ICS 07.040
A 75

Social Organization Standard

T/ CAGIS 1—2019

Data format for spatial 3D model

空间三维模型数据格式

(English Translation)

Issue date: 2019 - 07 - 19 Implementation date: 2019 - 08 - 31

Issued by China Associa for Geospatial Information Society

Contents

Foreword

SuperMap Software Co.,Ltd is in charge of this English translation. In case of any about the contents of English translation, the Chinese original shall be considered authoritative.

This standard is drafted in accordance with the rules given in the GB/T 1.1—2009 *Directives for standardization—Part 1:Structure and drafting of standards*.

Attention is drawn to the possibility that some of the elements of this standard may be the subject of patent rights. The issuing body of this document shall not be held responsible for identifying any or all such patent rights.

This standard was proposed and prepared management by the China Association of Geographic Information System (CAGIS).

Introduction

In recent years, the development of data acquisition techniques such as oblique photography and laser scanning has effectively reduced the cost of 3D spatial data, shortened the time period for producing 3D spatial data and improved the accuracy of data. As the amount of 3D spatial data are continuously growing, the works on efficient publishing, sharing and transmission for 3D data have formed the main directions to 3D GIS study.

This document defined an opened and extendable data format for 3D spatial model - Spatial 3D Model (S3M). It applies to transmission, exchange and sharing of 3D spatial model data which helps to solve the problem of data storage, high-efficient visualizing, data sharing, publishing and interoperating for multi-source 3D spatial data across different devices (mobile, browser, PC). It plays a key role in boost the process of more efficient and advanced applying with 3D geo-spatial data in China.

The following data types are covered by S3M standard:

a) Traditional data: Man-made model;
b) Realistic data: Oblique photogrammetry and point cloud;
c) Building Information Modelling (BIM): BIM produced by BIM design software;
d) GIS vector data: 2D point/line/region, 3D point/line/region, 3D pipeline.

Data Format for Spatial 3D Model

1 Scope

This standard specifies the file structure and storage regulation of a data format for spatial 3D model.

This standard is applicable for transmission, exchange and sharing of 3D spatial data to be used for 3D GIS related applications under online and offline environment and deployment among different end-terminals (mobile, browser, PC).

2 Normative references

The following referenced documents are indispensable for the application of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

GB/T 7408—2005 *Data elements and interexchange formats—Information interexchange—Representation of dates and times*

GB/T 16831—2013 *Standard representation of geographic point location by coordinates*

GB/T 30320—2013 *Geospatial database call-level interface*

GB/T 33187.1—2016 *Geographic information—Simple feature access—Part 1:Common architecture*

3 Terms and definitions

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

3.1

tile
each tile corresponds to a 2D polygon or 3D cuboid space, the space range between adjacent tiles may overlay

3.2

root node
the root node of TileTree

Note: A TileTree has only one root node; the spatial scope of the TileTree is the union of all the child-nodes.

3.3

tiletree
a tree spatial data structure with multiple levels of detail, which is divided by a unique root node from the top down, each node of the tree structure represents a tile

Note: the spatial scope of the parent node (tile) is the union of all its child-nodes (sub-tiles).

3.4

tiletreeset
a collection of one or more TileTrees

3.5

patchLOD
representing the set of patches over a specific LOD in a TileTree

Note: One LOD contains one or more PatchLOD.

3.6

patch
a piece of data in the PatchLOD

Note: A PatchLOD can contain one or more patches. Each patch contains zero or one parent-patch and zero or multiple child-patches; the spatial scope of parent-patch is the union of all its child-patches. The parent-child relationship of the patch constitutes a tree structure.

3.7

geode
the data package in a patch

Note: Each patch contains zero or multiple Geode. Geode contains information of model entity of Skeleton, Material and Texture.

3.8

model entity
the elementary structure of geode containing information of Skeleton, Material and Texture

3.9

skeleton
geomatic information including vertex, vertex indexing, texture coordinates, and texture coordinates indexing

Note: The Skeleton data containing texture information.

3.10

material
the collection of attributes that can be visualized on the surface of model objects, including surface color, texture, smoothness, reflectivity, refractive index and luminosity

3.11

texture
description of texture including width, height, compression approach and texture binary data

4 Symbols and abbreviations

4.1 Abbreviations

The following items apply for this document:

ESPG The European Petroleum Survey Group
LOD Level of Detail
UML Unified Modelling Language
WKT Well-known Text Representation

4.2 UML pictogram

The diagrams appearing in this standard are represented by UML static structure.

All Reference of all UML pictogram in this document are shown in Table 1.

Table 1 — UML pictogram comparison table

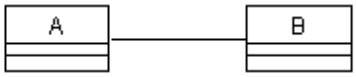
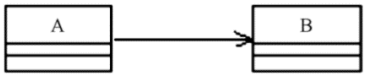
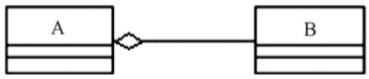
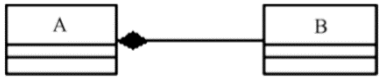
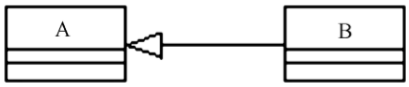
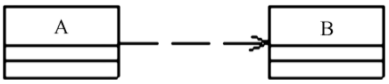
Symbol	Name	Description
	Bidirectional association	Represents the general relationship between the two classes A and B, both classes know the public properties and methods of the other class.
	Unidirectional association	Represents the relationship between the two classes A and B. Class A knows the public properties and methods of class B, but class B does not know the public properties and methods of class A.
	Polymerization	Represents the relationship between the two classes A and B. Class A knows the public properties and methods of class B, but class B does not know the public properties and methods of class A.
	Combination	Object A owns object B, object A can contain object B, but object B is not a part of object A, and their life cycles can be different.

Table 1 (continued)

Symbol	Name	Description
	Generalization	B object inherits A object, that is, B object is derived from A object.
	Dependency	Class A depends on Class B, and changes in Class B will affect Class A. If class A depends on class B, then B can be embodied as A's local variables, method parameters or static method calls.

4.3 Description of UML diversity

Meanings of UML diversity to this standard are shown in Table 2.

Table 2 — Meaning of UML diversity

Diversity	Meaning
0..1	0 or 1
1	Only 1
0..n	0 or more than 1
1..n	1 or more than 1

5 General regulation

5.1 Basic data type

Basic data type involved in this standard is shown in Table 3.

Table 3 — Numerical data type specification

Type	Number of bytes	Value range	Description
byte	1	[0,255]	Single byte
bool	1	0|1	Boolean
int16	2	[-32768,32767]	Short integer
uint16	2	[0,65535]	Unsigned short integer
int32	4	[-2147483648,2147483647]	Integer
uint32	4	[0,4294967295]	Unsigned short
int64	8	$[-2^{63},(2^{63}-1)]$	Long integer
uint64	8	$[0,(2^{64}-1)]$	Unsigned long integer
float	4	$[-3.4\times10^{38},3.4\times10^{38}]$	Single-precision float
double	8	$[-1.7\times10^{308},1.7\times10^{308}]$	Double-precision float
wchar	2	--	Wide character type

5.2 String type

String types involved in this standard are described by the String object, and encoded in Unicode. The character set is specified as UTF 8.

```
String {
    int 32 length ; //Number of bytes
    byte str [length]; //Data content
}
```

5.3 Json data

Json data in this standard, is encoded as UTF-8, without BOM prefix.

6 Organization structure

6.1 File organization

Files which formed data in this standard include: description file, data file, indexing tree file and attribute file.

Description file and data file are mandatory components. Description file contains one or more root node path of TileTree. Data file is organized by TileTree, each tile of TileTree corresponds one S3MB file. Indexing tree file is a collection of descriptions of each tile in TileTree, which can acquire oriented bounding box, LOD switching information and the path of hooking child-nodes file from each Tile of every LOD layers. The main role of indexing tree file is to accelerate the efficiency of Tile indexing. Attribute file includes descripted file of attributes and attribute data file.

The organizational form for each file is shown in Table 4.

Table 4 — Overview of file organization form

File type	Storage form	Storage regulation	Mandatory/ Optional
Description file	.scp	Description file stores described information of whole data in json format. File name is customizable, the extension is ".scp"	Mandatory
Data file	.s3mb	Data file stores data information of tiles. File name is customizable, the extension is ".s3mb"	Mandatory
Indexing tree file	.json	Indexing tree file stores tile file information of each LOD layer of TileTree. The file name is limited to the root node name of the tile, and the extension is limited to ".json" and under the same directory with the root node of TileTree	Optional
Attribute description file	attribute.json	Each dataset attribute description information in the TileTreeSet. The data set description see GB/T 30320-2013. The full name of the file is limited to "attribute.json" and the same directory as the .scp file	Optional

表1 Table 4 (continued)

File type	Storage form	Storage regulation	Mandatory/ Optional
Attribute data file	.s3md	Attribute data file stores attribute dara for all objects in TileTree. The file name is limited to the root node name of TileTree and the extension is limited to ".s3md". The same level as the root node of TileTree	Optional

6.2 Description of tree structure

UML of tree structure is given in Figure 1.

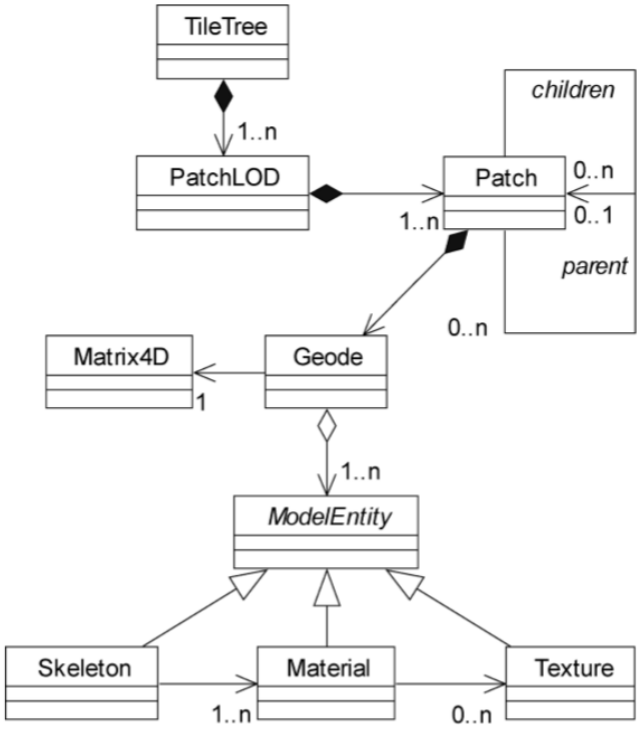


Figure 1 — Logical schema of the tree structure

7 Storage format

7.1 Description file

7.1.1 Overview of the description file

The description file (TilTeesetInfo object), is used to describe the basic information of data, the Logical schema of the associated object is shown in Figure 2.

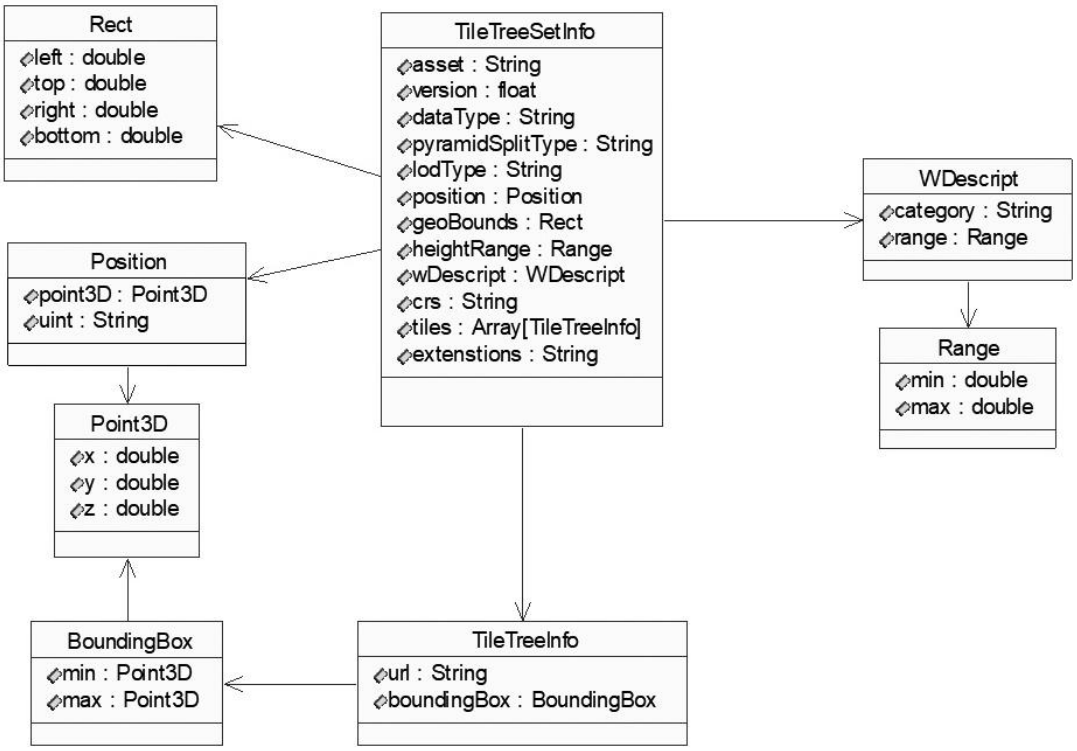


Figure 2 — Logical schema of the description file

7.1.2 Label information

Each label information of description file is shown in Table 5.

Table 5 — Labels in description file

Label Name	Type	Description
asset	String	Basic information about the data, such as production unit
version	Float	Version number. Value range: { '1.0' }
dataType	String	3D geo-spatial data type. Value range: { 'Vector', 'ObliquePhotogrammetry', 'ArtificialModel', 'BIM', 'PointCloud', 'PipeLine' }. Corresponding to Vector Data, Oblique Photogrammetric Model, Man-made Model, BIM, Point Cloud and Pipeline Data
pyramidSplitType	String	Spatial data structure. Value Rang: { 'QuadTree', 'Octree', 'RTree', 'K-DTree' }. Corresponding to Quadtree, octree, R tree, K-D tree
lodType	String	LOD type. Value range: { 'Add', 'Replace' }. Corresponding to add refinement and replace refinement
geoBounds	Rect	Geographic area of data, represented by Rect object, see Table 6

Table 5 (continued)

Label Name	Type	Description
heightRange	Range	Height range of data, represented by Range object, marking the maximum and minimum value of height, see Table 7
wDescript	WDescript	W description, represented by WDescript object, see Table 8
position	Position	Spatial coordinates of whole TileTreeSet, represented by Position object, containing spatial coordinates and unit, see Table 9
crs	String	Coordinate system. Representation: crs:{ 'type:content' }; crs is keyword; type can be either wkt or epsg; content is string. Representation format of epsg: crs:{ 'epsg:4326' }. Representation format of wkt: crs:{ 'wkt: wktcontent' }. Representation format of wkt: crs:{ 'wkt: wktcontent' }. wktcontent should meet the requirements of GB/T 33187.1-2016
tiles	Array<TileTreeInfo>	Tile information, represented by TileTreeInfo object, including the index file URL and enclosing sphere of each TileTree, see Table 11
extensions	User custom	User's extensions

Table 6 — Labels of Rect object

Label name	Type	Description
left	double	Left value of the geographic range of the data
top	double	Upper value of the geographic range of the data
right	double	Right value of the geographic range of the data
bottom	double	Bottom value of the geographic range of data

Table 7 — Labels of Range object

Label Name	Type	Description
min	Double	Maximum value
max	Double	Minimum value

Table 8 — Labels of WDescript object

Label Name	Type	Description
category	String	W-digit meaning description information
range	Range	W-digit value range, represented by Range object, including W-digit minimum and maximum values, see Table 7

Table 9 — Labels of Position object

Label Name	Type	Description
point3D	Point3D	The coordinate value of the spatial point is represented by the Point3D object and contains the X, Y, and Z coordinate values of the spatial point. See Table 10. The representation of longitude and latitude should comply with GB/T 16831-2013
unit	String	Unit of spatial coordinate system. Value range: { 'Degree', 'Meter' }. Corresponding to degree and meter respectively

Table 10 — Labels of Point 3D object

Label Name	Type	Description
x	Double	X value of spatial point
y	Double	Y value of spatial point
z	Double	Z value of spatial point

Table 11 — Labels of TileTreeInfo object

Label Name	Type	Description
url	String	Tile file path
boundingBox	Boundingbox	Area of tile, represented by Boundingbox, see Table 12

Table 12 — Labels of Boundingbox object

Label Name	Type	Description
max	Point3D	Maximum angle point of Boundingbox, represented by Point 3D, see Table 10
min	Point3D	Minimum angle point of Boundingbox, represented by Point 3D, see Table 10

7.2 Data file

7.2.1 Logical structure of s3mb file

7.2.1.1 Main structure

Data files formed by .s3mb (Spatial 3D Model Binary)files ,are key parts of data. The data of one PatchLOD is stored as one .s3mb file.

See Figure 3 for logical schema of objects in s3mb file. See Table 13 for the meanings of objects in s3mb file.

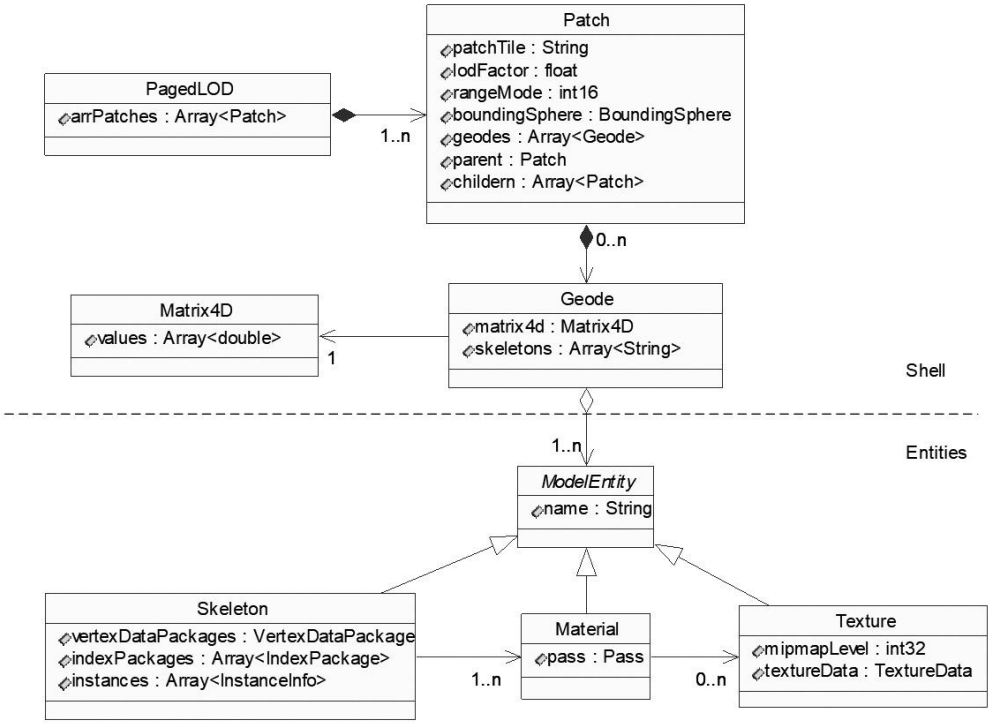


Figure 3 — Logical schema of objects in s3mb file

Table 13 — Objects in s3mb file

Object name	Type	Description
arrPatches	Array<Patch>	A collection of all patches in the PatchLOD. The patch description is in Table 14

Table 14 — Attributes of Patch object

Attribute name	Type	Description
patchTile	String	The S3MB file name where the patch is located
lodFactor	float	Switch factor, which is the threshold of LOD switches, using with switch mode
rangeMode	int16	Switch mode, see 7.2.2.2
boundingSphere	BoundingSphere	Boundingbox, represented by BoundingSphere object, see Table 15
geodes	Array<Geode>	Union of data package, see Table 16
parent	Patch	Parent-node
children	Array<Patch>	Array of child-nodes, all child-patches

Table 15 — Attributes of BoundingSphere object

Attribute name	Type	Description
x	double	X value of central point
y	double	Y value of central point

Table 15 (continued)

Attribute name	Type	Description
z	double	Z value of central point
r	double	Radius of boundingbox

Table 16 — Attributes of Geode object

Attribute name	Type	Description
matrix4d	Matrix4D	The matrix act on skeleton shape, represented by Matrix4D, see Table 17
skeletons	Array<String>	Array of skeleton name, see Table 18

Table 17 — Attributes of Matrix4D object

Attribute name	Type	Description
values	double[16]	4×4 texture matrix, represented by 16 doubles, row major order

Table 18 — Attributes of ModelEntity object

Attribute name	Type	Description
name	String	Entity name, the unique identification of entity object in TileTree. Object of Skeleton, Material,Texture inherited from ModelEntity object, see Table 19, 26, 32

7.2.1.2 Skeleton object

Skeleton object is made of one VertexDataPackage and one or more IndexPackages. VertexDataPackage is the description of each vertex, including coordinate, normal, color, texture coordinate, model object ID, instantiation information, ect. IndexPackage is the description of skeleton structure. Each IndexPackage has one or more Pass, which is used for identifying the rendering method of the vertex package.

See Table 4 for logical schema to Skeleton object; see Table 19 for the meanings of each Skeleton object.

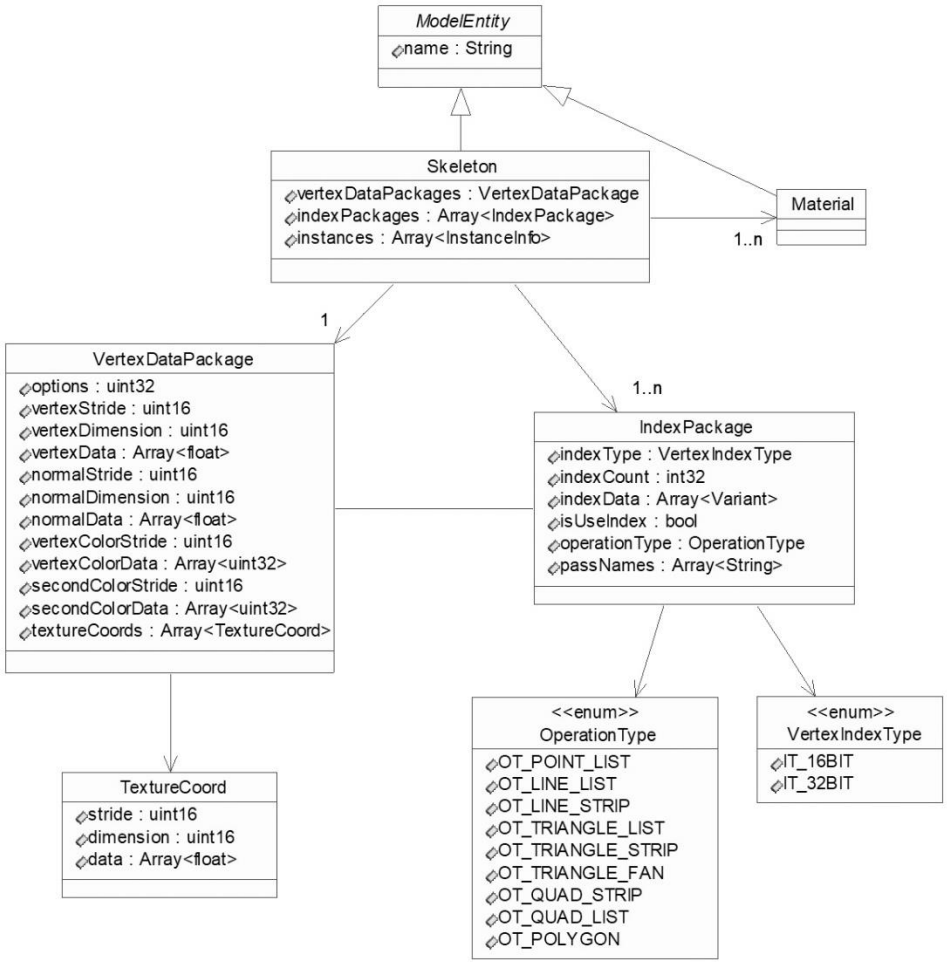


Figure 4 — Logical schema of objects in Skeleton

Table 19 — Attributes of Skeleton object

Attribute name	Type	Description
vertexDataPackages	VertexDataPackage	vertexDataPackages, represented by vertexDataPackage, see Table 20
indexPackages	Array<IndexPackage>	indexPackages, used by IndexPackage object, see Table 23
instanceInfos	Array<InstanceInfo>	Instantiation information group, represented by InstanceInfo object, can support point plug-in mode, see Table 22

Table 20 — Attributes of VertexDataPackage object

Attribute name	Type	Description
options	uit32	Data options, used for storing extended data
vertexStride	uint16	Coordinate offset in array

Table 20 (continued)

Attribute name	Type	Description
vertexDimension	uint16	Vertex dimension
vertexData	Array<float>	Vertex coordinates array
normalStride	uint16	The offset of the normal vector in the array
normalDimension	uint16	Normal vector dimension
normalData	Array<float>	Array composed by components of normal vector
vertexColorStride	uint16	Vertex color offset in array
vertexColorData	Array<uit32>	Array of vertex color
vertexAttributeStride	uint16	Vertex attribute offset in array
vertexAttributeData	Array<uit32>	Array of vertex attribute, use for storing ID information of model object
textureCoords	Array<TextureCoord>	Array of texture coordinates, represented by TextureCoord object, see Table 21

Table 21 — Attributes of TextureCoord object

Attribute name	Type	Description
stride	uint16	Offset
dimension	uint16	Dimension of texture coordinates
data	Array<float>	Array of texture coordinates

Table 22 — Attributes of InstanceInfo object

Attribute name	Type	Description
matrixValues	double[16]	4×4 Matrix
objectID	uint32	Object ID

Table 23 — Attributes of IndexPackage object

Attribute name	Type	Description
indexType	VertexIndexType	Vertex indexing type, represented by VertexIndexType object, see Table 24 for enumerates
indexCount	int32	Number of vertex indexing
indexData	Array<Variant>	Vertex data, it can be Short array or Integer array, depends on vertex indexing type
isUseIndex	bool	Whether to use indexing
operationType	OperationType	Indexing organization, represented by OperationType, see Table 25 for enumerate
passNames	Array<String>	Use the name of Pass object while rendering.

Table 24 — Enumerations of VertexIndexType object

Enumeration value	Type	Description
IT_16BIT	uint16	16 bytes unsigned integer
IT_32BIT	uint32	32 bytes unsinged integer

Generally, the enumeration value is selected according to the number of vertices. Vertex number greater than 65535, use IT_32BIT. Vertex number less than 65535, use IT_16BIT.

Table 25—Enumerations of OperationType object

Enumeration value	Type	Description
OT_POINT_LIST	int	Single point
OT_LINE_LIST	int	Two points line
OT_LINE_STRIP	int	Line string
OT_TRIANGLE_LIST	int	Triangle
OT_TRIANGLE_STRIP	int	Triangle strip
OT_TRIANGLE_FAN	int	Triangle fan
OT_QUAD_STRIP	int	Quadrilateral strip
OT_QUAD_LIST	int	Quadrilateral string without sharing edges
OT_POLYGON	int	Polygon

7.2.1.3 Matrial object

Material object is composed by Pass,which records the name of texture object used by Material and is expressed in JSON format. See Table 5 for logical schema of Material object, see Table 26 for the meanings of attributes.

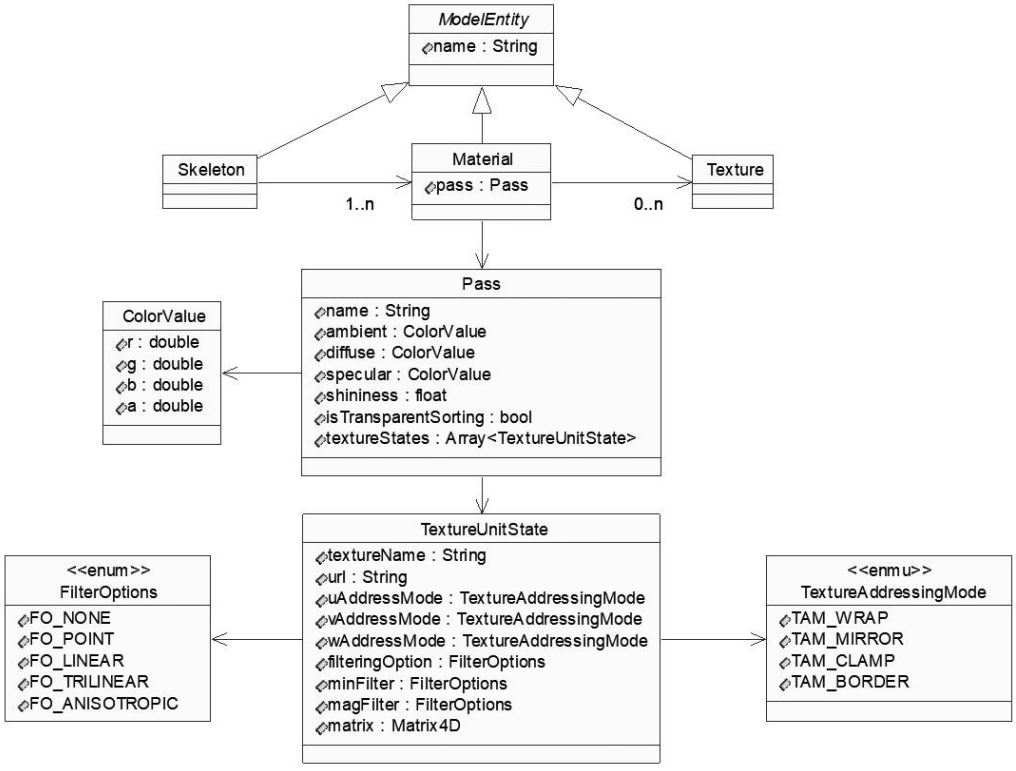


Figure 5 — Logical schema of Material object

Table 26 — Attributes of Material object

Attribute name	Type	Description
pass	Pass	Rendering pass, represented by Pass object, see Table 27

Table 27 — Attributes of Pass object

Attribute name	Type	Description
name	String	Pass name
ambient	ColorValue	Color of environment light, represented by ColorValue object, including components of r,g,b,a, see Table 28
diffuse	ColorValue	Scattered light, represented by ColorValue object, including components of r,g,b,a, see Table 28
specular	ColorValue	Specular light, represented by ColorValue object, including components of r,g,b,a, see Table 28
shininess	float	Gloss, which affects the intensity of the highlights of the reflected light
isTransparentSorting	bool	Whether to use transparent sorting. Range: { 'True', 'False'}. Corresponding to 'True', 'False'
textureStates	Array<TextureUnitState>	The texture information is represented by the texture information TextureUnitState object. Texture information, represented by TextureUnitState object, including textureName, url, uAddressMode, vAddressMode, wAddressMode, filteringOption, minFilter, magFilter, matrix, see Table 29

Table 28 — Attributes of ColorValue object

Attribute name	Type	Description
R	double	Red, value range 0.0 to 1.0
G	double	Green, value range 0.0 to 1.0
B	double	Blue, value range 0.0 to 1.0
A	double	Transparency, value range 0.0 to 1.0

Table 29 — Attributes of TextureUnitState object

Attribute name	Type	Description
textureName	String	Texture name
url	String	Path of texture resource
uAddressMode	TextureAddressingMode	Texture coordinate addressing mode in u direction, represented by TextureAddressingMode, see Table 30
vAddressMode	TextureAddressingMode	Texture coordinate addressing mode in texture coordinate v direction. Represented by TextureAddressingMode, see Table 30

Table 29 (continued)

Attribute name	Type	Description
wAddressMode	TextureAddressingMode	Texture coordinate addressing mode in w direction Represented by TextureAddressingMode, see Table 30
filteringOption	FilterOptions	Texture interpolation mode, represented by FilterOptions object, see Table 31
minFilter	FilterOptions	Interpolation mode used when texture is reduced, represented by FilterOptions object, see Table 31
magFilter	FilterOptions	Interpolation mode used when the texture is enlarged, represented by FilterOptions object, see Table 31
matrix	Matrix4D	Texture matrix, see Table 17

Table 30 — Enumerations of TextureAddressingMode object

Enumeration value	Type	Description
TAM_WRAP	int	Repeated texture
TAM_MIRROR	int	Symmetric reverse
TAM_CLAMP	int	Edge pixels to fill all texture coordinates greater than 1, edge stretched
TAM_BORDER	int	Border pixels to fill all texture coordinates greater than 1, the border is stretched

Table 31 — Enumerations of FilterOptions object

Enumeration value	Type	Description
FO_NONE	int	No filtering
FO_POINT	int	Proximity sampling
FO_LINEAR	int	Two-line filtering
FO_TRILINEAR	int	Three-line filtering
FO_ANISOTROPIC	int	Anisotropic filtering

7.2.1.4 Texture objects

See Table 6 for logical schema of Texture object. See Table 32 for meanings of Texture object.

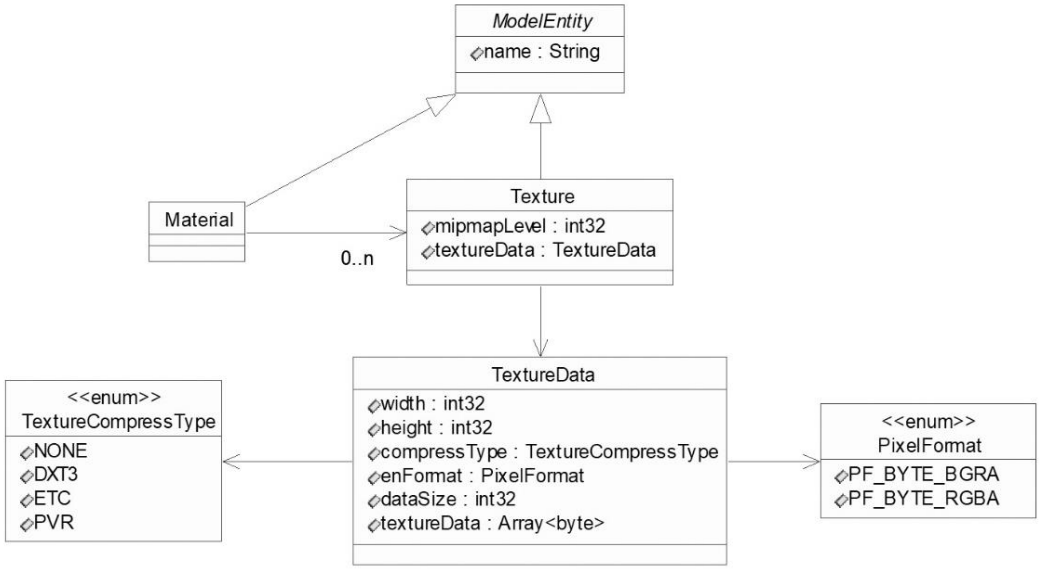


Figure 6 — Logical schema of Texture object

Table 32 — Attributes of Texture object

Attribute name	Type	Description
mipmapLevel	int32	Layer number of mipmap in texture object
textureData	TextureData	Texture data, represented by TextureData object, see Table 33

Table 33 — Attributes of TextureData object

Attribute name	Type	Description
width	int32	Number of horizontal pixels
height	int32	Number of vertical pixels
compressType	TextureCompressType	Texture compress method, represented by TextureCompressType, see Table 34
format	PixelFormat	Pixel format of texture, represented by PixelFormat, see Table 35
dataSize	int32	Binary stream size for textures
textureData	Array<byte>	Binary stream of texture data

Table 34 — Enumerations of TextureCompressType object

Enumeration value	Type	Description
NONE	int	No compressed format
DXT3	int	DXT3 texture compression format, suitable for PC
ETC	int	ETC texture compression format, suitable for Android
PVR	int	PVR texture compression format, suitable for iOS

Table 35 — Enumerations of PixelFormat object

Enumeration Value	Type	Description
PF_BYTE_BGRA	int	BGRA format
PF_BYTE_RGBA	int	RGBA format

7.2.2 Binary stream description of s3mb file

7.2.2.1 The main components of the s3mb file

The s3mb file is stored in the form of a binary stream, and the byte order is specified as Little-Endian, that is, the lower byte is discharged at the lower address end of the memory.

The content of the s3mb file stream is as follows:

```
S3MBFile{
  float header;            //S3mb file version number
  uint32 zippedSize;       // Bytes of zipped package
  byte* zippedPackage;     // Data compression package, length is zippedSize
};
```

After decompressing, the zippedPackage contains three parts: Reserved, Shell, and ModelEntities, see Figure 7. Reserved is the reserved four bytes. Shell stores PatchLOD, Patch, Geode objects. ModelEntities is entity data, including Skeleton, Material and Texture.

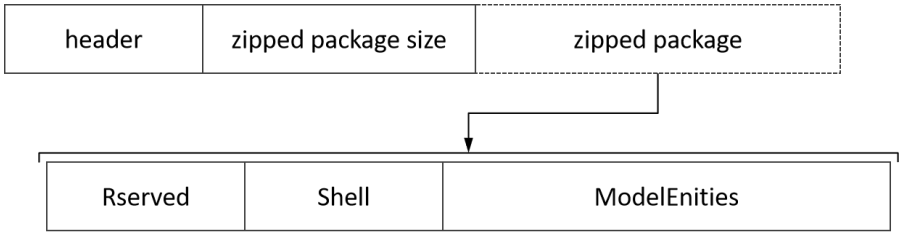


Figure 7 — Binary data package of s3mb file

7.2.2.2 Binary stream description of Shell

Binary stream of Shell and related objects meet the following specifications:

```
Shell{
  uint32 streamSize;               // Binary stream bytes of the shell
  PatchLOD patchLod;
};
PatchLOD {
  int32 patchCount;                 //Number of Patch objects
  Patch patches[patchCount];
};
Patch{
  float lodFactor;                  //Switching factor of LOD
  RangeMode rMode;                  //Switching mode of LOD, stored as int16
  BoundingSphere boundingSphere;    //Bounding Sphere
```

```
  String strChildTile;               // The relative path of the mounted subfile
  int32 geodeCount;                  // Number of Geode included
  Geode geodes[geodeCount];
};
RangeMode{
  Distance_From_EyePoint,    // Switching based on the distance from eyes point to camera
  Pixel_Size_OnScreen        // Switching based on the pixel size of screen projection
};
BoundingSphere{
  double x;                      //X coordinate to central point
  double y;                      //Y coordinate to central point
  double z;                      //Z coordinate to central point
  double r;                     //Radius of bounding sphere
};
Geode{
  Matrix4D matrix;
  int32 skeletonCount;
  String skeletonNames[skeletonCount];  //Skeleton name
};
Matrix4D{                       //4 × 4 matrix, row main order
  double values[16];
};
```

Only skeleton name is stored in Geode of Shell, and corresponding entities is stored in ModelEntites.

7.2.2.3 Binary stream of ModelEntities

Bindary stream of ModelEntites and related objects meet the following specifications:

```
ModelEntities{
  uint32 skeletonStreamSize;        // Binary length of skeleton data flow, unit is byte
  int32 skeletonCount;
  Skeleton skeletons[skeletonCount];
  uint32 textureStreamSize;
  int32 textureCount;                // Binary length of texture data flow, unit if byte
  Texture textures[textureCount];
  String materials;                    //String of material (JSON)
};
Skeleton{
  String name;
  VertexDataPackage dataPack;
  int32 indexpackCount;
  IndexPacakge indexPacks[indexpackCount];
};
```

```
VertexDataPackage{
  byte reserved[4];                 //Reserve
  uint32 vertexCount;               //Vertex
  uint16 vertexDimension;
  uint16 vertexStride;
  float vertexData[vertexCount * vertexDimension];
  uint32 normalCount;               //Normal
  uint16 normalDimension;
  uint16 normalStride;
  float normalData[normalCount * normalDimension];
  int32 vertexcolorCount;           //Vertex color
  uint16 vertexColorStride;
  byte reserved[2];
  uint32  vertexColorData[vertexcolorCount];       //  The  color  is  stored  in  uint32,
byte[0]~byte[4] represent the value of R, G, B, A respectively
  int32 vertexAttributeCount;                      // Vertex attribute
  uint16 vertexAttributeStride;
  byte reserved[2];
  uint32 vertexAttributeData[vertexAttributeCount];
  uint16 texturecoordCount;       //Texture coordinates
  byte reserved[2];
  TextureCoord textureCoords[texturecoordCount];
  uint16 instanceInfoCount;       //Instantiation information
  InstanceInfo instanceInfo[instanceInfoCount];
};
TextureCoord{
   uint32 coodsCount;
   uint16 dimension;
   uint16 stride;
   float data[coodsCount*dimension];
};
InstanceInfo{                   //Instantiation information
  double matrixvalues[16];      //Matrix, row main order
  uint32 objectID;              //Object ID
};
IndexPacakge{
  uint32 indexCount;
  VertexIndexType enIndexType;         //Store as byte
  byte reserved;
  OperationType opType;               //Store as byte
  byte reserved;
  variant  indexData[indexCount];   //  Indexing  value.  Iindicating:  If  IndexType  was
IT_16BIT,variant type is unit16; If IndexType was IT_32BIT, variant type is unit32
```

```
  int32 passCount;
  String passNames[passCount];
};
IndexType{
  IT_16BIT = 0,               // Indexing value represented by unit16
  IT_32BIT = 1                // Indexing value represented by uin32

};
OperationType{
  OT_POINT_LIST = 1,           //Single point
  OT_LINE_LIST  = 2,           //Two-points line
  OT_LINE_STRIP = 3,           //Line strip
  OT_TRIANGLE_LIST = 4,        //Triangle
  OT_TRIANGLE_STRIP = 5,       //Triangle strip
  OT_TRIANGLE_FAN = 6,         //Triangle fan
  OT_QUAD_STRIP = 8,           //Striped quadrilateral
  OT_QUAD_LIST = 9,            // Quadrilateral string, no shared edges
  OT_POLYGON = 10,             //Polygon
};
Texture{
  String strName;
  int32 mipMapLevel;
  TextureData texData;
};
TextureData{
  int32 width;
  int32 height;
  TextureCompressType compressType;     //Stored as uint32
  int32 datasize;
  PixelFormat pixelFormat;              // Stored as uint32
  byte data[datasize];
};
TextureCompressType{
  TC_NONE = 0,
  TC_DXT3 = 14,
};
PixelFormat{
  PF_BYTE_BGRA = 12,
  PF_BYTE_RGBA = 13,
};
```

The above is the binary stream specification for ModelEntities.

7.3 Indexing tree file

The indexing tree file is stored as a json file with .json extension. See Table 36 for the meanings of each label.

Table 36 — Labels of indexing tree

Label name	Type	Description
name	String	Tile name
tileInfo	TileInfo	Tile information, see Table 37
status	Status	Status of tile data, represented by Status object, including LODCount element (Sum of LOD levels) and TilesCount element (Sum of tiles), see Table 38

Table 37 — Labels of TileInfo object

Label name	Type	Description
lodNum	int	Number of LOD level that has root node, increasing in number from top to down, start number is 0
modelPath	String	Path of data file, relative to indexing file itself
rangeMode	String	Distance switching mode. Value Range: { 'distanceFromEyePoint', 'pixelSizeOnScreen' }. Correspomding to: { 'Distance from viewpoint to tile', 'Pixel number of screen projection'}
rangeValue	double	Child-node switches threshold value
boundingBox	BoundingBox	Boundingbox of data, represented by BoundingBox object, see Table 12
children	Array<TileInfo>	Child-node information

Table 38 — Labels of Status object

Label name	Type	Description
lodCount	int	Sum of LOD number of tile
tilesCount	int	Sum of tiles

7.4 Attribute file

7.4.1 Structure

Attribute file contains attribute description file and attribute data file. The name of attribute description file is specified as attribute.json, located in the same level of directory as description file (.scp). Attribute data file takes name of the root node file of TileTree, extension is .s3md (Spatial 3D Model Description). One root node corresponds to one attribute data of .s3md file, which is located in the same level of directory with data file (.s3mb).

7.4.2 Description file

Attribute description file describes ID range and filed information for objects of every layer, in json format. See Table 39 for labels of attribute description file.

Table 39 — Labels of attribute description file

Label name	Type	Description
layerInfos	Array<LayerInfo>	Attribute description for every layers, represented by LayerInfos object, including LayerInfo which is attribute description element of single layer, see Table 40.

Table 40 — Labels of LayerInfo object

Label name	Type	Description
layerName	String	Layer name
iDRange	IDRange	ID range, represented by IDRange, including min element which is ID minimum value and max element which is ID maximum value
fieldInfos	Array<FieldInfo>	Collection of file information, represented by FieldInfo, see Table 42

Table 41 — Labels of IDRange object

Label name	Type	Description
min	int32	Minimum value of ID for the layer object
max	int32	Maximum value of ID for the layer object

Table 42 — Labels of FieldInfo object

Label name	Type	Description
name	String	Filed name
alias	String	Filed alias
type	String	Field type. Value range: { ‘bool’, ‘int16’, ‘uint16’, ‘int32’, ‘uint32’, ‘int64’, ‘uint64’, ‘float’, ‘double’, ‘wchar’, ‘text’, ‘date’, ‘time’, ‘timestamp’}. Text is String type, date is date type, time is time type, timestamp is timedate type. The representation of date and time shall be as specified in GB/T 7408-2005. For the meaning and range of other types, see 5.1 and 5.2
size	int32	Field length
isRequired	bool	Required. Value Range: { ‘True’, ‘False’}. Corresponding to true and false

7.4.3 Data File

Attribute data file contains attribute description information of every layers and attribute value of every object, storage in json format, compression in zip, see Figure 8.

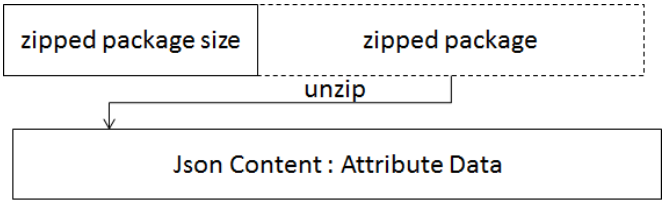


Figure 8 — Binary stream structure of attribute data file

Binary stream structure is as follows:

```
AttributeData{
  uint32 nZippedSize;       //Bytes of zipped package
  byte* zippedPackage;      //Compressed attribute data
}
```

After decompression, data is string type in json format. See Table 43 for labels of attribute data.

Table 43 — Labels of attribute data file

Label name	Type	Description
layer	Array<LayerInfo>	Layer information, represented by LayerInfo object, including IDRange, FieldInfos and Records, see Table 44

Table 44 — Labels of Layerinfo object

Label name	Type	Description
idRange	IDRange	ID range, represents the max and min ID of corresponding range of tiles, represented by IDRange, see Table 41
fieldInfos	Array<FieldInfo>	Union of filed information, represented by FieldInfo, see Table 42
records	Array<Record>	Union of record information for attribute data, represented by Record object, see Table 45

Table 45 — Labels of Record object

Label name	Type	Description
id	int32	Object ID
values	Array<Value>	Detailed description of attribute value for all filed, represented by Value object, see Table 46

Table 46 — Labels of Value object

Label name	Type	Description
name	String	Filed name
value	Variants	File value

Appendix A
(Information Annex)
Sample code

A.1 Example of description file

Take "Bird-Nest.scp" as an example, it contains two TileTrees named Tile_-7281_21185_0000 and Tile_-7282_21183_0000, the specific content is as follows:

```
{
  "asset":"SuperMap",
  "version":1.0,
  "dataType":"BIM",
  "pyramidSplitType":"QuadTree",        //Quadtree split
  "lodType":"Replace",                  //LOD is Replace mode
  "position":                           //Insertion point in degrees
    {
     "x":116.36,
     "y":39.99,
     "z":0.0,
     "units":"Degree"
     },
  "geoBounds":                  //Geospatial range
    {
     "left":116.3635,
     "top":40.0018,
     "right":116.3755,
     "bottom":39.9932
    },
  "heightRange":                    //Height range
    {
     "min":9.4875,
     "max":119.9612
    },
  "wDescript":                  // W-bit description
    {
     "category":"",
     "range":
        {
         "min":0.0,
         "max":0.0
        }
    },
   "tiles":                    // The root node file path and boundingbox corresponding to each TileTree (local coordinate system)
```

```
  [
  {                        // Information for the first root node
     "url":"./Tile_-7281_21185_0000/Tile_-7281_21185_0000.s3mb",
     "boundingbox":
        {
         "min":
            {
             "x":245.36567664297159,
             "y":-534.7293082718718,
             "z":-34.66962171293413
            },
        "max":
            {
             "x":443.1873785885407,
             "y":-336.9076063263026,
             "z":163.152080232635
            }
        }
    },
   {                                // Information about the second root node
     "url":"./Tile_-7282_21183_0000/Tile_-7282_21183_0000.s3mb",
     "boundingbox":
        {
         "min":
            {
             "x":-604.2845700298257,
             "y":92.21901333930407,
             "z":-190.14669717375353
            },
        "max":
            {
             "x":-147.10063304583208,
             "y":549.4029503232977,
             "z":267.03723981024009
            }
        }
   }
 ]
}
```

A.2 Example of indexing tree file

One of the TileTrees named Tile_-7281_21185_0000 has an indexing tree file Tile_-7281_21185_0000.json in the folder.

```
{
 "lodTreeExport":
 {
  "name":"Tile_-7281_21185_0000",
  "tileInfo":
  {
   "lodNum":0,                                  // LOD layer number, layer 0 is the root node
   "modelPath":"Tile_-7281_21185_0000.s3mb", // File path corresponding to the root node
(relative to the index file)
  "rangeMode":"pixelSizeOnScreen",            // Distance switching mode of LOD
  "rangeValue":1.0,                            // Threshold for child-node switching
  "boundingBox":                       // Boundingbox of root node (local coordinate system)
  {
   "min":
   {
    "x":-68.02222442626953,
    "y":-43.73067092895508,
    "z":9.495752334594727
   },
   "max":
   {
    "x":68.02222442626953,
    "y":43.73127746582031,
    "z":119.96125030517578
   }
  },
  "children":                   // Child-node information
  [
   {
    "tileInfo":
    {
     "lodNum":1,
     "modelPath":"Tile_-7281_21185_0000_0004_0000.s3mb",
     "rangeMode":"pixelSizeOnScreen",
     "rangeValue":2.0,
     "boundingBox":
     {
      "min":
      {
       "x":-68.02222442626953,
     "y":-43.73067092895508,
     "z":9.495752334594727
```

```
    },
   "max":
   {
    "x":68.02222442626953,
    "y":43.73127746582031,
    "z":119.96125030517578
   }
 },
 "children":[…]           //Child-node information of this node (recursive)
 }
 }
 ],
 "status":              // General description of TileTree
 {
  "lodCount":5,
  "tilesCount":6
 }
 }
 }
}
```

A.3 Material content

The material content is contained in the .s3mb file, see the content in 7.2.1.3 and 7.2.2.3. The sample data is as follows:

```
{
 "materials":         //Material object description information collection
    [
     {
     "material":            // The first material object
        {
         "id":"0_10710_Sec_0005_-7281_21185_0000_0000_0",  // ID of the material object
         "ambient": {"r":1.0,"g":1.0,"b":1.0,"a":1.0},
         "diffuse":{"r":1.0,"g":1.0,"b":1.0,"a":1.0},
         "specular":{"r":1.0,"g":1.0,"b":1.0,"a":1.0},
         "shininess":0.0,
         "transparentsorting":false,
         "textureunitstates":       // Included texture  information
          [
           {
            "textureunitstate":
             {
             "id":"0_10710_Sec_0005_-7281_21185_0000_0000", // ID of the texture object
```

```
            "url":"",   // URL is empty, through ID, texture data associates with the texture data that stored in S3MB file.

            "addressmode":{"u":0,"v":0,"w":0},
            "filteringoption":-842150451,
            "filtermin":2,
            "filtermag":2,
            texmodmatrix":[1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0]
            }
          }
       ]
      }
     }
],
}
```

A.4 Attribute description file

Attribute-related data is optional. If attribute data exists , the name of the attribute description file is limited to attribute.json and is in the same directory as the "Bird-Nest.scp" file, see Table 39. The sample data is as follows:

```
{
 "layerInfos":
  [
    {
     "layerName":"Building_Sub",          //Corresponding original dataset name
     "idRange":{"minID":1,"maxID":10},  //ID range of dataset objects contained in TileTreeSet
    "fieldInfos":                        //Field description
     [
        {
         "name":"SmID",
         "alias":"SmID",
         "type":'int32',
         "size":4,
         "isRequired":true
         },
        {
         "name":"MODELNAME",
         "alias":"ModelName",
         "type":'String',
         "size":30,
       "isRequired":false
     },
```

```
    ]
  }
 ]
}
```

A.5 Attribute data file

Attribute-related data is optional. If attribute data exists, in addition to the attribute description file used to describe the relevant attribute information of the layer, in each TileTree folder, there is also a file with the same name as the root node with the extension of .s3md, which is used to store all attribute data under this TileTree, as shown in Table 43.

Take Tile_-7281_21185_0000 as an example, the attribute data is the Tile_-7281_21185_0000.s3md file in the folder, the specific content is as follows:

```
{
 "layerInfos":              //Collection of attribute dataset
    [
        {
         "idRange":     //The ID range of object in TileTree
            {
             "minID":1,
             "maxID":1
            },
          "fieldInfos":     //Information described by each field (SmID, MODELNAME)
            [
                {
                 "name":"SmID",
                 "alias":"SmID",
                 "type":" int32" ,
                 "size":4,
                 "isRequired":true
                 },
                {
                 "name":"MODELNAME",
                 "alias":"ModelName",
                 "type":" String" ,
                 "size":30,
                "isRequired":false
                },
            ]
          "records":    //Collection of individual filed value
            [
             {
              "id":1,
```

```
            "values":     //The value of each field of the object with ID = 1 (SmID,
MODELNAME)
            [
               {
                "name":"SmID",
                "value":1
               },
               {
                "name":" ModelName",
                "value ":"Bird-Nest"
               },
              ]
           }
        ]
      }
   ]
}
```

Bibliography

[1]GB/T 30170—2013 *Geographic information—Spatial referencing by coordinates*

[2]GB/T 23707—2009 *Geographic information—Spatial schema*

[3]ISO 19101 *Geographic information—Reference model*