Lib3MF

Open Source Toolkit for the 3D Manufacturing Format



|  |  |
| --- | --- |
| **Version** | 1.8.1 |
| **Status** | Released |

THESE MATERIALS ARE PROVIDED “AS IS.” The contributors expressly disclaim any warranties (express, implied, or otherwise), including implied warranties of merchantability, non-infringement, fitness for a particular purpose, or title, related to the materials. The entire risk as to implementing or otherwise using the materials is assumed by the implementer and user. IN NO EVENT WILL ANY MEMBER BE LIABLE TO ANY OTHER PARTY FOR LOST PROFITS OR ANY FORM OF INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES OF ANY CHARACTER FROM ANY CAUSES OF ACTION OF ANY KIND WITH RESPECT TO THIS DELIVERABLE OR ITS GOVERNING AGREEMENT, WHETHER BASED ON BREACH OF CONTRACT, TORT (INCLUDING NEGLIGENCE), OR OTHERWISE, AND WHETHER OR NOT THE OTHER MEMBER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Contents

[About this Library 2](#_Toc512329848)

[General Architecture 3](#_Toc512329849)

[Versioning 4](#_Toc512329850)

[Examples 5](#_Toc512329851)

[COM Interfaces 6](#_Toc512329852)

[1.1. Plain C Interfaces 7](#_Toc512329853)

[1.2. Class Reference – Core Specification 8](#_Toc512329854)

[1.3. Class Reference – BeamLattice specification 26](#_Toc512329855)

[1.4. Class Reference – Slice-specification 29](#_Toc512329856)

[License 31](#_Toc512329857)

[Contributors 32](#_Toc512329858)

[Open API Points and Roadmap 32](#_Toc512329859)

## About this Library

Lib3MF is a C++ implementation of the 3D Manufacturing Format file standard.

As 3MF shall become an universal 3D Printing standard, its quick adoption is very important. This library shall lower all barriers of adoption to any possible user, let it be software providers, hardware providers, service providers or middleware tools.

Its aim is to offer an open source way to integrate 3MF reading and writing capabilities, as well as conversion and validation tools for input and output data. The 3MF Library shall provide a clean and easy-to-use API to speed up the development and keep integration costs at a minimum.

While the current code is primarily made for a Microsoft Visual Studio Environment, a lot of energy has been put into keeping it as platform independent as far as possible. For example, it compiles well with the GCC compiler, but there is some work left to recode a few platform specific functionalities, which are now covered by the WinRT platform (like XML parsing and ZIP compression). As described below, we are looking for contributors with extensive experience in this field.

To understand this documentation in full extent, it is important to have taken a look at the 3MF specification 1.\*, available for free at <https://github.com/3MFConsortium/spec_core>.

For the source code of the library code, please visit <https://github.com/3MFConsortium/lib3mf>.

Example code can be found at <https://github.com/3MFConsortium/lib3mf-examples>.

For any issues, hints, feedback or contributions, please use the issues tab on <https://github.com/3MFConsortium/lib3mf/issues>.

## General Architecture

The 3MF library API compiles into two different flavours, which are basically just bindings to the same class model:

1. A COM-like DLL Interface, which exposes the same API as the COM Interface, but comes without the need for global CLSID registration. This allows to link the library natively to many unmanaged object-oriented languages (e.g. C++ or Delphi)
2. A plain C Interface, which wraps the class structure of the API into pseudo-object-oriented. This works very well for C programs and other low level languages, as well as other operating systems.

In the first releases, the DLLs are only compiling in Microsoft Windows. Porting it to other platforms is planned. More information can be found in the corresponding headers in

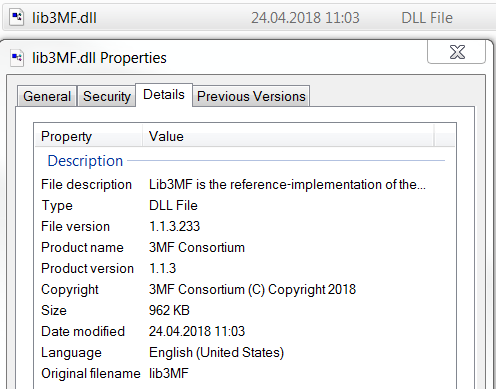
1. Include/Model/COM/NMR\_COMInterfaces.h
2. Include/Model/COM/NMR\_COMFactory.h
3. Include/Model/COM/NMR\_DLLInterfaces.h

## Versioning

The lib3MF library contains an internal version number, which should be checked by the consuming program when loading it (see Example applications).

Lib3MF follows a common, hierarchical schema for versioning:

lib3MF.major.minor.mirco.buildnumber

To identify lib3MF’s version under windows, look for “File version” in DLL’s details.  


Both examples are for version 1.1.3, build 233 of lib3MF.

Lib3MF’s API is stable in the following way:

* Lib3MF-libraries with different *major version* are not guaranteed to be compatible.
* Lib3MF-libraries with matching *major version* and different *minor versions* are backwards compatible:  
  A consumer who compiled against the API of version A.B can safely use the binary of version A.C for C>=B.  
  I.e., functions will not be removed within the range of a constant *major version*.
* Lib3MF-libraries with matching values of both *major* and *minor version* have an identical API.

Under Unix use a tool like objdumd to find out the so-name:  
This corresponds to the *major version* of the lib3MF. To find out the *minor* or *micro* *version*, use the GetInterfaceVersion-function of the library described below.

## Examples

Currently, there are several examples which show how to use the library:

* *Cube*: a simple example how to create an empty 3MF document and add custom geometry to it.
* *Converter*: a simple program to convert 3MFs into (binary) STLs and back.
* *Components*: explains component handling in 3MF
* *ExtractInfo*: shows how to import a 3MF and navigate through the in memory representation of the model.
* *TextureCube*: demonstrates the handling of texures in 3MF and shows the usage of the progress callback of lib3MF.

Please note, that you might need a proper understanding of the 3MF Specification in order to get the most out of the example code.

## COM Interfaces

For the 3MF Core spec, the following interfaces specify an in memory representation of the 3MF Document. For a detailed description, please refer to Include/Model/COM/NMR\_COMInterfaces.h.

|  |  |  |
| --- | --- | --- |
| **Interface** | **derived from** | **Description** |
| ILib3MFBase | IUnknown | ILib3MFBase is a base interface, which serves as parent for all interfaces related to the 3MF Library |
| ILib3MFModelWriter | ILib3MFBase | ILib3MFModelWriter encapsulates a writer class for writing the model into a specific file type. |
| ILib3MFModelReader | ILib3MFBase | ILib3MFModelReader encapsulates a reader class for reading a model from a specific file type. |
| ILib3MFModelResource | ILib3MFBase | ILib3MFModelResource is a base interface for all 3MF Resources. |
| ILib3MFModelResourceIterator | ILib3MFBase | ILib3MFModelResourceIterator is a helper class to iterate through arbitrary lists of 3MF resources |
| ILib3MFPropertyHandler | ILib3MFBase | ILib3MFPropertyHandler encapsulates all methods for handling 3MF mesh properties. |
| ILib3MFDefaultPropertyHandler | ILib3MFBase | ILib3MFDefaultPropertyHandler encapsulates all methods for handling 3MF default object properties. |
| ILib3MFModelBaseMaterial | ILib3MFModelResource | ILib3MFModelBaseMaterial implements the Base Material Group Resources of a 3MF model stream, and allows direct access to the base material information |
| ILib3MFModelAttachment | ILib3MFBase | ILib3MFModelAttachment implements the Model Attachments of a 3MF model stream, and allows direct access to direct binary data. |
| ILib3MFModelTexture2D | ILib3MFModelResource | ILib3MFModelTexture2D implements the Texture2D Resources of a 3MF model stream, and allows direct access to the texture properties and the image data. |
| ILib3MFModelMeshObject | ILib3MFModelObjectResource | ILib3MFModelMeshObject encapsulates all methods for handling 3MF mesh objects. |
| ILib3MFModelComponent | ILib3MFBase | ILib3MFModelComponent encapsulates one component node of a 3MF component object. |
| ILib3MFModelComponentsObject | ILib3MFModelObjectResource | ILib3MFModelComponentsObject encapsulates all methods for handling 3MF component objects. |
| ILib3MFModelBuildItem | ILib3MFBase | ILib3MFModelBuildItem encapsulates all methods for handling 3MF build items. |
| ILib3MFModelBuildItemIterator | ILib3MFBase | ILib3MFModelBuildItemIterator is a helper class to iterate through arbitrary lists of 3MF build items. |
| ILib3MFModel | ILib3MFBase | ILib3MFModel is the basic instance owning all In-Memory elements of a 3MF file. |
| ILib3MFModelFactory | ILib3MFBase | ILib3MFModelFactory is a factory interface for ILib3MFModel |
| ILib3MFModelMeshBeamSet | ILib3MFBase | ILib3MFModelMeshBeamSet is a class that holds the references that contain to a beamset. It is part of the  beamlattice extension to 3MF. |
| ILib3MFSlice | ILib3MFBase | ILib3MFSlice encapsulates all slice functionality for handling slices in 3mf |
| ILib3MFSliceStack | ILib3MFBase | ILib3MFSliceStack encapsulates all methods for handling slice stacks in 3MF |

## Plain C Interfaces

In order to address a wider user base, the object-oriented interfaces above are also compiled into a DLL for emulating pseudo objects in a procedural language. There are very valid use cases for it.

Please note, that this wrapper is a lot less type-safe than the COM interface.

**Example**: The following code in COM

pModel->QueryWriter („3mf“, &pModelWriter);

will be translated into

lib3mf\_model\_querywriter (pModel, „3mf“, &pModelWriter);

Moreover, the following type/interface in COM

LIB3MFINTERFACE(ILib3MFModel, ILib3MFBase, CLSID\_Lib3MF\_Model)

is translated to

typedef PLib3MFBase PLib3MFModel;

All these translations follow the same pattern. For more information, please compare the corresponding header files

1. Include/Model/COM/NMR\_DLLInterfaces.h
2. Include/Model/COM/NMR\_COMInterfaces.h

## Class Reference – Core Specification

The following list shall give a short class overview of the library. The interfaces are all defined centrally in one header file. Please read Include/Model/COM/NMR\_COMInterfaces.h for all details.

All Methods return a HRESULT defining the success of the operation. A successful operation always returns 0 (S\_OK), and a windows error code otherwise. If no success is returned, the output parameters might be in an undefined state.

1. **ILib3MFBase**

ILib3MFBase is a base interface, which serves as parent for all interfaces related to the 3MF Library.

Parent interface: *IUnknown*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| GetLastError | DWORD\* **pErrorCode**: returns error code  LPCSTR **pErrorMessage**: Returns pointer to the error message string, NULL if no error. | Returns detailed information of the last known error an object method. The error information is available for every method returning a LIB3MF\_FAILED constant. |

1. **ILib3MFModelWriter**

ILib3MFModelWriter encapsulates a writer class for a writing the model into a specific file type. Current implementations include (binary) STL and 3MF.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| WriteToFile | LPCWSTR **pwszFilename**: Filename to write into (UTF16 encoded) | Writes out the model as file. The file type is specified by the Model Writer class |
| WriteToFileUTF8 | LPCSTR **pwszFilename**: Filename to write into (UTF8 encoded) | Writes out the model as file. The file type is specified by the Model Writer class |
| WriteToStream | IStream\* **pStream***:* IStream to write into | Writes out the model into a COM IStream. Only available on Windows. |
| WriteToCallback | void\* **pWriteCallback**: Callback to call for writing a data chunk.  void\* **pSeekCallback**: Callback for seeking in the write data stream.  void\* **pUserData**: Userdata that is passed to the callback function | Writes out the model and passes the data to a provided callback function. The file type is specified by the Model Writer class |
| SetProgressCallback | void\* **callback**: pointer to the callback function. If the callback returns "false" the original function call will be aborted and set the error NMR\_USERABORTED.  void\* **userData**: pointer to arbitrary user data that is passed without modification to the callback. | Set the progress callback for calls to this writer |

1. **ILib3MFModelReader**

ILib3MFModelReader encapsulates a reader class for reading a model from a specific file type. Current implementations include (binary) STL and 3MF.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| ReadFromFile | LPCWSTR **pwszFilename***:* Filename to read from | Reads a model from a file. The file type is specified by the Model Reader class. |
| ReadFromFileUTF8 | LPCSTR **pszFilename***:* Filename to read from as UTF8 string | Reads a model from a file. The file type is specified by the Model Reader class. |
| ReadFromStream | IStream\* **pStream***:* IStream to read from | Reads a model from a COM IStream. Only available on Windows. |
| GetWarningCount | DWORD\* **pnWarningCount***:* filled with the count of the occurred warnings. | Returns warning and error count of the read process. |
| GetWarning | DWORD **nIndex***:* Index of the Warning. Valid values are 0 to WarningCount – 1.  DWORD\* **pErrorCode***:* filled with the error code of the warning  LPWSTR **pwszBuffer***:* filled with the error message, may be NULL  DWORD **cbBufferSize*:***size of pwszBuffer (including trailing 0).  DWORD\* **pcbNeededChars***:* filled with the count of the written bytes, or needed buffer size. | Returns warning and error information of the read process |
| GetWarningUTF8 | DWORD **nIndex***:* Index of the Warning. Valid values are 0 to WarningCount – 1.  DWORD\* **pErrorCode***:* filled with the error code of the warning  LPSTR **pszBuffer***:* filled with the error message, may be NULL  DWORD **cbBufferSize*:***size of pwszBuffer (including trailing 0).  DWORD\* **pcbNeededChars***:* filled with the count of the written bytes, or needed buffer size. | Returns warning and error information of the read process (UTF8) |
| SetStrictModeActive | BOOL **bStrictModeActive***:* flag whether strict mode is active or not. | Activates (deactivates) the strict mode of the reader. If active, all warnings are reported as errors. Otherwise, they are reported as warnings. By default, it is deactivated. |
| GetStrictModeActive | BOOL\* **pbStrictModeActive***:* flag whether strict mode is active or not. | Queries the whether the strict mode of the reader is active or not |
| SetProgressCallback | void\* **callback**: pointer to the callback function. If the callback returns "false" the original function call will be aborted and set the error NMR\_USERABORTED.  void\* **userData**: pointer to arbitrary user data that is passed without modification to the callback. | Set the progress callback for calls to this reader |

1. **ILib3MFModelResource**

ILib3MFModelResource is a base interface for all 3MF Resources.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| GetResourceID | DWORD\* **pnResourceID**: Filled with the ID of the Resource Instance | Retrieves the ID of a Model Resource Instance |

1. **ILib3MFModelResourceIterator**

ILib3MFModelResourceIterator is a helper class to iterate through arbitrary lists of 3MF resources.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| MoveNext | BOOL\* **pbHasNext**: returns, if there is a resource to use | Iterates to the next resource in the list. |
| MovePrevious | BOOL\* **pbHasPrevious**: returns, if there is a resource to use | Iterates to the previous resource in the list. |
| GetCurrent | ILib3MFModelResource\*\* **ppResourceInstance**: returns the resource instance | Returns the resource the iterator points at. |
| Clone | ILib3MFModelResourceIterator\*\* **ppIterator**: returns the cloned Iterator instance | Creates a new resource iterator with the same resource list. |

1. **ILib3MFPropertyHandler**

ILib3MFPropertyHandler encapsulates all methods for handling 3MF mesh properties.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| RemoveProperty | DWORD **nIndex**: Index of the triangle (0-based) | Removes all properties of a specific triangle. |
| RemoveAllProperties | - | Removes all properties of the triangle mesh. |
| GetPropertyType | DWORD **nIndex**: Index of the triangle (0-based)  eModelPropertyType\* **pnPropertyType**: Returns the property type of the triangle | Returns the property type of the specific triangle |
| GetBaseMaterial | DWORD **nIndex**: Index of the triangle (0-based)  ModelResourceID\* **pnMaterialGroupID**: returns the material group id, per triangle. A return group id of 0 means either no property at all or a non-material property.  ModelResourceIndex\* **pnMaterialIndex**: returns the material index, per triangle. Returns 0, if no base material is assigned. | Returns the base material of a specific triangle. |
| GetBaseMaterialArray | ModelResourceID\* **pnMaterialGroupIDs**: will be filled with the material group ids of the triangles. Array must have trianglecount entries. A return group id of 0 means either no property at all or a non-material property.  ModelResourceIndex\* **pnMaterialIndices**: will be filled with the material group indices of the triangles. Array must have trianglecount entries. | Returns the base materials of all triangles. If a triangle property is not a material, the returned material group ID will be 0. |
| SetBaseMaterial | DWORD **nIndex**: Index of the triangle (0-based)  ModelResourceID **nMaterialGroupID**: Group ID of the Material Group  ModelResourceIndex **nMaterialIndex**: Index of the Material in the Group | Sets the material of a triangle to a specific single value. All other Triangle properties are removed. This must be a base material . |
| SetBaseMaterialArray | ModelResourceID\* **pnMaterialGroupIDs**: array of the material Group IDs. Must have trianglecount entries. If a group ID of 0 is specified.  ModelResourceIndex\* **pnMaterialIndices**:  array of the corresponding material indices. Must have trianglecount entries. | Sets the materials of all triangles to specific values. |
| GetColor | DWORD **nIndex**: Index of the triangle (0-based)  MODELMESH\_TRIANGLECOLOR\_SRGB\* **pColor**:  returns the color values of the three nodes of the triangle. (#00000000) means no property or a different kind of property! | Returns the color of a specific triangle. |
| GetColorArray | MODELMESH\_TRIANGLECOLOR\_SRGB\* **pColors**:  returns the color values of the three nodes of each triangle. Must have at least trianglecount array entries. | Returns the color array of all triangles |
| SetSingleColor | DWORD **nIndex**: Index of the triangle (0-based)  MODELMESHCOLOR\_SRGB\* **pColor**: new color value of the triangle. (#00000000) means no color property. | Sets the specific triangle to a single color. All other properties are removed. Mixing properties needs the property extension API. |
| SetSingleColorRGB | DWORD **nIndex**: Index of the triangle  BYTE **bRed**: Red component of the color value  BYTE **bGreen**: Green component of the color value BYTE **bBlue**: Blue component of the color value | Sets the specific triangle to a single color. All other properties are removed. Mixing properties needs the property extension API. Value range is from 0 to 255. Alpha will be 255. |
| SetSingleColorRGBA | DWORD **nIndex**: Index of the triangle (0-based)  BYTE **bRed**: Red component of the color value  BYTE **bGreen**: Green component of the color value  BYTE **bBlue**: Blue component of the color value  BYTE **bAlpha**: Alpha component of the color value | Sets the specific triangle to a single color. All other properties are removed. Mixing properties needs the property extension API. #00000000 means no color. Value range is from 0 to 255. |
| SetSingleColorFloatRGB | DWORD **nIndex**: Index of the triangle (0-based)  FLOAT **fRed**: Red component of the color value  FLOAT **fGreen**: Green component of the color value  FLOAT **fBlue**: Blue component of the color value | Sets the specific triangle to a single color. All other properties are removed. Mixing properties needs the property extension API. Value range is from 0.0 to 1.0. Alpha value will be set to 1.0 |
| SetSingleColorFloatRGBA | DWORD **nIndex**: Index of the triangle (0-based)  FLOAT **fRed**: Red component of the color value  FLOAT **fGreen**: Green component of the color value  FLOAT **fBlue**: Blue component of the color value  FLOAT **fAlpha**: Alpha component of the color value | Sets the specific triangle to a single color. All other properties are removed. Mixing properties needs the property extension API. #00000000 means no color. Value range is from 0.0 to 1.0 |
| SetSingleColorArray | MODELMESHCOLOR\_SRGB\* **pColors**: new color values for the triangles. (#00000000) means no color property.. Must have at least trianglecount array entries. | Sets the (single) color of all triangles. All other properties are removed. |
| SetGradientColor | DWORD **nIndex**: Index of the triangle (0-based)  MODELMESH\_TRIANGLECOLOR\_SRGB\* **pColor**:  new color values of the three nodes of the triangle. (#00000000) means no color property is set. | Sets the specific triangle to a color per vertex. All other properties are removed. |
| SetGradientColorArray | MODELMESH\_TRIANGLECOLOR\_SRGB\* **pColors**:  pColors returns the color values of the three nodes of each triangle. Must have at least trianglecount array entries. (#00000000) means no color property is set. | Sets the (gradient) color of all triangles. All other properties are removed. |
| GetTexture | DWORD **nIndex**: Index of the triangle (0-based)  MODELMESHTEXTURE2D\* **pTexture**: returns the UV texture values of the three nodes of the triangle. texture ID 0 means no property or a different kind of property. | Returns the 2D texture information of a specific triangle. |
| GetTextureArray | MODELMESHTEXTURE2D\* **pTextures**: returns the UV texture values of the three nodes of all triangles. Must have at least trianglecount array entries. | Returns the 2D texture information of all triangles. |
| SetTexture | DWORD **nIndex**: Index of the triangle (0-based)  MODELMESHTEXTURE2D\* **pTexture**: new UV texture values of the three nodes of the triangle. texture ID 0 means no property or a different kind of property. | Sets the 2D texture information of a specific triangle. |
| SetTextureArray | MODELMESHTEXTURE2D\* **pTexture**: new UV texture values of the three nodes of all triangles. Must have at least trianglecount array entries. | Sets the 2D texture information of all triangles. |

1. **ILib3MFDefaultPropertyHandler**

ILib3MFDefaultPropertyHandler encapsulates all methods for handling 3MF default object properties.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| RemoveProperty | - | Removes the default property of the object. |
| GetPropertyType | eModelPropertyType\* **pnPropertyType**: Returns the property type of the triangle | Returns the default property type of the object |
| GetBaseMaterial | ModelResourceID\* **pnMaterialGroupID**: returns the material group id, per triangle. A return group id of 0 means either no property at all or a non-material property.  ModelResourceIndex\* **pnMaterialIndex**: returns the material index, per triangle. Returns 0, if no base material is assigned. | Returns the base material the object |
| SetBaseMaterial | ModelResourceID **nMaterialGroupID**: Group ID of the Material Group  ModelResourceIndex **nMaterialIndex**: Index of the Material in the Group | Sets the material of an object to a specific single value. This must be a base material. |
| GetColor | MODELMESH\_TRIANGLECOLOR\_SRGB\* **pColor**:  returns the color values of the three nodes of the triangle. (#00000000) means no property or a different kind of property! | Returns the default property color of an object. |
| SetColor | MODELMESHCOLOR\_SRGB\* **pColor**: new color value of the triangle. (#00000000) means no color property. | Sets the default property of an object to a single color. Mixing properties needs the property extension API. |
| SetColorRGB | BYTE **bRed**: Red component of the color value  BYTE **bGreen**: Green component of the color value BYTE **bBlue**: Blue component of the color value | Sets the default property of an object to a single color. |
| SetColorRGBA | BYTE **bRed**: Red component of the color value  BYTE **bGreen**: Green component of the color value  BYTE **bBlue**: Blue component of the color value  BYTE **bAlpha**: Alpha component of the color value | ets the default property of an object to a single color. |
| SetFloatColorRGB | FLOAT **fRed**: Red component of the color value  FLOAT **fGreen**: Green component of the color value  FLOAT **fBlue**: Blue component of the color value | Sets the default property of an object to a single color. |
| SetFloatColorRGBA | FLOAT **fRed**: Red component of the color value  FLOAT **fGreen**: Green component of the color value  FLOAT **fBlue**: Blue component of the color value  FLOAT f**Alpha**: Alpha component of the color value | Sets the default property of an object to a single color. |
| SetGradientColor | DWORD **nIndex**: Index of the triangle (0-based)  MODELMESH\_TRIANGLECOLOR\_SRGB\* **pColor**:  new color values of the three nodes of the triangle. (#00000000) means no color property is set. | Sets the specific triangle to a color per vertex. All other properties are removed. |
| GetTexture | MODELMESHTEXTURE2D\* **pTexture**: returns the UV texture values of the three nodes of the triangle. texture ID 0 means no property or a different kind of property.  FLOAT \* **pfU**: Returns the default U value of the object.  FLOAT \* **pfV**: Returns the default V value of the object. | Returns the default 2D texture information of an object. |
| SetTexture | MODELMESHTEXTURE2D\* **pTexture**: new UV texture values of the three nodes of the triangle. texture ID 0 means no property or a different kind of property. FLOAT **fU**: Sets the default U value of the object.  FLOAT **fV**: Sets the default V value of the object. | Sets the default 2D texture information of an object. |

1. **ILib3MFModelObjectResource**

ILib3MFModelObjectResource is a base interface for all 3MF Object Resources, i.e. Mesh Objects and Component Objects.

Parent interface: *ILib3MFModelResource*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| GetType | DWORD\* **pObjectType**: returns object type constant. See ModelTypes.h for more information. | Retrieves an object's type. |
| SetType | DWORD **ObjectType*:*** object type constant. See ModelTypes.h for more information | Sets an object's type. |
| GetName | LPWSTR **pwszBuffer*:*** buffer to fill  ULONG **cbBufferSize*:*** size of buffer to fill. needs to be at least string length + 1  ULONG\* **pcbNeededChars*:*** returns needed characters in buffer | Retrieves an object's name string. |
| SetName | LPCWSTR **pwszName*:*** new name of the object. (e.g. "Car") | Sets an object's name string. |
| GetNameUTF8 | LPSTR **pszBuffer*:*** buffer to fill  ULONG **cbBufferSize*:*** size of buffer to fill. needs to be at least string length + 1  ULONG\* **pcbNeededChars*:*** returns needed characters in buffer | Retrieves an object's name string in UTF8. |
| SetNameUTF8 | LPCSTR **pszName*:*** new name of the object. (e.g. "Car") | Sets an object's name string in UTF8. |
| SetPartNumber | LPCWSTR **pwszPartNumber*:*** new part number string for referencing parts from an outside context. | Sets an object's part number string. |
| SetPartNumberUTF8 | LPCWSTR **pszPartNumber*:*** new part number string for referencing parts from an outside context. | Sets an object's part number string in UTF8 |
| GetPartNumber | LPWSTR **pwszBuffer*:*** buffer to fill  ULONG **cbBufferSize*:*** size of buffer to fill. needs to be at least string length + 1  ULONG\* **pcbNeededChars*:*** returns needed characters in buffer | Retrieves an object's oartnumber string. |
| GetPartNumberUTF8 | LPSTR **pszBuffer*:*** buffer to fill  ULONG **cbBufferSize*:*** size of buffer to fill. needs to be at least string length + 1  ULONG\* **pcbNeededChars*:*** returns needed characters in buffer | Retrieves an object's oartnumber string in UTF8. |
| IsMeshObject | BOOL\* **pbIsMeshObject**: returns, if the object is a mesh object | Retrieves, if an object is a mesh object |
| IsComponentsObject | BOOL\* **pbIsComponentObject**: returns, if the object is a components object | Retrieves, if an object is a component object |
| IsValidObject | BOOL\* **pbIsValid**: returns, if the object is a valid object description. | Retrieves, if the object is valid according to the core spec. For mesh objects, we distinguish between the type attribute of the object:   * In case of object type "other", this always means "false" * In case of object type "support", this always means "true" * In case of object type "model", this means, if the mesh suffices all requirements of the core spec chapter 4.1   A component objects is valid if and only if it contains at least one component - and all child components are valid objects. |
| CreateDefaultPropertyHandler | ILib3MFDefaultPropertyHandler \* **ppPropertyHandler**: returns a default property handler instance for the object | creates a default property handler for the object |
| CreateDefaultMultiPropertyHandler | ILib3MFDefaultMultiPropertyHandler \* **ppPropertyHandler**: returns a default property handler instance for the object | creates a default property handler for a specific multiproperty channel of an object |
| GetThumbnailPathUTF8 | LPSTR\* **pszBuffer**: buffer to fill  ULONG **cbBufferSize**: size of the buffer to fill. needs to be at least string length + 1  ULONG\* **pcbNeededChars**: returns needed characters in buffer | Retrieves the path used as thumbnail for an object (UTF8). Returns "" if none is set |
| SetThumbnailPathUTF8 | LPCSTR\* **pszName**: pszPath path where to look for the thumbnail (e.g. "/Textures/thumbnail.png"). Call will NULL to clear the thumbnail. | Sets an object's thumbnail package path (UTF8) |

1. **ILib3MFModelBaseMaterial**

ILib3MFModelBaseMaterial implements the Base Material Group Resources of a 3MF model stream, and allows direct access to the base material information.

Parent interface: *ILib3MFModelResource*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| GetCount | DWORD\* **pcbCount**: returns the count of base materials. | Retrieves the count of base materials in the material group. |
| AddMaterial | LPCWSTR **pwszName**: new name of the base material. (e.g. "ABS red")  BYTE **bRed**: New red value of display color (0-255)  BYTE **bGreen**: New red value of display color (0-255)  BYTE **bBlue**: New red value of display color (0-255)  DWORD\* **pnResourceIndex**: returns new Index of the material in the material group | Adds a new material to the material group |
| RemoveMaterial | DWORD **nIndex**: Index of the material in the material group | Removes a material from the material group |
| GetName | DWORD **nIndex**: Index of the material in the material group  LPWSTR **pwszBuffer**: buffer to fill  ULONG **cbBufferSize**: size of buffer to fill. needs to be at least string length + 1  ULONG\* **pcbNeededChars** returns needed characters in buffer | Retrieves a base material's name |
| SetName | DWORD **nIndex**: Index of the material in the material group  LPCWSTR **pwszName**: new name of the base material. (e.g. "ABS red") | Sets a base material's name |
| SetDisplayColorRGB | DWORD **nIndex**: Index of the material in the material group  BYTE **bRed**: New red value (0-255)  BYTE **bGreen**: New red value (0-255)  BYTE **bBlue**: New blue value (0-255) | Sets a base material's display color. Alpha is set to 255. |
| SetDisplayColorRGBA | DWORD **nIndex**: Index of the material in the material group  BYTE **bRed**: New red value (0-255)  BYTE **bGreen**: New red value (0-255)  BYTE **bBlue**: New blue value (0-255)  BYTE **bAlpha**: New alpha value (0-255) | Sets a base material's display color. |
| SetDisplayColorFloatRGB | DWORD **nIndex**: Index of the material in the material group  FLOAT **bRed**: New red value (0.0-1.0)  FLOAT **bGreen**: New red value (0.0-1.0)  FLOAT **bBlue**: New blue value (0.0-1.0) | Sets a base material's display color. Alpha is set to 1.0. |
| SetDisplayColorFloatRGBA | DWORD nIndex: Index of the material in the material group  FLOAT **bRed**: New red value (0.0-1.0)  FLOAT **bGreen**: New red value (0.0-1.0)  FLOAT **bBlue**: New blue value (0.0-1.0)  FLOAT **bAlpha**: New alpha value (0.0-1.0) | Sets a base material's display color. |
| GetDisplayColor | DWORD **nIndex**: Index of the material in the material group  BYTE\* **pbRed**: Returns red value (0-255)  BYTE\* **pbGreen**: Returns green value (0-255)  BYTE\* **pbBlue**: Returns blue value (0-255)  BYTE\* **pbAlpha**: Returns alpha value (0-255) | Returns a base material's display color. |

1. **ILib3MFModelAttachment**

ILib3MFModelAttachment implements the Model Attachments of a 3MF model stream, and allows direct access to direct binary data.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| GetPath | LPWSTR **pwszBuffer**: buffer to fill  ULONG **cbBufferSize**: size of buffer to fill. needs to be at least string length + 1.  ULONG\* **pcbNeededChars**: returns needed characters in buffer | Retrieves a attachment's package path |
| SetPath | LPCWSTR **pwszPath**: new path of the attachment. (e.g. "/Textures/logo.png") | Sets a attachment's package path |
| GetRelationshipType | LPWSTR **pwszBuffer**: buffer to fill  ULONG **cbBufferSize**: size of buffer to fill. needs to be at least string length + 1.  ULONG\* **pcbNeededChars**: returns needed characters in buffer | Retrieves a attachment's package relationship type |
| SetRelationshipType | LPCWSTR **pwszRelationShipType**: new relationship type attachment. (e.g. "/Data/data.xml") | Sets a attachment's package relationship type |
| GetStreamSize | ULONG64 **pcbStreamSize**: Returns the stream size | Retrieves the size of the attachment stream. |
| WriteToFile | LPCWSTR **pwszFilename**: Filename to write into | Writes out the attachment as file. |
| WriteToBuffer | BYTE **pBuffer**: Buffer to write into  ULONG64 **cbBufferSize**: Size of the buffer in bytes | Writes out the attachment into a buffer. Buffer size must be at least the size of the stream. |
| WriteToStream | IStream **pStream**: IStream to write into. | Writes out the attachment into a COM IStream. Only available on Windows. |
| WriteToCallback | VOID\* **pWriteCallback**: Callback to call for writing a data chunk.  VOID\* **pUserData**: Userdata that is passed to the callback function | Writes out the attachment and passes the data to a provided callback function. The file type is specified by the Model Writer class |
| ReadFromFile | LPCWSTR **pwszFilename**: Filename to read from | Reads a attachment from a file. |
| ReadFromBuffer | BYTE\* **pBuffer**: Buffer to read from  ULONG64 **cbBufferSize**: Size of the buffer in bytes | Reads a attachment from a memory buffer. |
| ReadFromStream | IStream **pStream**: IStream to read from | Reads a attachment from a COM IStream. Only available on Windows. |

1. **ILib3MFModelTexture2D**

ILib3MFModelTexture2D implements the Texture2D Resources of a 3MF model stream, and allows direct access to the texture properties and the image data.

Parent interface: *ILib3MFModelResource*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| GetPath | LPWSTR **pwszBuffer*:*** buffer to fill  ULONG **cbBufferSize*:*** size of buffer to fill. needs to be at least string length + 1  ULONG\* **pcbNeededChars*:*** returns needed characters in buffer | Retrieves a texture’s package path. |
| SetPath | ILib3MFModelAttachment **ppTextureAttachment**: new path of the texture. (e.g. "/Textures/logo.png") | Sets a texture's package path. |
| GetAttachment | LPCWSTR **pwszPath**: new | Retrieves the attachment located at the path of the texture |
| SetAttachment | ILib3MFModelAttachment\* **pTextureAttachment**: attachment that holds the texture's image information | Sets the texture's package path to the path of the attachment. |
| GetContentType | eModelTexture2DType\* **peContentType**: returns content type enum | Retrieves a texture's content type |
| SetContentType | eModelTexture2DType **eContentType**:  new Content Type | Sets a texture's content type |
| GetTileStyleUV | eModelTextureTileStyle\* **peTileStyleU**: returns tilestyle type enum  eModelTextureTileStyle\* **peTileStyleV**: returns tilestyle type enum | Retrieves a texture's tilestyle type |
| SetTileStyleUV | eModelTextureTileStyle **eTileStyleU**: new tilestyle type enum  eModelTextureTileStyle **eTileStyleV**: new tilestyle type enum | Sets a texture's tilestyle type |
| GetFilter | eModelTextureFilter\* **peFilter**: returns the filter type enum | Retrieves a texture's filter type |
| SetFilter | eModelTextureFilter **eFilter**: new filter type enum | Sets a texture's filter type |
| ClearBox2D | - | Clears a texture's box2D coordinates. |
| GetStreamSize | ULONG64\* **pcbStreamSize**: Returns the stream size | Retrieves the size of the texture stream. |
| WriteToFile | LPCWSTR **pwszFilename**: Filename to write into | Writes out the texture as file. |
| WriteToBuffer | BYTE\* **pBuffer**: Buffer to write into  ULONG64 **cbBufferSize**: Size of the buffer in bytes | Writes out the texture into a buffer. Buffer size must be at least the size of the stream. |
| WriteToStream | IStream\* **pStream**: IStream to write into. | Writes out the texture into a COM IStream. Only available on Windows. |
| WriteToCallback | void\* **pWriteCallback**: Callback pointer to call for writing a data chunk.  void\* **pUserData**: Userdata that is passed to the callback function | Writes out the texture and passes the data to a provided callback function. |
| ReadFromFile | LPCWSTR pwszFilename: Filename to read from | Reads a texture from a file. |
| ReadFromBuffer | BYTE \* **pBuffer**: Buffer to read from  ULONG64 **cbBufferSize**: Size of the buffer in bytes | Reads a texture from a memory buffer. |
| ReadFromStream | IStream \* **pStream**: IStream to read from | Reads a texture from a COM IStream. Only available on Windows. |

1. **ILib3MFModelMeshObject**

ILib3MFModelMeshObject encapsulates all methods for handling 3MF mesh objects.

Parent interface: *ILib3MFModelObjectResource*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| GetVertexCount | DWORD\***pnVertexCount**: filled with the vertex count | Returns the vertex count of a mesh object. |
| GetTriangleCount | DWORD\* **pnTriangleCount**: filled with the triangle count | Returns the triangle count of a mesh object. |
| GetVertex | DWORD **nIndex**: Index of the vertex (0 to vertexcount - 1)  MODELMESHVERTEX\* **pVertex**: filled with the vertex coordinates | Returns coordinates of a single vertex of a mesh object. |
| SetVertex | DWORD **nIndex**: Index of the vertex (0 to vertexcount - 1)  MODELMESHVERTEX\* **pVertex**: contains the vertex coordinates | Sets the coordinates of a single vertex of a mesh object. |
| AddVertex | MODELMESHVERTEX\* **pVertex**: contains the vertex coordinates  DWORD\* **pnIndex**: filled with the new Index of the vertex | Adds a single vertex to a mesh object. |
| GetTriangle | DWORD **nIndex**: Index of the triangle (0 to trianglecount - 1)  MODELMESHTRIANGLE\* **pTriangle**: filled with the triangle indices | Returns indices of a single triangle of a mesh object. |
| SetTriangle | DWORD **nIndex*:*** Index of the triangle (0 to trianglecount - 1)  MODELMESHTRIANGLE\* **pTriangle**: contains the triangle indices | Sets the indices of a single triangle of a mesh object |
| AddTriangle | MODELMESHTRIANGLE\* **pTriangle**: contains the triangle indices  DWORD\* **pnIndex**: filled with the new Index of the vertex | Adds a single triangle to a mesh object. |
| GetVertices | MODELMESHVERTEX\* **pVertices**: buffer filled with the vertex coordinates  DWORD **nBufferSize**: size of the buffer in elements, must be at least vertexcount  DWORD\* **pnVertexCount**: returns how many vertices have been written | Retrieves all vertex coordinates of a mesh object. |
| GetTriangleIndices | MODELMESHTRIANGLE\* **pIndices**: buffer filled with the triangle indices  DWORD **nBufferSize**: size of the buffer in elements, must be at least trianglecount  DWORD\* **pnTriangleCount**: returns how many triangles have been written | Retrieves all triangle indices of a mesh object. |
| SetGeometry | MODELMESHVERTEX\* **pVertices**: Array of vertex coordinates  DWORD **nVertexCount**: Size of the vertex array  MODELMESHTRIANGLE\* **pTriangles**: Array of triangle indices  DWORD **nTriangleCount**: Size of the triangle array | Sets the whole geometry of a mesh object. |
| CreatePropertyHandler | ILib3MFPropertyHandler\*\* **ppPropertyHandler**: returns a property handler instance for the mesh. | creates a property handler for the mesh |
| CreateMultiPropertyHandler | DWORD **nChannel**: Channel Index  ILib3MFPropertyHandler\*\* **ppPropertyHandler**: returns a property handler instance for the mesh. | creates a property handler for a specific multiproperty channel of a mesh |
| IsManifoldAndOriented | BOOL\* **pbIsOrientedAndManifold**: returns, if the object is oriented and manifold | Retrieves, if an object describes a topologically oriented and manifold mesh, according to the core spec |

1. **ILib3MFModelComponent**

ILib3MFModelComponent encapsulates one component node of a 3MF component object. It links to other object resources of the same model.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| GetObjectResourceID | DWORD\* **pnResourceID**: returns the associated resource ID | Returns the associated resource ID of the component. |
| GetObjectResource | ILib3MFModelObjectResource\*\***ppResource**: returns the associated resource instance | Returns the associated resource Instance of the component. |
| GetTransform | MODELTRANSFORM\* **pmTransformation**: filled with the component transformation matrix. | Returns the transformation matrix of the component. |
| SetTransform | MODELTRANSFORM\* **pmTransformation**: new transformation matrix | Sets the transformation matrix of the component. |
| HasTransform | BOOL\* **pbHasTransform**: if true is returned, the transformation is not equal to the identity | Returns, if the component has a different transformation than the identity matrix. |

1. **ILib3MFModelComponentsObject**

ILib3MFModelComponentsObject encapsulates all methods for handling 3MF component objects.

Parent interface: *ILib3MFModelObjectResource*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| AddComponent | ILib3MFModelObjectResource\* **pObject**: object to add as component. May not lead to circular references!  MODELTRANSFORM\* **pmTransform**: optional transform matrix for the component  ILib3MFModelComponent\*\* **ppComponent**: returns new component instance | Adds a new component to a component object. |
| GetComponent | DWORD **nIndex**: index of the component to retrieve (0 to componentcount - 1)  ILib3MFModelComponent\*\* **ppComponent**: retrieves component instance | Retrieves a component from a component object. |
| GetComponentCount | DWORD\* **pComponentCount**: returns the component count | Retrieves the component count of a component object. |

1. **ILib3MFModelBuildItem**

ILib3MFModelBuildItem encapsulates all methods for handling 3MF build items.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| GetObjectResourceID | DWORD\* **pnID**: returns the associated resource ID | Retrieves the object resource associated to a build item. |
| GetObjectResource | ILib3MFModelObjectResource\*\***ppResource**: returns the associated resource instance | Returns the associated resource Instance of the build item. |
| GetObjectTransform | MODELTRANSFORM\* **pmTransformation**: filled with the component transformation matrix. | Returns the transformation matrix of the build item. |
| SetObjectTransform | MODELTRANSFORM\* **pmTransformation**: new transformation matrix | Sets the transformation matrix of the build item. |
| HasObjectTransform | BOOL\* **pbHasTransform**: if true is returned, the transformation is not equal to the identity | Returns, if the build item has a different transformation than the identity matrix. |
| GetPartNumber | LPWSTR **pwszBuffer**: buffer to fill  ULONG **cbBufferSize**: size of buffer to fill. needs to be at least string length + 1.  ULONG\* **pcbNeededChars**: returns needed characters in buffer | Retrieves a build item's part number string. |
| GetPartNumberUTF8 | LPSTR **pszBuffer**: buffer to fill  ULONG **cbBufferSize**: size of buffer to fill. needs to be at least string length + 1.  ULONG\* **pcbNeededChars**: returns needed characters in buffer | Retrieves a build item's part number string in UTF8. |
| SetPartNumber | LPCWSTR **pwszPartNumber**: new part number string for referencing parts from the outside world. | Sets a build item's part number string |
| SetPartNumberUTF8 | LPCSTR **pszPartNumber**: new part number string for referencing parts from the outside world. | Sets a build item's part number string in UTF8 |
| GetHandle | DWORD\* **pHandle**: returns the handle | Retrieves an internal handle of the build item. This 32bit number is unique throughout the model, but only valid for in-memory use of this instance. |

1. **ILib3MFModelBuildItemIterator**

ILib3MFModelBuildItemIterator is a helper class to iterate through arbitrary lists of 3MF build items.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| MoveNext | BOOL\* **pbHasNext**: returns, if there is a build item to use | Iterates to the next build item in the list. |
| MovePrevious | BOOL\* **pbHasPrevious**: returns, if there is a build item to use | Iterates to the previous build item in the list. |
| GetCurrent | ILib3MFModelBuildItem\*\* **ppBuildItemInstance**: returns the build item instance | Returns the build item the iterator points at. |
| Clone | ILib3MFModelBuildItemIterator\*\* **ppIterator**: returns the cloned Iterator instance | Creates a new build item iterator with the same build item list. |

1. **ILib3MFModel**

ILib3MFModel is the basic instance owning all In-Memory elements of a 3MF file.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| SetUnit | DWORD **Unit**: enum value for the model unit (see NMR\_ModelTypes.h for details) | sets the units of a model |
| GetUnit | DWORD\* **pUnit**: enum value for the model unit (see NMR\_ModelTypes.h for details) | retrieves the units of a model |
| SetLanguage | LPCWSTR **pwszLanguage**: Language string identifier | sets the language of a model |
| SetLanguageUTF8 | LPCSTR **pszLanguage**: Language string identifier | sets the language of a model (UTF8) |
| GetLanguage | LPWSTR **pwszBuffer**: buffer to fill  ULONG **cbBufferSize**: size of buffer to fill. needs to be at least string length + 1.  ULONG\* **pcbNeededChars**: returns needed characters in buffer | retrieves the language of a model |
| GetLanguageUTF8 | LPSTR **pszBuffer**: buffer to fill  ULONG **cbBufferSize**: size of buffer to fill. needs to be at least string length + 1.  ULONG\* **pcbNeededChars**: returns needed characters in buffer | retrieves the language of a model (UTF8) |
| QueryWriter | LPCWSTR **pwszWriterClass**: string identifier for the file (currently "stl" and "3mf")  ILib3MFModelWriter\*\* **ppWriter**: returns the writer instance | creates a model writer instance for a specific file type |
| QueryReader | LPCWSTR **pwszReaderClass**: string identifier for the file (currently "stl" and "3mf")  ILib3MFModelReader \*\* **ppReader**: returns the reader instance | creates a model reader instance for a specific file type |
| GetResourceByID | DWORD **nResourceID**: Resource ID  ILib3MFModelResource\*\* **ppResource**: returns the resource instance | finds a model resource by its id |
| GetTexture2DByID | DWORD **nResourceID**: Resource ID  ILib3MFModelTexture2D\*\* **ppTexture**:  returns the texture resource instance | finds a model 2d texture by its id |
| GetBaseMaterialByID | DWORD **nResourceID**: Resource ID  ILib3MFModelBaseMaterial\*\* **ppMaterial**: returns the base material instance | finds a base material by its id |
| GetMeshObjectByID | DWORD **nResourceID**: Resource ID  ILib3MFModelMeshObject \*\* **ppMeshObject**: returns the resource instance | finds a mesh object resource by its id |
| GetComponentsObjectByID | DWORD **nResourceID**: Resource ID  ILib3MFModelComponentsObject\*\* **ppComponentsObject**: returns the resource instance | finds a components object resource by its id |
| GetBuildItems | ILib3MFModelBuildItemIterator\*\* **ppIterator**: returns the iterator instance | creates a build item iterator instance with all build items |
| GetResources | ILib3MFModelResourceIterator\*\* **ppIterator**: returns the iterator instance | creates a resource iterator instance with all resources |
| GetObjects | ILib3MFModelResourceIterator\*\* **ppIterator**: returns the iterator instance | creates a resource iterator instance with all object resources |
| GetMeshObjects | ILib3MFModelResourceIterator\*\* **ppIterator**: returns the iterator instance | creates a resource iterator instance with all mesh object resources |
| GetComponentsObjects | ILib3MFModelResourceIterator\*\* **ppIterator**: returns the iterator instance | creates a resource iterator instance with all component object resources |
| Get2DTextures | ILib3MFModelResourceIterator\*\* **ppIterator**: returns the iterator instance | creates a resource iterator instance with all 2D texture resources |
| GetBaseMaterials | ILib3MFModelResourceIterator\*\* **ppIterator**: returns the iterator instance | creates a resource iterator instance with all base material resources |
| MergeToModel | ILib3MFModel\*\* **ppMergedModel**: returns the merged model instance | merges all components and objects which are referenced by a build item. The memory is duplicated and a new model is created. |
| AddMeshObject | ILib3MFModelMeshObject\*\* **ppMeshObject**: returns the mesh object instance | adds an empty mesh object to the model |
| AddComponentsObject | ILib3MFModelComponentsObject\*\* **ppComponentsObject**: ppComponentsObject returns the component object instance | adds an empty component object to the model |
| AddTexture2DFromAttachment | ILib3MFModelAttachment\* **pTextureAttachment**: attachment containing the image data  ILib3MFModelTexture2D \*\* **ppTextureInstance**: returns the new texture instance | adds a texture2d resource to the model. Its path is given by that of an existing attachment. |
| AddTexture2D | LPCWSTR **pwszPath**: Package path of the texture  ILib3MFModelTexture2D \*\* **ppTextureInstance**: returns the new texture instance | adds a texture2d resource to the model. |
| AddTexture2DUTF8 | LPCSTR **pszPath**: Package path of the texture  ILib3MFModelTexture2D \*\* **ppTextureInstance**: returns the new texture instance | adds a texture2d resource to the model (UTF8). |
| AddBaseMaterialGroup | ILib3MFModelBaseMaterial\*\* **ppBaseMaterialInstance**: returns the new base material instance | adds an empty basematerials resource to the model |
| AddBuildItem | ILib3MFModelObjectResource\*\* **pObject**: Object instance associated with the build item  MODELTRANSFORM\* **pTransform**: Transformation matrix to be used  ILib3MFModelBuildItem\*\* **ppBuildItem**: returns the build item instance | adds a build item to the model |
| GetPackageThumbnail Attachment | ILib3MFModelObjectResource\*\* **pObject**: Object instance associated with the build item  MODELTRANSFORM\* **pTransform**: Transformation matrix to be used  ILib3MFModelBuildItem\*\* **ppBuildItem**: returns the build item instance | Get the attachment to the OPC package containing the package thumbnail |
| RemovePackageThumbnail Attachment |  | Remove the attachment to the OPC package containing the package thumbnail |

1. **ILib3MFModelFactory**

ILib3MFModelFactory is the global factory class for model instances.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| CreateModel | ILib3MFModel\*\* **ppModel**: returns created model instance | creates an empty model instance |
| GetSpecVersion | DWORD\* **pMajorVersion**: returns the major version of the specification  DWORD\* **pMinorVersion**: returns the major version of the specification | retrieves the current version of the 3MF implementation and specification |
| GetInterfaceVersion | DWORD\* **pInterfaceVersionMajor**: returns the major version of the shared library  DWORD\* **pInterfaceVersionMinor**: returns the minor version of the shared library  DWORD\* **pInterfaceVersionMicro**: returns the micro version of the shared library | retrieves the current interface version of the library (build version) this version will increment with each release of the library, and should be used to ensure API compatibility.  (See … ) |
| QueryExtension | LPCWSTR\* **pwszExtensionUrl**: pwszExtensionUrl URL of extension to check  BOOL\* **pbIsSupported**: returns whether the extension is supported or not  DWORD \* **pExtensionInterfaceVersion**: pInterfaceVersion returns the interface version of of the extensions (if extension is supported) | checks whether a extension is supported by the DLL and which version of the interface is used. This extension version will increment with each change of the API of the extension and may be used to ensure API compatibility. |
| QueryExtensionUTF8 | LPCSTR\* **pszExtensionUrl**: pwszExtensionUrl URL of extension to check (UTF8)  BOOL\* **pbIsSupported**: returns whether the extension is supported or not  DWORD \* **pExtensionInterfaceVersion**: pInterfaceVersion returns the interface version of of the extensions (if extension is supported) | checks whether a extension is supported by the DLL and which version of the interface is used. This extension version will increment with each change of the API of the extension and may be used to ensure API compatibility. |
| RetrieveProgressMessage | int **progressIdentifier**: the progress identifier that is passed to the callback function LPCSTR\* **progressMessage**: English text for the progress identifier | Return an English text for a progress identifier  Note: this is the only function you can call from your callback function. |

## Class Reference – BeamLattice specification

The following list gives an overview of the classes that implement the beamlattice specification. Moreover, it explains the functionality added to core-specification classes.

**12. ILib3MFModelMeshObject (continuation)**

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| GetBeamLatticeMinLength | DOUBLE\* **pdMinLength**: minlength of the beamlattice | Returns the minimal member length for the beamlattice |
| SetBeamLatticeMinLength | DOUBLE **dMinLength**: minlength of the beamlattice | Sets the the minimal member length for the beamlattice |
| GetBeamLatticeRadius | DOUBLE\* **pdRadius**: default radius of the beams in the beamlattice | Returns the default radius for the beamlattice |
| SetBeamLatticeRadius | DOUBLE **dRadius**: default radius of the beams in the beamlattice | Sets the default radius for the beamlattice |
| GetBeamLatticeCapMode | eModelBeamLatticeCapMode\* **peCapMode**: default eModelBeamLatticeCapMode of the beamlattice  (MODELBEAMLATTICECAPMODE\_SPHERE, MODELBEAMLATTICECAPMODE\_HEMISPHERE, MODELBEAMLATTICECAPMODE\_BUTT) | Returns the default capping mode for the beamlattice |
| SetBeamLatticeCapMode | eModelBeamLatticeCapMode **eCapMode**: default eModelBeamLatticeCapMode of the beamlattice | Sets the default capping mode for the beamlattice |
| GetBeamLatticeClipping | eModelBeamLatticeClipMode\* **peClipMode**: default eModelBeamLatticeClipMode of the beamlattice (MODELBEAMLATTICECLIPMODE\_NONE, MODELBEAMLATTICECLIPMODE\_INSIDE, MODELBEAMLATTICECLIPMODE\_OUTSIDE)  DWORD\* **pnRessourceID:** filled with the resourceID of the clipping mesh-object or a undefined value if pClipMode is MODELBEAMLATTICECLIPMODE\_NONE | Returns the clipping mode and the clipping-mesh for the beamlattice of this mesh |
| SetBeamLatticeClipping | eModelBeamLatticeClipMode **eClipMode**: default eModelBeamLatticeCapMode of the beamlattice  DOUBLE **nRessourceID** the resourceID of the clipping mesh-object. This mesh-object has to be defined before setting the Clipping | Sets the clipping mode and the clipping-mesh for the beamlattice of this mesh |
| GetBeamLatticeRepresentation | BOOL\* **pbHasRepresentation**: flag whether the beamlattice has a representation mesh.  DWORD\* **pnRessourceID:** filled with the resourceID of the representation mesh-object. | Returns the representation-mesh for the beamlattice of this mesh |
| SetBeamLatticeRepresentation | DOUBLE **nRessourceID** the resourceID of the representation mesh-object. This mesh-object has to be defined before setting the representation mesh. Set "0" to unset the representation mesh. | S ets the representation-mesh for the beamlattice of this mesh |
| GetBeamCount | DWORD\* **pnBeamCount**: filled with the beam count | Returns the beam count of a mesh object |
| GetBeam | DWORD **nIndex**: Index of the beam (0 to beamcount - 1)  MODELMESHBEAM \* **pBeam**: filled with the node indices, radii and capmodes | Returns indices of a single beam of a mesh object |
| SetBeam | DWORD **nIndex**: Index of the beam (0 to beamcount - 1)  MODELMESHBEAM \* **pBeam**: contains the node indices, radii and capmodes | Sets the indices, radii and capmodes of a single beam of a mesh object |
| AddBeam | MODELMESHBEAM \* **pBeam**: contains the node indices, radii and capmodes  DWORD\* **nIndex**: filled with the new Index of the beam | Adds a single beam to a mesh object |
| SetBeamIndices | MODELMESHBEAM \* **pIndices**: buffer with the beam indices  DWORD **nBufferSize**: size of the buffer in elements | Sets all beam indices, radii and capmodes of a mesh object |
| GetBeamIndices | MODELMESHBEAM \* **pIndices**: buffer filled with the beam indices  DWORD **nBufferSize**: size of the buffer in elements, must be at least beam count  DWORD\* **pnBeamCount**: returns how many beams have been written | Retrieves all beam indices of a mesh object |
| GetBeamSetCount | DWORD\* **pnBeamSetCount**: filled with the beamset count | Returns the number of beamsets of a mesh object |
| AddBeamSet | ILib3MFModelMeshBeamSet\*\* **ppBeamSet**: pointer to the new beamset | Adds an empty beamset to a mesh object |
| GetBeamSet | DWORD **nIndex**: index of the requested beamset (0 ... beamsetcount-1)  ILib3MFModelMeshBeamSet\*\* **ppBeamSet**: pointer to the requested beamset | Returns a beamset of a mesh object |

**21. ILib3MFModelMeshBeamSet**

ILib3MFModelMeshBeamSet is a class that holds the references that contain to a beamset.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| SetName | LPCWSTR\* **pwszName**: new name of the BeamSet as UTF16 string. (e.g. "Car") | Sets a beamset's name string |
| SetIdentifier | LPCWSTR\* **pwszIdentifier**: new identifier of the BeamSet as UTF16 string. (e.g. "Car") | Sets a beamset's identifier string |
| GetNameUTF8 | LPSTR **pszBuffer*:*** buffer to fill  ULONG **cbBufferSize*:*** size of buffer to fill. needs to be at least string length + 1  ULONG\* **pcbNeededChars*:*** returns needed characters in buffer | Retrieves a BeamSet's name string (UTF8) |
| GetIdentifierUTF8 | LPSTR **pszBuffer*:*** buffer to fill  ULONG **cbBufferSize*:*** size of buffer to fill. needs to be at least string length + 1  ULONG\* **pcbNeededChars*:*** returns needed characters in buffer | Retrieves a BeamSet's identifier string (UTF8) |
| GetRefCount | DWORD\* **pnCount*:*** returns the reference count | Retrieves the reference count of a BeamSet |
| SetRefs | DWORD\* **pRefs*:*** buffer containing the indices of all beams in this beamset  DWORD **nRefCount*:*** number of references to be set | Sets the references of a BeamSet |
| GetRefs | DWORD\* **pRefs*:*** buffer filled with beam references (indices of beams)  DWORD **nBufferSize*:*** size of the buffer in elements, must be at least refcount  DWORD\* **pnRefCount*:*** returns how many references have been written | Retrieves all references of a BeamSet |

## Class Reference – Slice-specification

ILib3MFModelMeshBeamSet is a class that holds the references that contain to a beamset.

Parent interface: *ILib3MFBase*

**12. ILib3MFModelMeshObject (continuation)**

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| SetSliceStackId | DWORD **nSliceStackId**: id of the slice stack to link | Link the mesh object to a slice stack |
| GetSliceStackId | DWORD\* **nSliceStackId**: id of the slice stack that is linked to the Mesh | Returns to which slice stack the mesh is linked |
| SetSlicesMeshResolution | eModelSlicesMeshResolution **eSlicesMeshResolution**: meshresolution of this mesh object | set the meshresolution of the mesh object |
| GetSlicesMeshResolution | eModelSlicesMeshResolution \* p**eSlicesMeshResolution**: meshresolution of this mesh object | get the meshresolution of the mesh object |

**14. ILib3MFModelComponentsObject (continuation)**

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| SetSliceStackId | DWORD **nSliceStackId**: id of the slice stack to link | Link the components object to a slice stack |
| GetSliceStackId | DWORD\* **nSliceStackId**: id of the slice stack that is linked to the Mesh | Returns to which slice stack the components is linked |
| SetSlicesMeshResolution | eModelSlicesMeshResolution **eSlicesMeshResolution**: meshresolution of this components object | set the meshresolution of the components object |
| GetSlicesMeshResolution | eModelSlicesMeshResolution \* p**eSlicesMeshResolution**: meshresolution of this components object | get the meshresolution of the components object |

1. **ILib3MFModel (continuation)**

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| AddSliceStack | FLOAT **nBottomZ**: Model instance to add slicestack to ILib3MFSliceStack\* **ppSliceStack**: returns the new slice stack object | Adds a slicestack to a model |

1. **ILib3MFSlice**

ILib3MFSlice encapsulates all slice functionality for handling slices in 3mf.

Parent interface: *ILib3MFBase*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| AddVertex | MODELSLICEVERTEX\* **pVertex**: holds the vertex coordinates DWORD\* **pnIndex**: returns the index of the vertex. Needed to reference the vertex later on in a polygon | Add a single vertex to a slice |
| BeginPolygon | DWORD\* **pnIndex**: index of the newly created polygon | Begin a polygon |
| AddPolygonIndices | DWORD **nPolygonIndex**: index of the polygon to add the indices to  DWORD\* **pnVertexIndices**: array of the indices for the polygon  DWORD **nBufferSize**: number of elements in the vertex array  DWORD\* **nPolygonVertexIndex**: returns the start index of the added indices | Add indices to a polygon |
| GetPolygonCount | DWORD\* **pnPolygonCount**: returns the number of polygons in the slice | Get the number of polygons in the slice |
| GetVertexCount | DWORD\* **pnVertexCount**: returns the number of vertices in the slice | Get the number of vertices in the slice |
| GetTopZ | FLOAT\* **pfTopZ**: returns the upper Z coordinate of the slice | Get the upper Z coordinate of the slice |
| GetIndexCountOfPolygon | DWORD **nPolygonIndex**: the index of the polygon  DWORD\* **pnPolygonCount**: returns the number of indices in a polygon of the slice | Get the number of indices of a polygon in the slice |
| GetPolygonIndices | DWORD **nPolygonIndex**: the index of the polygon to query  DWORD\* **pPolygonIndices**: an array to be filled with the polygon indices  DWORD **nBufferCount**: number of elements in "pPolygonIndeces". Should match the number of indices in the polygon (GetIndexCountOfPolygon), if less not all indices are returned, if greater memory is wasted | Get the indices of a polygon |
| GetVertex | DWORD **nIndex**: index of the vertex to getMODELSLICEVERTEX\* **pVertex**: MODELSLICEVERTEX structure to be filled | Get a vertex of the slice |

1. **ILib3MFSliceStack**

ILib3MFSliceStack encapsulates all methods for handling slice stacks in 3MF.

Parent interface: *ILib3MFModelResource*

|  |  |  |
| --- | --- | --- |
| Method | Parameters | Description |
| AddSlice | FLOAT **fTopZ**: upper Z coordinate of the slice ILib3MFSlice\*\* **ppSliceObject**: the newly created slice object | Adds a slice to the slicestack |
| GetSlice | DWORD **nSliceIndex**: the index of the slice to query  ILib3MFSlice\*\* **ppSliceObject**: returns the slice | Query a slice from the slice stack |
| GetSliceCount | DWORD\* **pnSliceCount**: returns the number of slices on the slice stack | Get the number of slices on the slice stack |
| GetBottomZ | FLOAT\* **pfBottomZ**: the lower Z-Coordinate the slice stack | Get the lower Z-Coordinate of the slice stack |
| GetSliceStackId | DWORD\* **pnSliceStackId**: returns the slice stack id | Obtain the slice stack id |
| GetResourceID | DWORD\* **pnResourceID**: returns the resource id | Get the resource id of the slice stack |
| SetSliceRef | LPWSTR **pwszSliceRef**: the name of the entity for the slice stack | Set the reference of the slice stack, i.e. the slice stack will be stored in a separate entity within the 3mf file |
| GetSliceRef | LPWSTR **pwszSliceRef**: string for the slice reference  DWORD **nBufferSize**: length of the string passed in pwszSliceRef  DWORD\* **pnNeededChars**: number of characters needed to store the slice reference | Get the reference of the slice stack, i.e. the slice stack is stored in a separate entity within the 3mf file |

## License

The 3MF Library is released under the simplified BSD License (<http://opensource.org/licenses/BSD-2-Clause>). In short, this means it is allowed to use this code in closed source applications, open source applications and web services at no charge.

Redistribution and use in source and binary forms, with or without modification,are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.

2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

## Contributors

The current code base is maintained by the 3MF consortium, and is assembled with contributions from the following consortium members:

* Autodesk Inc., San Rafael, CA
* Microsoft Corporation, Redmond, WA
* netfabb GmbH, Lupburg, Germany

The development has just begun. We invite everyone interested to contribute test results, bug reports, suggestions or code contributions under the simplified BSD License. We are actively looking for testers on all different platforms and in all different programming languages.

If you are making language bindings for your favourite language, we plead with you to release it public into the Lib3MF repository, as this will enable others to spread the format in an easy way.

The current version of this document and the library code can be obtained from github at <https://github.com/3MFConsortium/lib3mf>.

For more information, contact the 3MF Working Group at <http://3mf.io>.

## Open API Points and Roadmap

* Remove Components from ComponentsObjects
* Remove Build Items from Model
* Signature support
* Metadatagroups
* Metadata checking (partially)
* Component Dependency Checking
* ID Reference Checking
* Transformations with negative determinant
* Compositematerials and multiproperties