3DML Open Format Specifications

Version: 1.0.1

Date: 1/7/2019

Overview

3DML is designed for streaming and rendering massive 3D geospatial content such as terrain surfaces, 3D city models, individual models and BIM/CAD. 3DML is based on the open source SQLite database with optional SpatialLite support and defines a database schema that contains a hierarchical data structure of the models, associated feature information and project information tables.

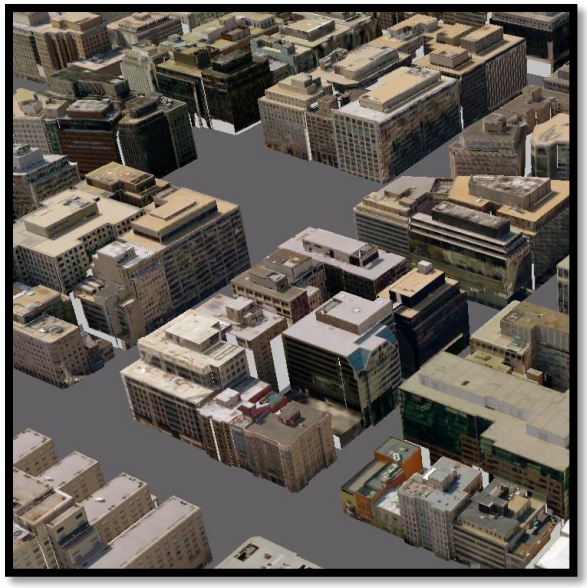
The 3DML database specification, associated block formats, and the associated feature information are open formats that are not dependent on any vendor-specific solution, technology, or products.

The 3DML database can be used as a local resource for 3D models or as an online database serving remote clients.

Main Uses for 3DML

3DML is mainly used for

- Terrain mesh models
- Urban area mesh models
- Individual models
- BIM and CAD models



3DML File Format

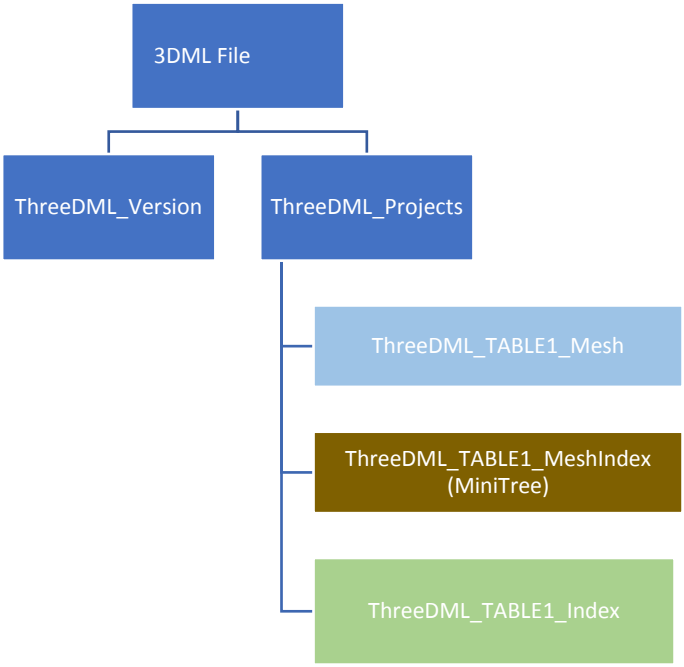
The 3DML file includes metadata information as well as the mesh and feature layers. The multiresolution tiles are held in a tree structure, where each node in the tree is split into several nodes (child nodes). The information in the child nodes represents the same information as the parent node but with more detail (higher-resolution).

The 3DML file is constructed from a few tables.

- Version – Stores 3DML version information.
- Projects table – Stores information on the 3DML dataset, including:
  - Bounding box
  - Unique GUID
  - Coordinate system of the dataset in well-known-text format
- Index table – Stores indexing information on the mesh models in Mini-tree format.
- Mesh table – Stores node blocks of mesh models including the geometry, texture and the ground surface.
- Feature table – Spatial table stores the classification polygons and their attribute information (not covered in this document).

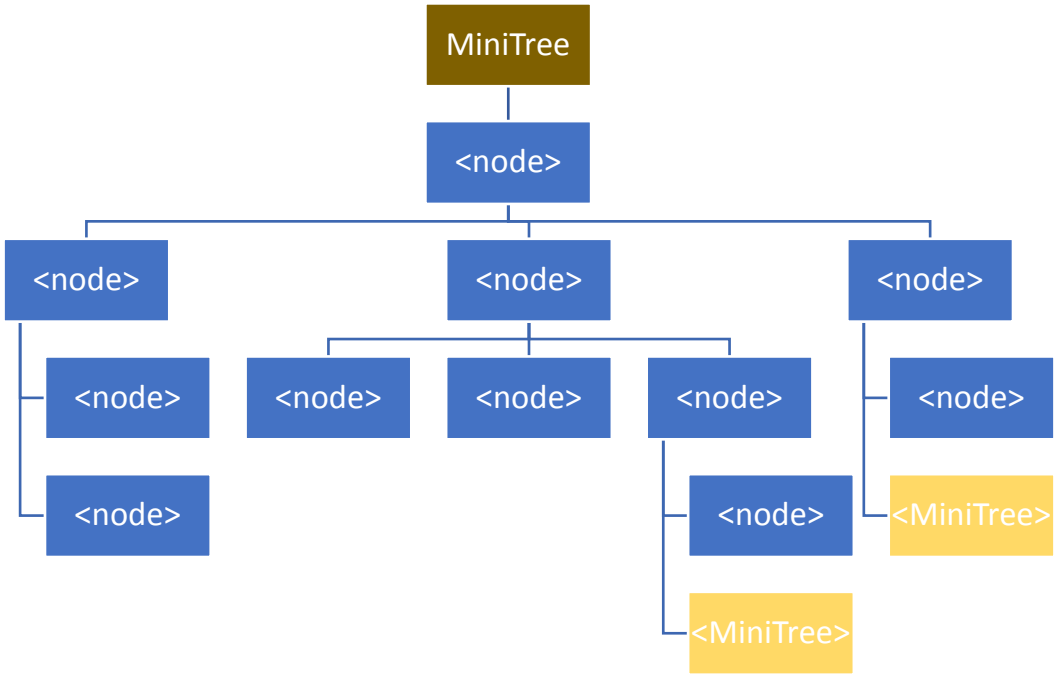
Detailed information about the tables and contents is found in the Tables chapter.

A schematic structure of a 3DML:

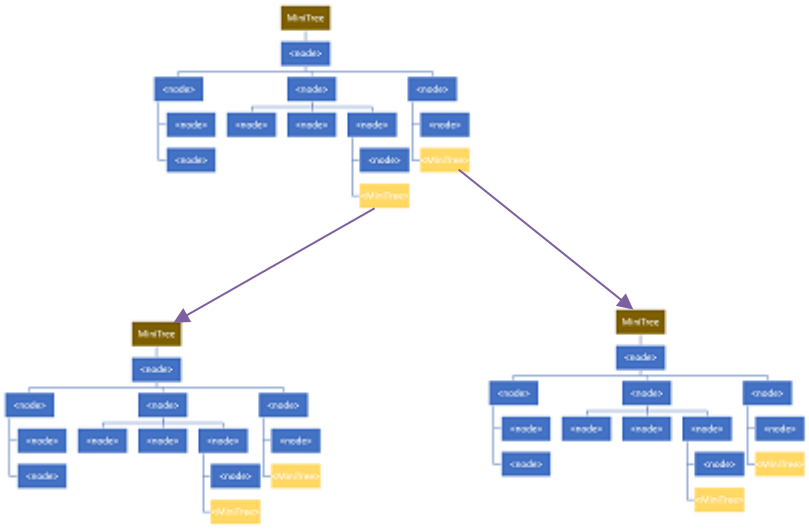


Mini-tree's are binary data that store the hierarchical structure of the Mesh Layer. Each mini-tree record holds general information for the node, e.g., Bounding Box, size in pixels to display the block, and reference, and references to child nodes

A schematic structure of a Mini-tree:



A schematic structure of linked Mini-trees:

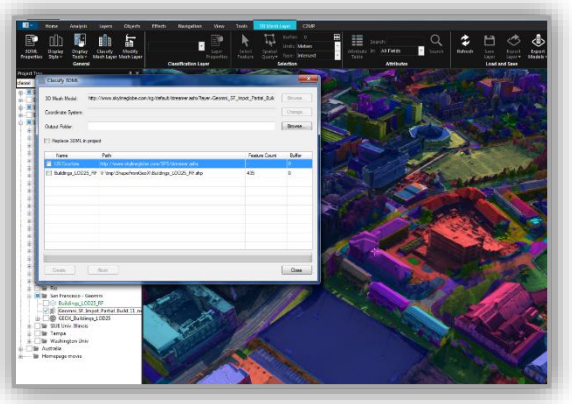


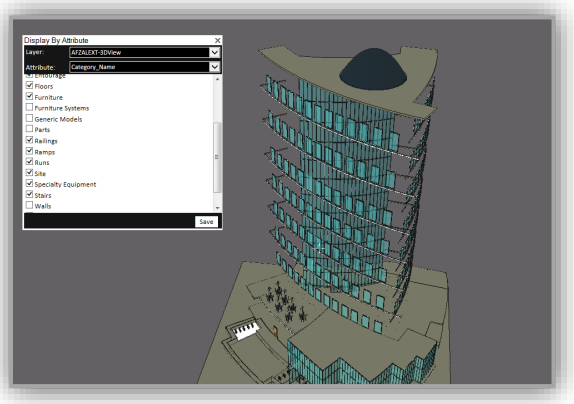
3DML Advantages

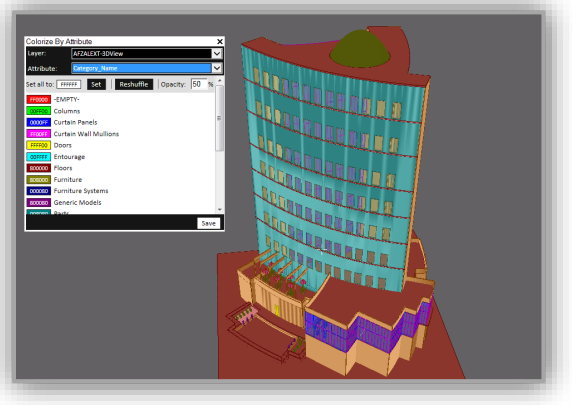
Classification of the 3DML

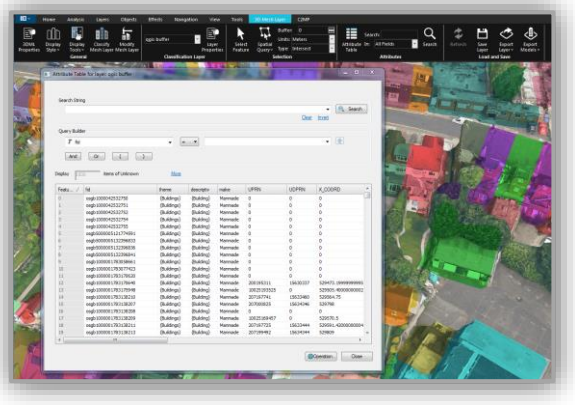
The 3DML format stores mesh classification information that transforms mesh into powerful geospatial data. Classified 3DML support access to attribute data and a range of feature layer operations, including:

- Spatial and attribute queries.
- Fast data access.
- Colorization of the mesh model and display of tooltips based on attribute value.
- Display of selected areas of the mesh model matching a specific attribute value.
- Creation of new layers based on spatial and attribute queries.









Ground Surface

The 3DML file format provides separate access to the mesh layer's ground information. This enables the mesh layer to replace the terrain imagery and elevation in the area of the 3DML so that on-terrain objects, such as road polylines, can be drawn directly on the 3DML ground surface, with an optional mesh ground offset.

Compact Storage

The 3DML format is based on an SQLite database, with optional SpatiaLite extension, to store very large 3D geospatial datasets that can be used in mobile, web and desktop clients. The 3DML is a single binary file that stores all information on the 3DML dataset including bounding box, coordinate system, blocks of mesh models, and classification polygons all in a single file that can be easily copied and managed.

Stream Optimization

The mesh is stored in multiple Levels of Detail (LOD) file database. This enables the initial loading of only low-resolution levels, while higher-resolution levels are dynamically loaded when zooming in on an area, allowing fast data loading and viewing. It also allows fast retrieval of non-sequential data from the file.

Tables

All 3DML's include the following general tables:

- ThreeDML_Version
- ThreeDML_Projects

And for each Mesh Layer, the following tables:

- ThreeDML_Table1_Mesh
- ThreeDML_Table1_Index

General Tables

ThreeDML_Version

The current version of the 3dml 1.0.

SQL:

```
CREATE TABLE IF NOT EXISTS `ThreeDML_Version` (
        `Major` INTEGER NOT NULL,
        `Minor` INTEGER NOT NULL
);
INSERT INTO `ThreeDML_Version` VALUES (1,0);
```

	Type	**Description**
Major	INTEGER	Major version
Minor	INTEGER	Minor version

ThreeDML_Projects

This table stores general information on the 3DML dataset, including:

- Bounding box
- Unique GUID
- Coordinate system of the dataset in well-known-text format
- Version
- Mini-tree root node

In general 3DML files can store more than one 3D Mesh Layers, but in practice we will use only one. This means that the Projects table will hold only one project (one line).

This project have an id of 1 and all of its data tables will be stored with tables names using the prefix ThreeDML_Table1 (ThreeDML_TABLE1_Mesh, ThreeDML_TABLE1_Index, ThreeDML_TABLE1_Meshindex).

SQL:

```
CREATE TABLE IF NOT EXISTS `ThreeDML_Projects` (
	`Id`	INTEGER NOT NULL,
	`Name`	TEXT NOT NULL,
	`Prefix`TEXT NOT NULL,
	`MinX` REAL NOT NULL,
	`MaxX`REAL NOT NULL,
	`MinY` REAL NOT NULL,
	`MaxY`REAL NOT NULL,
	`MinZ` REAL NOT NULL,
	`MaxZ`REAL NOT NULL,
	`BestResolution`	REAL NOT NULL,
	`WorstResolution`	REAL NOT NULL,
	`Projection`	TEXT NOT NULL,
	`Version`	INTEGER NOT NULL,
	`GUID`guid NOT NULL,
	`Tags` TEXT NOT NULL,
	`Description`	TEXT NOT NULL,
	`StartMeshIndexId`	INTEGER NOT NULL,
	`State` INTEGER NOT NULL,
	`CheckSum`	TEXT NOT NULL,
	`EngineVersion`	TEXT NOT NULL,
	`WKTFull`	TEXT,
	PRIMARY KEY(`Id`)
);
```

	Type	**Description**
Id	INTEGER	ID of the mesh layer.
Name	TEXT	Name of the mesh layer.
Prefix	TEXT	Prefix of tables storing the mesh information. Use “ThreeDML_Table1”.
MinX	REAL	Minimum X coordinate of stored data, in “Projection” coordinate system.
MaxX	REAL	Maximum X coordinate of stored data, in “Projection” coordinate system.
MinY	REAL	Minimum Y coordinate of stored data, in “Projection” coordinate system.
MaxY	REAL	Maximum X coordinate of stored data, in “Projection” coordinate system.

MinZ	REAL	Minimum Z (Altitude) coordinate of stored data, in “Projection” coordinate system.
MaxZ	REAL	Maximum Z (Altitude) coordinate of stored data, in “Projection” coordinate system.
BestResolution	REAL	Best mesh resolution, in units/pixel, used as metadata. E.g., 0.02.
WorstResolution	REAL	Worst mesh resolution, in units/pixel, used as to calculate the distance from which to start loading the 3DML into the 3D view. E.g., 4.
Projection	TEXT	Simple representation of the coordinate system of the dataset in well-known-text format. Without COMPD_CS[].
Version	TEXT	Use 2 for Version 1.0 of this specification.
GUID	guid	Unique ID of the mesh. Should be changed when 3DML is updated. E.g., 12345678-1234-1234-1234-123456789abc.
Tags	TEXT	Optional, comma separated metadata tags. Leave empty.
Description	TEXT	Optional, description of the mesh layer. Leave empty.
StartMeshIndexId	INTEGER	The index of the root node in the [Prefix]_MeshIndex table (E.g., ThreeDML_Table1_MeshIndex).
State	INTEGER	Use 65535 for Version 1.0 of this specification.
CheckSum	TEXT	Optional, can leave empty.
EngineVersion	TEXT	Use 7.0.0.1836 for Version 1.0 of this specification
WKTFulll	TEXT	Optional, complete representation of the coordinate system of the dataset in well-known-text format., including vertical datum defined using COMPD_CS[], and VERT_CS[]. If

		vertical datum is not provided, should be the same as “Projection”.

Mesh Layer Tables

ThreeDML_Table1_Mesh

This table stores mesh nodes.

Each node in this table can contain either:

3D-Mesh (geometry, textures, FLID) – See Mesh SLBLOB

3D-Floor (geometry) - not in the scope of this specification.

The relations of the nodes (parent-child) is store in a Mini-tree structure as described in ThreeDML_Table1_MeshInder and Mini-tree SLBLOB . The Mini-tree is a small tree structure that points to the nodes using ID (Node ID) or point to a other tree using ID (Tree ID).

SQL:

```
CREATE TABLE IF NOT EXISTS `ThreeDML_Table1_Mesh` (
        `Id`    INTEGER NOT NULL,
        `Data`  Blob NOT NULL,
        PRIMARY KEY(`Id`)
);
```

	Type	**Description**
Id	INTEGER	Id of the node.
Data	Blob	Node’s binary SLBLOB blob data***

ThreeDML_Table1_MeshIndex

This table stores the Mini-tree structures, describing the hierarchical relation of the nodes. The Mini-tree stores references to child node ID’s or other Mini-tree ID’s, to describe the full hierarchy of all nodes in the mesh layer.

SQL:

```
CREATE TABLE IF NOT EXISTS `ThreeDML_Table1_MeshIndex` (
        `Id`    INTEGER NOT NULL,
        `Data`  Blob NOT NULL,
        PRIMARY KEY(`Id`)
```

);

	Type	Description
Id	INTEGER	Id of the mini-tree.
Data	Blob	Mini-tree binary SLBLOB blob data

ThreeDML_Table1_Index

This table stores the metadata information for the mesh layer, describing each node.

SQL:

```
CREATE TABLE IF NOT EXISTS `ThreeDML_Table1_Index` (
        `Id`    INTEGER NOT NULL,
        `TreeId`        INTEGER NOT NULL,
        `Radius`        REAL NOT NULL,
        `Resolution`    REAL NOT NULL,
        `CenterX`       REAL NOT NULL,
        `CenterY`       REAL NOT NULL,
        `CenterZ`       REAL NOT NULL,
        `RadiusMinX`    REAL NOT NULL,
        `RadiusMaxX`    REAL NOT NULL,
        `RadiusMinY`    REAL NOT NULL,
        `RadiusMaxY`    REAL NOT NULL,
        `RadiusMinZ`    REAL NOT NULL,
        `RadiusMaxZ`    REAL NOT NULL,
        `MinX`  REAL NOT NULL,
        `MaxX`REAL NOT NULL,
        `MinY`  REAL NOT NULL,
        `MaxY`REAL NOT NULL,
        `MinZ`  REAL NOT NULL,
        `MaxZ`REAL NOT NULL,
        `MinRange`      REAL NOT NULL,
        `MeshSize`      INTEGER NOT NULL,
        `ParentId`      INTEGER NOT NULL,
        `MeshId`        INTEGER NOT NULL,
        `Source`        INTEGER NOT NULL,
        `PredictableLayerUID`           TEXT NOT NULL,
        `PredictableItemUID`    TEXT NOT NULL,
        PRIMARY KEY(`Id`)
);
```

	Type	Description
Id	INTEGER	Id of the mesh node.

TreeId	TreeId	Id of the mini-tree pointing to the mesh node.
Radius	REAL	The radius of the node, in meters/CS units.
Resolution	REAL	The resolution of the mesh node, In meters.
CenterX	REAL	Center X coordinate of mesh node, in mesh layer's coordinate system.
CenterY	REAL	Center Y coordinate of mesh node, in mesh layer's coordinate system.
CenterZ	REAL	Center Z (altitude) coordinate of mesh node, in mesh layer's coordinate system.
RadiusMinX	REAL	Radius X minimum size of mesh node, in mesh layer's coordinate system.
RadiusMaxX	REAL	Radius X Maximum size of mesh node, in mesh layer's coordinate system.
RadiusMinY	REAL	Radius Y minimum size of mesh node, in mesh layer's coordinate system.
RadiusMaxY	REAL	Radius Y maximum size of mesh node, in mesh layer's coordinate system.
RadiusMinZ	REAL	Radius Z (Altitude) minimum size of mesh node, in mesh layer's coordinate system.
RadiusMaxZ	REAL	Radius Z (Altitude) maximum size of mesh node, in mesh layer's coordinate system.
MinX	REAL	Mesh node minimum X coordinate, in mesh layer's coordinate system.
MaxX	REAL	Mesh node maximum X coordinate, in mesh layer's coordinate system.
MinY	REAL	Mesh node minimum Y coordinate, in mesh layer's coordinate system.
Maxy	REAL	Mesh node maximum Y coordinate, in mesh layer's coordinate system.

MinZ	REAL	Mesh node minimum Z (Altitude) coordinate, in mesh layer's coordinate system.
MaxZ	REAL	Mesh node maximum Z (Altitude) coordinate, in mesh layer's coordinate system.
MinRange	REAL	Minimum range in meters from which to display the mesh node.
MeshSize	INTEGER	Mesh node size in KB.
ParentId	INTEGER	The id of the mesh node's parent.
MeshId	INTEGER	The id of the mesh node.
Source	INTEGER	Use 1 for Version 1.0 of this specification.
PredictableLayerUID	TEXT	Leave empty for Version 1.0 of this specification.
PredictableItemUID	TEXT	Leave empty for Version 1.0 of this specification.

SLBLOB data

SLBLOB data is written into the ThreeDML_Table1_Meshindex and the ThreeDML_Table1_Mesh tables using compressed stream format:

SLBLOB = [marker][xor_7zdata]

The [7zdata] is compressed using regular 7zip compressor (lzma16) include 1 or more files in a byte array _7zdata.

In case the data has more than 64 bytes a xor operation is applied to the byte array using this method:

```
std::vector<byte>  _7zdata

if (_7zdata.size()>64)
  {
    DWORD* Data;
    DWORD* Mask;
    for (int i=0;i<8;i++)
    {
      Data=(DWORD*)(_7zdata.data()+i*sizeof(DWORD));
      Mask=(DWORD*)(_7zdata.data()+(8+i)*sizeof(DWORD));
      *Data=*Data ^ *Mask;
    }
  }
```

```
[xor_7zdata]=xor([7zdata])
```

Mesh SLBLOB

The mesh SLBLOB is identified using this marker.

[marker]= dword(0x31637078) [0x78,0x70,0x63,0x31] 'xpc1'

Xpc1 mesh data format: Mesh Node SLBLOB.

Mini-tree SLBLOB

The mini-tree SLBLOB is identified using this marker.

[marker]= dword(0x30304449) [0x49,0x44,0x30,0x30] 'ID00'

Mini-tree data is binary data in the following format:

The "id" file is built in from [marker], [root_link] and list of [node]

The [marker] is 4 bytes

The [root_link] point to the first node in the file (this should be the offset in bytes of one of the nodes in the list)

Each node contains [node_data] – this is the node information and optional [son_link]

[son_link] is a link to another node in this file (node_link_struct) or another mini-tree in other record in the database (tree_link_struct)

Here are the different streams of each struct:

[marker] = dword(0x30304449) or unsigned char [0x49,0x44,0x30,0x30] -> 'ID00'

[root_link] word(0x0004),word(0x0004),dword(first_node_offset)

[node] word(0x0001),word(0x0001/0x0002) [node_data] + optinal ([son_link])

[node_data] word(0x0002),word(0x0034), data_struct

[son_link] word(0x0003),word(num), node_link_struct X num | tree _link_struct X num

data_struct

```
{
  double Center[3];     // position of mesh pivot
  float Radius;         // radius =
sqrt(SizeDiv2[0]*SizeDiv2[0]+SizeDiv2[1]*SizeDiv2[1]+SizeDiv2[2]*SizeDiv2[2])
  float SizeDiv2[3];    // half size bounding box
  float MinRange;       // size in pixels when this node should be use
  LONGLONG Fid;         // use -1 to have no mesh
}
node_link_struct
{
   LONGLONG Offset;        // Offset in bytes for the node
   DWORD    Type;          // 0x00000003 - Offset in file
}
tree_link_struct
{
   LONGLONG TID;           // Tree ID
   DWORD    Type;          // 0x00000001 - Tree ID
}
```

Each Mini-tree SLBLOB is push into the ThreeDML_Table1_Meshindex, and have its own ID. This id is used in the tree_link_struct as TID and link from one Mini-tree to another.

Mesh Node SLBLOB

3DMesh Node Format:

3DMesh Node is an SLBLOB data that includes few files:

“1.sdkmesh” (see SDKMESH description)

“1.dds”/jpeg/png (see Texture Formats)

“1.sdkmesh” is a binary file holding the geometry of the mesh – sdkmesh files can hold meshes with more than one material, and more than one texture. In most cases we use only one geometry and one texture.

“1.dds” is a binary file holding the texture of the mesh.

In case of more than one texture, texture names will be 2…..n.dds/jpg/png

Splitting mesh into features is not in this scope (FLID)

Texture Format:

dds - DXT1 compression texture format. (No alpha)

DXT3 compression texture format. (With alpha)

In most cases 3dml includes dds files, you can also use png or jpg but the desktop client will convert them into dxt1/dxt3 in real time, while a browser client will use it as is.

Serving 3dml on a server optimization (TBD)

SDKMESH

TBD