

Group Standard

T/ CAGIS 1—2019

Data format for spatial 3D model

Release Date: 2019 – 07 – 19

Implement Date: 2019 – 08 – 31

Directory

Preface.....	II
Introduction.....	III
1 Scope.....	1
2 Normative References.....	1
3 Terms and Definitions.....	1
4 Symbols and Acronym.....	2
4.1 Acronym: the following acronyms apply for this document.....	2
4.2 UML Pictogram.....	2
4.3 Description of UML diversity.....	3
5 Basic Regulation.....	3
5.1 Basic Data Type.....	3
5.2 String Type.....	4
5.3 Json Format for Storage.....	4
6 Organization Structure.....	4
6.1 File Organization.....	4
6.2 Description of Tree Structure.....	5
7 Storage Format.....	6
7.1 Description File.....	6
7.2 Data File.....	10
7.3 Index Tree File.....	23
7.4 Attribute File.....	24
Appendix A (Information Annex) Sample Code.....	27
References.....	34

Preface

This standard is drafted in accordance with the specifications given in GB/T1.1-2009. Please note that some contents may involve patents.

The issuing organization of this document is not responsible for identifying these patents. This standard is proposed and centralized management by the China Association of Geographic Information System (CAGIS).

This standard was drafted by following units:

- the Information Center of the Ministry of Natural Resources ,
- the National Geomatics Center of China,
- Beijing SuperMap Software co.,Ltd,
- China Academy of Urban Planning and design,
- China Academy of Building Research,
- China Construction Technology Consulting Co.,Ltd,
- China Architecture Design and Research Group,
- Hebei third surveying and mapping institute,
- Guangzhou HongPeng Helicopter Remote Sensing Technology Co., Ltd,
- Chengdu RYD Information Technology Co.,Ltd.

The main drafters of this standard: Zhong Ershun, Song Guanfu, Feng Zhenhua, Zhou Qin, Liu Juhai, Gao Pi, Jia Wenyu, Zhang Jingbo, Xu Hui, Yang Tao, Ma Encheng, Wei Lai, Yu Jie, Li Yungui, Xu Peng, Sun Wei, Cai Wenwen, li Meng, He Qian.

Introduction

In recent years, the development of data acquisition techniques such as oblique photogrammetry and laser scanning has effectively reduced the cost and time period of acquisition with higher accuracy for 3D spatial data. As the amount of massive 3D spatial data are continuously growing, which efficient publishing, sharing and exchange are an important aspect to 3D GIS research. This standard defines an open and scalable data format for spatial 3D model—Spatial 3D Model(S3M), which are designed to apply for data transmission, exchange and sharing in order to solve the difficulties in the geographic information platforms such as data storage, high-efficient visualizing, sharing and interoperating among multi-source 3D spatial data on different devices (mobile, browser, PC), which plays a key role in the acceleration of data sharing and further application of 3D geo-spatial data.

At present, S3M is containing below data types:

- a) Traditional model: manual 3D model data;
- b) Realistic 3D data: including oblique photography and points cloud data;
- c) BIM data: 3D model data adopted the BIM design software;
- d) Vector data: including 2D point/line/region data, 3D point/line/region data and 3D pipelines data.

Data Format for Spatial 3D Model

1 Scope

The standard defines the file structure and storage criterial for the spatial 3D model data format.

The standard applies for data transmission and sharing in both online and off-line environment; also applies for the 3D GIS-related applications of on multi-ends (mobile, browser and PC).

2 Normative References

The following documents are indispensable for the application of this paper. For dated references, only the dated version applies to this document. For undated references, the latest edition (including all amendments) applies to this document.

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 Standard Representation

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 additional terms and definitions apply.

3.1 Tile

Each tile corresponds to a 2D polygon or 3D rectangular space, spatial overlaying is available between adjacent tiles.

3.2 Root node

The root node of TileTree.

Note: every TileTree has a root node, the spatial scope of the TileTree is the union of which all child-nodes.

3.3 Tiletree

A spatial data structure is composed of which the root node divided into multi-levels from the top down, each node of the TileTree represents a tile.

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

3.4 Tiletreeset

A collection which is composed of one or more tiletrees.

3.5 Patch LOD

Representing the set of patches over a specific LOD under a tiletree.

Note: One LOD contains one or more patches.

3.6 Patch

A piece of data range in the PatchLOD

Note: A PatchLOD contains one or more patches.

Each patch contains zero or one parent-patch, contains zero or multiple child-patches;

The spatial scope of parent-patch is the union of which all child-patches.

The parent-child relation of the patch is tree structured.

3.7 Geode

Representing the data package to 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 abbreviated terms

4.1 Abbreviated terms

For the purpose of this Technical Specification, the following acronyms apply for this document.

EPSG The European Petroleum Survey Group

LOD Level of detail

UML Unified Modelling Language

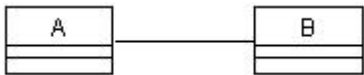
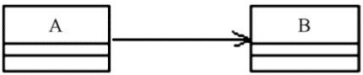
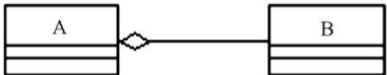
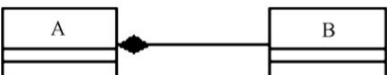
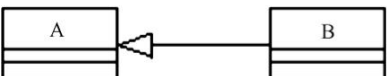
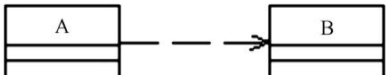
WKT Well-known Text Representation

4.2 UML Pictogram

The diagrams in explaining this standard are represented by UML pictogram.

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

Table 1—UML pictogram comparison table

Symbol	Name	Instruction
	Bi-directional	Represents the general relationship between the two classes A and B, both classes know the public properties and methods of the other class.
	Uni-directional	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.
	Aggregation	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.
	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

Multiple meanings of UML to this standard is in Table 2.

Table 2—Multiple meaning of UML

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 Basic Regulation

5.1 Basic Data Type

Basic data type involved in this standard is 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	$[-3.4 \times 10^{38}, 3.4 \times 10^{38}]$	Single-precision float
double	8	$[-1.7 \times 10^{308}, 1.7 \times 10^{308}]$	Double-precision float
wchar	2	--	Wide character type

5.2 String Type

String types involved in this standard are described by the String object, encoding by Unicode, the rule of string set.

```
UTF 8. String {
    Int 32 length ; //Number of bytes
    Bytestr [length]; //Data content
}
```

5.3 Json Format for Storage

Json format involved in this standard, encoding by UTF-8, without BOM prefix.

6 Organization Structure

6.1 File Organization

Files which formed data in this standard including; description file, data file, indexing tree file and attribute file.

Description file and data file are fundamental 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 is to accelerate the efficiency of Tile indexing; attribute file includes described file of attributes and attribute data file. The organizational form for each file is in Table 4.

Table 4—Organizational form of files

File type	Storage form	Storage regulation	Necessity
Description file	.scp	Described information of whole data in json format File name is customizable, the extension is “.scp”	Yes
Data type	.s3mb	File name is customizable, the extension is “.s3mb”	Yes
Indexing tree	.json	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.	No
Attribute description file	attribute.json	Each dataset attribute description information in the TileTreeSet. The data set description can be found in the GB/T 30320-2013. The full name of the file is limited to “attribute.json” and the same directory as the .scp file.	No
Attribute data file	.s3md	Attribute data 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	No

6.2 Description of Tree Structure

UML of tree structure is in Figure 1.

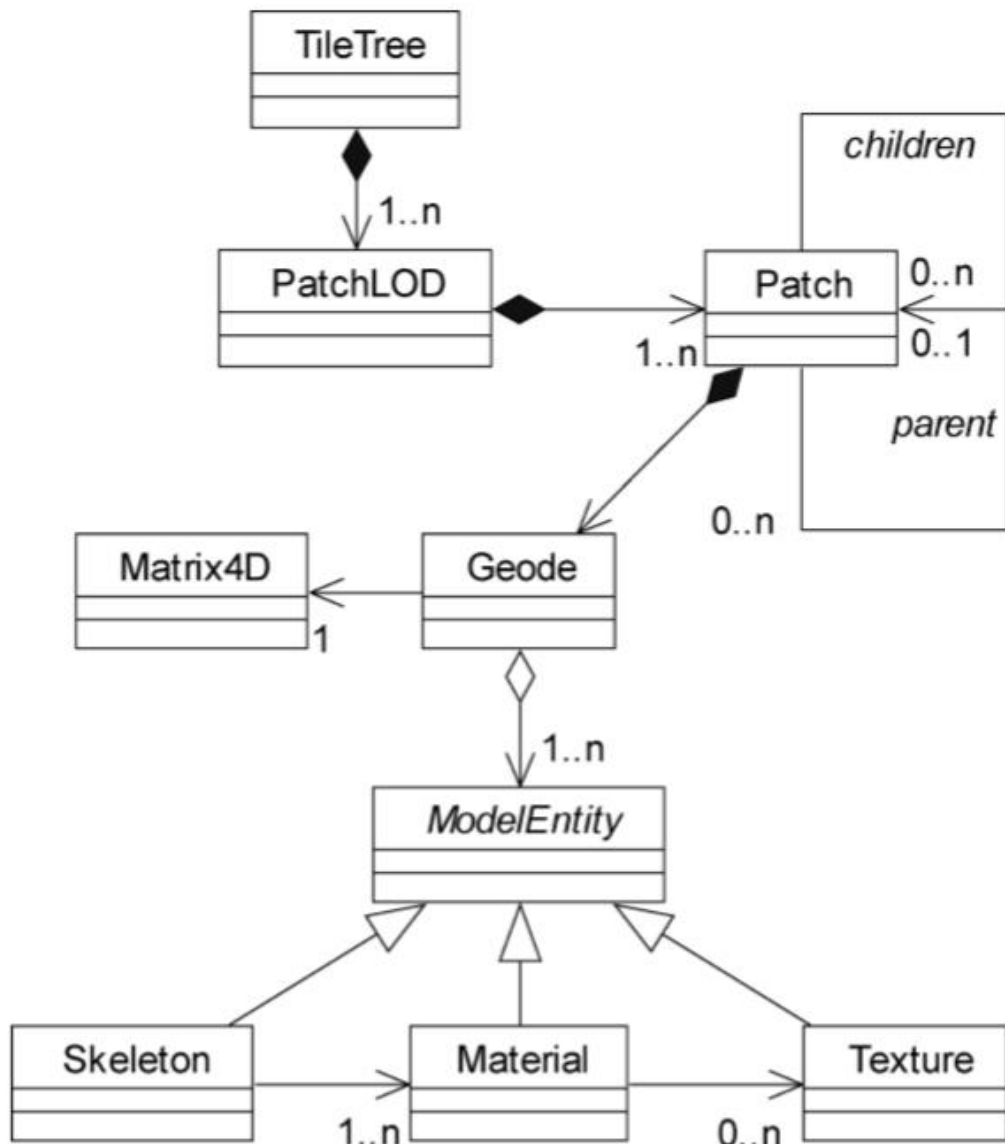


Figure 1—Tree structure UML diagram

7 Storage Format

7.1 Description File

7.1.1 Overview of the description file

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

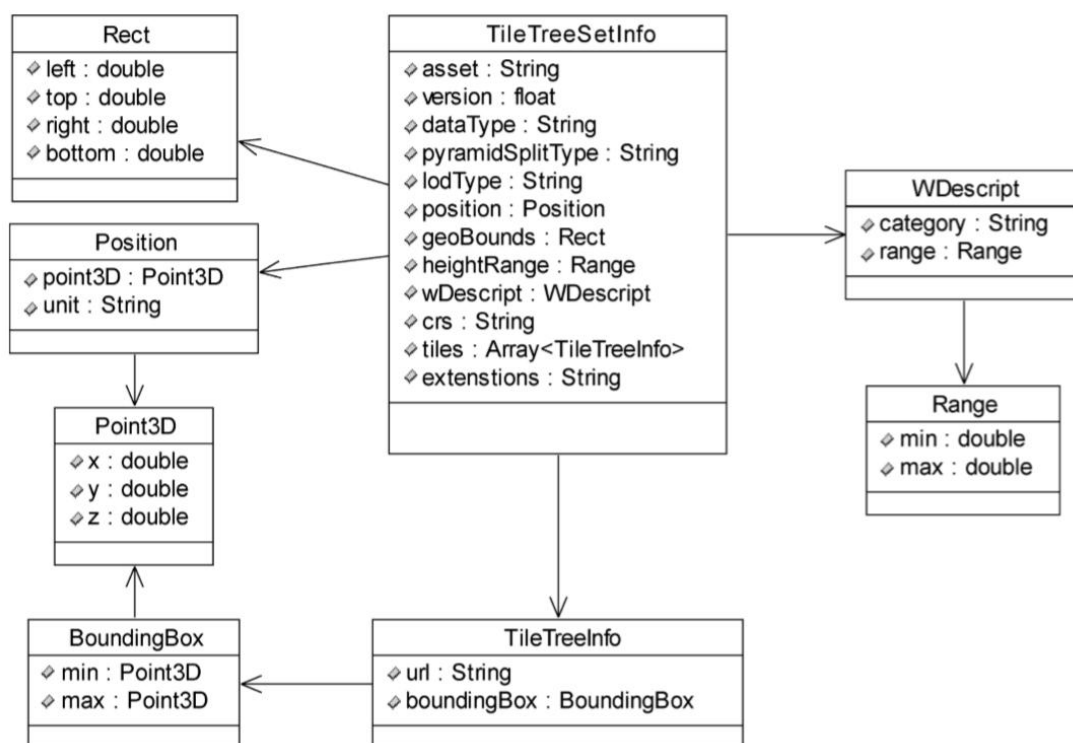


Figure 2— Description file UML diagram

7.1.2 Label information

Each label information of description file is in Table 5.

Table 5—Label information of description file

Label Name	Type	Description
asset	String	Basic information about the data, such as production vendor.
version	Float	Version number. Value range: { '1.0' } Value range: { '1.0' }.
dataType	String	3D geo-spatial data type. Value range: { 'Vector' , 'ObliquePhotogrammetry' , 'ArtificialModel' , 'BIM' , 'PointCloud' , 'PipeLine' } For 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' } For Quadtree, octree, R tree, K-D.
lodType	String	LOD type.

		Value range: { 'Add' , 'Replace' } For Add refinement, replace refinement.
geoBounds	Rect	Geographic area of data represented by Rect object, see Table 6.
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. crs:{ 'epsg:4326' } Representation format of epsg: { 'epsg:4326' } : 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

Figure 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

Figure 7—Labels of Range object

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

Figure 8—Labels of WDescript object

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

Figure 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 the provisions of GB/T 16831-2013
unit	String	Unit of spatial coordinate system Value range: { 'Degree' , 'Meter' } Corresponding to degree and meter respectively

Figure 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

Figure 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

Figure 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 are key parts in the structure formed by .S3MB (Spatial 3D Model Binary)files. One PatchLOD corresponding to one .S3MB files.

See Figure 3 for UML diagram.

See Table 13 for object meaning in S3MB files.

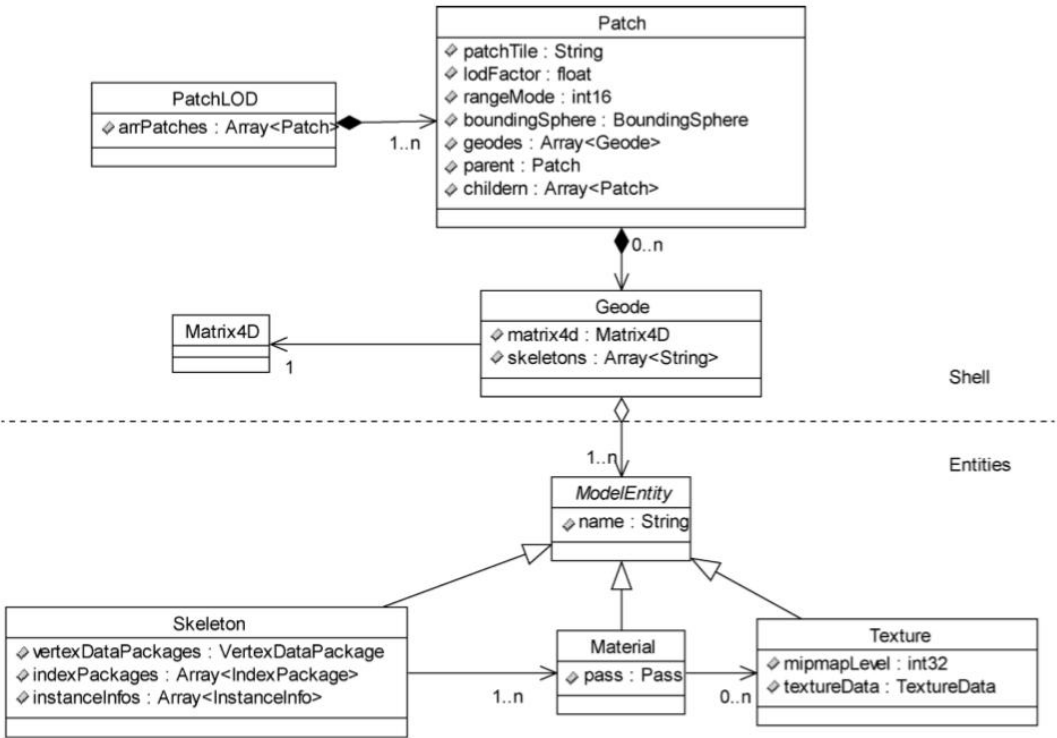


Figure 3—Objects in S3MB file

Table 13—Object of 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, LOD switches threshold, use 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.
z	double	Z value of central point.
r	double	Radius of boundingbox

Table 16—Attributes of Geode object

Attribute name	Type	Description
matrix4d	[no space]	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 only identifier under 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 and instantiation information;

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 related UML to skeleton object; see Table 19 for the meaning of each skeleton object.

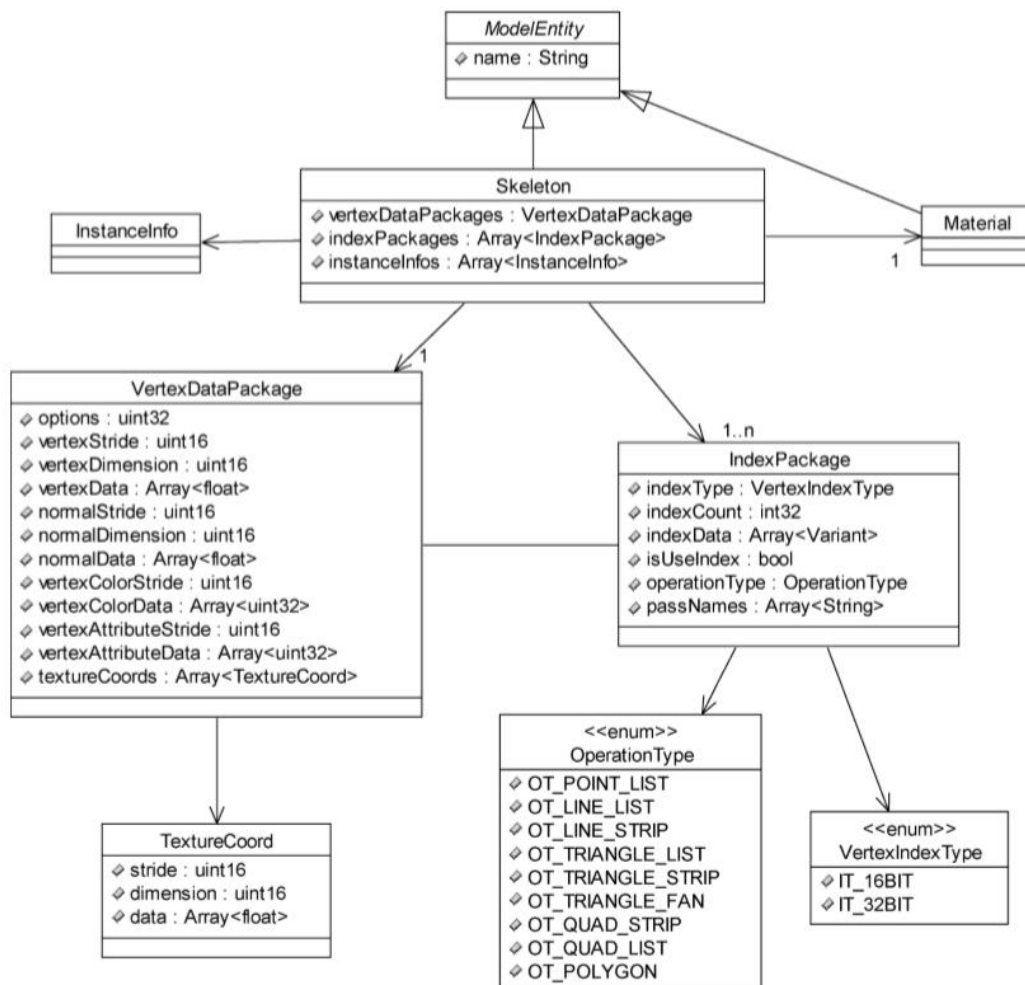


Figure 4—UML to skeleton objects

Table 19—Reference of skeleton objects

Attribute name	Type	Description
vertexDataPacka ges	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—Meanings of VertexDataPackage objects

Attribute name	Type	Description
options	uit32	Data options, used for storing extended data
vertexStride	uint16	Coordinate offset in array
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—Meaning of TextureCoor objects

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

Table 22—Reference of InstanceInfo objects

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

Table 23—Reference of IndexPackage objects

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—Reference of enumerates

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

Vertex number greater than 65535, use IT_32BIT; Less than 65535, use IT_16BIT;

Table 25—Reference of OperationType objects

Enumerate	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 composed by Pass, material object name is adopted in Pass, json format. See Table 5 for UML to material object, see Table 26 for attribute meanings.

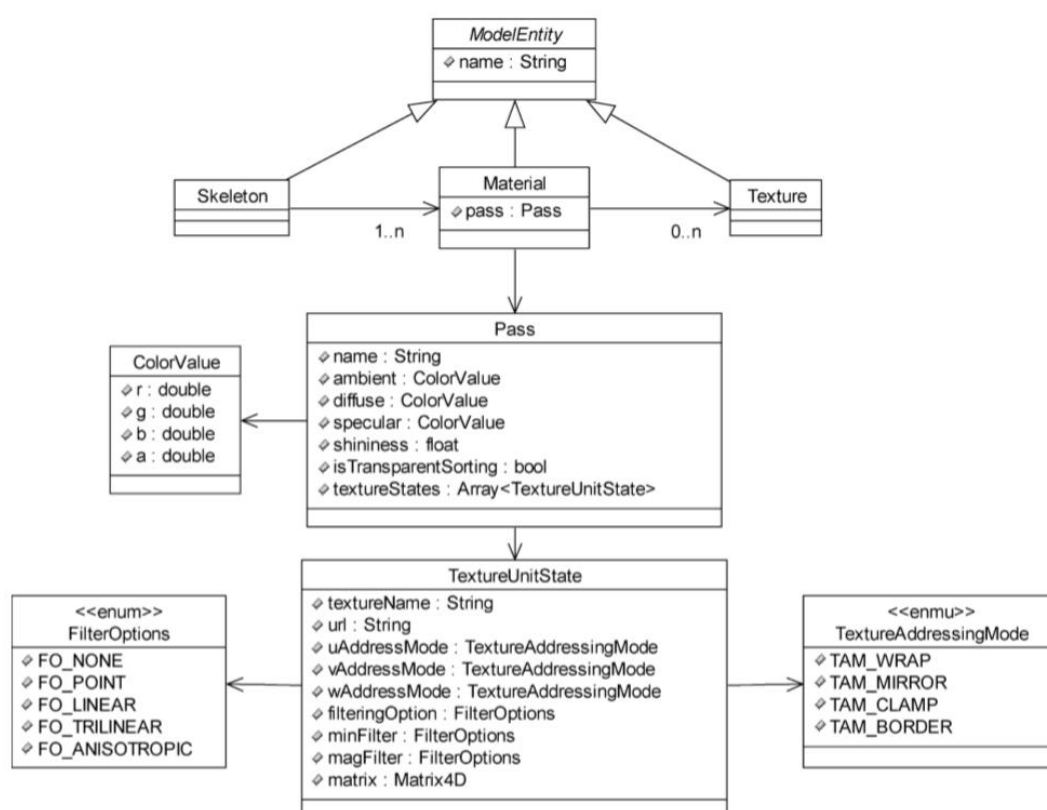


Figure 5—UML to material objects

Table 26—Reference of material objects

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

Table 27—Reference 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' } For '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. Contains textureName, url, uAddressMode, vAddressMode, wAddressMode, filteringOption, minFilter, magFilter, matrix and other elements, see Table 29

Table 28—Reference of ColorValue objects

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—Reference of TextureUnitState objects

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.
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—Enumerate Reference of TextureAddressingMode objects

Enumerate	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—Enumerate Reference of FilterOptions object

Enumerate	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 UML to Texture objects. See Table 32 for meanings of texture objects.

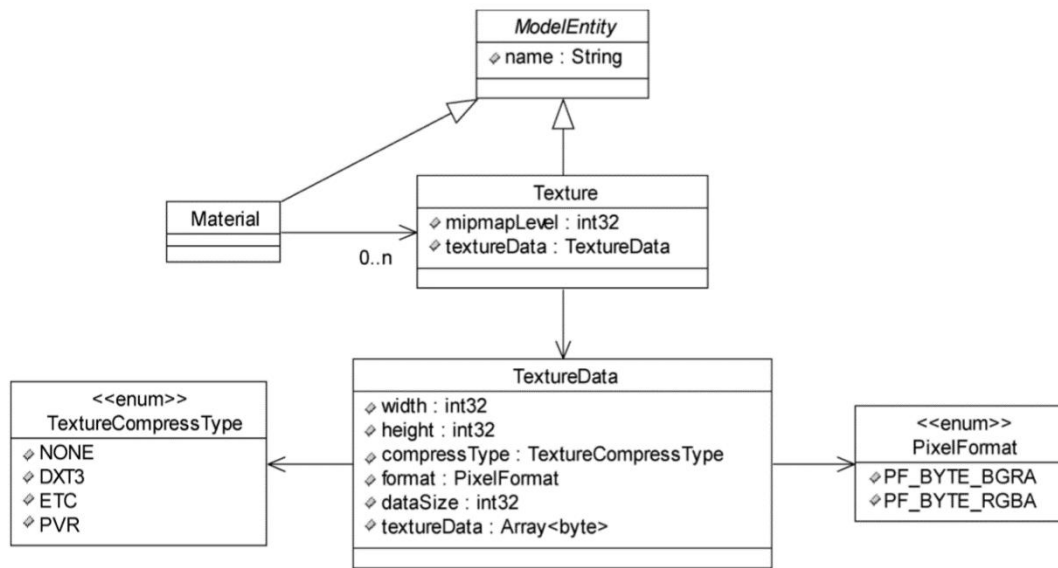


Figure 6—UML to Texture objects

Table 32—Reference 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—Reference 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—Reference of TextureCompressType object

Enumerate	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—Reference of PixelFormat object

Enumerate	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;   // zippedSize Data compression package, length is zippedSize
};
```

After the zippedPackage is decompressed, it 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.

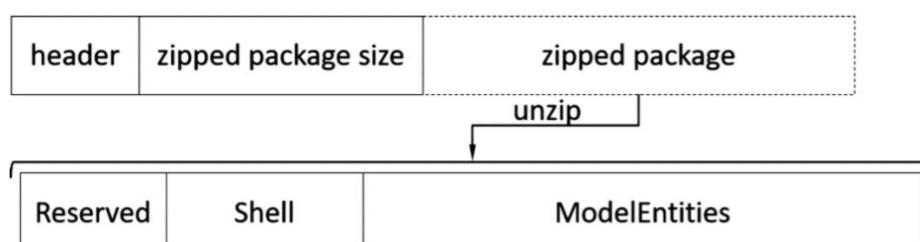


Figure 7—Binary data package of S3MB file

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 {
    int32 patchCount;          //Number of Patch objects
```

```

    Patch patches[patchCount];
};
Patch{
    float lodFactor;                //Switching factor of LOD
    RangeMode rMode;                //Switching mode of LOD
int16 LOD switching mode, stored as int16
    BoundingBoxSphere boundingSphere; //BoundingBoxSphere
    String strChildTile;            // The relative path of the mounted subfile
    int32 geodeCount;                // Number of Geode included
    Geode geodes[geodeCount];
};
Enum RangeMode{
    Distance_From_EyePoint, // Switching based on the distance from eyes point to camera
    Pixel_Size_OnScreen, // Switch according to the pixel size projected on the screen
    Switching based on the pixel size of screen projection
};
BoundingBoxSphere{
    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];
};

```

Shell is the only skeleton name in Geode, it's corresponding entities stored in ModelEntities.

7.2.2.3 Binary stream of ModelEntities

Binary stream of ModelEntities and related objects meet the following regulations:

```

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
Color adopted uin32 for storing, byte[0]~byte[4] represent 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;                                //ID Object ID
};

```

```

IndexPacakge{
    uint32 indexCount;
    IndexType enIndexType;          //byte Store as byte
    byte reserved;
    OperationType opType;          //byte Store as byte
    byte reserved;
    variant indexData[indexCount]; // Indexing value, indicating: 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; //uint32 Stored as uint32
    int32 datasize;
    PixelFormat pixelFormat;          //uint32 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 Index Tree File

The index tree file is stored in a json file with a .json extension. See Table 36 for the meaning of each label. The meaning of each label in the index tree file.

Table 36—Reference of indexing tree

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

Table 37—Reference 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' } { '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—Reference 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 corresponding to one attribute data of .S3MD file, 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 of every layers, in json format, see Table 39.

Table 39—Reference 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—Reference of Layer Info 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—Reference 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—Reference 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 should comply with the provisions of GB / T 7408-2005 For the meaning and range of other types, see Sections 5.1 and 5.2
size	int32	Field length
isRequired	bool	Required Value Range: { 'True' , 'False' }

7.4.3 Data File

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

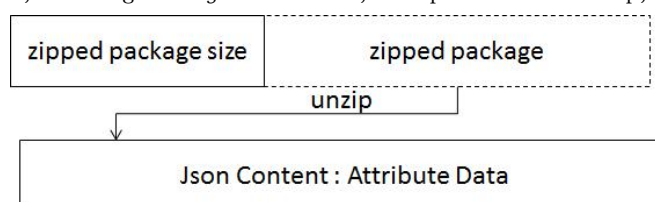


Figure 8—Binary stream structure of attribute data file

Binary stream structure:

```

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

Compressed data is string type in json format, see Table 43 for reference of attribute data

Table 43—Reference 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—Reference 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—Reference 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—Reference 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's 鸟巢.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",      //Quad tree 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 index tree file

For one of the TileTrees named Tile_-7281_21185_0000, there is an index 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)
      件)
      "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,
```

T/ CAGIS 1—2019

```
"boundingBox":
{
  "min":
  {
    "x":-68.02222442626953,
    "y":-43.73067092895508,
    "z":9.495752334594727
  },
  "max":
  {
    "x":68.02222442626953,
    "y":43.73127746582031,
    "z":119.96125030517578
  },
  "children":[...] //Children 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 the sections 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 data 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":[1.0,0.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0,0.0,0.0,0.0,0.0,1.0]
        }
      }
    ]
  }
},
]
}

```

A.4 Attribute description file

Attribute-related data is optional. If there is attribute data, the name of the attribute description file is limited to attribute.json and is in the same directory as the "鸟巢.scf" 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 describing the relevant attribute information of the layer,

In each TileTree folder, there is a file with the same name as the root node and an extension of .s3md.

Used to store all attribute data under this TileTree, see 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": // (SmID、MODELNAME)
The value of each field of the object with ID = 1 (SmID, MODELNAME)
      [
        {
          "name": "SmID",
          "value": 1
        },
        {
          "name": " ModelName",
          "value ": "鸟巢"
        },
      ],
    ]
  ]
}

```

References

- [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
-