ~~aspects, descriptors,~~ aspects or ~~elements~~descriptors of the ~~data.~~ data in an element.

2.1.4.1 *Discussion*—

In ~~object-oriented systems,~~ the XML framework, attributes are characteristics of ~~objects. In XML, attributes are characteristics of~~ elements.

2.1.5 ~~*comments, comments—n*—~~all text comments associated with any data within the ~~additive manufacturing file (AMF) not containing core relevant, technical, or administrative data and not containing pointers to references external to the AMF.~~Additive Manufacturing File Format (AMF) to be ignored by import software.

2.1.5.1 *Discussion*—

Comments are used for enhancing human readability of the file and for debugging purposes.

2.1.6 ~~*domain-specific applications, element—n*—additional, optional sets of AMF data elements specific to such areas as novel additive manufacturing processes, enterprise workflow, and supply chain management.~~ information unit within an XML document consisting of a start tag, an end tag, the content between the tags, and any attributes.

2.1.6.1 *Discussion*—

~~Data sets for optional AMF domain-specific applications will be developed and balloted separately from this specification.~~In the XML framework, an element can contain data, attributes and other elements.

2.1.7 *extensible markup language, ~~XML,~~ XML—n—*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.

2.1.7.1 *Discussion—*

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). **ISO/ASTM 52900-2015**

2.1.8 *STL ~~(file format),~~ stereolithography—n—*file format ~~native to the stereolithography computer-aided drafting (CAD) software that is supported by many software packages; it is widely used for rapid prototyping and computer-aided manufacturing.~~ 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.

2.1.8.1 *Discussion—*

~~STL files describe only the surface geometry of a three-dimensional object as a tessellation of triangles without any representation of color, texture, or other common CAD model attributes. The STL format specifies both the American Standard Code for Information Interchange (ASCII) and binary representations.~~ 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. **ISO/ASTM 52900-2015**

3. Key ~~Considerations~~considerations

3.1 ~~*There*~~ *General:*~~is a natural a tradeoff 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, this file format is designed to address the following concerns:~~

3.1.1 There is a natural tradeoff 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—~~independence—*The ~~file format shall describe~~ AMF describes an object in such a general way ~~such~~ 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 ~~properties~~features that describe capabilities only certain advanced machines support (for example, ~~color, multiple materials, and so forth),~~ colour, multiple materials), but these are defined in such a way as to avoid exclusivity.

3.1.3 *Simplicity*—The AMF ~~file~~ format is easy to implement and understand. The format can be read and debugged in a simple ~~ASCII~~ text viewer to encourage ~~understanding~~comprehension and adoption. ~~No identical~~ Identical information is not stored in multiple places.

3.1.4 *Scalability*—The file ~~format~~ size and processing time scales well with the increase in part complexity and ~~size and~~ with the improving resolution and accuracy of manufacturing equipment. This includes being able to handle large arrays of identical objects, complex ~~repeated~~periodic internal features (for example, ~~meshes),~~ meshes and lattices), and smooth curved surfaces ~~with fine printing resolution, and multiple components arranged in an optimal packing for printing.~~when fabricated with very high resolution.

3.1.5 *Performance*—The ~~file format should enable~~ 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 ~~Appendix X1~~Annex A2.

3.1.6 *Backwards ~~Compatibility—~~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 ~~algorithm~~algorithms and code infrastructure already in existence.

3.1.7 *Future ~~Compatibility—~~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 ~~homogenous~~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~~this specification

4.1 ~~*Information*~~*Format*—Information specified throughout this specification is stored in XML 1.0 format. XML is ~~an ASCII~~ a text file comprising a list of elements and attributes. Using this widely accepted data format ~~opens the door to a rich host of~~ allows for the use of many tools for creating, viewing, manipulating, ~~parsing,~~parsing and storing AMF files. XML is ~~human readable,~~ 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 ~~*Another*~~*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, ~~this~~ the use of XML namespaces allows new features to be added without ~~needing to update~~ 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, ~~a parsing program~~an AMF consumer should read and process real numbers in double precision (64 ~~bit).~~bits).

4.4 *Future ~~Amendments~~amendments and ~~Additions—~~additions—* ~~Additional~~ While additional XML elements can be added provisionally to any AMF file for ~~any purpose but~~ internal purpose, such additions will not be considered part of this specification. An unofficial AMF element ~~can~~may be ignored by any ~~reader~~ AMF consumer and ~~does~~may not ~~need to~~ be stored or reproduced ~~on output.~~ by an editor application. An element becomes official only when it is formally accepted into this specification.

5. General ~~Structure~~structure

5.1 The AMF file ~~begins~~ 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 ~~Blank lines~~Whitespace characters and standard XML comments ~~can~~may be interspersed in the file and ~~will~~shall be ignored by any interpreter, for example:

<!-- ignore this comment -->

5.3 The remainder of the file ~~is~~ shall be enclosed between ~~an opening~~ start </amf> ~~element~~ and ~~a closing~~ end </amf> ~~element. These elements are necessary to denote the file type, as well as to fulfill~~ element tags. This element denotes the file type and fulfills the requirement that all XML files have a ~~single root~~ single root element. ~~The version~~ A version attribute denoting the version of the AMF standard ~~as well as all standard XML namespace declarations can~~ 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 ~~can~~may also be specified ~~(mm,~~(millimetre, inch, ~~ft, meters,~~foot, metre, or ~~micrometers).~~micron). In absence of a unit specification, ~~millimeters are~~ the attribute value millimetres is assumed.

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

<amf unit="millimeter" version="1.0" xml:lang="en"
xmins:amf="www.astm.org/Standards/F2915–14">

5.4 ~~Within~~ Enclosed within the ~~AMF~~<amf/> ~~brackets,~~ element start- and end-tags, there are five top level ~~elements:~~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 ~~are associated with~~might also reference a material ~~identification (ID) for printing.~~ 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 ~~or more materials for printing with~~ 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 ~~or more images or textures for color or texture mapping each with~~ image or texture for colour or texture mapping, each of which declares an associated texture ID. The texture ID thus 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 ~~will~~shall be imported with no relative position data. The ~~parsing program can~~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 ~~Specifications~~specification

~~6.1 The top level <object> element specifies a unique id and contains two child elements: <vertices> and <volume>. The <object> element can optionally specify a material.~~

~~6.2 The required <vertices> element lists all vertices that are used in this object. Each vertex is implicitly assigned a number in the order in which it was declared starting at zero. The required child element <coordinates> gives the position of the point in three-dimensional (3D) space using the <x>,<y> , and <z> elements.~~

~~6.3 After the vertex information, at least one <volume> element shall be included. Each volume encapsulates a closed volume of the object. Multiple volumes can be specified in a single object. Volumes may share vertices at interfaces but may not have any overlapping volume.~~

6.1 ~~Within each volume, the child element <*triangle*>~~*General:* ~~shall be used to define triangles that tessellate the surface of the volume. Each <triangle> element will list three vertices from the set of indices of the previously defined vertices. The indices of the three vertices of the triangles are specified using the <v1>,<v2>, and <v3> elements. The order of the vertices shall be according to the right-hand rule such that vertices are listed in counter-clockwise order as viewed from the outside. Each triangle is implicitly assigned a number in the order in which it was declared starting at zero (see Fig. 1).~~

6.1.1 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.

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.


```
<?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 Containing Only a List of Vertices and Triangles—This Structure Is Compatible with the STL Standard~~file

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 Fig. 1).

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

6.2 *Smooth ~~Geometry:~~geometry:*

6.2.1 By default, all triangles ~~are~~ shall be assumed to be flat and all triangle edges ~~are~~ shall be assumed to be straight lines connecting their two vertices. However, curved triangles and curved edges ~~can~~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.

6.2.2 During ~~read,~~import, a curved triangle patch shall be recursively subdivided into four triangles ~~by the parsing program~~ to generate a final temporary set of flat ~~triangles at any desired resolution for manufacturing or display.~~ triangles. The depth of recursion shall be ~~determined by the parsing program, but a minimal level of four is recommended (that is, convert~~ exactly five (that is, a single curved triangle ~~into 256~~will be converted into 1024 flat triangles).

6.2.3 During ~~write,~~production, the ~~encoding software~~ producing software that generates curved triangles shall determine automatically the ~~minimum~~ number of curved triangles required to specify the target geometry to the desired tolerance, ~~assuming~~knowing that the ~~parser~~ consuming software will perform ~~at least four~~ five levels of subdivision for any curved triangle.

6.2.4 To specify curvature, a vertex ~~can optionally~~ may contain a child element <normal> to specify the desired surface normal at the ~~location of the~~ vertex. The normal should be unit length and pointing outwards. If this normal is specified, all triangle edges meeting at that vertex ~~should~~shall be curved so that they are perpendicular to that normal and in the plane defined by the normal and the original straight edge. ~~If a vertex is referenced by two volumes, the normal is considered separately for each volume, and its direction should be interpreted as consistent with the volume in consideration (pointing outwards). Vertices that have an ambiguous normal because they are common to multiple surfaces, should not specify a normal.~~

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 ~~a~~ the volume's surface at a vertex is undefined (for example, at a cusp, ~~corner,~~corner or edge), an <edge> element ~~can~~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 ~~will~~shall take precedence in case of a conflict with the curvature implied by a <normal> element.

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

6.2.9 When interpreting normal and tangents, second degree Hermite interpolation ~~will~~shall be used. See ~~Annex A3~~A1.3 for formulae for carrying out this interpolation.

~~6.5.8 The geometry shall not be used to describe support structure. Only the final target structure shall be described.~~

6.3 *Restrictions on ~~Geometry—~~geometry—*All geometry shall comply with the following restrictions:

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

6.3.2 Triangles ~~may~~shall not intersect or overlap except at their common edges or common vertices.

6.3.3 Volumes shall enclose a closed contiguous space with ~~nonzero~~non-zero volume.

6.3.4 Volumes ~~may~~shall not overlap.

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

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

6.3.7 ~~No~~Any two vertices ~~can~~ shall not have identical coordinates. The tolerance used to define equality is 10^{-8} units.

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

7. Material ~~Specifications~~specification

~~7.1 Materials are introduced using the <material> element. Any number of materials may be defined using the <material> element. Each material is assigned a unique id. Geometric volumes are associated with materials by specifying a materialid within the <volume> element. Any number of materials may be defined. The materialid "0" is reserved for no material (void) (see Fig. 2).~~

7.1 ~~Material attributes are contained within each <material>.~~*General:* ~~The element <color> is used to specify the red/green/blue/alpha (RGBA) appearance of the material (see Section 8 on color). Additional material properties can 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 Annex A1 for more information (see Fig. 3).~~

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 Fig. 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 Section 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 Fig. 3. See Annex A1 for a description of the AMF elements.

7.2 *Mixed and ~~Graded Materials~~graded materials and ~~Substructures:~~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*~~,*y* and *z* coordinates. A constant mixing proportion will lead to a ~~homogenous~~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 ~~nonperiodic~~non-periodic substructure. The proportion formula can also refer to a texture map using the tex ~~`tex(textureid,x,y,z)`~~ `(textureid,x,y,z)` function (see Annex A1).

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

7.2.3 Any ~~number of materials can be specified. Any~~ negative material proportion value ~~will~~shall be interpreted as a zero proportion. Material proportions shall be normalized to a sum of 1.0 to determine actual ratios.

~~7.3.3 Although the <composite> element could theoretically be used to describe the complete geometry of an object as a single function or texture, such use is discouraged (but see Appendix X2). The intended use of the <composite> element is for the description of cellular mesostructures.~~

7.3 ~~*Porous Materials—*~~

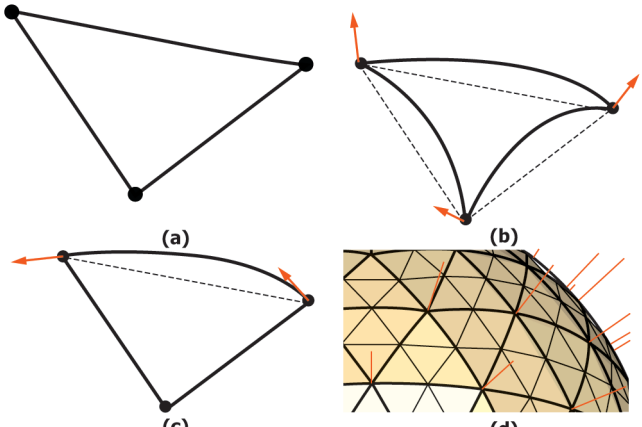
~~Reference to `materialid` "0" (void) can be used to specify porous structures. The proportion of void can be either 0 or 1 only. Any fractional value will be interpreted as 1 (that is, any fractional void will be assumed fully void).~~Porous materials:


```
<?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—~~An~~ The figure shows an AMF file containing five materials. Material 3 is a ~~40/60 % homogenous~~ 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 ~~Homogenous~~Homogeneous and ~~Composite Materials~~composite materials




```
<?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)

(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

FIG. 2 ~~(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, and (e) AMF File Containing Curved Geometry~~Types of triangles used in a mesh

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

7.4 *Stochastic ~~Materials—~~materials—*Reference to the `rand(x,y,z)` function ~~can~~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 ~~can~~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)~~[0,1] that is persistent across function calls (see ~~Annex A4~~A1.4).

8. ~~Color Specification~~colour specification

8.1 ~~Colors are introduced using the <*color*>~~*General:* ~~element by specifying the RGBA (transparency) values in a specified color space. By default, the color space shall be sRGB (2) but alternative profiles could be specified using the metadata tag in the root <amf> element (see Annex A1). The <color> element can be inserted at the material level to associate a color with a material, the object level to color an entire object, the volume level to color an entire volume, a triangle level to color a triangle, or a vertex level to associate a color with a particular vertex (see Fig. 4).~~

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 Annex A1). 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 Fig. 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.

~~8.2 Object color overrides material color specification, a volume color overrides an object color, vertex colors override volume colors, and triangle coloring overrides a vertex color.~~

8.2 ~~*Graded Colors and Texture Mapping:*~~*colour gradations and texture mapping:*

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


```
<?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<1vtex3>
          </texmap>
        </triangle>
      </volume>
    </mesh>
  </object>
</amf>
```


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

FIG. 4 ~~Color Specification~~Colour specification

8.2.2 When referring to a formula, the ~~<color>~~<colour> element ~~can~~may specify a ~~color~~colour that depends on the coordinates such as a ~~graded color or a spotted color.~~ colour gradation. Any mathematical expression that ~~combined~~combines the functions described in ~~Annex A2~~A1.2 ~~can~~may be used. For example, use of the rand function ~~can~~would allow for pseudo-random ~~color~~colour patterns. The tex function ~~can~~would allow the ~~color~~colour to depend on a texture map or image. To specify a ~~full-color~~full-colour graphic, typically three textures ~~will~~would be needed, one for each ~~color~~colour channel. To create a monochrome graphic, ~~typically only~~ one texture is typically sufficient.

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

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

9.1 The <texture> element ~~can be~~ is used to associate a textureid with ~~a~~ particular texture data. The texture map size ~~will~~shall be specified and both two-dimensional (2D) and ~~3D maps~~ three-dimensional (3D) textures are supported. The data ~~will be an encoded string of bytes in Base64 encoding as grayscale values. Grayscale will be encoded as a string of individual bytes, one per pixel, specifying the grayscale level in the 0-255 range.~~ 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 ~~data will start with the top left corner and proceeding left to right then top to bottom. A 3D texture will specify the first layer initially and repeat for all subsequent depths into the screen according to the right-hand rule. The data will be truncated or appended with zero values as needed to meet the specified texture size.~~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*,~~*v* and (optionally) *w* coordinates for each vertex of this triangle. If the texture's "tiled" property is ~~true,~~"true", any *u*,*v*,*w* mappings outside of the [0,1] range ~~will~~shall be determined according to the coordinate modulo 1. If the texture's tiled property is ~~false, and~~ not "true", any mappings that fall outside of the [0,1] range ~~will return transparent. Textures shall be linearly interpolated for each triangle. A triangle shall include only a single~~ shall return zero. ~~<texmap> element. Overlapping textures shall be combined into a single texture before being mapped onto a mesh.~~

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. ~~Print~~ Constellations

10.1 Multiple objects ~~can~~may be arranged together using the <constellation> element (see Fig. 5). A constellation ~~can~~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 ~~needs to undergo into its position~~ shall be transformed to position it in the constellation. The displacement and rotation ~~are always~~shall be defined ~~relatively~~relative to the original position and orientation ~~in which~~ of the object when it was originally defined. Rotation angles ~~are~~ shall be specified in degrees and ~~are applied first to rotation~~shall first apply rotations about the *x* axis, then the *y* ~~axis,~~axis and then the *z* axis.

10.2 A constellation ~~can refer to~~ may include another constellation, with multiple levels of hierarchy. However, ~~recursive or~~ cyclic definitions of constellations ~~are~~shall not ~~allowed.~~ be used.

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