The Gbr2OasFrac Library

API Reference

ARTWORK CONVERSION SOFTWARE INC.

415 Ingalls St. Santa Cruz CA 90560
Phone: (831) 426-6163
Fax: (831) 426-2824
Email: info@artwork.com
Web: http://www.artwork.com

INDEX

Index > **General Information**

1. Supported Platforms

1. 32bit Windows VC 6.0 (x86)
2. 32bit Windows VC 2008 (x86)
3. 64bit Windows VC 2008 (amd64)
4. 32bit Solaris 8 (sparc)
5. 64bit Solaris 8 (sparc)
6. 32bit Red Hat E3 Linux (x86)
7. 64bit Red Hat E3 Linux (x86_64)

2. Key Component Files

a. Windows

gbr2oasfrac.dll	The Gbr2OasFrac Library
igbr2oasfrac.h	Gbr2OasFrac Library C++ API header file
igbr2oasfrac_c.h	Gbr2OasFrac Library C API header file
gbr2oasfracshell.dsw/.sln	Project to build a sample program that uses the Gbr2OasFrac Library

b. Unix/Linux

gbr2oasfrac.so	The Gbr2OasFrac Library
igbr2oasfrac.h	Gbr2OasFrac Library C++ API header file
igbr2oasfrac_c.h	Gbr2OasFrac Library C API header file
makefile	Makefile to build a sample program that uses the Gbr2OasFrac Library

Index > API Reference > **C API** > **Usage & Flow**

Sequence of calls to Gbr2OasFrac API Functions :-

1. NGbr2OasFrac_initializeOnce
2. NGbr2OasFrac_setProgressCallbackFn
3. NGbr2OasFrac_convertGBRToFracturedOAS
4. NGbr2OasFrac_convertGBRToFracturedOAS_byCmd
5. NGbr2OasFrac_closeOnce

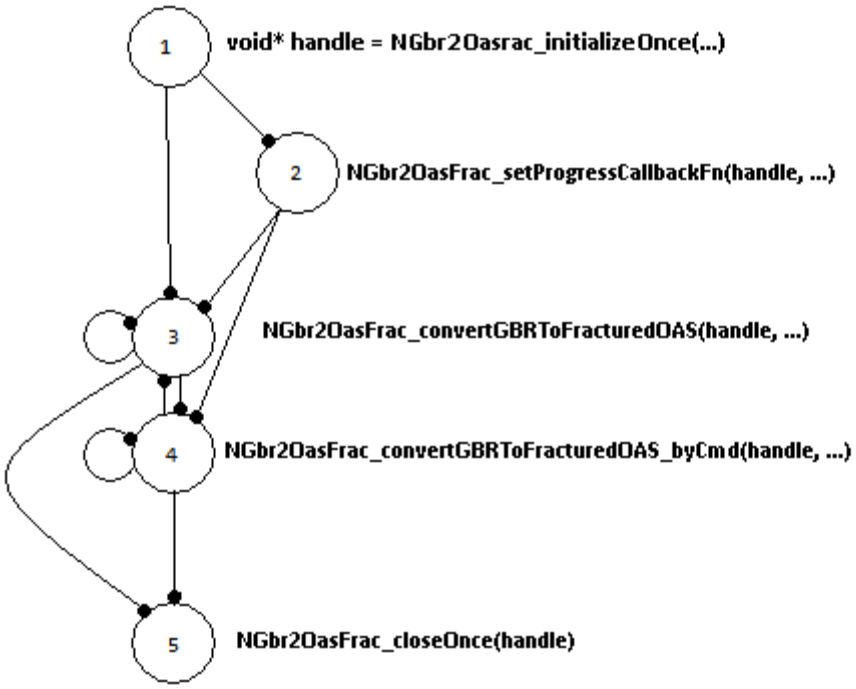

Figure 1: C API Flow

Index > API Reference > C API > **Data Structures** > **_sGBR2OASFRACOPTIONS**

Objective:

To store optional arguments to be used by Gbr2OasFrac during execution.

Members:

mModalOAS (unsigned char)	1: Generate modal OASIS output. 0: Generate regular OASIS output. (default)
mKeepTempFiles (unsigned char)	1: Retain temporary and intermediate files after execution. 0: Remove temporary and intermediate files after execution. (default)
mNoProgress (unsigned char)	1: Do not launch progress dialog. 0: Launch progress dialog. (default)
mArcRes (double)	Specify arc resolution in degrees. (default: 9.0)
mArcSag (double)	Specify chord error in Gerber input units. (default: 0.0)
mNewGrid (double)	Specify the grid of output data. (default: 0.001)
mWorkDir (const char*)	Specify working directory. (default: current working directory)
mEngThreads (unsigned char)	Specify the number of threads to be used by the Gbr2OasFrac Engine for flattening the Gerber data. (default 1)
mSortXY (unsigned char)	1: Sort the polygons in the output by their position (Y first, then X) 0: Process the polygons in the order in which they are read from the input file.
mTiles::mEnable (unsigned char)	1: Enable tiling. Use mTiles::mSize[0..1] to specify tile size. 0: Disable tiling. mTiles::mSize and mCellsTop2Bottom and any transformation will be ignored.
mTiles::mSize (double[2])	If tiling is enabled, specify the size (in user units) for each tile.
mCellsTop2Bottom (unsigned char)	1: Write cell (representing each tile) references in the order of the position of the tile (represented by that cell) from Top to Bottom. 0: Do not order cell references.
mTiles::mMirror (unsigned char)	Used only if mTiles::mEnable is set (to 1). mirror_X: Perform mirror transformation on each tile along the X-axis (about Y-axis) mirror_Y: Perform mirror transformation on each tile along the Y-axis (about X-axis) mirror_NONE: Do not perform mirror transformation. (default)
mTiles::mAngle (unsigned char)	Used only if mTiles::mEnable is set (to 1). angle_90: Perform rotation of +90 degrees (counter clock-wise) on each tile. angle_180: Perform rotation of +180 degrees on each tile. angle_270: Perform rotation of +270 degrees on each tile. angle_0: Do not perform any rotation. (default)
mTiles::mScale (double)	Used only if mTiles::mEnable is set (to 1). Scale each tile in both X and Y by the specified factor. (default 1.0)
mTiles::mTranslate (double[2])	Used only if mTiles::mEnable is set (to 1). Specify translation distance along X (index 0) and Y (index 1) respectively. This is used only if mTiles::mTranslatePolicy is set to translate_EXPLICIT.
mTiles::mTranslatePolicy (unsigned char)	Used only if mTiles::mEnable is set (to 1). translate_NONE: Do not perform translation on each tile.

	(default) translate_AUTO: Translate each tile appropriately so that the centroid of the tile preserved as it was before applying the transformations (mirror, scale, rotation). translate_EXPLICIT: Specify translation distance along X and Y using the mTiles::mTranslate member.
mEngineArgs (const char*)	Specify additional options for the Gbr2OasFrac engines.

Notes:

1. The function _sGBR2OASFRACOPTIONS_DEFAULT can be used to set an instance of this struct to it's default values. It is recommended to do this everytime aninstance of this struct is created.
2. *mEngineArgs is an advanced option that is currently provided only as a placeholder for future enhancements and fine-tuning the performance. Please consult with Artwork for more information about this feature*.

Index > API Reference > C API > Data Structures > **_sGBR2OASFRACERROR**

Objective:

To store information about an error that may have occurred during execution.

Members:

mCode (int)	Stores the error code. (Refer to each function/method for a list of error codes returned)
mMsg (const char*)	Pointer to a message string containing details about the error.

Notes:

1. The information referred by the mMsg pointer is not persistant across function calls. If this information would be required later, it must be copied immediately to another persistant buffer.

Index > API Reference > C API > Data Structures > **NGbr2OasFracProgressCbFnPtr**

Objective:

A type that represents a function pointer with the signature:-
int GBR2OASFRAC_CALLDECL functionName(int percentDone, const char *msg);

Notes:

1. GBR2OASFRAC_CALLDECL refers to the calling convention used by the function in question. For Windows, it is '__stdcall'. For Unix, it is empty. Any function that is to be passed on as this type must be defined with the GBR2OASFRAC_CALLDECL calling convention.
2. This type is used to pass a function as a callback to the NGbr2OasFrac_setProgressCallbackFn or IGbr2OasFrac:: setProgressCallbackFn function/methods.

See Also:

NGbr2OasFrac_setProgressCallbackFn
IGbr2OasFrac:: setProgressCallbackFn

Sample Code:

```
int GBR2OASFRAC_CALLDECL onProgressUpdate(int percent, const char* updateMsg); //declaration

int someFunction()
{
  ...
  void* libHandle = NGbr2OasFrac_initializeOnce(...);
  ...
  NGbr2OasFrac_setProgressCallbackFn(libHandle, onProgressUpdate);
  ...
}
```

Index > API Reference > C API > **Functions** > **_sGBR2OASFRACOPTIONS_DEFAULT**

```
void _sGBR2OASFRACOPTIONS_DEFAULT(_sGBR2OASFRACOPTIONS* instance);
```

Objective:

To set the members of the _sGBR2OASFRACOPTIONS struct to default values.

Parameters:

instance (_sGBR2OASFRACOPTIONS*) Address of the object to be initialized (set to default)

Return:

None.

Notes:

1. Use this function to initialize objects of type _sGBR2OASFRACOPTIONS.
2. It is recommended that objects of this type be initialized before use.
3. Refer to _sGBR2OASFRACOPTIONS to know the default values.

See Also:

_sGBR2OASFRACOPTIONS

Sample Code:

```
...
_sGBR2OASFRACOPTIONS options;
_sGBR2OASFRACOPTIONS_DEFAULT(&options);
...
```

Index > API Reference > C API > **Functions** > **NGbr2OasFrac_initializeOnce**

```
void* NGbr2OasFrac_initializeOnce(const char* execPath, const char* appName,
                                  _sGBR2OASFRACERROR* errorO);
```

Objective:

To create and initialize an instance of the Gbr2OasFrac library.

Parameters:

execPath (const char*)	Complete path of the program using the Gbr2OasFrac library.
appName (const char*)	Name of the application using the Gbr2OasFrac Library. This name will be used to display the title of the Gbr2OasFrac progress dialog.
errorO (_sGBR2OASFRACERROR*)	Address of an object to get error information if this function fails. (can be set to NULL if the error information is not required).

Return:

valid pointer (handle): success
NULL: failure. If errorO is not NULL, errorO->mCode will have one of the following values:-
cIOErr_MEMALLOC (Memory allocation failure)
cIOErr_INVPROGDIR (execPath is empty or NULL)
cIOErr_VERSION (The program is using a different API header file than the one the library was built with)
cIOErr_LICENSE (License failure)

Notes:

1. The 'handle' returned by this function represents ONE INSTANCE of the Gbr2OasFrac library.
2. The handle returned by this function must be remembered since it will be required for calling other functions.
3. Multiple instances of the library can be active in parallel by calling this function once for each instance.
4. None of the library functions can be used without the 'handle' and therefore without calling this function.
5. This function must be called only ONCE for each instance of the library. An for each such call, there must be a corresponding NGbr2OasFrac_closeOnce.
6. The library and its dependent binaries must be present in the same directory as the program using the library.

See Also:

NGbr2OasFrac_closeOnce
_sGBR2OASFRACERROR
Usage & Flow

Sample Code:

```
void myConvertGBRToFracturedOAS()
{
  ...
  _sGBR2OASFRACERROR libError;
  void* libHandle = NGbr2OasFrac_initializeOnce(execPath, "My Gbr2OasFrac Client", &libError);
  if(libHandle == NULL)
```

```
  {
    myReportError(libError.mMsg);
    return;
  }

 ...
  sGBR2OASFRACOPTIONS options;
  sGBR2OASFRACOPTIONS_DEFAULT(options);
 ...
 options.mWorkDir = getMyWorkingDir();
 success = NGbr2OasFrac_convertGBRToFracturedOAS(libHandle, myInputFile, myOutputFile,
                                                 &options, &libError);
 if(success == 0)
 {
   myReportError(libError.mMsg);
 }
 ...
 NGbr2OasFrac_closeOnce(&libHandle);
 ...
}
```

Index > API Reference > C API > Functions > **NGbr2OasFrac_closeOnce**

```
void NGbr2OasFrac_closeOnce(void** instanceHandle);
```

Objective:

To close an instance of the Gbr2OasFrac library and free all allocated resources associated with it.

Parameters:

instanceHandle (void**) Address of the handle to the instance of the library.

Return:

None.

Notes:

1. Every NGbr2OasFrac_initializeOnce must have a corresponding NGbr2OasFrac_closeOnce.
2. Once this function is called, the instance handle is invalidated (set to NULL) and the resources associated with it are de-allocated. Therefore, this handle cannot be used for making any more NGbr2OasFrac_* function calls.

See Also:

NGbr2OasFrac_initializeOnce
Usage & Flow

Sample Code:

```
//refer to NGbr2OasFrac_initializeOnce sample code.
```

Index > API Reference > C API > Functions > **NGbr2OasFrac_setProgressCallbackFn**

```
void NGbr2OasFrac_setProgressCallbackFn(void* instanceHandle,
                                        NGbr2OasFracProgressCbFnPtr callbackFnPtr);
```

Objective:

To specify a callback function to get progress updates from the Gbr2OasFrac library as the input GBR file is being translated to fractured OASIS.

Parameters:

instanceHandle (void*)	Address of the handle to the instance of the library.
callbackFnPtr (NGbr2OasFracProgressCbFnPtr)	Pointer/Name of the callback function to be called when the Gbr2OasFrac library has progress updates.

Return:

None.

Notes:

1. If the client program wishes to receive progress updates while the fractured OASIS output is being generated, this function must be called to set the callback function.
2. This function must be called before NGbr2OasFrac_convertGBRToFracturedOAS or NGbr2OasFrac_convertGBRToFracturedOAS_byCmd.
3. If progress updates are not desired for subsequent executions, pass NULL for 'callbackFnPtr'.
4. The callback function must have the following signature:-
   int GBR2OASFRAC_CALLDECL functionName(int percentDone, const char *msg);
5. The callback function if set, will be called everytime the Gbr2OasFrac library has progress updates.
6. 'percentDone' parameter reports the percentage of execution complete and 'msg' parameter reports the execution phase.
7. If the program wants to interrupt the execution at this point, this callback function must return -1. Otherwise, Gbr2OasFrac will continue with the translation.

See Also:

Usage & Flow
NGbr2OasFrac_convertGBRToFracturedOAS
NGbr2OasFrac_convertGBRToFracturedOAS_byCmd
NGbr2OasFracProgressCbFnPtr

Sample Code:

```
int GBR2OASFRAC_CALLDECL onProgressUpdate(int percent, const char* updateMsg)
{
  myUpdateProgressDialog(updateMsg, percent);
  if(checkStopTranslation() == 1)
    return -1; //return -1 to inform the library to stop translation.
  else
    return 0;
}

int myConvertGBRToFracturedOAS ()
{
  ...
  void* libHandle = NGbr2OasFrac_initializeOnce(...);
  ...
```

```
  NGbr2OasFrac_setProgressCallbackFn(libHandle, onProgressUpdate); //get progress updates

  const char* cmdString = "\"-i:C:\Gerber Files\input.gbr\" \"-o:C:\Temp Dir\output.oas\" \
                          -grid_addr:0.001 \"-workdir:C:\Temp Dir\" \
                          -modal_oas -keep -silent";
  int success = NGbr2OasFrac_convertGBRToFracturedOAS_byCmd(libHandle, cmdString, &libError);

  if(success == 0)
  {
    myReportError(libError.mMsg);
  }

  NGbr2OasFrac_setProgressCallbackFn(libHandle, NULL); //disable progress updates

  success = NGbr2OasFrac_convertGBRToFracturedOAS_byCmd(libHandle, cmdString, &libError);

  ...
}
```

Index > API Reference > C API > Functions > **NGbr2OasFrac_convertGBRToFracturedOAS**

```
int NGbr2OasFrac_convertGBRToFracturedOAS(void* instanceHandle, const char* inputGBRFile,
                                          const char* outputOASFile,
                                          const _sGBR2OASFRACOPTIONS* options,
                                          _sGBR2OASFRACERROR* errorO);
```

Objective:

To generate a fractured OASIS file from a Gerber file.

Parameters:

instanceHandle (void*)	Handle to the instance of the library obtained from NGbr2OasFrac_initializeOnce.
inputGBRFile (const char*)	Path to a valid Gerber file to be translated.
outputOASFile (const char*)	Location of the output OASIS file that will be created when this function returns successfully.
options (const _sGBR2OASFRACOPTIONS*)	A pointer to a set of options to control the output and execution of the translation.
errorO (_sGBR2OASFRACERROR*)	Address of an object to get error information if this function fails. (can be set to NULL if the error information is not required).

Return:

1: success
0: failure. If errorO is not NULL, errorO->mCode will have one of the following values:-
cCGTFOErr_MEMALLOC (Memory allocation failure)
cCGTFOErr_EXECPHASE1 (Phase 1: Unionizing Gerber Data into GDSII, failed)
cCGTFOErr_EXECPHASE2 (Phase 2: Converting GDSII into Fractured OASIS, failed)
cCGTFOErr_LICENSE (License error)
cCGTFOErr_USERINTR (User hit cancel while execution was in progress)

errorO->mMsg would point to a string with details about the error.

Notes:

1. This function converts an input Gerber file to a fractured OASIS file in 2 phases; unionizing Gerber Data into GDSII and converting GDSII into Fractured OASIS.
2. A few intermediate files are created in the 'working directory' (specified by the options.mWorkDir member or the default working directory). If options.mKeep is set, these intermediate files will not be deleted upon completion, otherwise they will be deleted.
3. If options.mNoProgress is set, all phases will be executed in 'silent' (background) mode, otherwise, a progress dialog will display the current progress updates.
4. If options.mModalOAS is set, the output OASIS file have modality, otherwise it will not.
5. The data grid of the output file will be in accordance with the options.mNewGrid member or 0.001 by default.
6. If options.mTiles.mEnable is set (to 1), the data to be translated is chopped into 'tiles' of size options.mTiles.mSize. Each of these tiles is repesented in the output file as a unique sub-structure
7. Other options such as options.mCellsTop2Bottom and various transformations are used only if tiling is enabled.
8. Unless options.mEngineArgs is set, all phases will be executed with default settings. Please refer to _sGBR2OASFRACOPTIONS or consult Artwork for information about these 'advanced' settings.
9. Each call to this function is self-contained and independent of other calls. Each call would create new intermediate files and output.
10. This function differs from NGbr2OasFrac_convertGBRToFracturedOAS_byCmd only in its interface.

See Also:

- NGbr2OasFrac_initializeOnce
- _sGBR2OASFRACOPTIONS
- _sGBR2OASFRACERROR
- NGbr2OasFrac_convertGBRToFracturedOAS_byCmd
- Usage & Flow

Sample Code:

```
void myConvertGBRToFracturedOAS()
{
  ...
  _sGBR2OASFRACERROR libError;
  void* libHandle = NGbr2OasFrac_initializeOnce(execPath, "My Gbr2OasFrac Client", &libError);
  if(libHandle == NULL)
  {
    myReportError(libError.mMsg);
    return;
  }

  ...
  _sGBR2OASFRACOPTIONS options;
  _sGBR2OASFRACOPTIONS_DEFAULT(options);
  ...
  options.mNewGris = 0.0001;
  options.mWorkDir = getMyWorkingDir();
  success = NGbr2OasFrac_convertGBRToFracturedOAS(libHandle, myInputFile, myOutputFile,
                                                  &options, &libError);
  if(success == 0)
  {
    myReportError(libError.mMsg);
  }
  ...
  NGbr2OasFrac_closeOnce(&libHandle);
  ...
}
```

Index > API Reference > C API > Functions >

NGbr2OasFrac_convertGBRToFracturedOAS_byCmd

```
int NGbr2OasFrac_convertGBRToFracturedOAS_byCmd(void* instanceHandle, const char* cmdString,
                                                _sGBR2OASFRACERROR* errorO);
```

Objective:

To generate a fractured OASIS file from a Gerber file.

Parameters:

instanceHandle (void*)	Handle to the instance of the library obtained from NGbr2OasFrac_initializeOnce.
cmdString (const char*)	Translation parameters specified as a command string.
errorO (_sGBR2OASFRACERROR*)	Address of an object to get error information if this function fails. (can be set to NULL if the error information is not required).

Command String Syntax: (All arguments containing 'space' must be enclosed in 'double quotes')

```
SYNTAX:-
cmdString := "<inputFile> <outputFile> <options>"

inputFile := "-i:<input Gerber file path>"
outputFile := "-o:<output OASIS file path>"
options := "[<arcRes>] [<arcSag>] [<newGrid>] [<workDir>] [<modalOAS>] [<keep>] [<silent>]
             [<otherArgs>] [<threads>] [<sortxy>] [<tile> [<top2bottom>]
             [<transformation>]]"

arcRes := "-arcres:<angle in degrees>
arcSag := "-arcsag:<chord error in Gerber input units>
newGrid := "-grid_addr:<output data grid>"
workDir := "-workdir:<path to working dir>"
modalOAS := "-modal_oas"
keep := "-keep"
silent := "-silent"
threads := "-eng_threads:<1-16>"
otherArgs := "-other_args:<arg list>"
tile := "-tilesz:<size_in_x>,<size_in_y>"
top2bottom := "-top2bottom_cells"
sortxy := "-sort_xy"
transformation := [<mirror>] [<rotation>] [<translation>] [<scaling>]
mirror := "-mirror:x" | "-mirror:y"
rotation := "-angle:90" | "-angle:180" | "-angle:270"
translation := "-translate:auto" | "-translate:<dx>,<dy>"
scaling := "-scale:<scale-factor>"



SAMPLE:-
const char* cmdString = "\"-i:C:\Gerber Files\input.gbr\" \"-o:C:\Temp Dir\output.oas\"
                        -grid_addr:0.001 \"-workdir:C:\Temp Dir\" -modal_oas -keep -silent
                        -arcres:4.0 -eng_threads:2 -tilesz:1000,1000 -top2bottom_cells
                        -sort_xy -mirror:x -angle:270 -translate:100,150 -scale:2.0;


CO-RELATION WITH _sGBR2OASFRACOPTIONS (options)
-arcres ⇔ options.mArcRes
-arcsag ⇔ options.mArcSag
```

```
-grid_addr ⇔ options.mNewGrid
-workdir ⇔ options.mWorkDir
-modal_oas ⇔ options.mModalOAS
-keep ⇔ options.mKeepTempFiles
-silent ⇔ options.mNoProgress
-other_args ⇔ options.mEngineArgs
-eng_threads ⇔ options.mEngThreads
-tilesz ⇔ options.mTiles.mSize, options.mTiles.mEnable = 1
-top2bottom_cells ⇔ options.mCellsTop2Bottom
-sort_xy ⇔ options.mSortByXY
-mirror ⇔ options.mTiles.mMirror
-angle ⇔ options.mTiles.mAngle
-translate ⇔ options.mTiles.mTranslatePolicy + options.mTiles.mTranslate
-scale ⇔ options.mTiles.mScale
```

Return:

1: success
0: failure. If errorO is not NULL, errorO->mCode will have one of the following values:-
cCGTFOErr_MEMALLOC (Memory allocation failure)
cCGTFOErr_EXECPHASE1 (Phase 1: Unionizing Gerber Data into GDSII, failed)
cCGTFOErr_EXECPHASE2 (Phase 2: Converting GDSII into Fractured OASIS, failed)
cCGTFOErr_LICENSE (License error)
cCGTFOErr_USERINTR (User hit cancel while execution was in progress)
cCGTFOCErr_INVALIDCOMMAND (Empty or NULL cmdString)
cCGTFOCErr_PARSEINVQUOTES (Double Quotes Mismatch in cmdString)
cCGTFOCErr_PARSEINVINPUT (-i: command absent, no input file specified)
cCGTFOCErr_PARSEINVOUTPUT (-o: command absent, no output file specified)

errorO->mMsg would point to a string with details about the error.

Notes:

1. This function converts an input Gerber file to a fractured OASIS file in 2 phases; unionizing Gerber data into GDSII and Converting GDSII into Fractured OASIS.
2. A few intermediate files are created in the 'working directory' (specified by the –workdir command or the default working directory). If -keep command is present, these intermediate files will not be deleted upon completion, otherwise they will be deleted.
3. If -silent command is present, all phases will be executed in 'silent' (background) mode, otherwise, a progress dialog will display the current progress updates.
4. If –modal_oas command is present, the output OASIS file have modality, otherwise it will not.
5. The data grid of the output file will be in accordance with the –grid_addr command or 0.001 by default.
6. If -tilesz is specified, the data to be translated is chopped into 'tiles'. Each of these tiles is repesented in the output file as a unique sub-structure
7. Other options such as –top2bottom_cells and various transformations are used only if tiling is enabled.
8. If –eng_threads command specifies a number that is > 0 and <= 16, the Gbr2OasFrac Engine will use that many number of threads for flattening Gerber data during phase 1 of translation.
9. Unless –other_args is present, all phases will be executed with default settings. Please refer to _sGBR2OASFRACOPTIONS or consult Artwork for information about these 'advanced' settings.
10. Each call to this function is self-contained and independent of other calls. Each call would create new intermediate files and output.
11. This function differs from NGbr2OasFrac_convertGBRToFracturedOAS only in its interface.

See Also:

NGbr2OasFrac_initializeOnce
_sGBR2OASFRACOPTIONS
_sGBR2OASFRACERROR
NGbr2OasFrac_convertGBRToFracturedOAS
Usage & Flow

Sample Code:

```
void myConvertGBRToFracturedOAS()
{
  ...

  _sGBR2OASFRACERROR libError;
  void* libHandle = NGbr2OasFrac_initializeOnce(execPath, "My Gbr2OasFrac Client", &libError);
  if(libHandle == NULL)
  {
    myReportError(libError.mMsg);
    return;
  }

  const char* cmdString = "\"-i:C:\Gerber Files\input.gbr\" \"-o:C:\Temp Dir\output.oas\" \
                          -grid_addr:0.001 \"-workdir:C:\Temp Dir\" \
                          -modal_oas -keep -silent";
  success = NGbr2OasFrac_convertGBRToFracturedOAS_byCmd(libHandle, cmdString, &libError);

  if(success == 0)
  {
    myReportError(libError.mMsg);
  }

 ...
  NGbr2OasFrac_closeOnce(&libHandle);
 ...

}
```

Sequence of calls to IGbr2OasFrac Member Functions :-

1. IGbr2OasFrac::initializeOnce
2. IGbr2OasFrac::setProgressCallbackFn
3. IGbr2OasFrac::setCallbackHandler
4. IGbr2OasFrac::convertGBRToFracturesOAS (by parameter values)
5. IGbr2OasFrac::convertGBRToFracturesOAS (by command string)

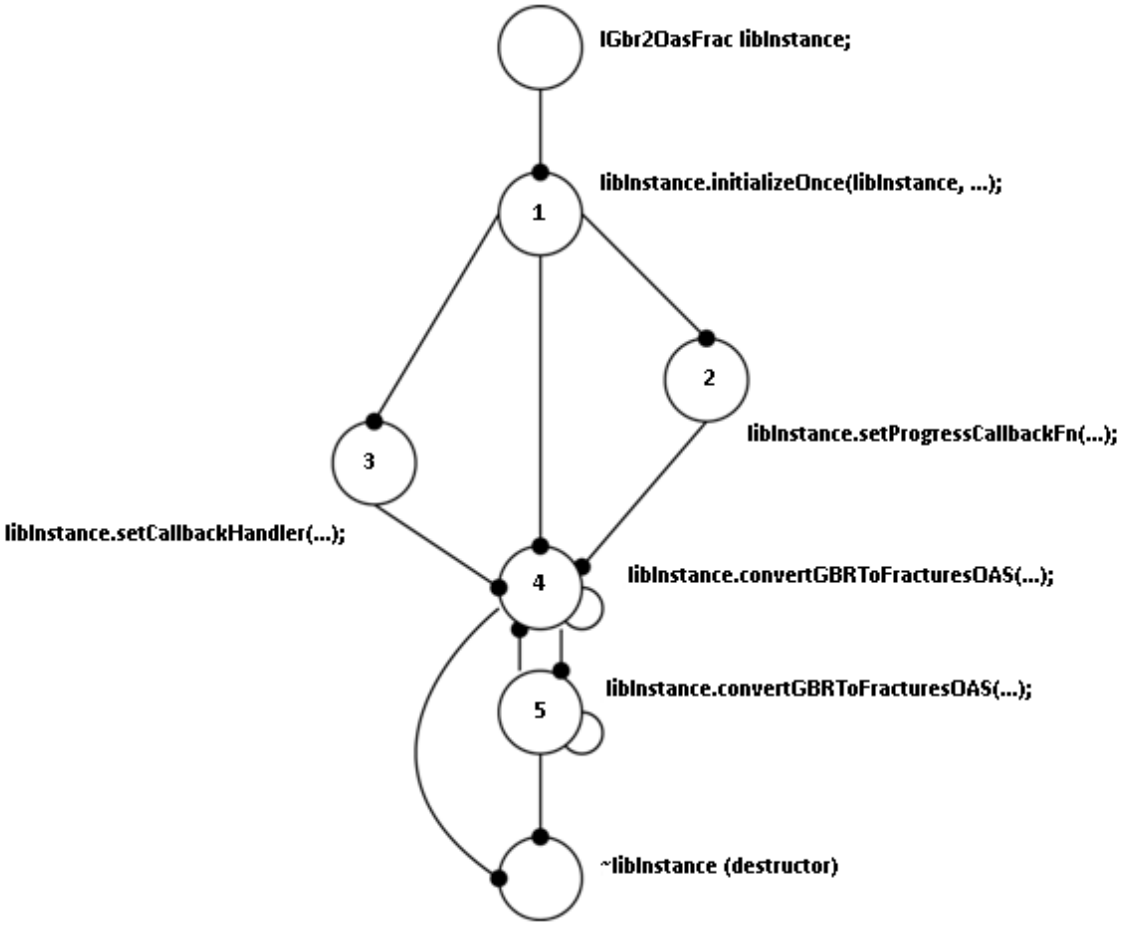

Figure 2: C++ API Flow

Objective:

To store optional arguments to be used by Gbr2OasFrac during execution.

Members:

mModalOAS (bool)	true: Generate modal OASIS output. false: Generate regular OASIS output. (default)
mKeepTempFiles (bool)	true: Retain temporary and intermediate files after execution. false: Remove temporary and intermediate files after execution. (default)
mNoProgress (bool)	true: Do not launch progress dialog. false: Launch progress dialog. (default)
mArcRes (double)	Specify arc resolution in degrees. (default: 9.0)
mArcSag (double)	Specify chord error in Gerber input units. (default: 0.0)
mNewGrid (double)	Specify the grid of output data. (default: 0.001)
mWorkDir (const char*)	Specify working directory. (default: current working directory)
mEngThreads (unsigned char)	Specify the number of threads to be used by the Gbr2OasFrac Engine for flattening the Gerber data. (default 1)
mSortXY (bool)	true: Sort the polygons in the output by their position (Y first, then X) false: Process the polygons as they are read from the input file.
mTiles::mEnable (unsigned char)	1: Enable tiling. Use mTiles::mSize[0..1] to specify tile size. 0: Disable tiling. mTiles::mSize and mCellsTop2Bottom will be ignored.
mTiles::mSize (double[2])	If tiling is enabled, specify the size (in user units) for each tile.
mCellsTop2Bottom (bool)	1: Write cell (representing each tile) definitions in the order of the position of the tile (represented by that cell) from Top to Bottom. 0: Do not order cell definitions.
mTiles::mMirror (unsigned char)	Refer to _sGBR2OASFRACOPTIONS::mTiles::mMirror
mTiles::mAngle (unsigned char)	Refer to _sGBR2OASFRACOPTIONS::mTiles::mAngle
mTiles::mScale (double)	Refer to _sGBR2OASFRACOPTIONS::mTiles::mScale
mTiles::mTranslate (double[2])	Refer to _sGBR2OASFRACOPTIONS::mTiles::mTranslate
mTiles::mTranslatePolicy (unsigned char)	Refer to _sGBR2OASFRACOPTIONS::mTiles::mTranslatePolicy
mEngineArgs (std::string)	Specify additional options for the Gbr2OasFrac engines.

Methods:

Copy constructor(const _sGBR2OASFRACOPTIONS*)	Copies data from a C _sGBR2OASFRACOPTIONS data object
Assignment operator(const _sGBR2OASFRACOPTIONS*)	Assigns data from a C _sGBR2OASFRACOPTIONS data object

Notes:

1. *mEngineArgs is an advanced option that is currently provided only as a placeholder for future enhancements and fine-tuning the performance. Please consult with Artwork for more information about this feature*.

See Also:

IGbr2OasFrac::convertGBRToFracturedOAS (by values)

IGbr2OasFrac::convertGBRToFracturedOAS_(by command)

Index > API Reference > C++ API > **Classes** > **IGbr2OasFracClient**

Objective:

To define the Gbr2OasFrac Library client (callback).

Methods:

onProgressUpdate	A callback for receiving progress updates from the Gbr2OasFrac Library.

Notes:

1. This class serves as an interface for providing notifications from the Gbr2OasFrac Library.
2. In order to receive notifications, the client program must derive a class (notification handler) from the IGbr2OasFracClient and implement it's methods.
3. In order to set the notification handler, call IGbr2OasFrac::setCallbackHandler method before making any calls to any of the two IGbr2OasFrac::convertGBRToFracturedOAS.
4. In order to stop receiving notifications from subsequent executions, call IGbr2OasFrac::setCallbackHandler(NULL).

See Also:

IGbr2OasFrac::setCallbackHandler
IGbr2OasFrac::convertGBRToFracturedOAS (by values)
IGbr2OasFrac::convertGBRToFracturedOAS (by command)

Index > API Reference > C++ API > Classes > IGbr2OasFracClient > **onProgressUpdate**

```
int IGbr2OasFracClient::onProgressUpdate(int percentDone, const char* msg);
```

Objective:

To provide a callback method to receive progress updates from the Gbr2OasFrac Library.

Parameters:

percentDone (int)	Report percentage of total execution complete.
msg (const char*)	Report current phase of execution.

Return:

any integer: inform the Gbr2OasFrac Library to continue execution.
-1: inform the Gbr2OasFrac Library to stop execution and return with code CGTFOErr_USERINTR.

Notes:

1. In order to receive notifications, the client program must derive a class (notification handler) from the IGbr2OasFracClient and implement it's methods.
2. In order to set the notification handler, call IGbr2OasFrac::setCallbackHandler method before making any calls to any of the two IGbr2OasFrac::convertGBRToFracturedOAS.
3. If the program wants to interrupt the execution at this point, this callback method must return -1. Otherwise, Gbr2OasFrac will continue with the translation.

See Also:

IGbr2OasFrac::setCallbackHandler
IGbr2OasFrac::convertGBRToFracturedOAS (by values)
IGbr2OasFrac::convertGBRToFracturedOAS (by command)

Sample Code:

```
class myGbr2OasFracClient: public IGbr2OasFracClient
{
 public:
   int onProgressUpdate(int percentDone, const char* msg)
   {
     printf("%s\n",msg);
     if(::checkStopTanslation())
       return -1; //inform library to stop execution
     else
       return 0; //inform library to continue execution
   }
};

void myGBR2OASFracTranslator::translate(...)
{
  IGbr2OasFrac libObj;
  if(! IGbr2OasFrac::initializeOnce(libObj, ...))
  {
    this->reportError(...);
    return;
  }
  ...
  myGbr2OasFracClient cbHandler;

  //receive progress updates ..
```

```
  libObj.setCallbackHandler(dynamic_cast< IGbr2OasFracClient*>(&cbHandler));

  libObj.convertGBRToFracturedOAS(...);

  //stop receiving progress updates ..
  libObj.setCallbackHandler(NULL);

  libObj.convertGBRToFracturedOAS(...);

  ...
}
```

Index > API Reference > C++ API > Classes > **IGbr2OasFrac**

Objective:

To define the interface class to the Gbr2OasFrac Library.

Methods:

initializeOnce	Initialize an instance of the Gbr2OasFrac Library and allocate necessary resources.
setProgressCallbackFn	Set a “C” callback function for receiving progress updates.
setCallbackHandler	Set a C++ object to handle notifications (callbacks).
convertGBRToFracturedOAS	Create a fractured Oasis file from an input Gbrii file by specifying execution parameters as values.
convertGBRToFracturedOAS	Create a fractured Oasis file from an input Gbrii file by specifying execution parameters as a command string.

Notes:

1. This class defines the interface for accessing the Gbr2OasFrac Library.
2. In order to access the library features the client program must create an instance of this class.
3. Every such instance must be first initialized by calling the initializeOnce static member function. Once this is done, the instance a.k.a library object (libObj) is ready for use.
4. The client program can then be accessed by calling one or more member functions using the libObj instance.

See Also:

C++ Usage & Flow

Index > API Reference > C++ API > Classes > IGbr2OasFrac > **initializeOnce**

```
static bool initializeOnce(IGbr2OasFrac& obj, const char* execPath, const char* appName,
                           _sGBR2OASFRACERROR* error = 0,
                           const char* verStr = GBR2OASFRAC_API_VER);
```

Objective:

To initialize an instance of the Gbr2OasFrac library interface class (IGbr2OasFrac).

Parameters:

obj (IGbr2OAsFrac&)	Instance of the IGbr2OasFrac class to be initialized.
execPath (const char*)	Complete path of the program using the Gbr2OasFrac library.
appName (const char*)	Name of the application using the Gbr2OasFrac Library. This name will be used to display the title of the Gbr2OasFrac progress dialog.
error (_sGBR2OASFRACERROR*)	Address of an object to get error information if this function fails. (can be set to NULL if the error information is not required).
verStr (const char*)	Passes the API version string to ensure that the client program is using the same header file as the Gbr2OasFrac Library. This parameter should not be used and left as it is.

Return:

true: success
false: failure. If errorO is not NULL, errorO->mCode will have one of the following values:-
IOErr_MEMALLOC (Memory allocation failure)
IOErr_INVPROGDIR (execPath is empty or NULL)
IOErr_VERSION (The program is using a different API header file than the one the library was built with)
IOErr_LICENSE (License failure)

Notes:

1. The 'obj' used by this method represents ONE INSTANCE of the Gbr2OasFrac library.
2. Every such instance of the Gbr2OasFrac Library must be initialized using this method before use.
3. Multiple instances of the library can be active in parallel by instantiating multiple instances of the IGbr2OasFrac class and calling this method once for each such instance.
4. The IGbr2OasFrac instance and any of the other IGbr2OasFrac methods cannot be used without calling this method once at the beginning.
5. This method must be called only ONCE for each instance of the library. All resources associated with this instance will be released when the instance goes out of scope or is deleted.
6. The library and its dependent binaries must be present in the same directory as the program using the library.

See Also:

C++ Usage & Flow
IGbr2OasFrac
_sGBR2OASFRACERROR

Sample Code:

```
void myGBR2OASFracTranslator::translate(...)
{
  IGbr2OasFrac libObj;
  _sGBR2OASFRACERROR libError;
```

```
  if(! IGbr2OasFrac::initializeOnce(libObj, execPath, "My Gbr2OasFrac Client", &libError))
  {
    this->reportError(libError.mMsg);
    return;
  }
  ...

  myGbr2OasFracClient cbHandler;

  libObj.convertGBRToFracturedOAS(...);

  ...
}
```

Index > API Reference > C++ API > Classes > IGbr2OasFrac > **setProgressCallbackFn**

```
void IGbr2OasFrac::setProgressCallbackFn(NGbr2OasFracProgressCbFnPtr callbackFnPtr);
```

Objective:

To specify a callback function to get progress updates from the Gbr2OasFrac library as the input GBR file is being translated to fractured OASIS.

Parameters:

callbackFnPtr (NGbr2OasFracProgressCbFnPtr)	Pointer/Name of the callback function to be called when the Gbr2OasFrac library has progress updates.

Return:

None.

Notes:

1. If the client program wishes to receive progress updates while the fractured OASIS output is being generated, this method must be called to set the callback function.
2. This function must be called before either of the convertGBRToFracturedOAS methods.
3. If progress updates are not desired for subsequent executions, pass NULL for 'callbackFnPtr'.
4. The callback function must have the following signature:-
   int GBR2OASFRAC_CALLDECL functionName(int percentDone, const char *msg);
5. The callback function if set, will be called everytime the Gbr2OasFrac library has progress updates.
6. 'percentDone' parameter reports the percentage of execution complete and 'msg' parameter reports the execution phase.
7. If the program wants to interrupt the execution at this point, this callback function must return -1. Otherwise, Gbr2OasFrac will continue with the translation.

See Also:

C++ Usage & Flow
IGbr2OasFrac::convertGBRToFracturedOAS
NGbr2OasFracProgressCbFnPtr

Sample Code:

```
int GBR2OASFRAC_CALLDECL onProgressUpdate(int percent, const char* updateMsg)
{
  myUpdateProgressDialog(updateMsg, percent);
  if(checkStopTranslation() == 1)
    return -1; //return -1 to inform the library to stop translation.
  else
    return 0;
}

void myGBR2OASFracTranslator::translate(...)
{
  IGbr2OasFrac libObj;
  if(! IGbr2OasFrac::initializeOnce(libObj, ...))
  {
    this->reportError(...);
    return;
  }
  ...
```

```
  //receive progress updates ..
  libObj.setProgressCallbackFn(onProgressUpdate);

  libObj.convertGBRToFracturedOAS(...);

  //stop receiving progress updates ..
  libObj.setProgressCallbackFn(NULL);

  libObj.convertGBRToFracturedOAS(...);

  ...
}
```

Index > API Reference > C++ API > Classes > IGbr2OasFrac > **setCallbackHandler**

```
void IGbr2OasFrac::setCallbackHandler(IGbr2OasFracClient* cbHandler);
```

Objective:

To specify a callback function to get progress updates from the Gbr2OasFrac library as the input GBR file is being translated to fractured OASIS.

Parameters:

cbHandler (IGbr2OasFracClient*)	Pointer/Name of the callback object to be used to handle the Gbr2OasFrac library notifications.

Return:

None.

Notes:

1. If the client program wishes to receive notifications such as progress updates while the fractured OASIS output is being generated, this method must be called to set the notification handler.
2. This function must be called before either of the convertGBRToFracturedOAS methods.
3. If progress updates are not desired for subsequent executions, pass NULL for 'cbHandler'.
4. The cbHandler must be an object of a class derived from the IGbr2OasFracClient class.

See Also:

C++ Usage & Flow
IGbr2OasFrac::convertGBRToFracturedOAS
IGbr2OasFracClient

Sample Code:

```
//Refer to the IGbr2OasFracClient::onProgressUpdate sample code.
```

Index > API Reference > C++ API > Classes > IGbr2OasFrac > **convertGBRToFracturedOAS (by values)**

```
bool IGbr2OasFrac::convertGBRToFracturedOAS(const char* inputGBRFile,
                                            const char* outputOASFile,
                                            const sGBR2OASFRACOPTIONS& options,
                                            _sGBR2OASFRACERROR* error = 0);
```

Objective:

To generate a fractured OASIS file from a Gerber file.

Parameters:

inputGBRFile (const char*)	Path to a valid Gerber file to be translated.
outputOASFile (const char*)	Location of the output OASIS file that will be created when this function returns successfully.
options (const sGBR2OASFRACOPTIONS&)	A reference to a set of options to control the output and execution of the translation.
errorO (_sGBR2OASFRACERROR*)	Address of an object to get error information if this function fails. (can be set to NULL if the error information is not required).

Return:

true: success
false: failure. If errorO is not NULL, errorO->mCode will have one of the following values:-
CGTFOErr_MEMALLOC (Memory allocation failure)
CGTFOErr_EXECPHASE1 (Phase 1: Unionizing Gerber Data into GDSII, failed)
CGTFOErr_EXECPHASE2 (Phase 2: Converting GDSII into Fractured OASIS, failed)
CGTFOErr_LICENSE (License error)
CGTFOErr_USERINTR (User hit cancel while execution was in progress)

errorO->mMsg would point to a string with details about the error.

Notes:

1. This method converts an input Gerber file to a fractured OASIS file in 2 phases; unionizing Gerber Data into GDSII and converting GDSII into Fractured OASIS.
2. A few intermediate files are created in the 'working directory' (specified by the options.mWorkDir member or the default working directory). If options.mKeep is set, these intermediate files will not be deleted upon completion, otherwise they will be deleted.
3. If options.mNoProgress is set, all phases will be executed in 'silent' (background) mode, otherwise, a progress dialog will display the current progress updates.
4. If options.mModalOAS is set, the output OASIS file have modality, otherwise it will not.
5. The data grid of the output file will be in accordance with the options.mNewGrid member or 0.001 by default.
6. If options.mTiles.mEnable is set (to 1), the data to be translated is chopped into 'tiles' of size options.mTiles.mSize. Each of these tiles is repesented in the output file as a unique sub-structure
7. Other options such as options.mCellsTop2Bottom and various transformations are used only if tiling is enabled.
8. If options.mEngThreads is > 0 and <= 16, the Gbr2OasFrac Engine will use that many number of threads for falttening Gerber data during phase 1 of translation.
9. Unless options.mEngineArgs is set, all phases will be executed with default settings. Please refer to sGBR2OASFRACOPTIONS or consult Artwork for information about these 'advanced' settings.
10. Each call to this function is self-contained and independent of other calls. Each call would create new intermediate files and output.

11. This method is overloaded by the convertGBRToFracturedOAS method to support a command string based interface.

See Also:

C++ Usage & Flow
_sGBR2OASFRACERROR
sGBR2OASFRACOPTIONS
convertGBRToFracturedOAS

Sample Code:

```
void myGBR2OASFracTranslator::translate(...)
{
  IGbr2OasFrac libObj;
  _sGBR2OASFRACERROR libError;
  if(! IGbr2OasFrac::initializeOnce(libObj, execPath, "My Gbr2OasFrac Client", &libError))
  {
    this->reportError(libError.mMsg);
    return;
  }
  ...

  sGBR2OASFRACOPTIONS options;

  if(! libObj.convertGBRToFracturedOAS(myInputFile, myOutputFile, options, &libError);
  {
    this->reportError(libError.mMsg);
  }
  ...
}
```

Index > API Reference > C++ API > Classes > IGbr2OasFrac > **convertGBRToFracturedOAS** (by command)

```
bool IGbr2OasFrac::convertGBRToFracturedOAS(const char* cmdString,
                                            _sGBR2OASFRACERROR* error = 0);
```

Objective:

To generate a fractured OASIS file from a Gerber file.

Parameters:

cmdString (const char*)	Translation parameters specified as a command string.
errorO (_sGBR2OASFRACERROR*)	Address of an object to get error information if this function fails. (can be set to NULL if the error information is not required).

Command String Syntax: (All arguments containing 'space' must be enclosed in 'double quotes')

```
SYNTAX:-
cmdString := "<inputFile> <outputFile> <options>"

inputFile := "-i:<input Gerber file path>"
outputFile := "-o:<output OASIS file path>"
options := "[<arcRes>] [<arcSag>] [<newGrid>] [<workDir>] [<modalOAS>] [<keep>] [<silent>]
              [<otherArgs>] [<threads>] [<sortxy>] [<tile> [<top2bottom>]
              [<transformation>]]"

arcRes := "-arcres:<angle in degrees>
arcSag := "-arcsag:<chord error in Gerber input units>
newGrid := "-grid_addr:<output data grid>"
workDir := "-workdir:<path to working dir>"
modalOAS := "-modal_oas"
keep := "-keep"
silent := "-silent"
threads := "-eng_threads:<1-16>"
otherArgs := "-other_args:<arg list>"
tile := "-tilesz:<size_in_x>,<size_in_y>"
top2bottom := "-top2bottom_cells"
sortxy := "-sort_xy"
transformation := [<mirror>] [<rotation>] [<translation>] [<scaling>]
mirror := "-mirror:x" | "-mirror:y"
rotation := "-angle:90" | "-angle:180" | "-angle:270"
translation := "-translate:auto" | "-translate:<dx>,<dy>"
scaling := "-scale:<scale-factor>"



SAMPLE:-
const char* cmdString = "\"-i:C:\Gerber Files\input.gbr\" \"-o:C:\Temp Dir\output.oas\"
                        -grid_addr:0.001 \"-workdir:C:\Temp Dir\" -modal_oas -keep -silent
                        -arcres:4.0 -eng_threads:2 -tilesz:1000,1000 -top2bottom_cells
                        -sort_xy" -mirror:x -angle:270 -translate:100,150 -scale:2.0";



CO-RELATION WITH sGBR2OASFRACOPTIONS (options)
-arcres ⇔ options.mArcRes
-arcsag ⇔ options.mArcSag
-grid_addr ⇔ options.mNewGrid
```

```
-workdir ⇔ options.mWorkDir
-modal_oas ⇔ options.mModalOAS
-keep ⇔ options.mKeepTempFiles
-silent ⇔ options.mNoProgress
-other_args ⇔ options.mEngineArgs
-eng_threads ⇔ options.mEngThreads
-tilesz ⇔ options.mTiles.mSize, options.mTiles.mEnable = 1
-top2bottom_cells ⇔ options.mCellsTop2Bottom
-sort_xy ⇔ options.mSortByXY
-mirror ⇔ options.mTiles.mMirror
-angle ⇔ options.mTiles.mAngle
-translate ⇔ options.mTiles.mTranslatePolicy + options.mTiles.mTranslate
-scale ⇔ options.mTiles.mScale
```

Return:

true: success
false: failure. If errorO is not NULL, errorO->mCode will have one of the following values:-
CGTFOErr_MEMALLOC (Memory allocation failure)
CGTFOErr_EXECPHASE1 (Phase 1 Unionizing Gerber Data into GDSII, failed)
CGTFOErr_EXECPHASE2 (Phase 2: Converting GDSII into Fractured OASIS, failed)
CGTFOErr_LICENSE (License error)
CGTFOErr_USERINTR (User hit cancel while execution was in progress)
CGTFOCErr_INVALIDCOMMAND (Empty or NULL cmdString)
CGTFOCErr_PARSEINVQUOTES (Double Quotes Mismatch in cmdString)
CGTFOCErr_PARSEINVINPUT (-i: command absent, no input file specified)
CGTFOCErr_PARSEINVOUTPUT (-o: command absent, no output file specified)

errorO->mMsg would point to a string with details about the error.

Notes:

1. This method converts an input Gerber file to a fractured OASIS file in 2 phases; unionizing Gerber Data into GDSII and converting GDSII into Fractured OASIS.
2. A few intermediate files are created in the 'working directory' (specified by the –workdir command or the default working directory). If -keep command is present, these intermediate files will not be deleted upon completion, otherwise they will be deleted.
3. If -silent command is present, all phases will be executed in 'silent' (background) mode, otherwise, a progress dialog will display the current progress updates.
4. If –modal_oas command is present, the output OASIS file have modality, otherwise it will not.
5. The data grid of the output file will be in accordance with the –grid_addr command or 0.001 by default.
6. If -tilesz is specified, the data to be translated is chopped into 'tiles'. Each of these tiles is repesented in the output file as a unique sub-structure
7. Other options such as –top2bottom_cells and various transformations are used only if tiling is enabled.
8. If –eng_threads command specifies a number that is > 0 and <= 16, the Gbr2OasFrac Engine will use that many number of threads for falttening Gerber data during phase 1 of translation.
9. Unless –other_args is present, all phases will be executed with default settings. Please refer to sGBR2OASFRACOPTIONS or consult Artwork for information about these 'advanced' settings.
10. Each call to this function is self-contained and independent of other calls. Each call would create new intermediate files and output.
11. This method is overloaded by the convertGBRToFracturedOAS method to support a command string based interface.

See Also:

C++ Usage & Flow
_sGBR2OASFRACERROR
convertGBRToFracturedOAS
IGbr2OasFrac

Sample Code:

```
void myGBR2OASFracTranslator::translate(...)
{
  IGbr2OasFrac libObj;
  _sGBR2OASFRACERROR libError;
  if(! IGbr2OasFrac::initializeOnce(libObj, execPath, “My Gbr2OasFrac Client”, &libError))
  {
    this->reportError(libError.mMsg);
    return;
  }
  ...

  const char* cmdString = “\“-i:C:\Gerber Files\input.gbr\” \“-o:C:\Temp Dir\output.oas\” \
                          -grid_addr:0.001 \”-workdir:C:\Temp Dir\” \
                          -modal_oas -keep -silent”;

  if(! libObj.convertGBRToFracturedOAS(cmdString, &libError);
  {
    this->reportError(libError.mMsg);
  }
  ...
}
```

Index > **Version History**

v1.0 (Apr 06, 2011) API Version: v1.0 (Apr 06, 2011)

1. First Release.