Artwork Conversion Software, Inc.

WMLib and WMView API

July 6, 2021

Purpose

This document describes the functionality of the WMLib and WMView API which can be licensed for customers to use in their own C++ and .NET applications.

WMLib

ACS has developed WMLib for opening, transforming, editing, converting, and saving of wafer map files. The API defines a set of core functions and more functions will be added incrementally as needed. The library comes in a version for C++ (WMClib) and .NET (WMNLib).

WMView

ACS has developed WMView which permits viewing of wafer map data loaded into WMLib. The viewer is lightweight, but with built-in functionality to configure color and display attributes and includes interactive selection and highlighting of wafer map elements. The library comes in a version for C++ (WMCView) and .NET (WMNView).

Frameworks

All libraries are built using Visual Studio 2015. The .NET libraries, WMNLib and WMNView, target version 4.7 of the .NET framework. All libraries are defined within an ACS namespace which is omitted for the purposes of making the documentation more clear and concise.

WMLib API

createWMCLib/createWMNLib

Create an instance of the wafer map library. The instanceFile and instanceData parameters are for future use and should be passed empty strings. The status string will contain information if the function fails and returns a null pointer.

```
C++
ACS_WMCLIB_DECLSPEC WMCLib* createWMCLib(const char* instanceFile,
                                         const char* instanceData,
                                         char* status,
                                         int statusLength);
C#
public static WMNLib createWMNLib(string instanceFile,
                                  string instanceData,
                                  ref string status);

VB
Public Shared Function createWMNLib(instanceFile As String,
                                    instanceData As String,
                                    ByRef status As String) As WMNLib
```

destroyWMCLib

Destroys the library object returned from **createWMCLib**. This function is only needed in a C++ client, which should call the function when finished with the library, to avoid memory leaks.

```
C++
ACS_WMCLIB_DECLSPEC void destroyWMCLib(WMCLib* wmLib);
```

initLib

Initializes an instance of WMLib with the directory to the WMLib (and WMView) configuration, the output directory for file operations, and the path to a file where information will be logged. This function should be called immediately after creating the WMLib object.

```
C++
void initLib(const char* configDirectory,
             const char* outputDirectory,
             const char* logFile);

C#
public void initLib(string configDirectory,
                    string outputDirectory,
                    string logFile);

VB
Public Sub initLib(configDirectory As String,
                   outputDirectory As String,
                   logFile As String)
```

getVersionInfo

Return version information about WMLib.

C++
```
const char* getVersionInfo() const;
```

C#
```
public string getVersionInfo();
```

VB
```
Public Function getVersionInfo() As String
```

getConfigDirectory

Return the library configuration directory.

C++
```
const char* getConfigDirectory() const;
```

C#
```
public string getConfigDirectory();
```

VB
```
Public Function getConfigDirectory() As String
```

getOutputDirectory

Return the library output directory.

C++
```
const char* getOutputDirectory() const;
```

C#
```
public string getOutputDirectory();
```

VB
```
Public Function getOutputDirectory() As String
```

setConfigDirectory

Sets the library configuration directory. The function will attempt to create the directory if needed and the return value will indicate if the directory exists at the end of the function call.

C++
```
bool setConfigDirectory(const char* directory);
```

C#
```
public bool setConfigDirectory(string directory);
```

VB
```
Public Function setConfigDirectory(directory As String) As Boolean
```

setOutputDirectory

Sets the library output directory. The function will attempt to create the directory if needed and the return value will indicate if the directory exists at the end of the function call.

C++
```
bool setOutputDirectory(const char* directory);
```

C#
```
public bool setOutputDirectory(string directory);
```

VB
```
Public Function setOutputDirectory(directory As String) As Boolean
```

setLogFile

Sets the library log file.

C++
```
bool setLogFile(const char* fileName,
                bool clear = true)
```

C#
```
public bool setLogFile(string fileName,
                       bool clear);
```

VB
```
Public Function setLogFile(fileName As String,
                           clear As Boolean) As Boolean
```

getSupportedFormatCount

Returns the number of wafer map formats supported by WMLib. This function is needed in C++ to initialize an array large enough to return all formats.

C++
```
int getSupportedFormatCount() const;
```

C#
```
public int getSupportedFormatCount();
```

VB
```
Public Function getSupportedFormatCount() as Integer
```

getSupportedFormats

Populates the provided array or list with the name of each supported wafer map format as defined in the configuration directory. The return value is the number of strings included in the list.

```
C++
int getSupportedFormats(char** formatNames,
                        int formatLength,
                        int formatCount) const;

C#
public getSupportedFormats(List<string> formatNames);

VB
Public Function getSupportedFormats(formatNames As List(Of String)) As Integer
```

isFormatSupported

Checks if the wafer map format named in the provided string is supported by the library. The return value is true if supported, otherwise false.

```
C++
bool isFormatSupported(const char* format) const;

C#
public bool isFormatSupported(string format);

VB
Public Function isFormatSupported(format As String) As Boolean
```

getLicenseStatus

Retrieves a string representation of the licensing status of each parser and writer for every available wafer map format.

```
C++
void getLicenseStatus(char* status,
                      int statusLength) const;

C#
public void getLicenseStatus(ref string status);

VB
Public Sub getLicenseStatus(ByRef status As String)
```

isFormatLicensed

Checks if the wafer map format named in the provided string is licensed. The two boolean references are used to check for both an input and output license. If a license is available, the bookean will be true, otherwise false.

```
C++
void isFormatLicensed(const char* format,
                      bool& inputLicense,
                      bool& outputLicense) const;

C#
public void isFormatLicensed(string format,
                             ref bool inputLicense,
                             ref bool outputLicense);

VB
Public Sub isFormatLicensed(format As String,
                            ByRef inputLicense As Boolean,
                            ByRef outputLicense As Boolean)
```

getFormatFileExtensions

Retrieves the default input file extensions (which could be more than one) and the default output file extension for the provided wafer map format. If the library is unable to find a format with this name in the configuration the return value will be **StatusFileError**, otherwise **StatusOK**.

```
C++
int getFormatFileExtensions(const char* format,
                            char* inputExtensions,
                            char* outputExtension,
                            int extensionLength) const;

C#
public int getFormatFileExtensions(string format,
                                   ref string inputExtensions,
                                   ref string outputExtension);

VB
Public Function getFormatFileExtensions(format As String,
                                        ByRef inputExtensions As String,
                                        ByRef outputExtension As String) As Integer
```

getOpenFileFilter

Returns a string describing all supported wafer map formats and their input file extensions. This string is suitable for use with the system open file dialog. The order of each wafer map type corresponds to the list in the **getSupportedFormats** function. This allows client applications to efficiently prompt users for both file name and wafer map type. The return value is the number of supported file formats.

```
C++
int getOpenFileFilter(char* filter,
                      int filterLength) const;

C#
public int getOpenFileFilter(ref string filter);

VB
Public Function getOpenFileFilter(ByRef filter As String) As Integer
```

getSaveFileFilter

Returns a string describing all supported wafer map formats and their output file extensions. This string is suitable for use with the system save file dialog. The order of each wafer map type corresponds to the list in the **getSupportedFormats** function. This allows client applications to efficiently prompt users for both file name and wafer map type. The return value is the number of supported file formats.

```
C++
int getSaveFileFilter(char* filter,
                      int filterLength) const;

C#
public int getSaveFileFilter(ref string filter);

VB
Public Function getSaveFileFilter(ByRef filter As String) As Integer
```

loadWaferMap

Load the wafer map (or maps) in the provided file, which is of type format, into the library. The return value represents the status of loading the file and format. The status will be **StatusOK** if successful, or an error code. See the error code documentation for further details.

```
C++
int loadWaferMap(const char* fileName,
                 const char* format);

C#
public int loadWaferMap(string fileName,
                        string format);

VB
Public Function loadWaferMap(fileName As String,
                             format As String) As Integer
```

getInputFormat

Retrieve the wafer map format of the input database. The return value will be **StatusOK** if successful, or an error code

```
C++
int getInputFormat(char* format,
                   int formatLength) const;

C#
public int getInputFormat(ref string format);

VB
Public Function getInputFormat(ByRef format As String) As Integer
```

getOutputFormat

Retrieve the wafer map format of the output database. The return value will be **StatusOK** if successful, or an error code.

```
C++
int getOutputFormat(char* format,
                    int formatLength) const;

C#
public int getOutputFormat(ref string format);

VB
Public Function getOutputFormat(ByRef format As String) As Integer
```

setOutputFormat

Sets the wafer map format of the output database. This initiates a conversion of the output database from its current wafer map format to the new wafer map format. Information related to bins, header values, and other formatting may be changed, or lost, if the format being converted to does not support those options. For wafer map formats that support multiple bin formats, this parameter can also be passed to ensure a specific output. The return value will be **StatusOK** if successful, or an error code.

While a license is not required to convert to a different wafer map format, a license is required to save the data to a file of that format.

```
C++
int setOutputFormat(const char* format,
                    const char* binFormat = "");

C#
public int setOutputFormat(string format,
                           string binFormat);

VB
Public Function setOutputFormat(format As String,
                                binFormat As String) As Integer
```

getOutputExtension

Retrieve the output file extension of the wafer map format in the output database. The return value will be **StatusOK** if successful, or an error code

```
C++
int getOutputExtension(char* extension,
                       int extensionLength) const;

C#
public int getOutputExtension(ref string extension);

VB
Public Function getOutputExtension(ByRef extension As String) As Integer
```

saveWaferMap

Save the output database to a wafer map file. If the database contains multiple wafer maps in a format that does not support multiple wafer maps, multiple files will be saved with a decoration at the end of the provided file name. The return value will be **StatusOK** if succesful, or an error code. A separate license is required to save each wafer map format.

```
C++
int saveWaferMap(const char* fileName);

C#
public int saveWaferMap(string fileName);

VB
Public Function saveWaferMap(fileName As String) As Integer
```

newDB

Create a new input and output database with the given wafer size and wafer map format. Any existing input and output databases will be removed from the library. It is the responsibility of the client application to check whether the output database is modified and could need saving before calling this function. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int newDB(const char* format,
          WMCLib::XY waferSize);

C#
public int newDB(string format,
                 XY waferSize);

VB
Public Function newDB(format As String,
                      waferSize As XY) As Integer
```

selectInputDB

Selects the input database as active for further API calls that query the database.

```
C++
void selectInputDB();

C#
public void selectInputDB();

VB
Public Sub selectInputDB()
```

selectOutputDB

Selects the output database as active for further API calls that query the database.

```
C++
void selectOutputDB();

C#
public void selectOutputDB();

VB
Public Sub selectOutputDB()
```

inputDBModified

Returns whether the input database has been modified (such as a new file being loaded) and resets the input modification state. This function is useful, for example, if an application using WMView needs to track when the input database contents have changed.

```
C++
bool inputDBModified();

C#
public bool inputDBModified();

VB
Public Function inputDBModified() As Boolean
```

outputDBModified

Returns whether the output database has been modified. This function can be used to check if file contents need to be saved before exiting.

```
C++
bool outputDBModified() const;

C#
public bool outputDBModified();

VB
Public Function outputDBModified() As Boolean
```

getWaferMapCount

Returns the number of wafer maps defined in the database. Some wafer map formats support defining multiple wafer maps in one file.

C++
```
int getWaferMapCount() const;
```

C#
```
public int getWaferMapCount();
```

VB
```
Public Function getWaferMapCount() As Integer
```

getWaferMapID

Returns the Wafer Map ID of the given wafer map in the database. The **index** parameter should be between zero and one less than the number of wafer maps returned from **getWafermapCount**. The return value will be **StatusOK** if succesful, or an error code.

C++
```
int getWaferMapID(int index,
                  char* waferID,
                  int waferIDLength) const;
```

C#
```
public int getWaferMapID(int index,
                         ref string waferID);
```

VB
```
Public Function getWaferMapID(index As Integer,
                              ByRef waferID As String) As Integer
```

setActiveWaferMap

Sets the active wafer map in the database. An index of -1 is used to set all wafer maps as active. All wafer specific API function calls made after this will affect the active wafer(s). The return value will be **StatusOK** if succesful, or an error code.

C++
```
int setActiveWaferMap(int index);
```

C#
```
public int setActiveWaferMap(int index);
```

VB
```
Public Function setActiveWaferMap(index As Integer) As Integer
```

rotateWaferMap

Rotates the wafer map the defined angle. A positive angle is counter-clockwise. This also changes device and step size, reference die parameters, flat location and angle, and the array of bins. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int rotateWaferMap(int angle);

C#
public int rotateWaferMap(int angle);

VB
Public Function rotateWaferMap(angle As Integer) As Integer
```

addWaferMap

Adds a new wafer map to the existing input and output databases. The wafer map will match the current wafer map format and wafer size of the input and output databases. The new wafer map will be a synthesized array using the provided device size and bin values. Devices which fall completely inside the wafer will be assigned the given **binValue** parameter. Those devices which fall completely outside the wafer will be assigned the default NULL bin. Those devices which straddle the wafer edge will be assigned the given **edgeBinValue** parameter.

```
C++
int addWaferMap(const char* waferID,
                WMCLib::Flat flat,
                WMCLib::XY deviceSize,
                const char* binFormat,
                const char* binValue,
                const char* edgeBinValue);

C#
public int addWaferMap(string waferID,
                       int flat,
                       XY deviceSize,
                       string binFormat,
                       string binValue,
                       string edgeBinValue);

VB
Public Function addWaferMap(waferID As String,
                            flat As Integer,
                            deviceSize As XY,
                            binFormat As String,
                            binValue As String,
                            edgeBinValue As String) As Integer
```

getWaferSize

Retrieve the size of the wafer, including the units, if defined. Only circular wafers are currently supported. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getWaferSize(WMCLib::XY& xy) const;

C#
public int getWaferSize(ref XY size);

VB
Public Function getWaferSize(ByRef size As XY) As Integer
```

setWaferSize

Sets the size of the wafer, including the units. Only circular wafers are currently supported. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int setWaferSize(WMCLib::XY xy);

C#
public int setWaferSize(XY size);

VB
Public Function setWaferSize(size As XY) As Integer
```

getOrigin

Retrieve the origin of the wafer map. The reference passed to the function will be one of the origin constants defined in the appendix. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getOriginValue(WMCLib::Origin& origin) const;

C#
public int getOriginValue(ref int origin);

VB
Public Function getOriginValue(ByRef origin As Integer) As Integer
```

getOriginValue

Retrieve the wafer map format specific origin value given one of the origin constants. The origin constants are defined in the appendix. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getOriginValue(WMCLib::Origin origin,
                   char* value,
                   int valueLength) const;

C#
public int getOriginValue(int origin,
                          ref string value);

VB
Public Function getOriginValue(origin As Integer,
                               ByRef value As String) As Integer
```

setOrigin

Sets the wafer map origin. The origin constants are defined in the appendix. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int setOrigin(WMCLib::Origin origin,
              char* value,
              int valueLength) const;

C#
public int setOrigin(int origin);

VB
Public Function setOrigin(origin As Integer) As Integer
```

getAxis

Retrieves the wafer map axis. The axis constants are defined in the appendix. Pass the **resolveOrigin** parameter as true to ensure that the retrieved axis is an absolute direction when the axis is the **WMLib::AxisOrigin** constant. This constant indciates the axis always faces towards the center of the wafer from the origin location. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getAxis(WMCLib::Axis& axis,
            bool resolveOrigin = true) const;

C#
public int getAxis(ref int axis,
                   bool resolveOrigin);

VB
Public Function getAxis(ByRef axis As Integer,
                        resolveOrigin As Boolean) As Integer
```

getFlat

Retrieves the wafer map flat side. The flat constants are defined in the appendix. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getFlat(WMCLib::Flat& flat,
            bool resolveOrigin = true) const;

C#
public int getFlat(ref int flat,
                   bool resolveOrigin);

VB
Public Function getFlat(ByRef flat As Integer,
                        resolveOrigin As Boolean) As Integer
```

getFlatAngle

Retrieves the wafer map specific label or angle for the given flat side. The flat constants are defined in the appendix. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getFlatAngle(WMCLib::Flat flat,
                 int& angle) const;

C#
public int getFlatAngle(int flat,
                        ref int angle);

VB
Public Function getFlatAngle(flat As Integer,
                             ByRef angle As Integer) As Integer
```

getDeviceSize

Retrieves the width, height and units of the device size. The device size and step size are treated as the same. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getDeviceSize(WMCLib::XY& xy) const;

C#
public int getDeviceSize(ref XY xy);

VB
Public Function getDeviceSize(ByRef xy As XY) As Integer
```

setDeviceSize

Sets the width, height and units of the device size. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int setDeviceSize(WMCLib::XY xy);

C#
public int setDeviceSize(XY xy);

VB
Public Function setDeviceSize(xy As XY) As Integer
```

getStepSize

Retrieves the width, height and units of the step size. The step size and device size are treated as the same. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getStepSize(WMCLib::XY& xy) const;

C#
public int getStepSize(ref XY xy);

VB
Public Function getStepSize(ByRef xy As XY) As Integer
```

setStepSize

Sets the width, height and units of the step size. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int setStepSize(WMCLib::XY xy);

C#
public int setStepSize(XY xy);

VB
Public Function setStepSize(xy As XY) As Integer
```

getBinFormatCount

Retrieve the number of bin formats supported by the database wafer map format. See the appendix for a description of the bin format structure. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getBinFormatCount(int& count) const;

C#
public int getBinFormatCount(ref int count);

VB
Public Function getBinFormatCount(ByRef count As Integer) As Integer
```

getBinFormat

Retrieve a bin format supported by the database wafer map format by index. The index should be between zero and one less than the number of supported bin formats. See the appendix for a description of the bin format structure. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getBinFormat(int index,
                 WMCLib::BinFormat& format) const;

C#
public int getBinFormat(int index,
                        ref BinFormat format);

VB
Public Function getBinFormat(index As Integer,
                             ByRef format As BinFormat) As Integer
```

getBinFormat

Retrieve the bin format currently in use by the database. See the append for a description of the bin format structure. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getBinFormat(WMCLib::BinFormat& format) const;

C#
public int getBinFormat(ref BinFormat format);

VB
Public Function getBinFormat(ByRef format As BinFormat) As Integer
```

setBinFormat

Set the bin format to one of those supported by the wafer map format by index. The index should be between zero and one less than the number of supported bin formats. See the appendix for a description of the bin format structure. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int setBinFormat(int index);

C#
public int setBinFormat(int index);

VB
Public Function setBinFormat(index As Integer) As Integer
```

getNullBinValue

Retrieves the value of the NULL bin in the current bin format. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getNullBinValue(char* value,
                    int length) const;

C#
public int getNullBinValue(ref string value);

VB
Public Function getNullBinValue(ByRef value As String) As Integer
```

getBinCount

Retrieves the number of unique bins in the wafer map. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getBinCount(int& count) const;

C#
public int getBinCount(ref int count);

VB
Public Function getBinCount(ByRef count As Integer) As Integer
```

getBinValue

Retrieves the value of a bin in the current bin format by index. The index should be between zero and one less than the number of bins. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getBinValue(int index,
                char* value,
                int length) const;

C#
public int getBinValue(int index,
                       ref string value);

VB
Public Function getBinValue(index As Integer,
                            ByRef value As String) As Integer
```

mapBin

Maps the input bin value to the output bin value. If the input bin value is found in the input database, the equivalent mapped bin in the output database will have its value changed. If the input value is found in the output database then that bin will have its value changed. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int mapBin(const char* input,
           const char* output);

C#
public int mapBin(string input,
                  string output);

VB
Public Function mapBin(input As String,
                       output As String) As Integer
```

getBinIndex

Retrieve the index of the given bin value. The index will be a value between zero and one less than the number of bins, or -1 if the bin value is not found in the wafer map. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getBinIndex(const char* value,
                int& index) const;

C#
public int getBinIndex(string value,
                       ref int index);

VB
Public Function getBinIndex(value As String,
                            ByRef index As Integer) As Integer
```

getBinDescription

Retrieve the description of the given bin index. The index should be between zero and one less than the number of bins. The bin description is an optional attribute found in some wafer map formats. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getBinDescription(int index,
                      char* description,
                      int length) const;

C#
public int getBinDescription(int index,
                             ref string description);

VB
Public Function getBinDescription(index As Integer,
                                  ByRef description As String) As Integer
```

setBinDescription

Sets the description of the given bin index. The index should be between zero and one less than the number of bins. The bin description is an optional attribute found in some wafer map formats. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int setBinDescription(int index,
                      const char* description);

C#
public int setBinDescription(int index,
                             string description);
VB
Public Function setBinDescription(index As Integer,
                                  description As String) As Integer
```

getBinQuality

Retrieves the quality of the given bin index. The index should be between zero and one less than the number of bins. See the appendix for a list of possible bin quality constants. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getBinQuality(int index,
                  WMCLib::BinQuality& quality) const;

C#
public int getBinQuality(int index,
                         ref int quality);

VB
Public Function getBinQuality(index As Integer,
                              ByRef quality As Integer) As Integer
```

setBinQuality

Sets the quality of the given bin index. The index should be between zero and one less than the number of bins. The string parameter should be a reasonable string representation of one of the bin quality constants. Eg. “PASS”, etc. See the appendix for a list of possible bin quality constants. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int setBinQuality(int index,
                  const char* quality);

C#
public int setBinQuality(int index,
                         string quality);

VB
Public Function setBinQuality(index As Integer,
                              quality As String) As Integer
```

addBin

Adds a new bin with the given value, description and quality. The value must adhere to the current bin format and not already exist. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int addBin(const char* value,
           const char* description = "",
           const char* quality = "");

C#
public int addBin(string value,
                  string description,
                  string quality);

VB
Public Function addBin(value As String,
                       description As String,
                       quality As String) As Integer
```

getColCount

Retrieve the number of columns in the active wafer map. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getColCount(int& count) const;

C#
public int getColCount(ref int count);

VB
Public Function getColCount(ByRef count As Integer) As Integer
```

getRowCount

Retrieve the number of rows in the active wafer map. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getRowCount(int& count) const;

C#
public int getRowCount(ref int count);

VB
Public Function getRowCount(ByRef count As Integer) As Integer
```

getArrayExtents

Retrieve the physical extents of the active wafer map. This is based on the number of columns and rows and the device or step size. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getArrayExtents(WMCLib::Box& extents) const;

C#
public int getArrayExtents(ref Box extents);

VB
Public Function getArrayExtents(ByRef extents As Box) As Integer
```

getArrayBinIndex

Gets the bin index (between zero and one less than the number of bins) at the given [col, row] position on the wafer using the specified coordinate system. See the appendix for coordinate system constants. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getArrayBinIndex(WMCLib::Coordinates coordinates,
                     int col,
                     int row,
                     int& index) const;
C#
public int getArrayBinIndex(int coordinates,
                            int col,
                            int row,
                            ref int index);

VB
Public Function getArrayBinIndex(coordinates As Integer,
                                 col As Integer,
                                 row As Integer,
                                 ByRef index As Integer) As Integer
```

setArrayBinIndex

Sets the bin index (between zero and one less than the number of bins) at the given [col, row] position on the wafer using the specified coordinate system. See the appendix for coordinate system constants. The return value will be **StatusOK** if succesful, or an error code.

C++
```
int setArrayBinIndex(WMCLib::Coordinates coordinates,
                     int col,
                     int row,
                     int index);
```

C#
```
public int setArrayBinIndex(int coordinates,
                            int col,
                            int row,
                            int index);
```

VB
```
Public Function setArrayBinIndex(coordinates As Integer,
                                 col As Integer,
                                 row As Integer,
                                 index As Integer) As Integer
```

fillArrayBinIndex

Fills the given region with the bin index (between zero and one less than the number of bins) using the specified coordinate system. See the appendix for coordinate system constants. If The **inside** parameter is set to false then all of the devices outside the given region will be filled. The return value will be **StatusOK** if succesful, or an error code.

C++
```
int fillArrayBinIndex(WMCLib::Coordinates coordinates,
                      int left,
                      int top,
                      int right,
                      int bottom,
                      int index,
                      bool inside = true);
```

C#
```
public int fillArrayBinIndex(int coordinates,
                             int left,
                             int top,
                             int right,
                             int bottom,
                             int index,
                             bool inside);
```

VB
```
Public Function fillArrayBinIndex(coordinates As Integer,
                                  left As Integer,
                                  top As Integer,
                                  right As Integer,
                                  bottom As Integer,
                                  index As Integer,
                                  inside As Boolean) As Integer
```

getDeviceCount

Retrieve the number of devies on the wafer which have the given bin index. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getDeviceCount(int index,
                   int& count) const;

C#
public int getDeviceCount(int index,
                          ref int count);

VB
Public Function getDeviceCount(index As Integer,
                               ByRef count As Integer) As Integer
```

addNullBins

Add entire columns and/or rows of null bins to the sides of the wafer specified. The **sides** parameter is a bitwise combination of constants representing the sides of the wafer. See the appendix for details of the sides constants. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int addNullBins(int sides);

C#
public int addNullBins(int sides);

VB
Public Function addNullBins(sides As Integer) As Integer
```

deleteNullBins

Delete all the entire columns and/or rows of null bins from the sides of the wafer specified. The **sides** parameter is a bitwise combination of constants representing the sides of the wafer. See the appendix for details of the sides constants. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int deleteNullBins(int sides);

C#
public int deleteNullBins(int sides);

VB
Public Function deleteNullBins(sides As Integer) As Integer
```

getHeaderValueCount

Retrieve the number of label-value pairs defined in the header of the wafer map. Some wafer map formats support fixed or arbitrary pairs of data. Eg. Lot ID. The label is a text string and the value may represent a text string, integer, floating-point value, or a measurement. In all cases the value is represented as a simple text string in the database. It is up to the client application to determine if the text string represents a specific value and parse it as required. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getHeaderValueCount(int& count) const;

C#
public int getHeaderValueCount(ref int count);

VB
Public Function getHeaderValueCount(ByRef count As Integer) As Integer
```

getHeaderLabelValue

Retrieve a label-value pair by index. The index should be between zero and one less than the number of label-value pairs. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int getHeaderLabelValue(int index,
                        char* label,
                        char* value,
                        int length) const

C#
public int getHeaderLabelValue(int index,
                               ref string label,
                               ref string value);

VB
Public Function getHeaderLabelValue(index As Integer,
                                    ByRef label As String,
                                    ByRef value As String) As Integer
```

setHeaderValue

Set a label-value pair. If the label already exists, its value will be modified to the given value. If the label does not already exist it will be added to the header. Not all wafer map formats support arbitrary header values. Knowledge of the output wafer map format may be required. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int setHeaderValue(const char* label,
                   const char* value);

C#
public int setHeaderValue(string label,
                          string value);

VB
Public Function setHeaderValue(label As String,
                               value As String) As Integer
```

stringToUnits

Retrieves a units constant based on the string representation provided. See the appendix for a description of units constants.

```
C++
void stringToUnits(const char* s,
                   WMCLib::Units& units) const;

C#
public void stringToUnits(string s,
                          ref int units);

VB
Public Sub stringToUnits(s As String,
                         ByRef units As Integer)
```

convertUnits

Calculates the scale factor between the source and target units. This allows easy conversion between values with associated units.

```
C++
double convertUnits(WMCLib::Units source,
                    WMCLib::Units target) const;

C#
public double convertUnits(int source,
                           int target);

VB
Public Function convertUnits(source As Integer,
                             target As Integer) As Double
```

formatToAbsolute

Translate a [col, row] position from wafer map format to absolute coordinates. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int formatToAbsolute(int forX,
                     int forY,
                     int& absX,
                     int& absY,
                     bool outputDB = true) const;

C#
public int formatToAbsolute(int forX,
                            int forY,
                            ref int absX,
                            ref int absY,
                            bool outputDB);

VB
Public Function formatToAbsolute(forX As Integer,
                                 forY As Integer,
                                 ByRef absX As Integer,
                                 ByRef absY As Integer,
                                 outputDB As Boolean) As Integer
```

absoluteToFormat

Translate a [col, row] position from absolute to wafer map format coordinates. The return value will be **StatusOK** if succesful, or an error code.

```
C++
int absoluteToFormat(int absX,
                     int absY,
                     int& forX,
                     int& forY,
                     bool outputDB = true) const;

C#
public int absoluteToFormat(int absX,
                            int absY,
                            ref int forX,
                            ref int forY,
                            bool outputDB);

VB
Public Function absoluteToFormat(absX As Integer,
                                 absY As Integer,
                                 ByRef forX As Integer,
                                 ByRef forY As Integer,
                                 outputDB As Boolean) As Integer
```

stringToBinQuality

Retrieves a bin quality constant based on the string representation provided. See the appendix for a description of bin quality constants.

C++
```
void stringToBinQuality(const char* s, WMCLib::BinQuality& quality) const;
```

C#
```
public void stringToBinQuality(string s, ref int quality);
```

VB
```
Public Sub stringToBinQuality(s As String, ByRef quality As Integer)
```

binQualityToString

Retrieves a string representation of the specified bin quality constant.

C++
```
void binQualityToString(WMCLib::BinQuality quality, char* buffer, int bufferLength) const;
```

C#
```
public void binQualityToString(int quality, ref string buffer);
```

VB
```
Public Sub binQualityToString(quality As Integer, ByRef buffer As String)
```

WMView API

createWMCView/createWMNView

Create an instance of the wafer map view library. The instanceFile and instanceData parameters are for future use and should be passed empty strings. The status string will contain information if the function fails and returns a null pointer.

```
C++
ACS_WMCLIB_DECLSPEC WMCView* createWMCView(const char* instanceFile,
                                           const char* instanceData,
                                           char* status,
                                           int statusLength);
C#
public static WMNView createWMNView(string instanceFile,
                                    string instanceData,
                                    ref string status);

VB
Public Shared Function createWMNView(instanceFile As String,
                                     instanceData As String,
                                     ByRef status As String) As WMNView
```

destroyWMCView

Destroys the view object returned from **createWMCView**. This function is only needed in a C++ client, which should call the function when finished with the view, to avoid memory leaks.

```
C++
ACS_WMCLIB_DECLSPEC void destroyWMCView(WMCView* wmView);
```

initView

Initializes an instance of WMView with an existing instance of WMLib and the window pointer where the view will be rendered. See the WMLib creation functions for details on how to create WMLib. This function should be called immediately after the WMView object is created.

```
C++
void initView(const WMCLib* wmLib,
              HWND window);

C#
public void initView(WMNLib wmLib,
                     IntPtr window);

VB
Public Sub initView(wmLib As WMNLib,
                    window As IntPtr)
```

clear

Clears the view using the currently defined background color. The return value is **StatusWindowInvalid** if the window Is invalid, otherwise **StatusOK**.

C++
```
int clear();
```

C#
```
public int clear();
```

VB
```
Public Function clear() As Integer
```

redraw

Redraws the view. Most of the functions in the WMView API do not automatically redraw the view. This allows for multiple changes to be made to the library and view, then control when the view should be updated. The return value is **StatusWindowInvalid** if the window Is invalid, otherwise **StatusOK**.

C++

```
int redraw();
```

C#
```
public int redraw();
```

VB
```
Public Function redraw() As Integer
```

displayDeviceOutlines

Sets whether to draw or not draw the device outlines for the whole wafer map.

C++
```
void displayDeviceOutlines(bool display);
```

C#
```
public void displayDeviceOutlines(bool display);
```

VB
```
Public Sub displayDeviceOutlines(display As Boolean)
```

displayDeviceLabels

Sets whether to draw or not draw the device labels for the whole wafer map.

C++
```
void displayDeviceLabels(bool display);
```

C#
```
public void displayDeviceLabels (bool display);
```

VB
```
Public Sub displayDeviceLabels (display As Boolean)
```

setBackgroundColor

Sets the background color of the view window.

```
C++
virtual void setBackgroundColor(WMCView::Color color);

C#
public void setBackgroundColor(Color color);

VB
Public Sub setBackgroundColor(color As Color)
```

setSelectionColor

Sets the color style of all selected devices. The **StyleItem** parameter can be a bitwise combination of values representing fill, line and text color. These constants are listed in the appendix.

```
C++
virtual void setSelectionColor(int styleItem,
                               WMCView::Color color);

C#
public void setSelectionColor(int styleItem,
                              Color color);

VB
Public Sub setSelectionColor(styleItem As Integer,
                             color As Color)
```

setQualityColor

Sets the color style of all the devices with the provided quality. For a list of possible quality options see the appendix. The **StyleItem** parameter can be a bitwise combination of values representing fill, line and text color. These constants are also listed in the appendix.

```
C++
void setQualityColor(WMCView::BinQuality quality,
                     int styleItem,
                     WMCView::Color color);

C#
public void setQualityColor(int quality,
                            int styleItem,
                            Color color);

VB
Public Sub setQualityColor(quality As Integer,
                           styleItem As Integer,
                           color As Color)
```

setBinColor

Sets the color style of all the devices with the provided bin. The **StyleItem** parameter can be a combination of values representing fill, line and text color. These constants are listed in the appendix.

```
C++
void setBinColor(const char* binValue,
                 int styleItem,
                 WMCView::Color color);

C#
public void setBinColor(string value,
                        int styleItem,
                        Color color);

VB
Public Sub setBinColor(value As String,
                       styleItem As Integer,
                       color As Color)
```

zoomAll

Fits the entire wafer map in the view so that all devices are visible.

```
C++
void zoomAll();

C#
public void zoomAll();

VB
Public Sub zoomAll()
```

zoomCenter

Zooms the view by the provided scale centered on the center of the view. A scale less than one zooms in and a scale greater than one zooms out.

```
C++
void zoomCenter(double scale);

C#
public void zoomCenter(double scale);

VB
Public Sub zoomCenter(scale As Double)
```

zoomPoint

Zooms the view by the provided scale centered on the provided center point (in window coordinates).

```
C++
void zoomPoint(double scale,
               int centerX,
               int centerY);

C#
public void zoomPoint(double scale,
                      int centerX,
                      int centerY);

VB
Public Sub zoomPoint(scale As Double,
                     centerX As Integer,
                     centerY As Integer)
```

zoom

Zooms the view to display all the devices inside the provided rectangle. The coordinates can be based on the window, the wafer map format, or an absolute coordinate system where the top left device is at column zero and row zero. See the appendix for all possible coordinate values.

```
C++
void zoom(WMCView::Coordinates coordinates,
          int x1,
          int y1,
          int x2,
          int y2);

C#
public void zoom(int coordinates,
                 int x1,
                 int y1,
                 int x2,
                 int y2);

VB
Public Sub zoom(coordinates As Integer,
                x1 As Integer,
                y1 As Integer,
                x2 As Integer,
                y2 As Integer)
```

pan

Pans the view by the provided translation vector. The coordinates can be based on the window, the wafer map format, or an absolute coordinate system where the top left device is at column zero and row zero. See the appendix for all possible coordinate values.

```
C++
void pan(WMCView::Coordinates coordinates,
         int dx,
         int dy);

C#
public void pan(int coordinates,
                int dx,
                int dy);

VB
Public Sub pan(coordinates As Integer,
               dx As Integer,
               dy As Integer)
```

select

Selects a single device in the view according to the selection mode and provided position. The selection mode can set the device selection state to on, off, or toggle its current state. Multiple devices can be selected in the view at the same time. The coordinates can be based on the window, the wafer map format, or an absolute coordinate system where the top left device is at column zero and row zero. See the appendix for all possible coordinate values.

```
C++
bool select(WMCView::SelectMode selectMode,
            WMCView::Coordinates coordinates,
            int x = 0,
            int y = 0,
            bool clear = true);

C#
public void select(int selectMode,
                   int coordinates,
                   int x,
                   int y,
                   bool clear);

VB
Public Sub [select](selectMode As Integer,
                    coordinates As Integer,
                    x As Integer,
                    y As Integer,
                    clear As Boolean)
```

selectRange

Selects a range of devices in the view according to the selection mode and provided region. The selection mode can set the device selection state to on, off, or toggle its current state. The coordinates can be based on the window, the wafer map format, or an absolute coordinate system where the top left device is at column zero and row zero. See the appendix for all possible coordinate values.

C++
```
bool selectRange(WMCView::SelectMode selectMode,
                 WMCView::Coordinates coordinates,
                 int x1 = 0,
                 int y1 = 0,
                 int x2 = 0,
                 int y2 = 0,
                 bool clear = true);
```

C#
```
public void selectRange(int selectMode,
                        int coordinates,
                        int x1,
                        int y1,
                        int x2,
                        int y2,
                        bool clear);
```

VB
```
Public Sub selectRange(selectMode As Integer,
                       coordinates As Integer,
                       x1 As Integer,
                       y1 As Integer,
                       x2 As Integer,
                       y2 As Integer,
                       clear As Boolean)
```

getSelectionCount

Returns the number of selected devices.

C++
```
int getSelectionCount() const;
```

C#
```
public int getSelectionCount();
```

VB
```
Public Function getSelectionCount() As Integer
```

getSelection

Retrieves the column and row (in absolute coordinates) of the selected device with the provided index. Index values can be between zero and one less than the number of selected devices. This function can be useful to iterate over all of the selected devices and perform some operation.

```
C++
bool getSelection(int& col,
                  int& row,
                  int deviceIndex = 0) const;

C#
public bool getSelection(ref int col,
                         ref int row,
                         int deviceIndex);

VB
Public Function getSelection(ByRef col As Integer,
                             ByRef row As Integer,
                             deviceIndex As Integer) As Boolean
```

clearSelection

Clears the selection so that no devices are selected.

```
C++
void clearSelection();

C#
public void getSelectionCount();

VB
Public Sub getSelectionCount()
```

windowToAbsoluteCoordinates

Transforms from window coordinates to absolute coordinates. This is useful to determine the [col, row] position on the wafer map under the mouse pointer.

```
C++
virtual void windowToAbsoluteCoordinates(int x,
                                         int y,
                                         int& col,
                                         int& row) const;

C#
public void windowToAbsoluteCoordinates(int x,
                                        int y,
                                        ref int col,
                                        ref int row);

VB
Public Sub windowToAbsoluteCoordinates(x As Integer,
                                       y As Integer,
                                       ByRef col As Integer,
                                       ByRef row As Integer)
```

API Appendix

These constants are static members of the WMLib framework.

Status

C++

```
static const int StatusOK;
static const int StatusUnknownFormat;
static const int StatusUnlicensed;
static const int StatusFileError;
static const int StatusBinInvalid;
static const int StatusBinDuplicate;
static const int StatusLoadFailed;
static const int StatusInputUnsupported;
static const int StatusOutputUnsupported;
static const int StatusNoWaferMaps;
static const int StatusRowInvalid;
static const int StatusDBInvalid;
static const int StatusMapInvalid;
static const int StatusBinFormatUnsupported;
static const int StatusColInvalid;
static const int StatusColRowInvalid;
static const int StatusFileNameValidated;
static const int StatusBinIndexInvalid;
static const int StatusHeaderIndexInvalid;
```

C#

```
public static int StatusOK;
public static int StatusUnknownFormat;
public static int StatusUnlicensed;
public static int StatusFileError;
public static int StatusBinInvalid;
public static int StatusBinDuplicate;
public static int StatusLoadFailed;
public static int StatusInputUnsupported;
public static int StatusOutputUnsupported;
public static int StatusNoWaferMaps;
public static int StatusRowInvalid;
public static int StatusDBInvalid;
public static int StatusMapInvalid;
public static int StatusBinFormatUnsupported;
public static int StatusColInvalid;
public static int StatusColRowInvalid;
public static int StatusFileNameValidated;
public static int StatusBinIndexInvalid;
public static int StatusHeaderIndexInvalid;
```

VB

```
Public Shared StatusOK As Integer
Public Shared StatusUnknownFormat As Integer
Public Shared StatusUnlicensed As Integer
Public Shared StatusFileError As Integer
Public Shared StatusBinInvalid As Integer
Public Shared StatusBinDuplicate As Integer
```

```
Public Shared StatusLoadFailed As Integer
Public Shared StatusInputUnsupported As Integer
Public Shared StatusOutputUnsupported As Integer
Public Shared StatusNoWaferMaps As Integer
Public Shared StatusRowInvalid As Integer
Public Shared StatusDBInvalid As Integer
Public Shared StatusMapInvalid As Integer
Public Shared StatusBinFormatUnsupported As Integer
Public Shared StatusColInvalid As Integer
Public Shared StatusColRowInvalid As Integer
Public Shared StatusFileNameValidated As Integer
Public Shared StatusBinIndexInvalid As Integer
Public Shared StatusHeaderIndexInvalid As Integer
```

Units

C++

```
static const int UnitsMM;
static const int UnitsUM;
static const int UnitsInch;
static const int UnitsMil;
static const int UnitsTenth;
```

C#

```
public static int UnitsMM;
public static int UnitsUM;
public static int UnitsInch;
public static int UnitsMil;
public static int UnitsTenth;
```

VB

```
Public Shared UnitsMM As Integer
Public Shared UnitsUM As Integer
Public Shared UnitsInch As Integer
Public Shared UnitsMil As Integer
Public Shared UnitsTenth As Integer
```

Coordinates

C++

```
typedef enum {
  CoordinatesAbsolute,
  CoordinatesFormat
} Coordinates;
```

C#

```
public static int CoordinatesAbsolute;
public static int CoordinatesFormat;
```

VB

```
Public Shared CoordinatesAbsolute As Integer
Public Shared CoordinatesFormat As Integer
```

Origin

C++

```
static const int OriginLowerLeft;
static const int OriginUpperLeft;
static const int OriginUpperRight;
static const int OriginLowerRight;
static const int OriginCenter;
```

C#

```
public static int OriginLowerLeft;
public static int OriginUpperLeft;
public static int OriginUpperRight;
public static int OriginLowerRight;
public static int OriginCenter;
```

VB

```
Public Shared OriginLowerLeft As Integer
Public Shared OriginUpperLeft As Integer
Public Shared OriginUpperRight As Integer
Public Shared OriginLowerRight As Integer
Public Shared OriginCenter As Integer
```

Axis

C++

```
static const int AxisRightUp;
static const int AxisRightDown;
static const int AxisLeftDown;
static const int AxisLeftUp;
static const int AxisOrigin;
```

C#

```
public static int AxisRightUp;
public static int AxisRightDown;
public static int AxisLeftDown;
public static int AxisLeftUp;
public static int AxisOrigin;
```

VB

```
Public Shared AxisRightUp As Integer
Public Shared AxisRightDown As Integer
Public Shared AxisLeftDown As Integer
Public Shared AxisLeftUp As Integer
Public Shared AxisOrigin As Integer
```

Flat

C++

```
static const int FlatBottom;
static const int FlatLeft;
static const int FlatTop;
static const int FlatRight;
```

C#

```
public static int FlatBottom;
public static int FlatLeft;
public static int FlatTop;
public static int FlatRight;
```

VB

```
Public Shared FlatBottom As Integer
Public Shared FlatLeft As Integer
Public Shared FlatTop As Integer
Public Shared FlatRight As Integer
```

Generic Side

C++

```
static const int SideBottom;
static const int SideLeft;
static const int SideTop;
static const int SideRight;
```

C#

```
public static int SideBottom;
public static int SideLeft;
public static int SideTop;
public static int SideRight;
```

VB

```
Public Shared SideBottom As Integer
Public Shared SideLeft As Integer
Public Shared SideTop As Integer
Public Shared SideRight As Integer
```

Bin Quality

C++

```
typedef enum {
  BinQualityNull,
  BinQualityPass
  BinQualityFail,
  BinQualityReference,
  BinQualityMirror,
  BinQualityEdge,
  BinQualitySkip,
  BinQualityUgly,
  BinQualityTest,
  BinQualityUnknown,
} BinQuality;
```

C#

```
public static int BinQualityEdge;
public static int BinQualityFail;
public static int BinQualityMirror;
public static int BinQualityNull;
public static int BinQualityPass;
public static int BinQualityReference;
public static int BinQualitySkip;
public static int BinQualityTest;
public static int BinQualityUgly;
public static int BinQualityUnknown;
```

VB

```
Public Shared BinQualityEdge As Integer
Public Shared BinQualityFail As Integer
Public Shared BinQualityMirror As Integer
Public Shared BinQualityNull As Integer
Public Shared BinQualityPass As Integer
Public Shared BinQualityReference As Integer
Public Shared BinQualitySkip As Integer
Public Shared BinQualityTest As Integer
Public Shared BinQualityUgly As Integer
Public Shared BinQualityUnknown As Integer
```

Bin Base

C++

```
typedef enum {
  BinBaseASCII
  BinBaseHexadecimal,
  BinBaseDecimal,
} BinBase;
```

C#

```
public static int BinBaseASCII;
public static int BinBaseHexadecimal;
public static int BinBaseDecimal;
```

VB

```
Public Shared BinBaseASCII As Integer
Public Shared BinBaseHexadecimal As Integer
Public Shared BinBaseDecimal As Integer
```

These structures/classes are members of the WMLib framework.

Co-ordinate pair/size with units

C++

```
struct XY {
  double x;
  double y;
  Units units;
};
```

C#

```
public class XY {
  public double x;
  public double y;
  public int units;
  public XY();
}
```

VB

```
Public Class XY
  Public x As Double
  Public y As Double
  Public units As Integer
  Public Sub New()
End Class
```

Co-ordinate bounding box with units

C++

```
struct Box {
  double x1;
  double y1;
  double x2;
  double y2;
  Units units;
};
```

C#

```
public class Box {
  public double x1;
  public double y1;
  public double x2;
  public double y2;
  public int units;
  public Box();
}
```

VB

```
Public Class Box
  Public x1 As Double
  Public y1 As Double
  Public x2 As Double
  Public y2 As Double
  Public units As Integer
  Public Sub New()
End Class
```

Bin format defining encoding type and length

C++

```cpp
struct BinFormat {
  BinBase base;
  int length;
};
```

C#

```csharp
public class BinFormat {
  public int @base;
  public int length;
  public BinFormat();
}
```

VB

```vb
Public Class BinFormat
  Public base As Integer
  Public length As Integer
  Public Sub New()
End Class
```

These constants are static members of the WMView framework.

Status

C++

```
static const int StatusOK;
static const int StatusWindowInvalid;
```

C#

```
public static int StatusOK;
public static int StatusWindowInvalid;
```

VB

```
Public Shared StatusOK As Integer
Public Shared StatusWindowInvalid As Integer
```

Style

C++

```
static const int StyleItemFill;
static const int StyleItemLine;
static const int StyleItemText;
```

C#

```
public static int StyleItemFill;
public static int StyleItemLine;
public static int StyleItemText;
```

VB

```
Public Shared StyleItemFill As Integer
Public Shared StyleItemLine As Integer
Public Shared StyleItemText As Integer
```

Coordinates

C++

```
typedef enum {
  CoordinatesAbsolute,
  CoordinatesFormat,
  CoordinatesWindow
} Coordinates;
```

C#

```
public static int CoordinatesAbsolute;
public static int CoordinatesFormat;
public static int CoordinatesWindow;
```

VB

```
Public Shared CoordinatesAbsolute As Integer
Public Shared CoordinatesFormat As Integer
Public Shared CoordinatesWindow As Integer
```

Select Mode

C++

```
typedef enum {
  SelectModeOn,
  SelectModeOff,
  SelectModeToggle
} SelectMode;
```

C#

```
public static int SelectModeOn;
public static int SelectModeOff;
public static int SelectModeToggle;
```

VB

```
Public Shared SelectModeOn As Integer
Public Shared SelectModeOff As Integer
Public Shared SelectModeToggle As Integer
```

These structures/classes are members of the WMView framework.

RGB Color Triplet

C++

```
typedef struct {
  int r;
  int g;
  int b;
} Color;
```

C#

```
public class Color {
  public int r;
  public int g;
  public int b;
  public Color();
  public Color(int red, int green, int blue);
}
```

VB

```
Public Class Color
  Public r As Integer
  Public g As Integer
  Public b As Integer
  Public Sub New()
  Public Sub New(red As Integer, green As Integer, blue As Integer)
End Class
```