FAV® (Fabricatable Voxel)
File Format Specification

Created by: Tomonari Takahashi 1), Atsushi Masumori 2), Masahiko Fujii 1), Hiroya Tanaka 2)

1) FUJIFILM Business Innovation Corp.

2) Social Fabrication Laboratory of Keio University at Shonan-Fujisawa Campus (SFC)

Issued on: Feb.18, 2019

Version: 1.1a

Contents

Introduction

The purpose of this specification document

This document describes the FAV (FAbricatable Voxel) file format specification.
This document is made publicly available so that:

- the FAV format may be used to achieve efficient management of the complex structures of 3D model data such as shapes and attributes (including internal structures)
- FAV format 3D model data may be utilized in hardware and software
- FAV files may be created, edited, and utilized within systems (interoperability)

3D model data formatted in accordance with this specification can be used on systems that support FAV files. By processing FAV files on FAV-supporting systems in accordance with this specification, it is possible to utilize a range of information from 3D model data stored in this format.

We hope that use of the FAV format will enhance the range of design and expression of 3D model data and contribute to further developments in the fabrication of and communication using 3D models.

Wording in this document

● **RFC 2119**

This document uses keywords and phrases as described in RFC 2119. Below is an excerpt from RFC 2119.

1. **MUST**
   This word, or the terms "REQUIRED" or "SHALL", mean that the definition is an absolute requirement of the specification.
2. **MUST NOT**
   This phrase, or the phrase "SHALL NOT", mean that the definition is an absolute prohibition of the specification.
3. **SHOULD**
   This word, or the adjective "RECOMMENDED", mean that there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a different course.
4. **SHOULD NOT**
   This phrase, or the phrase "NOT RECOMMENDED" mean that there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implication should be understood and the case carefully weighed before implementing any behavior described with this label.
5. **MAY**
   This word, or the adjective "OPTIONAL", mean that an item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option, though perhaps with reduced functionality. In the same vein an implementation which does not include a particular option MUST be prepared to interoperate with another implementation which does include the option
   (except, of course, for the feature the option provides.)

In this specification document, those specifications that contain the keywords and phrases defined above (1 to 5) are the requirements deemed necessary in fulfilling the purpose of the FAV format and achieving the intended operation of systems that handle FAV files. It must be fully understood that serious problems may occur in systems handling FAV files when FAV files are used without satisfying the specified requirements.

The use of FAV files that do not conform to this specification requires that a system be equipped with an appropriate error handling mechanism; however, it cannot be guaranteed that any one system will have such a mechanism. Therefore, it is recommended for the users of FAV format files to take utmost care that no unintended operations occur on a system handling FAV files.

● XML

FAV files are written in XML.

In this specification document, elements in XML are enclosed in angle brackets. e.g., <element>
The attributes of elements are enclosed in parentheses. e.g., (attribute)

In this document, XML code examples are enclosed in a box and begin with "e.g.,".

```
e.g., <element_label attribute_label=0.1>ELEMENT</element_label>
```

In this document, elements and attributes are detailed in tables as shown below.

Table 1: An example description of the specifications of an element and its attribute

Element	Attribute	Data type	Data [Default value]	Description	Condition
element_label	attribute_label	double	A double value [0.0]	This area describes what is specified as the attribute of an element and the meaning of the attribute.	This area describes conditions such as whether the attribute is required, the number of attributes required (e.g., one or more), etc.
	–	string	A character string [—]	This area describes what is specified as the data of the element and the meaning of the specified data. "–" is written when there is no default value.	

When elements are written in angle brackets separated by a period, the element on the left side of the period indicates a parent element, and the element on the right side of the period indicates a child element.

```
e.g., <element_parent.element_child> expresses the element structure shown below.

      <element_parent>
          <element_child> The data for this element is described here. </element_child>
      </element_parent>
```

License

The owners of the copyright and the license of this specification document are as follows:
License information:

This license statement is related to the copying and modification of this document. It is neither intended to restrict nor permit the use of data in FAV format and/or systems that handle FAV files, and does not assert any rights with regard to such data and systems.

The following are to be described in order to fulfill the obligations specified in the license that apply when this document is entirely or partially copied or used in other formats.

- The license information shown above
- The URL of this document
- Notation stating that the material is a copy or an excerpt of FAV (FAbricatable Voxel) File Format Specification (Ver 1.1)

1. The FAV file format

The FAV format is a file format for saving 3D model data.

1.1. Concept

FAV stands for “FAbricatable Voxel.”

In addition to a 3D object's external structure, a FAV file can store information on a range of attributes such as those defining the internal structure, materials to be used, colors, and connection strength of an object. The FAV file format enables designers to model both the exterior and interior of a 3D object right down to the finest requirements and to then save this data.

Using 3D model data in the FAV format furthers the range of expression possible with a fabricated object, thereby enabling us to conceive new ideas and fabricate new objects that have not been possible until now. These enhancements will contribute to the facilitation of better communication using 3D model data, and be especially beneficial in the creation of new applications capable of crossing between industry borders. For example, the FAV format can be expected to have a particularly large influence in the creation of applications capable of integrating the currently separate processes of image processing, management of scanned data in the medical field (MRI / CT), and fabrication of industrial products from composite materials.

3D model data in FAV format is "fabricatable," meaning that data is stored in a way that is optimized for fabrication. Data optimized for fabrication satisfies the following:

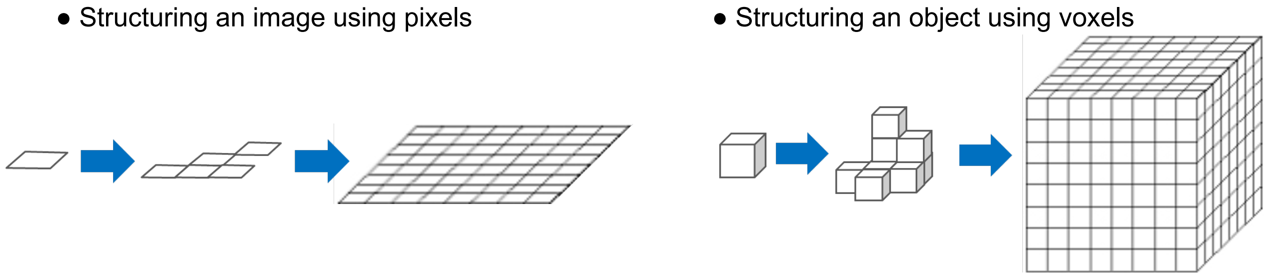
- The information required for fabrication (e.g., shape, material, color, connection strength) is clearly defined in three-dimensional space, for both the exterior and interior of an object.
- It allows the user to design (CAD), analyze (CAE), and inspect (CAM) 3D model data seamlessly in an integrated manner without having to convert data.

FAV format 3D model data satisfies all of the above.

1.2. Characteristics

FAV stands for “FAbricatable Voxel.” The FAV format expresses 3D data in the form of voxels.

Voxels are the three-dimensional equivalents of pixels. Similar to the way pixels are arranged to create a two-dimensional image, a three-dimensional object is structured by arranging voxels in three-dimensional space.



● Structuring an image using pixels

● Structuring an object using voxels

Fig. 1: Pixels and voxels

Defining three-dimensional objects using voxels makes it possible to design 3D objects of various structures. This can be done by using voxels of various designs, not including voxels in certain areas, etc.

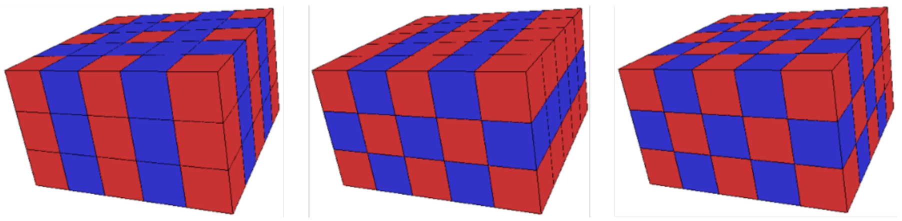
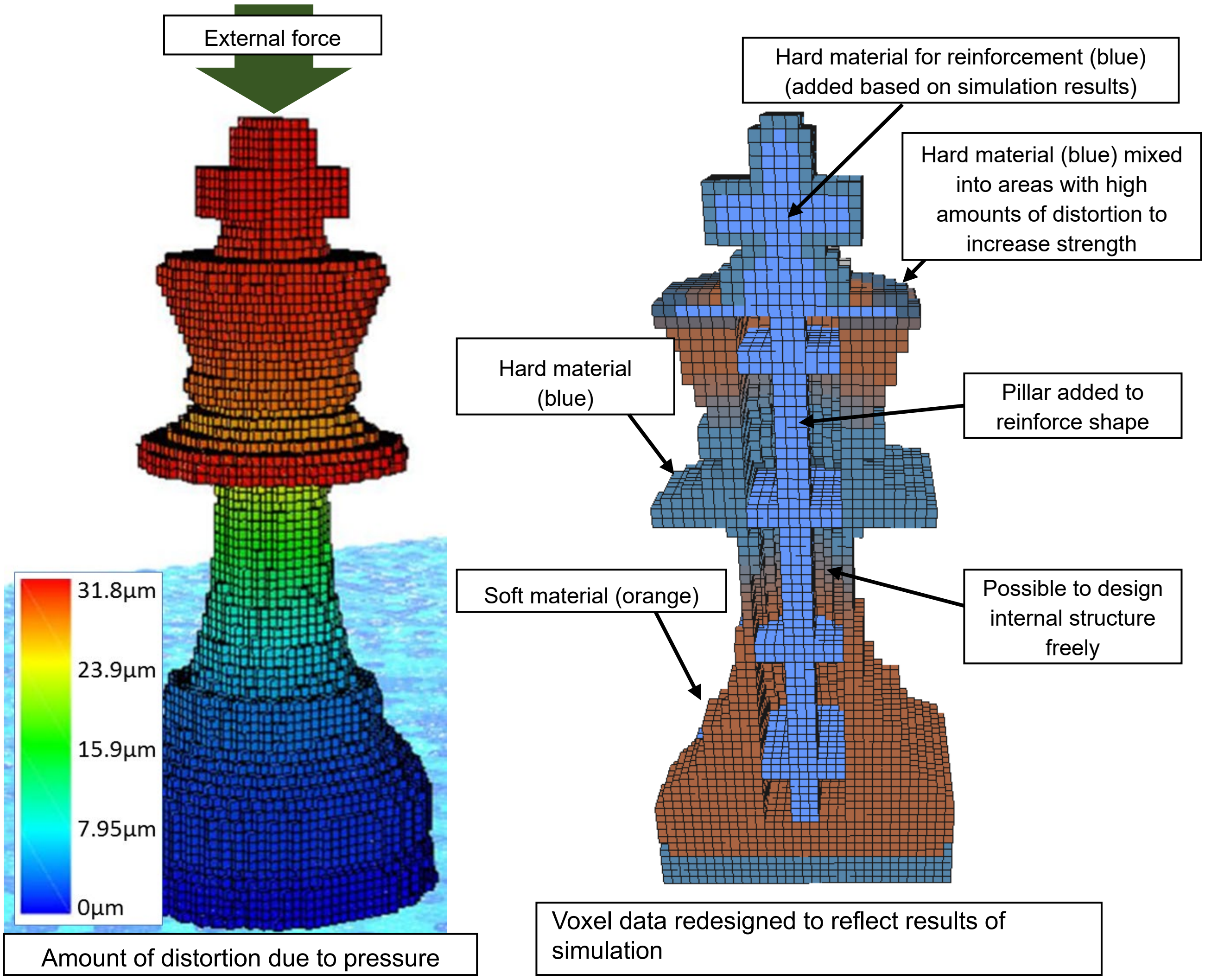


Fig. 2: Designing intricate structures using voxels

The following are the benefits of defining 3D objects by arranging voxels:

- The internal structure as well as the external structure of an object can be defined.
- Various attributes such as materials, colors, and connection strength can be defined together with the overall shape of the object.
- Voxels are finite elements. Therefore, voxel-based data can be used for simulation (CAE) using the same format as that used for the design phase (CAD). Also, it is possible to modify the design of each voxel based on the results of simulations.


External force
31.8μm
23.9μm
15.9μm
7.95μm
0μm
Amount of distortion due to pressure
Hard material for reinforcement (blue) (added based on simulation results)
Hard material (blue) mixed into areas with high amounts of distortion to increase strength
Hard material (blue)
Pillar added to reinforce shape
Soft material (orange)
Possible to design internal structure freely
Voxel data redesigned to reflect results of simulation

Simulation results can be reflected in voxel data for easy design changes

- Material changes (changing to harder materials, etc.)
- Structural changes (reinforcing shape, removing internal areas, etc.)

Fig. 3: The benefits of using voxel-based 3D model data

1.3. Environmental conditions

The specifications that follow assume that FAV-format data is handled in the environment described below.

- **The coordinate system**

In the coordinate system for FAV files, planes which form the layers of an object are defined on the x-y axes, while the height of an object is defined on the z-axis as a positive value. The coordinate system is required to be right-handed.

Taking into consideration the work areas of fabrication equipment such as 3D printers where 3D models are actually constructed, it is recommended that coordinates of 3D model data be plotted so that the z-axis value represents the height, the x-axis value the width, and the y-axis value the depth of a fabricated object when viewed from the origin point of the equipment to be used (which is equal to the origin point of the coordinates).

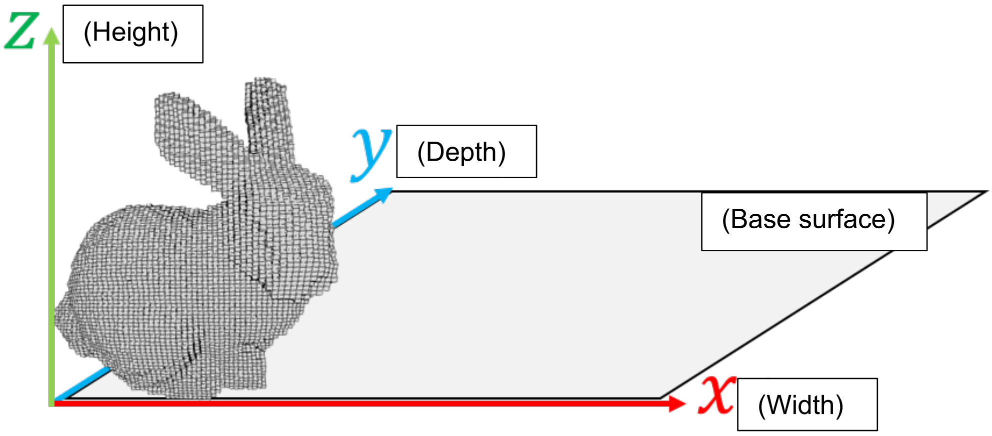


Fig. 4: The coordinate system used for FAV files

- **The units used for the coordinate system**

The numerical value 1 used in the coordinate system of the FAV format is required to represent 1 mm in a real-world object.

Different units (e.g., inches) may be used depending on the application. If a different unit is used, however, note that a coordinate value of 1 must still be equal to 1 mm in a real-world object when data is saved in FAV format. Note also that the same FAV file must not be interpreted in different units when handled by different environments or applications.

1.4. Structure

The elements of a FAV file form a tree structure as shown below.

<metadata>, <palette>, <voxel>, and <object> are the main four elements of a FAV file. <voxel> references <palette>, and <object> references <voxel>. Therefore, it is recommended that elements be defined in the following order: <palette>, <voxel>, <object>.Unless otherwise specified, the order of elements is discretional.

See each section for details of each element and attribute.

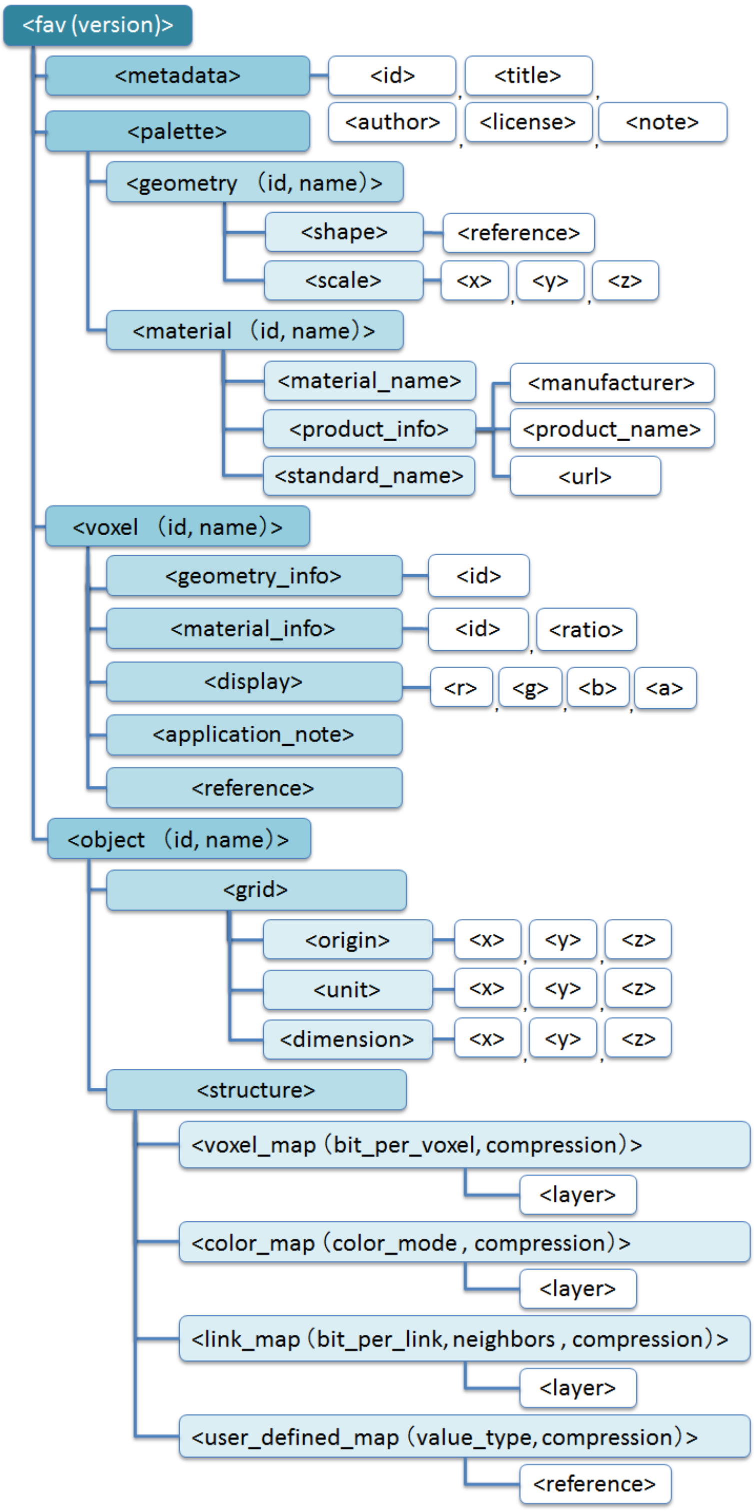


Fig. 5: The tree structure of elements that constitute an FAV file

1.5. Notation method

The labels of elements and attributes in a FAV file shall be in lower case.
When a label consists of multiple words, each word shall be concatenated by underscore.

```
e.g., <element_item attribute_item=”ATTRIBUTE”/>
```

1.6. Security

Security specifications for FAV-format data do not greatly differ from other general electronic data. This means that it is possible to apply security measures such as setting passwords, adding digital signatures for detecting falsification, and allowing users to browse FAV files only when they have certain keys.

FAV files that have been modified without the permission of the author or whose author information has been falsified must not be used. It must be fully understood that any usage in conflict with the rules stipulated in the license may be considered a violation of the law.

2. <fav>

Parent element: —; Link: <metadata>, <palette>, <voxel>, <object>

<fav> is the root element of a FAV file. 3D model data defined in the FAV format starts from the <fav> element. All pieces of data defined and managed in the FAV format are stored under <fav>.

<fav> has the following attribute.

Table 2: The description of <fav>

Element	Attribute	Data type	Data [Default value]	Description	Condition
fav	version	string	The version number of the FAV format [−]	Specifies the character string representing the version number of the FAV format used for the file.	Required

The following element may be described at the level under <fav>. For details, see the section <metadata>.

- <metadata>

Shown below is an example of XML code under <fav>.

```
      e.g.,
                <fav version="1.1a">
                  …
                </fav>
```

Fig. 6: Example of XML code under <fav>

3. <metadata>

Parent element: <fav>; Link: <material>, <object>, <user_defined_map>

<metadata> defines metadata for the various types of data defined in the FAV format.

<fav>, <material>, <object>, and <user_defined_map> can be defined as the parent elements of <metadata>. Regardless of the level at which the <metadata> element is defined, elements that can be defined under <metadata> always remain the same. It is only necessary to define required elements.

In <metadata>, the following elements may be defined.

- <id>
- <title>
- <author>
- <license>
- <note>

Shown below is an example of XML code under <metadata>.

```
 e.g.,
<metadata>
  <id>bc4affb5-9a53-4de7-9f27-721ef27e8f34</id>
  <title><![CDATA[FAV Ver1.1 Sample File]]></title>
  <author><![CDATA[FUJIFILM Business Innovation & Keio SFC]]></author>
  <license><![CDATA[CC BY]]></license>
  <note><![CDATA[This is a sample file in FAV format ver1.1.]]></note>
</metadata>
```

Fig. 7: Example of XML code under <metadata>

3.1. <id>

Parent element: <metadata>; Link: <fav>, <material>, <object>, <user_defined_map>

<id> specifies the ID used to identify the parent element of <metadata>. It is recommended that IDs that are guaranteed to be unique be used (e.g., GUID) to ensure the uniqueness of each piece of data in the FAV format.

<id> contains the following data.

Table 3: The description of <id>

Element	Attribute	Data type	Data [Default value]	Description	Condition
id	–	string	The ID of the parent element [–]	Specifies the character string of the ID that is used to identify the parent element of <metadata>.	Required

See Fig. 7 for an XML code example for <id>.

3.2. <title>

Parent element: <metadata>; Link: <fav>, <material>, <object>, <user_defined_map>

<title> specifies the name of the parent element of <metadata>. For example, <fav.metadata.title> specifies the name of <fav>, and <object.metadata.title> specifies the name of a single <object> element.

<title> contains the following data.

Table 4: The description of <title>

Element	Attribute	Data type	Data [Default value]	Description	Condition
title	–	CDATA	The name of the parent element [–]	Specifies the character string of the name of the parent element of <metadata>.	Required

See Fig. 7 for an XML code example for <title>.

3.3. <author>

Parent element: <metadata>; Link: <fav>, <material>, <object>, <user_defined_map>

<author> specifies the author of the parent element of <metadata>. For example, <fav.metadata.author> specifies the author of <fav>, and <object.metadata.author> specifies the author of a single <object> element.

<author> contains the following data.

Table 5: The description of <author>

Element	Attribute	Data type	Data [Default value]	Description	Condition
author	–	CDATA	The author of the parent element [–]	Specifies the character string of the author of the parent element of <metadata>.	Required

See Fig. 7 for an XML code example for <author>.

3.4. <license>

Parent element: <metadata>; Link: <fav>, <material>, <object>, <user_defined_map>

<license> is the license information of the parent element of <metadata>. For example, <fav.metadata.license> specifies the license for <fav>, and <object.metadata.license> specifies the license for a single <object> element.

Some example licenses are Creative Commons (CC), GNU General Public License (GPL), BSD, and X11 (MIT). If a unique license is to be defined, write the entire text under <license> and/or provide the link to the license information.

<license> contains the following data.

Table 6: The description of <license>

Element	Attribute	Data type	Data [Default value]	Description	Condition
license	–	CDATA	The license information of the parent element [–]	Specifies the character string describing the license information of the parent element of <metadata>.	Required

See Fig. 7 for an XML code example for <license>.

3.5. <note>

Parent element: <metadata>; Link: <fav>, <material>, <object>, <user_defined_map>

<note> is a memo for the parent element of <metadata>. For example, <fav.metadata.note> specifies the memo for <fav>, and <object.metadata.note> specifies the memo for a single <object> element.

<note> contains the following data.

Table 7: The description of <note>

Element	Attribute	Data type	Data [Default value]	Description	Condition
note	–	CDATA	The memo for the parent element [−]	Specifies the character string of a memo for the parent element of <metadata>.	

See Fig. 7 for an XML code example for <note>.

4. <palette>

Parent element: <fav>; Link: <voxel>, <geometry>, <material>

In <palette>, basic information such as the shape and material of a voxel is registered in preparation for fabricating a 3D object based on the information contained in a FAV file. 3D model data is represented in the FAV format by taking <voxel> elements built from the basic information registered in <palette> and using these to define <object>.

The following elements are described at the level under <palette>.

- <geometry>
- <material>

Shown below is an example of XML code under <palette>.

```
e.g.,
   <palette>
    <geometry id="1" name="NormalCube">
      <shape>cube</shape>
      <scale>
        <x>1</x>
        <y>1</y>
        <z>1</z>
      </scale>
    </geometry>
    <geometry id="2" name="Plate">
      <shape>cube</shape>
      <scale>
        <x>1</x>
        <y>1</y>
        <z>0.25</z>
      </scale>
    </geometry>
    <geometry id="3" name="Diamond">
      <shape>user_defined</shape>
      <reference><![CDATA[Diamond.stl]]></reference>
      <scale>
        <x>0.98</x>
        <y>0.98</y>
        <z>-1.05</z>
      </scale>
    </geometry>
    <material id="1" name="SoftMat1">
      <material_name><![CDATA[Some-soft-materials]]></material_name>
    </material>
    <material id="2" name="HardMat1">
      <product_info>
        <manufacturer><![CDATA[ABC Materials Co.]]></manufacturer>
        <product_name><![CDATA[ULTRA-HARD/007]]></product_name>
        <url><![CDATA[http://www.abcmaterial.com/ultra/hard/007]]></url>
      </product_info>
      <product_info>
        <manufacturer><![CDATA[ABC Materials Co.]]></manufacturer>
        <product_name><![CDATA[ULTRA-HARD/006a]]></product_name>
        <url><![CDATA[http://www.abcmaterial.com/ultra/hard/006/a]]></url>
      </product_info>
      <standard_name><![CDATA[JIS K6899-1 ABS]]>
      </standard_name >
    </material>
    </palette>
```

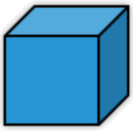
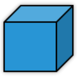
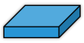
Fig. 8: Example of XML code under <palette>

4.1. <geometry>

Parent element: <palette>; Link: <voxel>, <shape>, <scale>

<geometry> defines the shape and scale of a voxel, the basic element of 3D model data. By arranging voxels that have their geometric properties defined in <geometry>, 3D model data can be structured.

Table 8: Example voxel shapes that can be defined in <geometry>

Image						
(id)	01	02	03	04	05	06
(name)	Cube01	Cube02	Plate	BigSphere	SmallSphere	Cylinder
<shape>	cube	cube	cube	sphere	sphere	user_defined
<scale>	2×2×2	1×1×1	1×1×0.3	1.5×1.5×1.5	0.25×0.25×0.25	3×1×1

<geometry> has the following attributes.

Table 9: The description of <geometry>

Element	Attribute	Data type	Data [Default value]	Description	Condition
geometry	id	positiveInteger	The ID of the geometry information [1]* *After this value has been used, the default value is increased by 1 each time.	Specifies the unique ID that is used to identify <geometry>. The same ID must not be used for multiple <geometry> elements.	Required
	name	string	The name to be defined for the geometry information [—]	Specifies the name of <geometry>. The same name should not be used for multiple <geometry> elements.	

The following elements are described at the level under <geometry>.

- <shape>
- <scale>

Shown below is an example of XML code under <geometry>.

```
e.g.,
   <geometry id="2" name="Plate">
     <shape>cube</shape>
     <scale>
       <x>1</x>
       <y>1</y>
       <z>0.25</z>
     </scale>
   </geometry>
   <geometry id="3" name="Diamond">
     <shape>user_defined</shape>
     <reference><![CDATA[Diamond.stl]]></reference>
     <scale>
       <x>0.98</x>
       <y>0.98</y>
       <z>-1.05</z>
     </scale>
   </geometry>
```

Fig. 9: Example of XML code under <geometry>

4.1.1. <shape>

Parent element: <geometry>; Link: <voxel>, <grid>

<shape> defines the shape of a voxel, the basic element of 3D model data. In addition to the standard shapes, it is possible to use other shapes by specifying external STL files.

When voxels with <shape> specified are arranged in <grid>, the center of each shape is positioned in the center of its cell.

<shape> contains the following data.

Table 10: The description of <shape>

Element	Attribute	Data type	Data [Default value]	Description	Condition
shape	–	string	cube / sphere / user_defined [cube]	Specifies the shape of a voxel. When "user_defined" is specified, <reference> must also be defined.	

When "user_defined" is specified in <shape>, <reference> must also be defined. The <reference> element specifies an external STL file that defines the shape of a voxel.

Table 11: The description of <reference>

Element	Attribute	Data type	Data [Default value]	Description	Condition
reference	–	CDATA	The path of the STL file defining the shape of a voxel [–]	Specifies the character string of the relative path to the external STL file defining the shape of a voxel.	

See Fig. 9 for an XML code example for <shape>.

4.1.2. <scale>

Parent element: <geometry>; Link: <voxel>, <shape>, <grid>

<scale> defines the scale of a voxel, the basic element of 3D model data. The scale is specified in relation to the cell size specified under <grid>, the element which defines the space in which voxels are arranged. The standard scale is 1×1×1.

When voxels with <scale> specified are arranged in <grid>, the center of each shape is positioned in the center of its cell.

<scale> contains the following elements.

Table 12: The description of <scale>

Element	Attribute	Data type	Data [Default value]	Description	Condition
x	–	double	The scale of the x-axis [1.0]	Specifies the scale for the x-axis of <shape> in relation to the cell size specified in <grid>. When a negative value is specified, the voxel is flipped. 0 must not be specified.	
y	–	double	The scale of the y-axis [1.0]	Specifies the scale for the y-axis of <shape> in relation to the cell size specified in <grid>. When a negative value is specified, the voxel is flipped. 0 must not be specified.	
z	–	double	The scale of the z-axis [1.0]	Specifies the scale for the z-axis of <shape> in relation to the cell size specified in <grid>. When a negative value is specified, the voxel is flipped. 0 must not be specified.	

See Fig. 9 for an XML code example for <scale>.

4.2. <material>

Parent element: <palette>; Link: <voxel>

<material> contains information on the material(s) that constitute each voxel, the basic element of 3D model data.
By arranging voxels for which the material make-up is defined under <material>, the overall material structure of the 3D model data is defined.

<material> has the following attributes.

Table 13: The description of <material>

Element	Attribute	Data type	Data [Default value]	Description	Condition
material	id	positiveInteger	The ID of the material information [1]* *After this value has been used, the default value is increased by 1 each time.	Specifies the unique ID that is used to identify <material>. The same ID must not be used for multiple <material> elements.	Required
	name	string	The name to be defined for the material information [－]	Specifies the name of <material>. The same name should not be used for multiple <material> elements.	

At the level under <material>, at least one of the following elements representing material information is defined.

- <material_name>
- <product_info>
- <standard_name>

It is possible to define multiples of these elements under <material> in case the user cannot use a specified material. In this case, the elements are defined in the order in which their use is recommended.

Also, the following element may be described at the level under <material>. For details, see the section <metadata>.

- <metadata>

Shown below is an example of XML code under <material>.

```
e.g.,
        <material id="2" name="HardMat1">
            <product_info>
                <manufacturer><![CDATA[ABC Materials Co.]]></manufacturer>
                <product_name><![CDATA[ULTRA-HARD/007]]></product_name>
                <url><![CDATA[http://www.abcmaterial.com/ultra/hard/007]]></url>
            </product_info>
            <product_info>
                <manufacturer><![CDATA[ABC Materials Co.]]></manufacturer>
                <product_name><![CDATA[ULTRA-HARD/006a]]></product_name>
                <url><![CDATA[http://www.abcmaterial.com/ultra/hard/006/a]]></url>
            </product_info>
            <standard_name><![CDATA[JIS K6899-1 ABS]]>
            </standard_name>
        </material>
```

Fig. 10: Example of XML code under <material>

4.2.1. <material_name>

Parent element: <material>; Link: –

<material_name> specifies a material by its name. A character string identifying the material (e.g., product name, common name, general material name such as ABS, PLA) is specified.

<material_name> contains the following data.

Table 14: The description of <material_name>

Element	Attribute	Data type	Data [Default value]	Description	Condition
material_name	–	CDATA	The material name [–]	Specifies a character string identifying the material used.	

4.2.2. <product_info>

Parent element: <material>; Link: –

<product_info> specifies a material by its product information. Product information identifying the material (e.g., manufacturer's name, product name, product code, product information URL) is specified.

<product_info> contains the following elements.

Table 15: The description of <product_info>

Element	Attribute	Data type	Data [Default value]	Description	Condition
manufacturer	–	CDATA	The name of the manufacturer of the material [–]	Specifies the character string of the name of the manufacturer of the material.	
product_name	–	CDATA	The product name of the material [–]	Specifies the character string of the product name, product code, etc. provided by the manufacturer.	
url	–	CDATA	The URL of the website providing material information [–]	Specifies the URL of the website providing material information.	

See Fig. 10 for an XML code example for <product_info>.

4.2.3. <standard_name>

Parent element: <material>; Link: –

<standard_name> specifies a material by its name as defined in a standard. In order to clearly specify the standard in which the material is defined, it is necessary to write it using the below format.

```
[standard type] [standard number] [material name specified in standard]
```

e.g.,

Acrylonitrile butadiene styrene (ABS) specified using ISO standard:	ISO 1043-1:2001 ABS
Polycarbonate (PC) specified using JIS standard:	JIS K 6899-1:2006 PC

<standard_name> contains the following element.

Table 16: The description of <iso_standard>

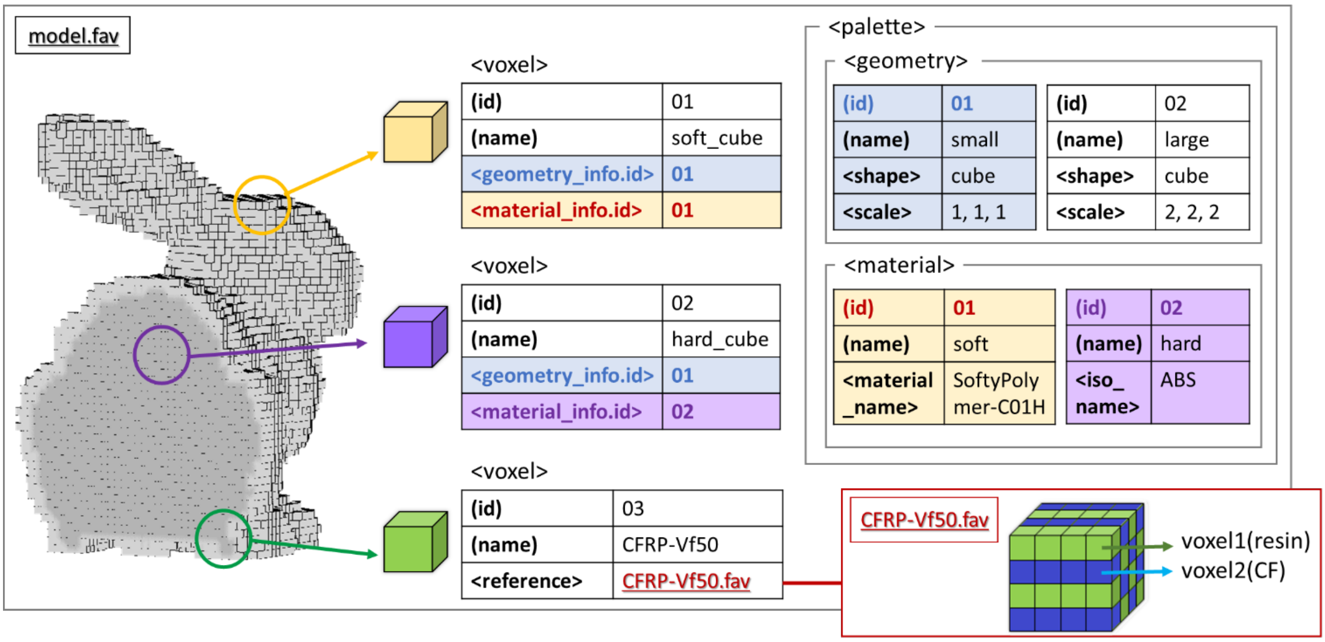
Element	Attribute	Data type	Data [Default value]	Description	Condition
standard_name	–	CDATA	Standard information for a material [–]	Specifies a material by its name as defined in a standard, written in the below format: [standard type] [standard number] [name of material specified in standard]	

See Fig. 10 for an XML code example for <standard_name>.

5. <voxel>

Parent element: <fav>; Link: <geometry>, <material>, <display>, <application_note>, <reference>

<voxel> defines the voxel, which is the basic element constituting 3D model data in the FAV format. By arranging voxels in three-dimensional space, the overall structure of 3D model data is defined. <voxel> references information such as <geometry> and <material>. Therefore, it is possible to define attributes such as colors and materials for each voxel of a 3D model as well as the overall shape of the model.



(id)	01
(name)	soft_cube
<geometry_info.id>	01
<material_info.id>	01

(id)	02
(name)	hard_cube
<geometry_info.id>	01
<material_info.id>	02

(id)	03
(name)	CFRP-Vf50
<reference>	CFRP-Vf50.fav

(id)	01	(id)	02
(name)	small	(name)	large
<shape>	cube	<shape>	cube
<scale>	1, 1, 1	<scale>	2, 2, 2

(id)	01	(id)	02
(name)	soft	(name)	hard
<material_name>	SoftyPolymer-C01H	<iso_name>	ABS

Fig. 11: Example of specifying the overall shape and attributes using <voxel>

<voxel> has the following attributes.

Table 17: The description of <voxel>

Element	Attribute	Data type	Data [Default value]	Description	Condition
voxel	id	positiveInteger	The ID of the voxel [1]* *After this value has been used, the default value is increased by 1 each time.	A positive integer of 1 or greater must be specified as the ID that is used to identify the <voxel> element. The same ID must not be used for multiple <voxel> elements.	Required
	name	string	The name to be defined for the voxel [–]	Specifies the name of the <voxel> element. The same name should not be used for multiple <voxel> elements.	

The (id) of <voxel> is required to be a positive integer. This is because the value is listed as a hexadecimal number in <voxel_map>, the element which defines the shape of a 3D object. Also, 0 must not be used for (id), because this value is reserved to denote the absence of a voxel.

The following elements are described at the level under <voxel>. Alternatively, by defining <reference> only, it is possible to specify an external FAV file, which allows a group of voxels to be treated as a single voxel.

- <geometry_info>
- <material_info>

- <display>
- <application_note>

Shown below is an example of XML code under <voxel>.

```
e.g.,
    <voxel id="1" name="soft_cube">
    <geometry_info>
      <id>1</id>
    </geometry_info>
    <material_info>
      <id>1</id>
      <ratio>1</ratio>
    </material_info>
  </voxel>
  <voxel id="2" name="hard_cube">
    <geometry_info>
      <id>1</id>
    </geometry_info>
    <material_info>
      <id>1</id>
      <ratio>0.15</ratio>
    </material_info>
    <material_info>
       <id>2</id>
      <ratio>0.85</ratio>
    </material_info>
    <application_note><![CDATA[HM-H01:Hybrid Hard Material Number 01]]></application_note>
    <application_note><![CDATA[FabAppAttr : application note]]></application_note>
  </voxel>
  <voxel id="3" name="reserved_cube">
    <geometry_info>
      <id>1</id>
    </geometry_info>
    <material_info>
      <id>0</id>
      <ratio>1</ratio>
    </material_info>
  </voxel>
  <voxel id="4" name="sparse_cube">
    <geometry_info>
      <id>1</id>
    </geometry_info>
    <material_info>
      <id>2</id>
      <ratio>0.6</ratio>
    </material_info>
    <material_info>
      <id>0</id>
      <ratio>0.4</ratio>
    </material_info>
  </voxel>
  <voxel id="5" name="Voxel01">
    <reference><![CDATA[Voxel01.fav]]></reference>
  </voxel>
```

Fig. 12: Example of XML code under <voxel>

5.1. <geometry_info>

Parent element: <voxel>; Link: <palette>, <geometry>

<geometry_info> specifies the shape and scale of a voxel, as defined under <geometry>.

The <id> value specified here must be that value which was specified in <geometry> registered under <palette>. When the shape of 3D model data is defined using voxels that have geometry IDs specified under <geometry_info>, the shapes and scales of each voxel at their respective positions will be the shapes and scales associated with the same IDs specified under <geometry> (Fig. 11)

<geometry_info> contains the following element.

Table 18: The description of <geometry_info>

Element	Attribute	Data type	Data [Default value]	Description	Condition
id	–	positiveInteger	The ID of a predefined shape information element [–]	Specifies the ID of a <geometry> element so that it can be referenced. The predefined (id) attribute of a <geometry> element must be specified here.	Required

See Fig. 12 for an XML code example for <geometry_info>.

5.2. <material_info>

Parent element: <voxel>; Link: <palette>, <material>

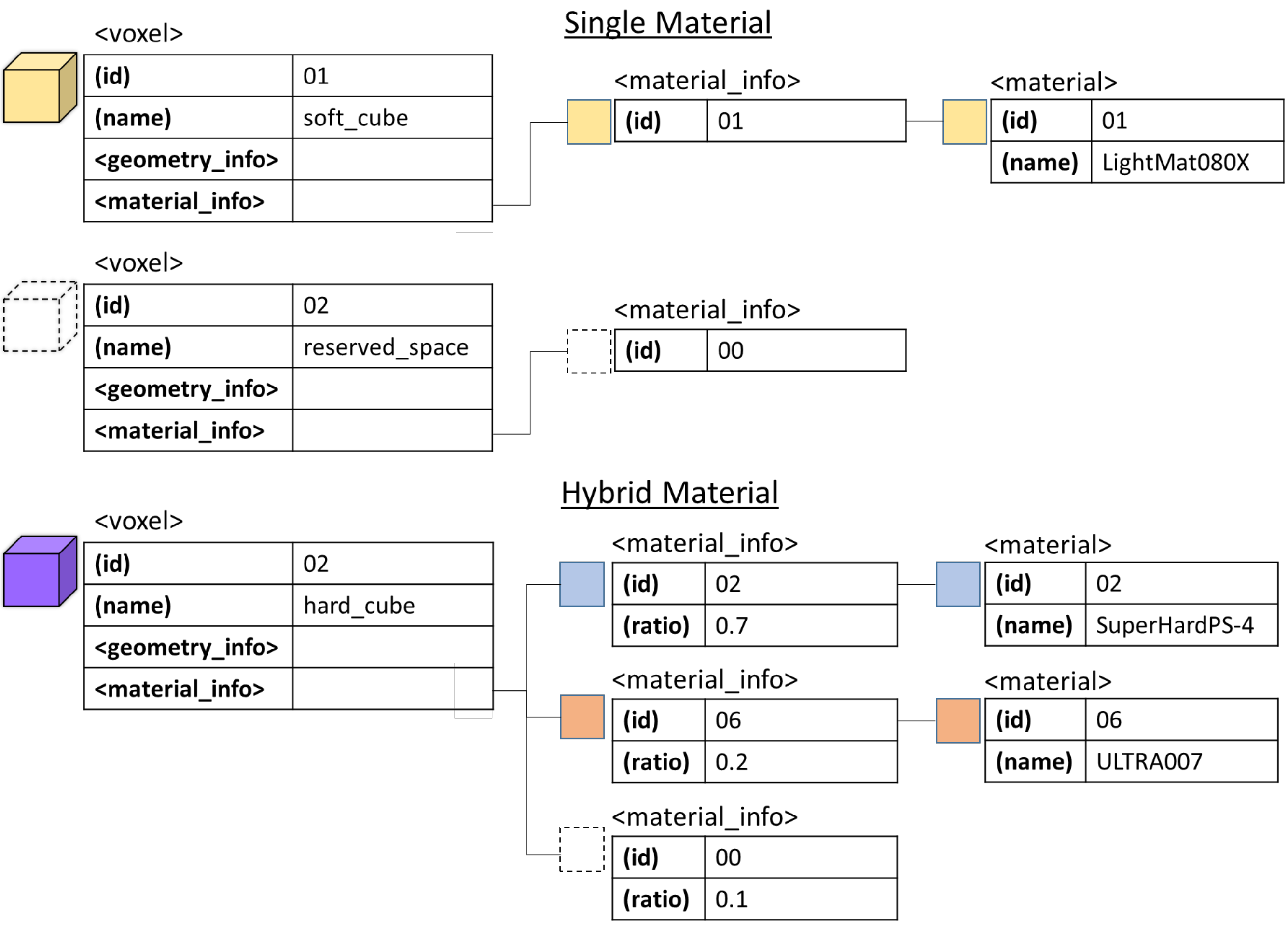
<material_info> specifies the material information of a voxel, as defined under <material>.

The <id> element of <material_info> is either specified to be the same as the (id) attribute of a <material> element (registered under <palette>) or specified as 0. When the (id) of a <material> element (registered under <palette>) is specified as the <id> element, it denotes that the specified <material> is the material composing the area where that voxel is located (Fig. 11).

When 0 is specified as the <id> element, it denotes that the location of that voxel has been secured as an empty space containing no material.

It is also possible to define a composite material by defining multiple <material_info> elements under <voxel> (Fig. 13). When a composite material is defined, the following pair of elements is specified under each <material_info> element defined: <id>, which references a material specified under the <material> element, and <ratio>, which specifies the proportion of that material. Values should be specified so that the sum of the <ratio> values of all <material_info> elements of a given <voxel> element totals to 1.0.

When the material for a voxel is specified as a <material_info> element with the <id> of 0 in combination with another <material_info> element or elements, it signifies that the voxel contains the specified <ratio> of empty space (containing no material) within it.



<voxel>

(id)	01
(name)	soft_cube
<geometry_info>	
<material_info>	

<material_info>

(id)	01

<material>

(id)	01
(name)	LightMat080X

<voxel>

(id)	02
(name)	reserved_space
<geometry_info>	
<material_info>	

<material_info>

(id)	00

<voxel>

(id)	02
(name)	hard_cube
<geometry_info>	
<material_info>	

<material_info>

(id)	02
(ratio)	0.7

<material>

(id)	02
(name)	SuperHardPS-4

<material_info>

(id)	06
(ratio)	0.2

<material>

(id)	06
(name)	ULTRA007

<material_info>

(id)	00
(ratio)	0.1

Fig. 13: Examples of voxels made up of single and composite materials

<material_info> contains the following elements.

Table 19: The description of <material_info>

Element	Attribute	Data type	Data [Default value]	Description	Condition
id	–	positiveI nteger	The ID of a predefined material information element [–]	Specifies the ID of a <material> element so that it can be referenced. Either the predefined (id) attribute of a <material> element or 0 must be specified here.	Required
ratio	–	double	The proportion of the material to be used when multiple materials are combined into a composite material [–]	A double value is used to specify what proportion the material makes up of the whole (defined under <material>). Values should be specified so that they total to 1.0 when added. The value must be greater than 0.	

See Fig. 12 for an XML code example for <material_info>.

5.3. <display>

Parent element: <voxel>; Link: <color_map>

<display> represents the color information used to visually distinguish a <voxel> element. This element is used so that differences in the attributes of voxels, such as the shape and material, can be visually confirmed.

If the <display> element has been defined under a <voxel> element at the time when the shape of the 3D model data was defined, the color of the voxel in the specified location of the 3D model data should be displayed using the color specified under <display>.

When an actual physical object is fabricated using the 3D model data, <color_map> shall be used as the information to define the color of the object.

<display> contains the following elements.

Table 20: The description of <display>

Element	Attribute	Data type	Data [Default value]	Description	Condition
r	–	nonNegativ eInteger	The red component of the display color of a voxel [–]	Specifies the red component of the display color of a voxel. A value less than 0 or a value exceeding 255 must not be specified.	
g	–	nonNegativ eInteger	The green component of the display color of a voxel [–]	Specifies the green component of the display color of a voxel. A value less than 0 or a value exceeding 255 must not be specified.	
b	–	nonNegativ eInteger	The blue component of the display color of a	Specifies the blue component of the display color of a voxel. A value less	

			voxel [–]	than 0 or a value exceeding 255 must not be specified.	
a	–	nonNegativeInteger	The alpha component of the display color of a voxel [–]	Specifies the alpha component of the display color of a voxel. A value less than 0 or a value exceeding 255 must not be specified.	

5.4. <application_note>

Parent element: <voxel>; Link: –

<application_note> defines properties to be stored in a <voxel> element.

It is necessary to add one <application_note> element for each property to be stored. Also, it is recommended that descriptions be added to assist users (both those who created the properties and those who use them) in distinguishing and selecting the necessary data from multiple <application_note> elements.

Note that it is not recommended for large data to be stored or for large numbers of <application_note> elements to be defined unnecessarily.

<application_note> contains the following data.

Table 21: The description of <application_note>

Element	Attribute	Data type	Data [Default value]	Description	Condition
application_note	–	CDATA	Property information to be stored in a voxel. [–]	Specifies the character string of a property to be stored in a <voxel> element. It is recommended that descriptions be added to help users distinguish and select the necessary data from multiple <application_note> elements.	

See Fig. 12 for an XML code example for <application_note>.

5.5. <reference>

Parent element: <voxel>; Link: −

<reference> can be used to specify an external FAV file, allowing a group of voxels to be used as a single voxel.

When a child FAV file (the external FAV file from which data is read) is specified to be a single voxel of the parent FAV file (the file into which data is read), the following relationships between the two should be maintained.

<unit.**x**> of parent FAV file = (<unit.**x**> of child FAV file) x (<dimension.**x**> of child FAV file)

<unit.**y**> of parent FAV file = (<unit.**y**> of child FAV file) x (<dimension.**y**> of child FAV file)

<unit.**z**> of parent FAV file = (<unit.**z**> of child FAV file) x (<dimension.**z**> of child FAV file)

Fig. 14 is an example of defining a voxel by referring to an external FAV file. The file Bunny.fav, which defines the actual model data, uses the file Voxel01.fav as one voxel. Voxel01.fav consists of a group of voxels with edges each measuring 1 mm arranged in a 10 x 10 x 10 voxel formation. Furthermore, the file Voxel01.fav uses the file SubVoxel01.fav as one voxel. SubVoxel01.fav consists of a group of voxels with edges each measuring 0.01 mm arranged in a 100 x 100 x 100 voxel formation.

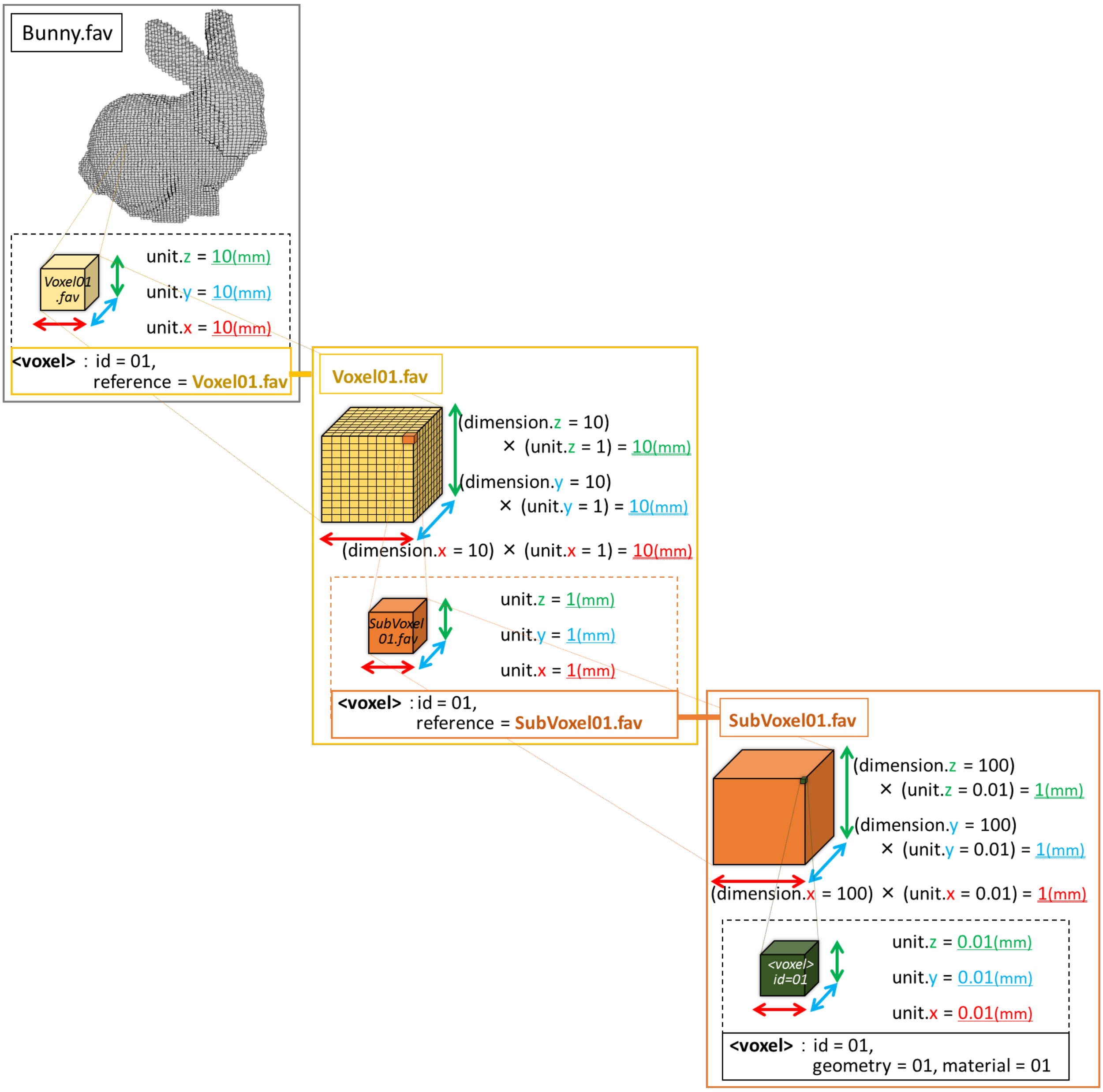


Fig. 14: Example of defining voxels by referring to external FAV files (1)

Fig. 15 shows an example in which multiple types of voxels are defined by referring to external FAV files.

The file that defines the actual model data, Bunny.fav, uses Voxel02.fav as one voxel. Voxel02.fav consists of a group of voxels with x-edges measuring 1 mm, y-edges measuring 2.5 mm, and z-edges measuring 0.5 mm arranged in a 10 x 4 x 20 voxel formation.

Furthermore, in Voxel02.fav, three different types of voxels are defined as follows.

- <voxel> id = 01 ... uses the external file SubVoxel_A.fav as one voxel. SubVoxel_A.fav consists of a group of voxels with x-edges measuring 0.05 mm, y-edges measuring 0.05 mm, and z-edges measuring 0.01 mm arranged in a 20 x 50 x 50 voxel formation.

- <voxel> id = 02 ... uses the external file SubVoxel_B.fav as one voxel. SubVoxel_B.fav consists of a group of voxels with x-edges measuring 0.01 mm, y-edges measuring 0.01 mm, and z-edges measuring 0.01 mm arranged in a 100 x 250 x 50 voxel formation.

- <voxel> id = 03 ... a voxel defined internally within Voxel02.fav. Has an x-edge measuring 1 mm, a y-edge measuring 2.5 mm, and a z-edge measuring 0.5 mm.

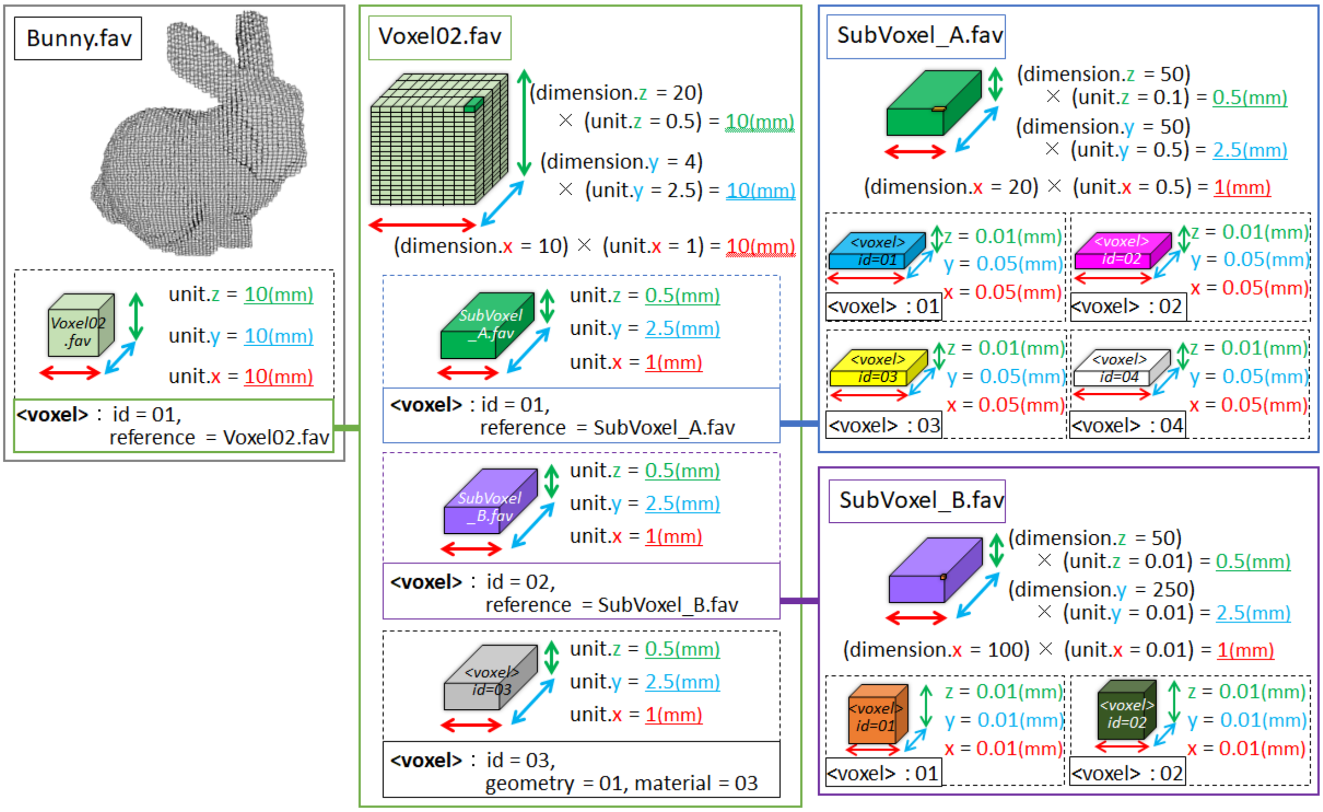


Fig. 15: Example of defining voxels by referring to external FAV files (2)

<reference> contains the following data.

Table 22: The description of <reference>

Element	Attribute	Data type	Data [Default value]	Description	Condition
reference	–	CDATA	The path of a FAV file defining a group of voxels to be treated as a single voxel [–]	Specifies the character string of the relative path to an external FAV file which is to be treated as a single voxel.	

See Fig. 12 for an XML code example for <reference>.

6. <object>

Parent element: <fav>; Link: <voxel>, <grid>, <structure>

<object> defines 3D model data in FAV format to create the actual form of the final 3D object.

Underneath <object> are the <grid> and <structure> elements. <grid> defines the space that stores 3D model data, and within this space, <structure> defines the structure of the 3D model data. 3D model data is further divided into <voxel_map>, which defines the shape, <color_map>, which defines color information, and <link_map>, which defines the connection strength between voxels. It is also possible to map a user's unique information (<user_defined_map>) onto the cells of <grid>.

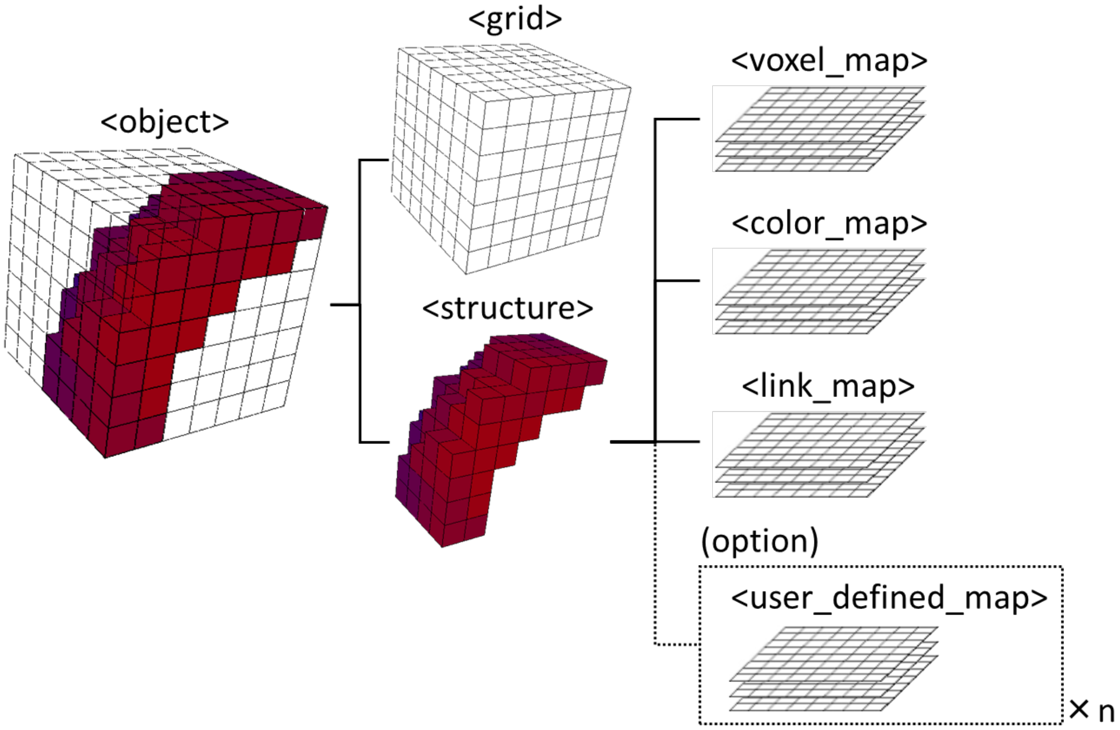


Fig. 16: The relationship between grid and structure

Multiple <object> elements can be defined in a FAV file. When multiple <object> elements are defined, <grid> and <structure> are defined for each <object> element.

<object> has the following attributes.

Table 23: The description of <object>

Element	Attribute	Data type	Data [Default value]	Description	Condition
object	id	positiveInteger	The ID of the object [1]* *After this value has been used, the default value is increased by 1 each time.	Specifies the ID that is used to identify the <object> element. The same ID must not be used for multiple <object> elements.	Required
	name	string	The name to be defined for the object [Object001~]	Specifies the name of the <object> element. The same name should not be used for multiple <object> elements.	

The following elements must be described at the level under <object>.

- <grid>
- <structure>

Also, the following element may be described at the level under <object>. For details, see the section <metadata>.

- <metadata>

Shown below is an example of XML code under <object>.

```
e.g.,
  <object id="1" name="SampleObject">
    <grid>
      <origin>
        <x>28.5</x>
        <y>-30</y>
        <z>0</z>
      </origin>
      <unit>
        <x>1</x>
        <y>1</y>
        <z>1</z>
      </unit>
      <dimension>
        <x>7</x>
        <y>7</y>
        <z>7</z>
      </dimension>
    </grid>
    <structure>
      <voxel_map compression="none" bit_per_voxel="8">
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
      </voxel_map>
      <color_map compression="none" color_mode="RGB">
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
      </color_map>
      <link_map bit_per_link="8" compression="none" neighbors="6">
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
        <layer><![CDATA[……]]></layer>
      </link_map>
      <user_defined_map value_type="byte" compression="none">
        <reference><![CDATA[StressHeatmap.favmapx]]></reference>
      <metadata>
        <id>fa23e6c1-e52e-4591-b354-e4cfa382571a</id>
        <title><![CDATA[Stress Heatmap]]></title>
        <author><![CDATA[FUJIFILM Business Innovation & Keio SFC]]></author>
        <license><![CDATA[CC BY]]></license>
        <note><![CDATA[This file is FAVMAP format on ver1.1.]]></note>
      </metadata>
    </user_defined_map>
    <user_defined_map value_type="float" compression="none">
      <reference><![CDATA[Thermography.favmapx]]></reference>
      <metadata>
        <id>84343e14-ad81-49e9-bf1e-884dcfa80c34</id>
        <title><![CDATA[Thermography sensing data]]></title>
        <author><![CDATA[FUJIFILM Business Innovation & Keio SFC]]></author>
        <license><![CDATA[CC BY]]></license>
      </metadata>
    </user_defined_map>
  </structure>
</object>
```

Fig. 17: Example of XML code under <object>

6.1. <grid>

Parent element: <object>; Link: <voxel>, <shape>, <scale>

<grid> defines the space in which 3D model data in FAV format is defined. The space is divided into cells of uniform size. The structure of 3D model data is defined based on the definition of <grid>. When multiple <object> elements are defined in a FAV file, <grid> is defined for each <object> element.

The following elements are specified at the level under <grid>.

- <origin>
- <unit>
- <dimension>

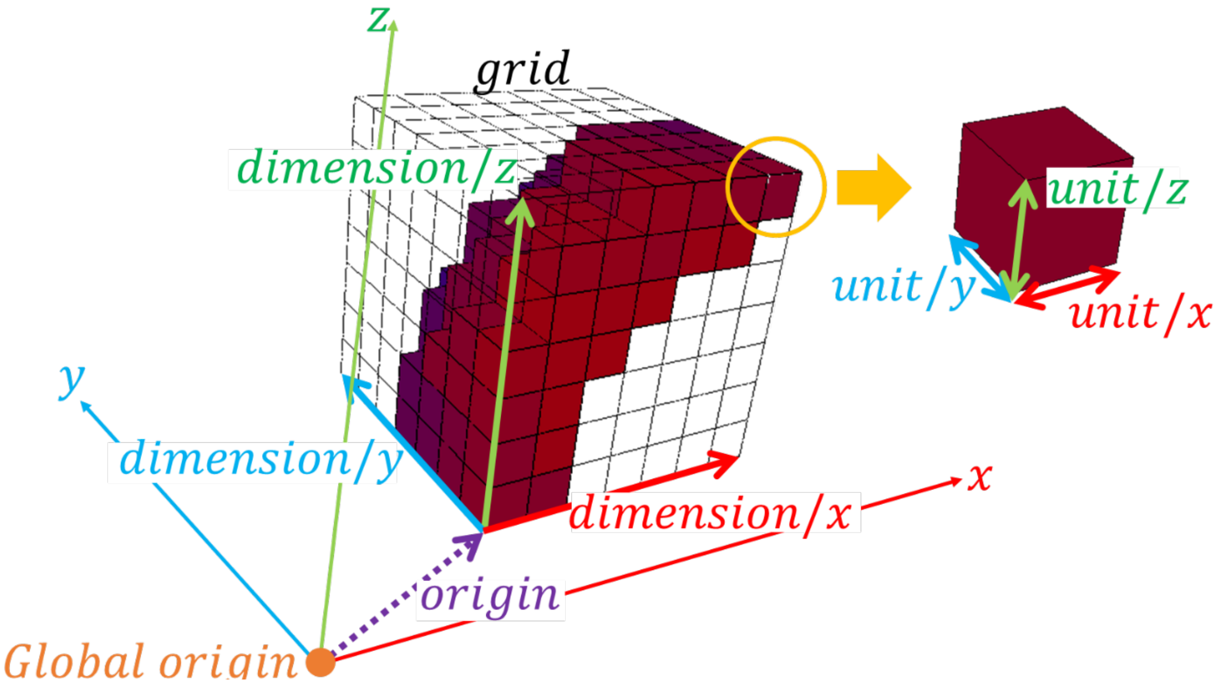


Fig. 18: Visual description of each element in <grid>

6.1.1. <origin>

Parent element: <grid>; Link: –

<origin> defines the distance from the origin of the global coordinate system to the origin of the coordinate system of <grid>, in so specifying the position of an object in the global coordinate system.

<origin> contains the following elements.

Table 24: The description of <origin>

Element	Attribute	Data type	Data [Default value]	Description	Condition
x	–	double	The distance to the origin of the coordinate system for the x-axis [0.0]	Specifies the distance from the origin of the global coordinate system to the origin of the coordinate system of <grid> for the x-axis.	
y	–	double	The distance to the origin of the coordinate system for the y-axis [0.0]	Specifies the distance from the origin of the global coordinate system to the origin of the coordinate system of <grid> for the y-axis.	
z	–	double	The distance to the origin of the coordinate system for the z-axis [0.0]	Specifies the distance from the origin of the global coordinate system to the origin of the coordinate system of <grid> for the z-axis.	

See Fig. 17 for an XML code example for <origin>.

6.1.2. <unit>

Parent element: <grid>; Link: <voxel>, <geometry>, <shape>

<unit> specifies the size of a cell within <grid>. The size can be specified for each of the axes: x, y, and z.

From <unit>, it is possible to calculate the number of voxels contained within a space of 1 mm (vpm = voxels per mm) as follows.

1 (voxel) / unit (mm) = vpm

e.g.,	unit = (1, 1, 1)	In this case:	x = 1 vpm,	y = 1 vpm,	z = 1 vpm
	unit = (10, 10, 5)	In this case:	x = 0.1 vpm,	y = 0.1 vpm,	z = 0.2 vpm
	unit = (0.01, 0.002, 0.05)	In this case:	x = 100 vpm,	y = 500 vpm,	z = 20 vpm

<unit> contains the following elements.

Table 25: The description of <unit>

Element	Attribute	Data type	Data [Default value]	Description	Condition

x	–	double	The length of each cell in the x-axis direction [1.0]	Specifies the length of a single cell in the direction of the x-axis. The value must be greater than 0.	
y	–	double	The length of each cell in the y-axis direction [1.0]	Specifies the length of a single cell in the direction of the y-axis. The value must be greater than 0.	
z	–	double	The length of each cell in the z-axis direction [1.0]	Specifies the length of a single cell in the direction of the z-axis. The value must be greater than 0.	

See Fig. 17 for an XML code example for <unit>.

6.1.3. <dimension>

Parent element: <grid>; Link: <voxel>, <unit>

<dimension> defines the overall size of a grid. The size is specified by determining the maximum number of voxels that can be arranged in a grid. The actual size of a grid in the global coordinate system can be calculated by multiplying the cell size defined in <unit> by the number of cells defined in <dimension>.

<dimension> contains the following elements.

Table 26: The description of <dimension>

Element	Attribute	Data type	Data [Default value]	Description	Condition
x	–	positiveInteger	The number of cells in the x-axis direction [–]	Specifies the maximum number of voxels that can be arranged in <grid> in the x-axis direction	Required
y	–	positiveInteger	The number of cells in the y-axis direction [–]	Specifies the maximum number of voxels that can be arranged in <grid> in the y-axis direction.	Required
z	–	positiveInteger	The number of cells in the z-axis direction [–]	Specifies the maximum number of voxels that can be arranged in <grid> in the z-axis direction.	Required

See Fig. 17 for an XML code example for <dimension>.

6.2. <structure>

Parent element: <object>; Link: <voxel_map>, <color_map>, <link_map>, <user_defined_map>

<structure> defines the structure of 3D model data in FAV format. The shape of a 3D object is defined by arranging voxels (defined by <voxel>) within a three-dimensional grid (defined by <grid>). Attributes such as the color, material, and connection strength of each arranged voxel are defined in each type of map element.

The following elements are described at the level under <structure>.

- <voxel_map>
- <color_map>
- <link_map>
- <user_defined_map>

By defining <user_defined_map>, it is possible to map a user's unique attributes onto the cells of <grid>. It is possible to define multiple <user_defined_map> elements.

The following sections explain each of the elements located under <structure>. To do so, an example is used in which the following 3D model data is defined.

See Fig. 30 for an XML code example for <structure>.

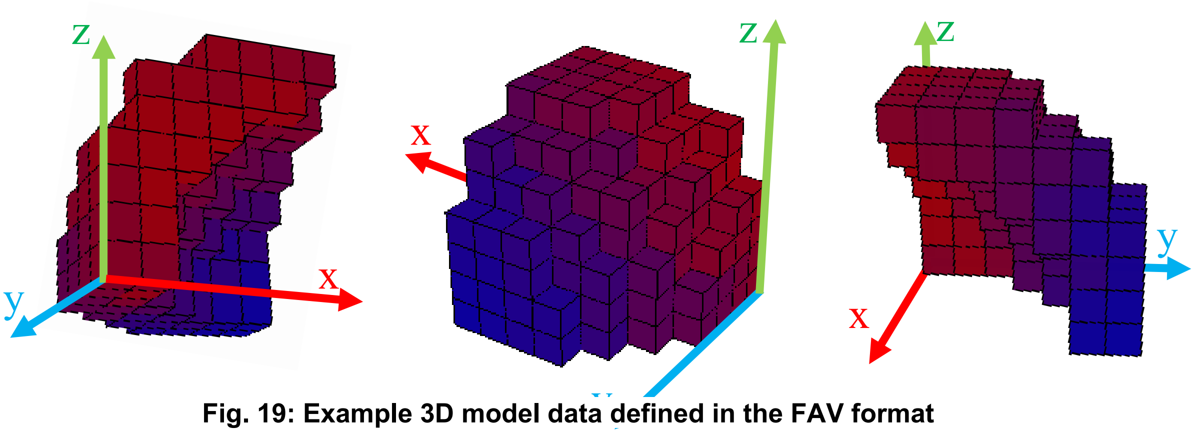


Fig. 19: Example 3D model data defined in the FAV format

6.2.1. <voxel_map>

Parent element: <structure>; Link: <grid>, <voxel>

<voxel_map> defines the overall shape of 3D model data in FAV format. In a voxel map, it is necessary to define whether a voxel is present or not at each voxel position of an x-y layer (horizontal layer defined on the x and y axes), for each layer of a three-dimensional grid. By defining the voxel arrangement in the <voxel_map> element for each layer and repeating this for the number of layers comprising the height of the object, the overall shape of a 3D object is defined.

<voxel_map> has the following attributes.

Table 27: The description of <voxel_map>

Element	Attribute	Data type	Data [Default value]	Description	Condition
voxel_map	bit_per_voxel	positiveInteger	4 / 8 / 16 [–]	Specifies a numerical value of 4, 8, or 16 (representing the length of hexadecimal characters) to define the number of bits in one voxel. (4, 8, and 16 each indicate one, two, and four hexadecimal characters respectively.)	Required
	compression	string	none / base64 / zlib / runlength [none]	Selects the compression method for <layer> from the following: none (no compression), base64, zlib, and runlength.	

<layer> elements are defined at the level under <voxel_map>. The number of <layer> elements to be defined is determined by <dimension.z>. <layer> elements are stored starting from the lowest layer on the z-axis (i.e., the base of the 3D object).

One <voxel_map> element is defined for each x-y layer (<layer>). When the (id) of <voxel> is written in a specific position in <voxel_map>, it indicates that the voxel associated to that ID is present at that specified position in the grid. When no voxel is to be present in a position, 0 is written in the corresponding position in <voxel_map>.

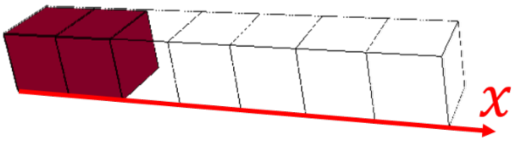
<voxel_map.layer> contains the following data.

Table 28: The description of <voxel_map.layer>

Element	Attribute	Data type	Data [Default value]	Description	Condition
layer	–	CDATA	The character string listing the ID of each voxel in the layer [–]	Lists the (id) values in <voxel> using the format specified in (bit_per_voxel). Each (id) value is concatenated into one line (the number of IDs to be concatenated can be calculated by multiplying <dimension.x> by <dimension.y>). When (compression) in <voxel_map> is set to that other than "none", the data in <layer> is compressed using the specified method and stored.	

Shown below is the procedure for defining the voxel map for the first layer of the 3D model data shown in Fig. 19. (The cells of the voxel maps to the right of the figures below are colored solely for the purpose of improving readability, and have no effect on the actual 3D model.)

① The presence or absence of a voxel is defined for each position along the x-axis (<dimension.x>) in <grid> by specifying the (id) values of <voxel>. A character string is formed by concatenating the (id) values of each <voxel> element. 0 is specified to denote the absence of a voxel at a certain position.

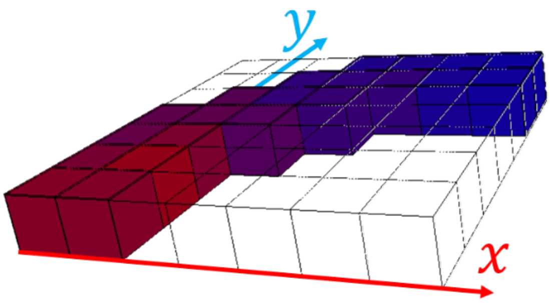


y_1	01	01	00	00	00	00	00
	x_1	x_2	x_3	x_4	x_5	x_6	x_7

(<dimension.x> = 7)

Fig. 20: <voxel_map> for the x-axis (<dimension.x>)

② The process in step ① (specifying (id) values for <dimension.x>) is carried out for each row on the y-axis (i.e., the number of rows defined by <dimension.y>) to create the voxel map.



y_7	00	00	00	02	02	02	02
y_6	00	00	01	02	02	02	02
y_5	00	01	01	01	00	00	00
y_4	01	01	01	00	00	00	00
y_3	01	01	00	00	00	00	00
y_2	01	01	00	00	00	00	00
y_1	01	01	00	00	00	00	00
	x_1	x_2	x_3	x_4	x_5	x_6	x_7

(<dimension.x> = 7, <dimension.y> = 7)

Fig. 21: <voxel_map> for one layer (<dimension.x> x <dimension.y>)

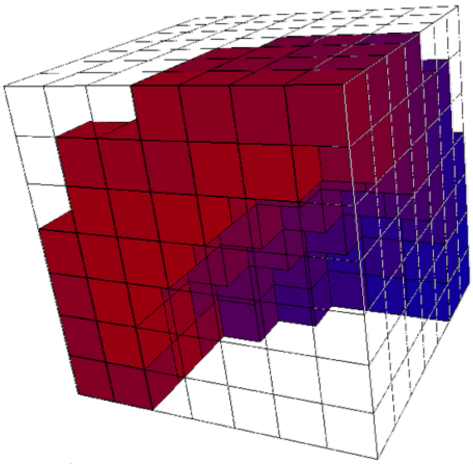
③ The data for a single <layer> element is obtained by concatenating the values defined for each position on the x-y layer (<dimension.x> x <dimension.y>) in <voxel_map>.

y_1							y_2							y_7						
01	01	00	00	00	00	00	01	01	00	00	…	02	02	00	00	00	02	02	02	02
x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_1	x_2	x_3	x_4		x_6	x_7	x_1	x_2	x_3	x_4	x_5	x_6	x_7

<layer>0101000000000001010000…020200000002020202</l

Fig. 22: The data of a single <layer> obtained by concatenating the values of <voxel_map>

④ The process shown above for creating a voxel map for a single layer is carried out for the number of layers comprising the z-axis (i.e., the number defined by <dimension.z>).



<layer>0101000000000001010000…020200000002020202</layer

…

<layer>0000000101010100000001010101000000…000000</layer

	y_1							y_2								y_7						
z_1	01	01	00	00	00	00	00	01	01	00	00	…	02	02	00	00	00	02	02	02	02
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_1	x_2	x_3	x_4		x_6	x_7	x_1	x_2	x_3	x_4	x_5	x_6	x_7

⋮ (<dimension.z> = 7)

	y_1							y_2							y_3						
z_7	00	00	00	01	01	01	01	00	00	00	01	01	01	01	00	00	00	…	00	00	00
	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_1	x_2	x_3	x_4	x_5	x_6	x_7	x_1	x_2	x_3		x_5	x_6	x_7

Fig. 23: <voxel_map> for the overall 3D model data

6.2.2. <color_map>

Parent element: <structure>; Link: <grid>, <voxel>, <display>

<color_map> defines the color information of 3D model data in FAV format.
Color information is specified for each <voxel> element listed in <voxel_map>.

<color_map> has the following attributes.

Table 29: The description of <color_map>

Element	Attribute	Data type	Data [Default value]	Description	Condition
color_map	color_mode	string	GrayScale / GrayScale16 / RGB / RGBA / CMYK [−]	Selects the color mode defined for each voxel from the following: GrayScale, GrayScale16, RGB, RGBA, CMYK.	Required
	compression	string	none / base64 / zlib / runlength [none]	Selects the compression method for <layer> from the following: none (no compression), base64, zlib, and runlength.	

The following are the color modes available for (color_mode).

- GrayScale (white to black) ... One byte per channel is used, allowing the use of 256 shades. The color information for one voxel is written in one byte (two hexadecimal characters).

- GrayScale16 (white to black)... Two bytes per channel are used, allowing the use of 65,536 shades. The color information for one voxel is written in two bytes (four hexadecimal characters).
- RGB ... One byte is used for each channel of red, green, and blue, allowing the use of 256 shades per channel.
  The color information for one voxel is written in three bytes (six hexadecimal characters).
- RGBA ... One byte is used for each channel of red, green, blue, and alpha, allowing the use of 256 shades per channel.
  The color information for one voxel is written in four bytes (eight hexadecimal characters).
- CMYK ... One byte is used for each channel of cyan, magenta, yellow, and key plate, allowing the use of 256 shades per channel.
  The color information for one voxel is written in four bytes (eight hexadecimal characters).

<layer> elements are defined at the level under <color_map>. The number of <layer> elements to be defined is determined by <dimension.z>. One <color_map> element is defined for each x-y layer (<layer>). In <color_map>, color information for each voxel listed in <voxel_map> is specified in the mode specified by (color_mode). When 0 is specified for a voxel in <voxel_map> (denoting the absence of a voxel at a certain position), the color information of the voxel is omitted in <color_map> and the values are left-aligned.

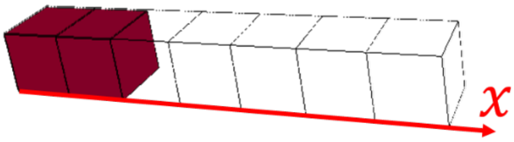
<color_map.layer> contains the following data.

Table 30: The description of <color_map.layer>

Element	Attribute	Data type	Data [Default value]	Description	Condition
layer	–	CDATA	A hexadecimal character string representing the color information of the 3D model data [–]	Specifies the color information of each voxel in <voxel_map> in the mode specified by (color_mode). When (compression) of <color_map> is set to that other than "none", data in <layer> is compressed using the specified method and stored.	One or more

Shown below is the procedure for defining the color map for the first layer of the 3D model data shown in Fig. 19. (The cells of the color maps to the right of the figures below are colored solely for the purpose of improving readability, and have no effect on the actual 3D model.)

① The color information of each voxel on the x-axis (<dimension.x>) in a grid is defined as specified by (color_mode). When this is specified as 0 (denoting the absence of a voxel), color information is omitted.



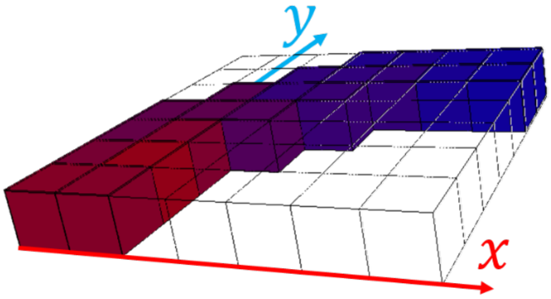
y_1	01	01	00	00	00	00	00
	x_1	x_2	x_3	x_4	x_5	x_6	x_7

$$\text{voxel\_}y_1x_1 = RGB(131,0,37) \rightarrow 830025$$
$$\text{voxel\_}y_1x_2 = RGB(129,0,39) \rightarrow 810027$$

The remaining voxel color information is omitted as the values are 0.
The value of <color_map> for the x-axis (<dimension.x>) is 830025810027.

Fig. 24: <color_map> for the x-axis (<dimension.x>)

② The process in ① (specifying color information for <dimension.x>) is carried out for each row of the y-axis (i.e., the number of rows defined by <dimension.y>) to create the color map. The data for a single <layer> is obtained by concatenating the values defined on the x-y layer (<dimension.x> × <dimension.y>) of <color_map>.



y_7	00	00	00	02	02	02	02
y_6	00	00	01	02	02	02	02
y_5	00	01	01	01	00	00	00
y_4	01	01	01	00	00	00	00
y_3	01	01	00	00	00	00	00
y_2	01	01	00	00	00	00	00
y_1	01	01	00	00	00	00	00
	x_1	x_2	x_3	x_4	x_5	x_6	x_7

An example of the color map data for 21 voxels in one layer
<Layer>8300258100277600329100176400457c002d5e004a5c004c500059560052330075370071300
0782f007a3100771800900f00991f00891c008c1300960c009c</Layer>

Fig. 25: <color_map> for one layer (<dimension.x> × <dimension.y>)

③ The process above for creating a color map for a single layer is carried out for the number of layers comprising the z-axis (i.e., the number defined by <dimension.z>).

6.2.3. <link_map>

Parent element: <structure>; Link: <grid>, <voxel>

<link_map> defines the link information of 3D model data in FAV format. Link information is that information which defines the degree of relation between each voxel (e.g., connection strength). Link information can be used to achieve more precise structure analysis and to generate tool paths of 3D printers, etc. Link information is defined for each voxel listed in <voxel_map>.

<link_map> has the following attributes.

Table 31: The description of <link_map>

Element	Attribute	Data type	Data [Default value]	Description	Condition
link_map	bit_per_link	positiveInteger	4 / 8 / 16 [−]	Specifies a numerical value of 4, 8, or 16 (representing one, two, and four hexadecimal characters respectively) to define how many bits (hexadecimal characters) are used to write each link information.	Required
	neighbors	positiveInteger	6 / 18 / 26 [−]	Specifies a numerical value of 6, 18, or 26 to define the number of neighboring voxels of the voxel in question for which link information is retained.	Required
	compression	string	none / base64 / zlib / runlength [none]	Selects the compression method for <layer> from the following: none (no compression), base64, zlib, and runlength.	

The following are three options for specifying the number of voxels that neighbor a certain voxel (neighbors) for which link information is retained (Fig. 26).

- 6 neighboring voxels ... Retains the link information between a specific voxel and the six neighboring voxels with which its six faces are in full and complete contact.
- 18 neighboring voxels ... Retains the link information between a specific voxel and the 18 neighboring voxels with which the voxel in question is in contact by face OR edge (excludes neighbors only in contact by corner).
- 26 neighboring voxels ... Retains the link information between a specific voxel and the 26 neighboring voxels with which the voxel in question is in contact by face OR edge OR corner (includes neighbors only in contact by corner).

The number of hexadecimal characters used to write the link information retained for a single voxel is as follows: the number of hexadecimal characters specified by (bit_per_link) multiplied by the number of neighboring voxels specified by (neighbors).
Each of the examples (1) and (2) below is an excerpt describing the link information for a single voxel within <link_map>.

```
(1) e.g.,  <link_map bit_per_link=4, neighbors=6>
                      <layer>...004af0...</layer>
           </link_map>
```

In the case of one hexadecimal character per link x six neighboring voxels,
the link information for a single voxel is written using six hexadecimal characters.

```
(2) e.g.,   <link_map bit_per_link=16, neighbors=26>
                      <layer>...
                                 000000000000004e01a90000000000000000000f
                                 0cbe0000100d0000005000000000000000000000
                                 00200ff6000000000000000...
                      </layer>
            </link_map>
```

In the case of four hexadecimal characters per link x 26 neighboring voxels, the link information for a single voxel is written using 104 hexadecimal characters.

<layer> elements are defined at the level under <link_map>. The number of <layer> elements to be defined is determined by <dimension.z>. Link information is defined for each x-y layer (<layer>). Within each <layer>, link information for each voxel listed in <voxel_map> is specified using the number of bytes required for the format specified by (neighbors). When the value 0 is specified for a voxel in <voxel_map> (denoting the absence of a voxel in that position), the link information of the voxel is omitted in <link_map> and the values are left-aligned.

When there are no neighboring voxels, 0 is specified for link information. Link information must be listed for each voxel defined as a neighbor, i.e., the number of bytes required for the format specified by (neighbors).

The order in which link information is defined in <link_map> is as shown below (Fig. 26). Link information is defined starting from the voxel with the lowest coordinate values under the axis ordering of z, y, x. The red voxel in the figure below indicates a reference voxel, and the numbers illustrate the order in which link information is defined.

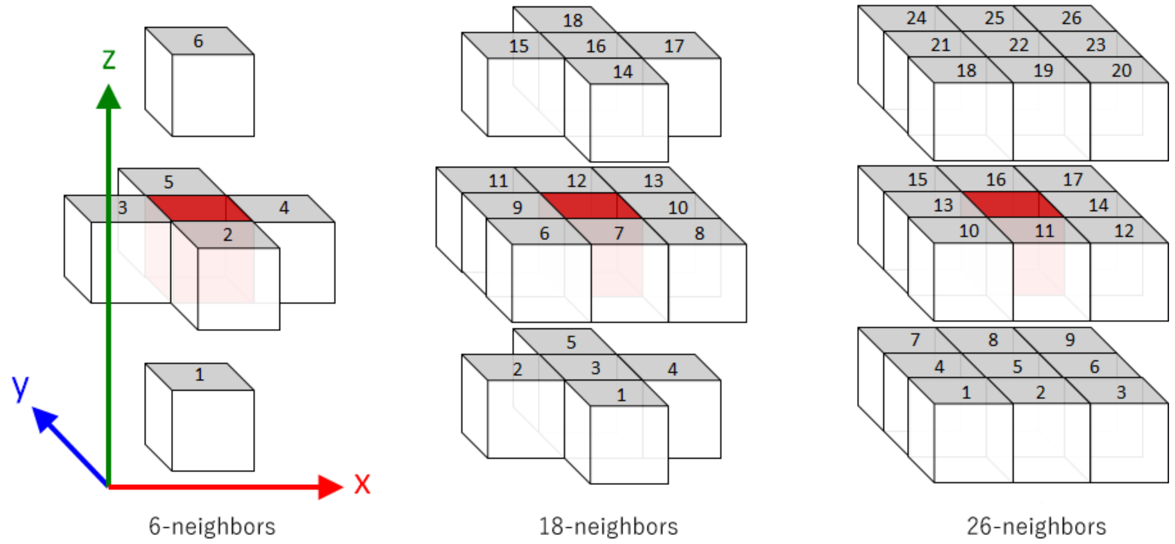


Fig. 26: The order in which link information is defined

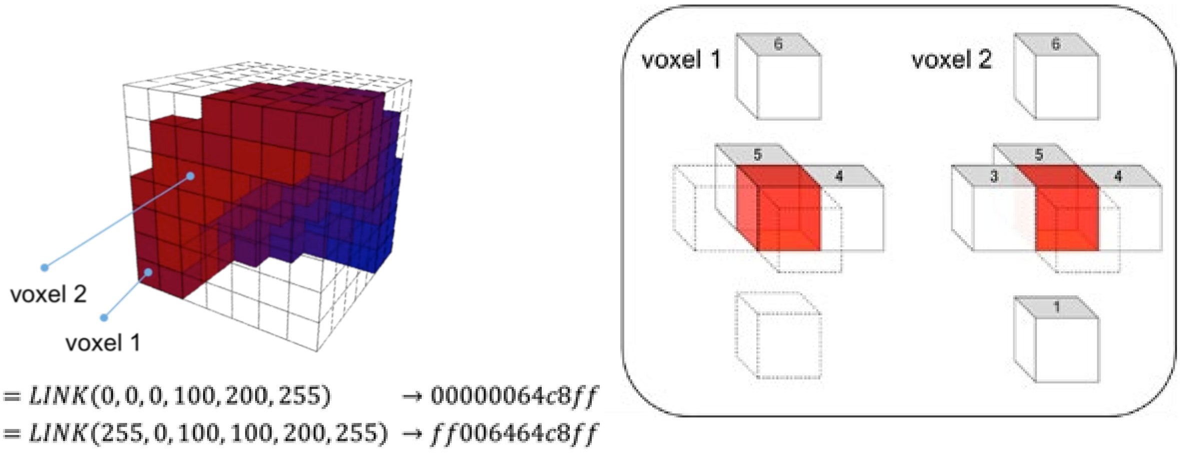
<link_map.layer> contains the following data.

Table 32: The description of <link_map.layer>

Element	Attribute	Data type	Data [Default value]	Description	Condition
layer	–	CDATA	A hexadecimal character string representing the link information of the 3D model data [–]	Specifies the link information in the order shown in Fig. 26 for each voxel in <voxel_map> as specified by (neighbors). When (compression) of <link_map> is set to that other than "none", data in <layer> is compressed using the specified method and stored.	One or more

Shown below is an example of the link map data specified for two voxels in the 3D model data of Fig. 19.

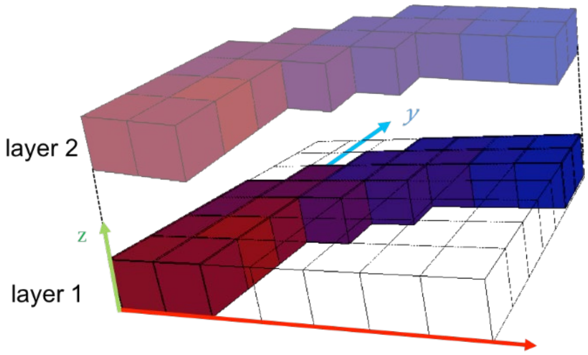
When (neighbors) is set to six and the x, y, and z coordinate values of the link information are 100, 200, and 255 respectively, the values of link information for voxel 1 and voxel 2 in Fig. 27 will be as shown below.



$$\mathrm{voxel1} = LINK(0,0,0,100,200,255) \rightarrow 00000064c8ff$$
$$\mathrm{voxel2} = LINK(255,0,100,100,200,255) \rightarrow ff006464c8ff$$

Fig. 27: Link information for neighboring voxels

The link information map for a single layer is defined by specifying the link information for each voxel in the layer. The number of voxels for which link information is defined can be calculated by multiplying <dimension.x> by <dimension.y>. The data of a single <layer> is obtained by concatenating the link information values of each voxel. To define the link information of the overall 3D model data, the link information map created for a single layer is created for each layer comprising the z-axis (i.e., the number defined by <dimension.z>).



An example of the link map data for 21 voxels in layer 1

<layer>00000064c8ff00000000c8ff00000064c8ff00000000c8ff00c80064c8ff00c80000c8ff00c8006400ff00c80064c8ff00006400c8ff00c8006400ff00c86464c8ff00006400c8ff00c8006400ff00c86464c8ff00006464c8ff00006464c8ff00006400c8ff00c8006400ff00c8646400ff00c8646400ff00c8640000ff</layer>

Fig. 28: <link_map> for one layer (<dimension.x> × <dimension.y>)

6.2.4. <user_defined_map>

Parent element: <structure>; Link: <.favmapx>, <.favmap>

<user_defined_map> defines a map used to add the user's unique attribute to each of the cells of a <grid> composing the 3D model data, which are defined by the FAV format. Here, the file to which user-defined attributes are applied is referred to as the parent file.

For a user-defined attribute, a single attribute value is mapped to each cell of <grid> of the parent file. By defining multiple maps for user-defined attributes, it is possible to map multiple attributes to each cell of <grid> of the parent file.

In <user_defined_map>, defining <reference> allows a user-defined attribute to be read from an external file. In this external file, the user-defined attribute can be written in either XML format (with the file extension .favmapx) or binary format (with the file extension .favmap). The details of user defined attribute files are explained in the following sections.

<user_defined_map> has the following attributes.

Table 33: The description of <user_defined_map>

Element	Attribute	Data type	Data [Default value]	Description	Condition
user_defined_map	value_type	string	byte / short / ushort / int / uint / float / double [byte]	Specifies the data type of the attribute value (used for one voxel) of the the user-defined attribute stored in the external file.	Required
	compression	string	none / base64 / zlib /runlength [none]	Selects the compression method for <layer> stored in the external file from the following: none (no compression), base64, zlib, and runlength. (Only available when a file in the XML format (.favmapx) is used.)	

In the level under <user_defined_map>, the following elements are defined.

- <reference>
- <metadata>

Defining <metadata> is recommended to make it possible for not only the original user who defined <user_defined_map> but also others who will use the data to determine the type of attribute defined / whether loading data is necessary.

Table 34: The description of <user_defined_map.reference> and <user_defined_map.metadata>

Element	Attribute	Data type	Data [Default value]	Description	Condition
reference	–	CDATA	The path of the external file containing the user-defined attribute [–]	Specifies the character string of the relative path to the external .favmap/.favmapx file containing the user-defined attribute.	Required
metadata	–	See <metadata>.		Describes information such as the ID used to identify the user-defined attribute, its name (Title), author (Author), license (License), and an explanantion of the attribute (note).	

6.2.4.1. User-defined attribute files: XML format (.favmapx)

Parent element: <user_defined_map>; Link: <grid>, <voxel>

This section describes the specifications of external .favmapx files containing user-defined attributes that are specified in <reference> of <user_defined_map>. Here, the .fav file that refers to the external .favmapx file is referred to as the parent file.

When a user-defined attribute is stored in XML format, it is written as follows.

```
e.g.,
<?xml version="1.0" encoding="utf-8"?>
    <fav version="1.1">
        <user_defined_map>
            <layer><![CDATA[······]]></layer>
            <layer><![CDATA[······]]></layer>
            <layer><![CDATA[······]]></layer>
            <layer><![CDATA[······]]></layer>
        </user_defined_map>
    </fav>
```

Fig. 29: Example of XML code under <geometry>

<layer> elements are defined at the level under <user_defined_map>. The number of <layer> elements to be defined is the same as <dimension.z> of the parent file. One user-defined attribute map element is defined for each x-y layer (<layer>). In each <layer>, attribute values of the data type specified in (value_type) of the parent file are listed. The number of attribute values to be listed can be calculated by multiplying <dimension.x> of the parent file by <dimension.y> of the parent file.

The data written for each of the data types that can be specified in (value_type) of the parent file is as follows.

- byte The user-defined attribute value for one cell is written as a value from **0 to 255**.
  The user-defined attribute value for one cell is written in 1 byte (**2 hexadecimal characters**).

- short The user-defined attribute value for one cell is written as a value from **-32,768 to 32,767**.
  The user-defined attribute value for one cell is written in 2 bytes (**4 hexadecimal characters**).

- ushort The user-defined attribute value for one cell is written as a value from **0 to 65,535**.
  The user-defined attribute or one cell is written in 2 bytes (**4 hexadecimal characters**).

- int The user-defined attribute value for one cell is written as a value from **-2,147,483,648 to 2,147,483,647**.
  The user-defined attribute value for one cell is written in 4 bytes (**8 hexadecimal characters**).

- uint The user-defined attribute value for one cell is written as a value from **0 to 4,294,967,295**.
  The user-defined attribute value for one cell is written in 4 bytes (**8 hexadecimal characters**).

- float The user-defined attribute value for one cell is written as a value from **$\pm10^{-37}$ to $\pm10^{38}$**.
  The user-defined attribute value for one cell is written in 4 bytes (**8 hexadecimal characters**).

- double The user-defined attribute value for one cell is written as a value from **$\pm10^{-307}$ to $\pm10^{308}$**.
  The user-defined attribute value for one cell is written in 8 bytes (**16 hexadecimal characters**).

Even when the value 0 (denoting the absence of a voxel in a certain position) has been specified for a cell in <voxel_map> of the parent file, attribute information for that cell is to be listed in <user_defined_map>. In cases when attribute information is not needed for a cell, 0 is to be specified for that cell in <user_defined_map>.

The following relationships should be maintained between a file in the .favmapx format and the parent file.

- The number of <layer> elements under <voxel_map> of the parent file should be equal to the number of <layer> elements under <user_defined_map>.
- The number of pieces of data listed in a <layer> of <voxel_map> of the parent file should be equal to the number of pieces of data listed in a <layer> of the <user_defined_map>. (The number of pieces of data can be calculated by multiplying <dimension.x> of the parent file by <dimension.y> of the parent file.)

Under <user_defined_map>, multiple <layer> elements are stored. Each <layer> contains the following data.

Table 35: The description of <user_defined_map.layer>

Element	Attribute	Data type	Data [Default value]	Description	Condition
layer	−	CDATA	The character string listing all the user-defined attribute values. [−]	Specifies attribute information for each cell of <grid> of the parent file listed in the format specified in (value_type) of the parent file. When (compression) in <user_defined_map> of the parent file is set to that other than "none", the data in <layer> is compressed using the specified method and stored.	

6.2.4.2. User-defined attribute files: Binary format (.favmap)

Parent element: <user_defined_map>; Link: <grid>, <voxel>

This section describes the specifications of external .favmap files containing user-defined attributes that are specified in <reference> of <user_defined_map>. Here, the .fav file that refers to the external .favmap file is referred to as the parent file.

When a user-defined attribute is stored in binary format, the data for each of the data types that can be specified in (value_type) of the parent file is as follows. The data is stored in order starting from the beginning of the binary file.

- byte The user-defined attribute value for one cell is written as a value from **0 to 255.**
  The user-defined attribute value for one cell is written in **1 byte**.

- short The user-defined attribute value for one cell is written as a value from **-32,768 to 32,767**.
  The user-defined attribute value for one cell is written in **2 bytes**.

- ushort The user-defined attribute value for one cell is written as a value from **0 to 65,535**.
  The user-defined attribute value for one cell is written in **2 bytes**.

- int The user-defined attribute value for one cell is written as a value from **-2,147,483,648 to 2,147,483,647**.
  The user-defined attribute value for one cell is written in **4 bytes**.

- uint The user-defined attribute value for one cell is written as a value from **0 to 4,294,967,295**.
  The user-defined attribute value for one cell is written in **4 bytes**.

- float The user-defined attribute value for one cell is written as a value from **$\underline{+}10^{-37}$ to $\underline{+}10^{38}$**.
  The user-defined attribute value for one cell is written in **4 bytes**.

- double The user-defined attribute value for one cell is written as a value from **$\underline{+}10^{-307}$ to $\underline{+}10^{308}$** .
  The user-defined attribute value for one cell is written in **8 bytes**.

Attribute information is stored as follows starting from the beginning of the binary file.

① Attribute information is stored starting from the lowest layer on the z-axis (i.e., the layer of <grid> of the parent file representing the base of the 3D object).
Attribute information for each position in a row along the x-axis (<dimension.x>) of <grid> of the parent file is stored in order starting from the beginning of the binary file.
In cases when there is no attribute information for a cell, the value 0 is stored.

② For the next row in the layer, attribute information for each position along the x-axis (<dimension.x>) is stored in the binary file.

③ Step ② is repeated for each row in the layer. The number of rows is the same as <dimension.y> of <grid> of the parent file. When the number of pieces of data stored is equivalent to <dimension.x> of the parent file multiplied by <dimension.y> of the parent file, this data is the user-defined attribute data for an entire x-y layer.

④ Steps ② and ③ are repeated to store data for each layer in order. When the number of pieces of data stored is equal to <dimension.x> multiplied by <dimension.y> multiplied by <dimension.z> of the parent file, this data is the user-defined attribute data for the entire <grid> of the parent file.

All the data contained in a binary file represents the user-defined attribute. There is no information used to separate pieces of data or information representing separations between <layer> elements.

The size of a .favmap file can be calculated according to the following formula.

Size (bytes) = (<dimension.x> of the parent file) x
(<dimension.y> of the parent file) x
(<dimension.z> of the parent file) x
(the number of bytes required for the data type represented by (value_type))

7. Conclusion

Shown below is the list of all elements and attributes in the FAV format described in this document.

Table 36: All elements and attributes in the FAV format

Element (The further left an element is positioned on the table, the further up that element is in the tree structure, and the further right, the further down.)			Attribute	Data type	Data [Default value]	Description	Condition
–		fav	–	–	–	The Root element of the FAV format. Defines <metadata>, <palette>, <voxel>, and <object>.	Required
			version	string	The version number of the FAV format [–]	Specifies the character string representing the version number of the FAV format used for the file.	Required
fav / material / object / user_defined_map	metadata	→	–	–	–	<id>, <title>, <author>, <license>, and <note> are defined in <metadata>.	
		id	–	string	The ID of the parent element [–]	Specifies the character string of the ID that is used to identify the parent element of <metadata>.	Required
		title	–	CDATA	The name of the parent element [–]	Specifies the character string of the name of the parent element of <metadata>.	Required
		author	–	CDATA	The author of the parent element [–]	Specifies the character string of the author of the parent element of <metadata>.	Required
		license	–	CDATA	The license information of the parent element [–]	Specifies the character string describing the license information of the parent element of <metadata>.	Required

				note	–	CDATA	A memo for the parent element [–]	Specifies the character string of a memo for the parent element of <metadata>.	
fav	palette	→			–	–	–	<geometry> and <material> are defined in <palette>.	
		geometry	→		–	–	–	<shape> and <scale> are defined in <geometry>.	Multiple may be defined
					id	positiveInteger	The ID of the geometry information [1]* *After this value has been used, the default value is increased by 1 each time.	Specifies the unique ID that is used to identify <geometry>. The same ID must not be used for multiple <geometry> elements.	Required
					name	string	The name to be defined for the geometry information [–]	Specifies the name of <geometry>. The same name should not be used for multiple <geometry> elements.	
			shape		–	string	cube / sphere / user_defined [cube]	Specifies the shape of a voxel. When "user_defined" is specified, <reference> must also be defined.	
			reference		–	CDATA	The path of the STL file defining the shape of a voxel [–]	Specifies the character string of the relative path to the external STL file defining the shape of a voxel.	
			scale	x	–	double	The scale of the x-axis [1.0]	Specifies the scale for the x-axis of <shape> in relation to the cell size	

								specified in <grid>. When a negative value is specified, the voxel is flipped. 0 must not be specified.	
				y	–	double	The scale of the y-axis [1.0]	Specifies the scale for the y-axis of <shape> in relation to the cell size specified in <grid>. When a negative value is specified, the voxel is flipped. 0 must not be specified.	
				z	–	double	The scale of the z-axis [1.0]	Specifies the scale for the z-axis of <shape> in relation to the cell size specified in <grid>. When a negative value is specified, the voxel is flipped. 0 must not be specified.	
		material	→		–	–	–	At least one of the following is defined in <material>: <material_name>, <product_info>, and <standard_name>.	Multiple may be defined
					id	positiveInteger	The ID of the material information [1]* *After this value has been used, the default value is increased by 1 each time.	Specifies the unique ID that is used to identify a <material> element. The same ID must not be used for multiple <material> elements.	Required
					name	string	The name to be defined for the material information [–]	Specifies the name of a <material> element. The same name should not be used for multiple <material> elements.	

			material_name	→	–	CDATA	The material name [–]	Specifies a character string identifying the material used.	Multiple may be defined
			product_info	→	–	–	–	Contains the elements below.	Multiple may be defined
				manufacturer	–	CDATA	The name of the manufacturer of the material [–]	Specifies the character string of the name of the manufacturer of the material.	
				product_name	–	CDATA	The product name of the material [–]	Specifies the character string of the product name, product code, etc. provided by the manufacturer.	
				url	–	CDATA	The URL of the website providing material information [–]	Specifies the URL of the website providing material information.	
			standard_name	→	–	CDATA	Standard information for a material [–]	Specifies a material by its name as defined in a standard.	Multiple may be defined
	voxel	→			–	–	–	<geometry_info> and <material_info> must be written in <voxel>. Alternatively, it is also possible to define	Multiple may be defined

							<reference> only. <display> and <application_note> may be defined.	
				id	positiveInteger	The ID of the voxel [1]* *After this value has been used, the default value is increased by 1 each time.	A positive integer of 1 or greater must be specified as the ID identifying the <voxel> element. The same ID must not be used for multiple <voxel> elements.	Required
				name	string	The name to be defined for the voxel [–]	Specifies the name of the <voxel> element. The same name should not be used for multiple <voxel> elements.	
		geometry_info	→	–	–	–		
			id	–	positiveInteger	The ID of a predefined shape information element [–]	Specifies the ID of <geometry> so that it can be referenced. The predefined (id) attribute of a <geometry> element must be specified here.	Required
		material_info	→	–	–	–		Multiple may be defined
			id	–	positiveInteger	The ID of a predefined material information element [–]	Specifies the ID of a <material> element so that it can be referenced. The predefined (id) attribute in <material> must be specified.	Required
			ratio	–	double	The proportion of the material to be used when multiple	A double value is used to specify what proportion the material makes up of the whole (defined under	

						materials are combined into a composite material [–]	<material>). Values are to be specified so that they total to 1.0 when added. The value must be greater than 0.	
		display	r	–	nonNegativeInteger	The red component of the display color of a voxel [–]	Specifies the red component of the display color of a voxel. A value less than 0 or a value exceeding 255 must not be specified.	
			g	–	nonNegativeInteger	The green component of the display color of a voxel [–]	Specifies the green component of the display color of a voxel. A value less than 0 or a value exceeding 255 must not be specified.	
			b	–	nonNegativeInteger	The blue component of the display color of a voxel [–]	Specifies the blue component of the display color of a voxel. A value less than 0 or a value exceeding 255 must not be specified.	
			a	–	nonNegativeInteger	The alpha component of the display color of a voxel [255]	Specifies the alpha component of the display color of a voxel. A value less than 0 or a value exceeding 255 must not be specified.	
		application_note	→	–	CDATA	Property information to be stored in a voxel [–]	Specifies the character string of a property to be stored in a <voxel> element. It is recommended that descriptions be added to help users distinguish and select the necessary data from multiple	

								<application_note> elements.	
		reference		–	–	CDATA	The path of a FAV file defining a group of voxels to be treated as a single voxel [–]	Specifies the character string of the relative path to an external FAV file which is to be treated as a single voxel.	
	object	→			–	–	–	<grid> and <structure> must be described in <object>.	Multiple may be defined
					id	positivel nteger	The ID of the object [1]* *After this value has been used, the default value is increased by 1 each time.	Specifies the ID that is used to identify the <object> element. The same ID must not be used for multiple <object> elements.	Required
					name	string	The name to be defined for the object [–]	Specifies the name of the <object> element. The same name should not be used for multiple <object> elements.	
		grid	→		–	–	–	In <grid>, <origin>, <unit>, and <dimension> may be defined.	
			origin	x	–	double	The distance to the origin of the coordinate system for the x-axis [0.0]	Specifies the distance from the origin of the global coordinate system to the origin of the coordinate system of <grid> for the x-axis.	

				y	–	double	The distance to the origin of the coordinate system for the y-axis [0.0]	Specifies the distance from the origin of the global coordinate system to the origin of the coordinate system of <grid> for the y-axis.	
				z	–	double	The distance to the origin of the coordinate system for the z-axis [0.0]	Specifies the distance from the origin of the global coordinate system to the origin of the coordinate system of <grid> for the z-axis.	
			unit	x	–	double	The length of each cell in the x-axis direction [1.0]	Specifies the length of a single cell in <grid> in the direction of the x-axis. The value must be greater than 0.	
				y	–	double	The length of each cell in the y-axis direction [1.0]	Specifies the length of a single cell in <grid> in the direction of the y-axis. The value must be greater than 0.	
				z	–	double	The length of each cell in the z-axis direction [1.0]	Specifies the length of a single cell in <grid> in the direction of the z-axis. The value must be greater than 0.	
			dimension	x	–	positiveInteger	The number of cells in the x-axis direction [–]	Specifies the maximum number of voxels that can be arranged in <grid> in the x-axis direction.	Required
				y	–	positiveInteger	The number of cells in the y-axis direction [–]	Specifies the maximum number of voxels that can be arranged in <grid> in the y-axis direction.	Required
				z	–	positiveInteger	The number of cells in the z-	Specifies the maximum number of voxels that can	Required

							axis direction [–]	be arranged in <grid> in the z-axis direction.	
		structure	→		–	–	–	In <structure>, <voxel_map>, <color_map>, and <link_map> may be defined. Alternatively, <user_defined_map> may be defined instead.	Required
			voxel_map	→	–	–	–		
					bit_per_voxel	positiveInteger	4 / 8 / 16 [–]	Specifies a numerical value of 4, 8, or 16 (representing the length of hexadecimal characters) to define the number of bits in one voxel. (4, 8, and 16 indicate one, two, and four hexadecimal characters respectively.)	Required
					compression	string	none / base64 / zlib / runlength [none]	Selects the compression method for <layer> from the following: none (no compression), base64, zlib, and runlength.	
				layer	–	CDATA	The character string listing the ID value of each voxel in the layer [–]	Lists the (id) values in <voxel> using the format specified in (bit_per_voxel). Each (id) value is concatenated into one line (the number of IDs to be concatenated can be calculated by multiplying <dimension.x> by <dimension.y>). When (compression) in <voxel_map> is set to that other than "none", the data in <layer> is compressed using the specified method	One or more

								and stored as CDATA.	
			color_map	→	–	–	–		
					color_mode	string	GrayScale / GrayScale16 / RGB / RGBA / CMYK [–]	Selects the color mode defined for a voxel from the following: GrayScale, GrayScale16, RGB, RGBA, CMYK.	Required
					compression	string	none / base64 / zlib / runlength [none]	Selects the compression method for <layer> from the following: none (no compression), base64, zlib, and runlength.	
				layer	–	CDATA	A hexadecimal character string representing the color information of the 3D model data [–]	Specifies the color information of each voxel in <voxel_map> in the format specified by (color_mode). When (compression) of <color_map> is set to that other than "none", data in <layer> is compressed using the specified method and stored as CDATA.	One or more
			link_map	→	–	–	–		
					bit_per_link	positiveInteger	4 / 8 / 16 [–]	Specifies a numerical value of 4, 8, or 16 (representing the length of hexadecimal characters) to define the number of bits per link information. (4, 8, and 16 indicate one, two, and four hexadecimal characters respectively.)	Required

					neighbors	positiveInteger	6 / 18 / 26 [–]	Specifies a numerical value of 6, 18, or 26 to define the number of neighboring voxels for which link information is retained for the voxel in question.	Required
					compression	string	none / base64 / zlib / runlength [none]	Selects the compression method for <layer> from the following: none (no compression), base64, zlib, and runlength.	
				layer	–	CDATA	A hexadecimal character string representing the link information of the 3D model data [–]	Specifies the link information in the order shown in Fig. 26 for each voxel in <voxel_map> as specified by (neighbors). When (compression) of <link_map> is set to that other than "none", data in <layer> is compressed using the specified method and stored as CDATA.	
			user_defined_map	→	–	–	–		
					value_type	string	byte / short / ushort / int / uint / float / double [byte]	Specifies the data type of a user-defined attribute value (for a single voxel) stored in an external file.	Required
					compression	string	none / base64 / zlib /runlength [none]	Selects the compression method for the <layer> data that is stored in the external file from the following: none (no compression), base64, zlib, and runlength. (Only available when an XML format (.favmapx) file	

								is used.)	
				reference	–	CDATA	The path of the external file containing a user-defined attribute [–]	Specifies the character string of the relative path to the external .favmap/.favmapx file in which a user-defined attribute is stored.	Required

e.g.,

```
<?xml version="1.0" encoding="utf-8"?>
<fav version="1.1">
  <metadata>
    <id>bc4affb5-9a53-4de7-9f27-721ef27e8f34</id>
    <title><![CDATA[FAV Ver1.1 Sample File]]></title>
    <author><![CDATA[FUJIFILM Business Innovation & Keio SFC]]></author>
    <license><![CDATA[CC BY]]></license>
    <note><![CDATA[This is a sample file in FAV format ver1.1.]]></note>
  </metadata>
  <palette>
    <geometry id="1" name="NormalCube">
      <shape>cube</shape>
      <scale>
        <x>1</x>
        <y>1</y>
        <z>1</z>
      </scale>
    </geometry>
    <geometry id="2" name="Plate">
      <shape>cube</shape>
      <scale>
        <x>1</x>
        <y>1</y>
        <z>0.25</z>
      </scale>
    </geometry>
    <geometry id="3" name="Diamond">
      <shape>user_defined</shape>
      <reference><![CDATA[Diamond.stl]]></reference>
      <scale>
        <x>0.98</x>
        <y>0.98</y>
        <z>-1.05</z>
      </scale>
    </geometry>
    <material id="1" name="SoftMat1">
      <material_name><![CDATA[Some-soft-materials]]></material_name>
    </material>
    <material id="2" name="HardMat1">
      <product_info>
        <manufacturer><![CDATA[ABC Materials Co.]]></manufacturer>
        <product_name><![CDATA[ULTRA-HARD/007]]></product_name>
        <url><![CDATA[http://www.abcmaterial.com/ultra/hard/007]]></url>
      </product_info>
      <product_info>
        <manufacturer><![CDATA[ABC Materials Co.]]></manufacturer>
        <product_name><![CDATA[ULTRA-HARD/006a]]></product_name>
        <url><![CDATA[http://www.abcmaterial.com/ultra/hard/006/a]]></url>
      </product_info>
      <standard_name> <![CDATA[JIS K6899-1 ABS]]>
      </standard_name>
    </material>
  </palette>
  <voxel id="1" name="soft_cube">
    <geometry_info>
      <id>1</id>
    </geometry_info>
    <material_info>
      <id>1</id>
     <voxel>
  <voxel id="2" name="hard_cube">
    <geometry_info>
      <id>1</id>
    </geometry_info>
    <material_info>
      <id>1</id>
      <ratio>0.15</ratio>
    </material_info <ratio>1</ratio>
    </material_info>
  <voxel/ >
    <material_info>
      <id>2</id>
      <ratio>0.85</ratio>
    </material_info>
    <application_note><![CDATA[HM-H01:Hybrid Hard Material Number 01]]></application_note>
    <application_note><![CDATA[FabAppAttr : application note]]></application_note>
  </voxel>
  <object id=″1″ name=″SampleObject″>
    <metadata>
      <id>cafed8bd-3bd9-4d7a-a67d-2df635d2d8f8</id>
      <title><![CDATA[]]></title>
      <author><![CDATA[Mr. Sample Creator]]></author>
      <license><![CDATA[No rights reserved]]></license>
    </metadata>
    <grid>
      <origin>
        <x>28.5</x>
        <y>-30</y>
        <z>0</z>
      </origin>
      <unit>
        <x>1</x>
        <y>1</y>
        <z>1</z>
      </unit>
      <dimension>
        <x>7</x>
        <y>7</y>
        <z>7</z>
      </dimension>
    </grid>
    <structure>
      <voxel_map bit_per_voxel=″8″ compression=″none″>
        <layer><![CDATA[01010000000000010100000000000101000000000001010100000000
              000101010000000000010101010100000001010101]]></layer>
        <layer><![CDATA[01010000000000010100000000000101000000000001010100000000
              000101010000000000010101010100000001010101]]></layer>
        <layer><![CDATA[01010000000000010100000000000101010000000000010101000000
              000101010100000000010101010100000000010101]]></layer>
    <layer><![CDATA[01010100000000001010100000000000010101000000000010101010000
            000001010101010000000101010100000000010101]]></layer>
    <layer><![CDATA[00010101000000000010101000000000010101010000000000101010101
            00000101010101000000000101010000000000000]]></layer>
    <layer><![CDATA[000101010101000000001010101010000010101010100000001010101
            000000000101010000000000000100000000000000]]></layer>
    <layer><![CDATA[000000010101010000000101010100000001010101000000000010101
            00000000000000000000000000000000000000000]]></layer>
  </voxel_map>
  <color_map compression=″none″ color_mode=″RGB″>
    <layer><![CDATA[8300258100277600329100176400457c002d5e004a5c004c500059560052
                    3300753700713000782f007a3100771800900f00991f00891c008c130096
                    0c009c]]></layer>
    <layer><![CDATA[8200269700117400348e001a6300466800405b004e59005055005355005
                    34000684200663b006d2c007c2400842d007a1700911f00891e008b14009
                    4100098]]></layer>
    <layer><![CDATA[8900209a000e8500237f002a8100277b002d6c003c7700315800515a004e
                    5300554900604200673d006b390070380070270081230086300079270081
                    19008f19008f]]></layer>
    <layer><![CDATA[9300159800109b000e8100278000288100277f0029720036700039750033
                    6e003b6f003a58005159004f3e006a4100684100674100673d006b2d007c
                    2200862000892c007c2100881f0089]]></layer>
    <layer><![CDATA[99000f9700119a000e9200169200168a001e7500337300347000386b003d
                    72003562004759004f6300465b004e6100475d004b55005342006639006f
                    4300662a007e270081]]></layer>
    <layer><![CDATA[9000189000188a001e9700129600128f001988001f8c001c7b002d7a002e
                    8700218600237700316f00396d003c5c004c6f003a6d003c5800505a004e
                    5300554f005939006f]]></layer>
  </color_map>
  <link_map bit_per_link=″8″ neighbors=″6″ compression=″none″ >
   <layer><![CDATA[00000064c8ff00000000c8ff00000064c8ff00000000c8ff00c80064c8ff
                   00c80000c8ff00c8006400ff00c80064c8ff00006400c8ff00c8006400ff
                   00c86464c8ff00006400c8ff00c8006400ff00c86464c8ff00006464c8ff
                   00006464c8ff00006400c8ff00c8006400ff00c8646400ff00c8646400ff
                   00c8640000ff]]></layer>
   <layer><![CDATA[00000064c8ff00000000c8ff00000064c8ff00000000c8ff00c80064c8ff
                   00c80000c8ff00c80064000000c80064c8ff00006400c8ff00c8006400ff
```

```
                        00c86464c8ff00006400c8ff00c8006400ff00c86464c8ff00006464c8ff
                        00006464c8ff00006400c8ff00c80064000000c8646400ff00c8646400ff
                        00c8640000ff]]></layer>
        <layer><![CDATA[ff000064c8ffff000000c8ffff000064c8ffff000000c8ffffc800640000
                        ffc80064c8ff00006400c8ffffc80064c8ffffc86464c8ff00006400c8ff
                        ffc800640000ffc86464c8ffffc86464c8ff00006400c8ffffc800640000
                        ffc8646400ffffc86464c8ffff006464c8ffff006400c8ffffc8006400ff
                        ffc8646400ffffc8640000ff]]></layer>
        <layer><![CDATA[ff000064c800ff000064c8ff00006400c8ffff0000640000ff000064c8ff
                        00006400c8ffffc80064c8ffffc86464c8ff00006400c8ffffc800640000
                        ffc86464c8ffffc86464c8ff00006400c8ffffc8006400ffffc86464c8ff
                        ffc86464c8ff00006464c8ff00006400c8ffffc800640000ffc86464c8ff
                        ffc86464c8ffffc86400c8ffffc800640000ffc864640000ffc864000000]]></layer>
        <layer><![CDATA[ff000064c8ffff006464c8ff00006400c8ffff000064c800ff006464c8ff
                        00006400c8ffffc800640000ffc86464c8ffffc86464c8ff00006400c8ff
                        ffc80064c800ffc86464c8ffffc86464c8ff00006464c8ff00006400c8ff
                        ffc800640000ffc864640000ffc86464c8ffffc86464c8ffffc86400c8ff
                        ffc800640000ffc864640000ffc8640000ff]]></layer>
        <layer><![CDATA[ff0000640000ff006464c800ff006464c8ff00006464c8ff00006400c8ff
                        ff000064c800ff006464c8ff00006464c8ff00006464c8ff00006400c8ff
                        ffc800640000ffc86464c8ffffc86464c8ff00c86464c8ff00c86400c8ff
                        ffc800640000ffc86464c8ffffc86464c8ffffc86400c8ffffc800640000
                        ffc864640000ffc86400c800ffc800000000]]></layer>
        <layer><![CDATA[ff000064c800ff006464c800ff006464c80000006400c800ff000064c800
                        ff006464c800ff006464c800ff006400c800ffc800640000ffc86464c800
                        ffc86464c800ffc86400c800ffc800640000ffc864640000ffc864000000]]></layer>
      </link_map>
      <user_defined_map value_type=″float″ compression=″none″>
        <reference><![CDATA[ExternalAttributes.favmap]]></reference>
      </user_defined_map>
    </structure>
  </object>
</fav>
```

Fig. 30: Example of the entire XML code of FAV format

8. Revision history

Revised on	Revised by	Revised section	Details of change
October 11, 2018	Tomonari Takahashi	Cover page	Registered trademark symbol was added after FAV.
		License	Removed the word “non-commercial”. Changed the licenser from individual members to Fuji Xerox.
		4. <voxel>	Added an explanation of the function for referring to an external .fav file and defining a group of voxels to be treated as a single voxel. Updated the diagrams.
		4.2. <material_info>	Added a function for specifying the value 0 for a voxel in order to secure an empty space containing no material. Updated the diagram.
		4.5. <reference>	Added a function for referring to an external .fav file and defining a group of voxels to be treated as a single voxel.
		5. <object>	Added an explanation of the function for mapping a user-defined attribute onto each cell of the grid of voxels. Updated the diagrams.
		5.1.2. <unit>	Added the formula to calculate the number of voxels contained within a space of 1 mm (vpm = voxel per mm).
		5.2. <structure>	Added an explanation of the function for mapping a user-defined attribute onto each cell of the grid of voxels.
		5.2.2. <color_map>	Corrected an error in the number of shades available. Error: 512 shades Correct: 65,536 shades
		5.2.3. <link_map>	Added a further explanation of how many bits are used to write link information (bit_per_link).
		5.2.4. <user_defined_map>	Added a function for mapping a user-defined attribute onto each cell of the grid of voxels.
		5.2.1. <voxel_map> 5.2.2. <color_map> 5.2.3. <link_map> 5.2.4. <user_defined_map>	Added a function to support runlength in (compression) of each <layer>.
February 18, 2019	Masahiko Fujii, Tomonari Takahashi	6.2.3 <link_map>	Changed the order in which link information for neighboring voxels is defined. (It is now defined starting from the voxel with the lowest coordinate values under the axis ordering of z, y, x.) Corrected the example of link map data.
		4. <palette> 6. <object> 7. Conclusion	Corrected the examples of code.
December 7, 2022	Tomonari Takahashi	Cover page	Changed the company name from "Fuji Xerox Co., Ltd." to "FUJIFILM Business Innovation Corp."
		3. <author> 6. <author> 7. <conclude>	Changed the description from "Fuji Xerox" to "FUJIFILM Business Innovation".