|  |  |
| --- | --- |
| 基础数据及结构体定义 |  |

|  |  |
| --- | --- |
| 类型 | 枚举 |
| 名称 | emImageDirection; |
| 描述 | 旋转图像显示的枚举值 |

值说明：

 DIRECTION\_0 //正常显示

 DIRECTION\_90 //顺时针90度旋转显示

 DIRECTION\_180 //顺时针180度旋转显示

 DIRECTION\_270 //顺时针270度旋转显示

|  |  |
| --- | --- |
| 类型 | 枚举 |
| 名称 | emADVInfoPos |
| 描述 | 文本显示的位置 |

值说明：

 POS\_LEFT\_TOP //文本显示于图像的左上

 POS\_RIGHT\_TOP //文本显示于图像的右上

 POS\_LEFT\_BOTTOM //文本显示于图像的左下

 POS\_RIGHT\_BOTTOM //文本显示于图像的右下

|  |  |
| --- | --- |
| 类型 | 枚举 |
| 名称 | emMarkStatus |
| 描述 | 缺陷状态 |

 \_MARK\_STATUS\_SELECT = 0, //缺陷为选中状态。.//调用方，请不要设置此值。（保留值）

 \_MARK\_STATUS\_INVALID = 1, // 无效的缺陷

 \_MARK\_STATUS\_VALID = 2, // 有效的缺陷

 \_MARK\_STATUS\_UNKONWN = 3 // 待定的缺陷

|  |  |
| --- | --- |
| 类型 | 值 |
| 名称 | 文本信息显示的可选项 |
| 描述 | 文本信息显示的可选项 |

值说明：

 #define PIC\_INFO\_TEXT\_IMAGE\_IDX 0x01 //图像序号

#define PIC\_INFO\_TEXT\_IMAGE\_SIZE 0x02 //图像尺寸

#define PIC\_INFO\_TEXT\_ZOOM\_VALUE 0x04 //缩放比例

#define PIC\_INFO\_TEXT\_MOUSE\_PT 0x08 //鼠标位置。（屏幕位置）

#define PIC\_INFO\_TEXT\_PIC\_POS 0x10 //图像当前位置

#define PIC\_INFO\_TEXT\_IMAGE\_EXT 0X20 //图像扩展信息

#define PIC\_INFO\_TEXT\_IMAGE\_TIME 0X40 //图像时间信息

#define PIC\_INFO\_TEXT\_IMAGE\_KM 0X80 //图像里程信息

#define PIC\_INFO\_TEXT\_CURRENT\_MODE 0X100 //是否显示当前模式；

|  |  |
| --- | --- |
| 类型 | 结构体 |
| 名称 | stImageTime; |
| 描述 | 控件使用的时间结构信息 |

typedef struct \_IMAGE\_TIME\_

{

 int nYear; //年

 int nMonth; //月

 int nDay; //日

 int nHour; //时

 int nMintue; //分

 int nSecond; //秒

}stImageTime;

|  |  |
| --- | --- |
| 类型 | 结构体 |
| 名称 | stImageBaseInfo;; |
| 描述 | 图像基础信息 |

typedef struct \_ADV\_IMAGE\_BASE\_INFO\_

{

 long nImageIndex; //图号

 long lKm; //图像里程。千米信息;

 long lM; //图像里程。米信息;

 int nImageWidth; //图像宽度

 int nImageHeight; //图像高度

 stImageTime \_time; //图像采集时间；

 std::string strInfoExt; //扩展信息，如果设置为显示图像信息，此信息将显示在信息的最后位置

}stImageBaseInfo;

|  |  |
| --- | --- |
| 类型 | 结构体 |
| 名称 | stMarkInfo |
| 描述 | 缺陷基础信息 |

typedef struct \_ADV\_MARK\_INFO\_

{

 long lFaultIndex; //缺陷的ID，全局唯一；

 long nImageIndex; // 缺陷所在图像的序号

 long nKM; // 缺陷里程信息;

 long nM; // 缺陷里程信息

 long iFaultX; // 缺陷位置x

 long iFaultY; // 缺陷位置y

 long iFaultCx; // 缺陷位置宽度

 long iFaultCy; // 缺陷位置高度

 emMarkStatus iFaultState; //缺陷状态

 long nFaultTypeID; //缺陷类型序号;

 std::string strFaultName; //缺陷名称

}stMarkInfo;

|  |  |
| --- | --- |
| 接口定义 |  |

|  |  |
| --- | --- |
| 名称 | IAveImageNotify  |
| 描述 | 需要接收控件回调的对象需要实现的接口 |

/\*

当绘图控件新增了一个缺陷标记的时候，回调此函数

 参数:

 UINT nID 控件ID

 StMarkInfo 当前新增加的缺陷的信息.

 返回值：

 TRUE,表示 要保留此缺陷，FALSE表示放弃此次标记。

 说明

通知上层.当前新添加的标记区域 ::上层需要填充:stFaultInfo::nImageIndex值

如果要保留此缺陷，需要在调里面填充：

 lFaultIndex

 nFaultTypeID

 strFaultName

为有效值。否者控件会抛弃此缺陷标记.

\*/

virtual BOOL OnControlInsertMark(UINT nID,stMarkInfo&) = 0;

/\*

通知上层，标记区域发生变化；

参数:

 UINT nID 控件ID

 StMarkInfo 当前发生参数变化的缺陷的信息

返回值：

 恒定为TRUE;

\*/

 virtual BOOL OnControlModifyMark(UINT nID,stMarkInfo&) = 0;

/\*

通知上层，当前图像的缩放比例发生变化；

参数:

 UINT nID 控件ID

 CurValue 当前图像的缩放比例

返回值：

 恒定为TRUE;

\*/

 virtual void OnZoomValueChange(UINT nID,const float& \_curValue) = 0;

/\*

通知上层，控件鼠标右键压下；

参数:

 UINT nID 控件ID

 lImageIdx 当前的图像序号

lMarkIdx 当前的缺陷序号

返回值：

 无

说明：

 当控件内鼠标右键压下时，会出发此回调到上层调用者。

 返回当前的图像需要在lImageIdx，如果当前没有任何显示的图像lImageIdx = -1；

 返回当前右键位置的缺陷在lMarkIdx ，如果当前没有缺陷被鼠标右键选中，返回-1;

\*/

 virtual void OnControlRButtonDown(UINT nID,long lImageIdx, long lMarkIdx) = 0;

/\*

通知上层，当前测量状态发生变化；

参数:

 UINT nID 控件ID

 nWidth 当前测量状态下，图像像素宽度

 nHeight 当前测量状态下，图像像素高度

 showInfo 上层传递给控件，需要显示的测量信息；

返回值：

 恒定为TRUE;

\*/

 virtual void OnControlMeasure(UINT nID,int nWidth, int nHeight, std::string& \_showInfo) = 0;

|  |  |
| --- | --- |
| 接口定义 |  |

|  |  |
| --- | --- |
| 名称 | IAdvImageShowControl |
| 描述 | 绘图控件主类 |

/\*

 控件全屏显示图像的方式

\*/

typedef enum \_FULLSCREEN\_MONITOR\_STATUS\_

 {

 MONITOR\_ALL = 0, //全屏显示在所有显示器上。

 MONITOR\_CURRENT = 1, //全屏显示在当前程序所在显示器上

 MONITOR\_DEFAULT = 2 //全屏显示在主显示器上;

 }emFullScreenMonitorStatus;

/\*

创建绘图控件

参数:

 hParent 创建控件的父窗口句柄

 x 控件在父窗口中的位置

 y 控件在父窗口中的位置

 cx 控件的宽度

 cy 控件的高度

 IAveImageNotify\* 接收控件回调通知的接口，如果为NULL，控件将不会触发通知回调。

 nControlID 控件的唯一ID（当同一父窗口创建多个控件时，ID必须要保持唯一）

 返回值：

 TRUE 成功

 FALSE 失败

\*/

 virtual BOOL Create(HWND hParent, int x, int y, int cx, int cy,IAveImageNotify\*,UINT nControlID) = 0;

/\*

返回控件ID

请参考 Create 函数

\*/

 virtual UINT GetControlID() = 0;

/\*

是否显示控件

参数:

 bShow = TRUE 显示控件

 bShow = FALSE 隐藏控件

返回值：

 无

\*/

 virtual void ShowControl(BOOL bShow) = 0 ;

/\*

获取控件是否显示的状态

参数:

 无

返回值：

 TRUE 显示控件

 FALSE 隐藏控件

\*/

 virtual BOOL IsControlVisible() = 0;

/\*

是否启用控件

参数:

 bEnable = TRUE 启用控件

 bEnable = FALSE 禁用控件

返回值：

 无

\*/

 virtual void EnableControl(BOOL bEnable) = 0;

/\*

获取控件是否启用的状态

参数:

 无

返回值：

 TRUE 启用控件

 FALSE 禁用控件

\*/

 virtual BOOL IsControlEnable() = 0;

/\*

设置控件在父窗口中的大小和位置

参数:

 x 控件在父窗口中的位置

 y 控件在父窗口中的位置

 cx 控件的宽度

 cy 控件的高度

返回值：

 无

\*/

 virtual void SetControlPos(int x, int y, int cx, int cy) = 0;

/\*

获取控件在父窗口中的大小和位置

参数:

x 控件在父窗口中的位置

 y 控件在父窗口中的位置

 cx 控件的宽度

 cy 控件的高度

返回值：

 无

\*/

virtual void GetContrlPos(int& x, int& y, int& cx, int& cy) = 0;

/\*

设置控件默认显示状态下的背景色

参数:

 COLORREF RGB颜色

返回值：

 无

\*/

 virtual void SetControlBackColor(COLORREF) = 0;

/\*

获取控件默认显示状态下的背景色

参数:

 COLORREF RGB颜色

返回值：

 无

\*/

virtual void GetControlBackColor(COLORREF&) = 0;

/\*

设置控件附件数据

参数:

 pData 附加数据

返回值：

 无

\*/

 virtual void SetControlData(DWORD\_PTR pData) = 0;

/\*

获取控件附件数据

参数:

 无

返回值：

 pData 附加数据

\*/

 virtual DWORD\_PTR GetControlData() = 0;

/\*

设置一个图像数据缓冲给控件

 参数:

 BYTE\* pImageBuffer, 包含图像数据的BUFFER

 long lBufferLength BUFFER长度

 stImageBaseInfo 参考stImageBaseInfo

返回值：

 TRUE 成功

 FALSE 失败

说明：

 如果控件当前为设置的第一张图，则会立即显示此图像

 如果控件当前已经设置了N张图，请参考：

 GetImageCount

 SeekToImage

\*/

virtual BOOL SetImageBuffer(BYTE\* pImageBuffer, long lBufferLength, const stImageBaseInfo&) = 0;

 /\*

设置一个图像数据文件给控件

 参数:

 std::string strFileName 有效的图像文件绝对路径。

 stImageBaseInfo 参考stImageBaseInfo

返回值：

 TRUE 成功

 FALSE 失败

说明：

 如果控件当前为设置的第一张图，则会立即显示此图像

 如果控件当前已经设置了N张图，请参考：

 GetImageCount

 SeekToImage

\*/

 virtual BOOL SetImageFile(std::string strFileName, stImageBaseInfo&) = 0;

/\*

设置指定图号图像的文字信息

 参数:

 long nImageIndex 有效图像序号 参考stImageBaseInfo。

 std::string 图像的文字信息

返回值：

 无

\*/

 virtual void UpdateImageInfo(long nImageIndex, std::string) = 0;

 //获取当前控件中有多少张图像;

 virtual int GetImageCount() = 0;

 //获取当前控件中的图像序号::按照添加顺序返回;

 virtual BOOL GetImageListByShowOrder(std::vector<long>&) = 0;

 //跳转到某图像；

 virtual BOOL SeekToImage(long lImageIdx) = 0;

 //删除指定的图像

 virtual BOOL DeleteImage(long lImageIdx) = 0;

 //删除所有图像;

 virtual void DeleteAllImage() = 0;

//向控件添加一个标记框，

 virtual void AddMarkInfo(stMarkInfo) = 0;

/\*

删除指定标记框

参数:

 long nImageIndex 效图像序号 参考stImageBaseInfo。

 long lMarkIdx 效图的图像上的缺陷ID

返回值：

 无

\*/

 virtual void DeleteMarkInfo(long nImageIndex,long lMarkIdx) = 0;

 //删除指定图像的标记框；

 virtual void DeleteMarkInfoByImageIdx(long lImageIdx) = 0;

 //删除所有的标记区域

 virtual void DeleteAllMarkInfo() = 0;

 //选中指定的图像上的标记框

 virtual void SelectMarkInfo(long lImageIdx, long lMarkIdx) = 0;

//获取当前系统的显示器数量;

 virtual int GetMonitorCount() = 0;

 //设置是否全屏;

 virtual void SetFullScreen(BOOL bFull) = 0;

 //判断当前是否全屏模式

 virtual BOOL IsFullScreen() = 0;

 //设置全屏的行为模式，所有显示器全屏，主显示器全屏，程序当前显示器全屏。（如果存在多显示器的情况）

 virtual void SetFullScreenMode(emFullScreenMonitorStatus) = 0;

 //获取全屏行为;

 virtual void GetFullScreenMode(emFullScreenMonitorStatus& \_status) = 0;

/\*

是否启用全屏退出按钮

参数:

 bEnable TRUE将显示一个设置的图片在图像右上角

返回值：

 无

说明：

 参考SetFullCloseImage

 当没有使用全屏退出按钮时，使用ESC按键可退出全屏状态。

\*/

 virtual void EnableFullCloseImage(BOOL bEnable) = 0;

/\*

获取启用全屏退出按钮的状态

返回值：

 TRUE 启用

 FALSE 未启用

\*/

 virtual BOOL IsEnableFullCloseImages() = 0;

/\*

设置全屏时退出全屏按钮图片，（设置后，可以显示一个退出按钮，点击退出全屏，否则使用ESC退出）

参数说明：

 std::string strFileName 绝对路径指定得全屏退出按钮图像文件，支持JPG,BMP.PNG

返回值：

 无

\*/

 virtual void SetFullCloseImage(std::string strFileName) = 0;

/\*

获取全屏退出图像得文件全路径名

参数说明：

 std::string& strFileName 返回得绝对路径指定得全屏退出按钮图像文件

返回值：

 无

\*/

 virtual void GetFullCloseImage(std::string& strFileName) = 0;

/\*

当前显示图像按照显示区域大小自动缩放适配显示(保持比例)

\*/

 virtual void ShowImageAdjustByVisibleRectMode() = 0;

/\*

当前显示图像按照图像原始大小显示

\*/

 virtual void ShowImageWidthRealSize() = 0;

/\*

设置当前图像显示得的缩放比例

参数说明:

 float flZoom 设置的图像缩放比例

说明：

 如果当前正在显示图像，当前图像将按照设置的比例显示。如果当前没有图像显示，下次图像加载将按照此比例显示。

\*/

 virtual void SetImageZoomValue(float flZoom) = 0;

/\*

获取当前图像显示的缩放比例

参数说明:

 Float& flZoom 返回当前设置的图像缩放比例

返回值：

 恒定为TRUE

说明：

 相关函数：

 IAveImageNotify：：OnZoomValueChange

\*/

 virtual BOOL GetImageZoomValue(float& flZoomValue) = 0;

/\*

设置图像显示的缩放比例范围

参数说明:

 float flMin 控件缩放比例的最小值

 float flMax 控件缩放比例的最大值

说明：

 相关函数：

 SetImageZoomValue

当设置的控件缩放比例大于最大值，或小于最小值时，将不会应用设置的超出范围的值。

\*/

 virtual void SetZoomRange(float flMin = 0.2, float flMax = 5.0) = 0;

/\*

获取像显示的缩放比例范围

参数说明:

 float flMin 返回控件缩放比例的最小值

 float flMax 返回控件缩放比例的最大值

\*/

 virtual void GetZoomRange(float& flMin, float& flMax) = 0;

/\*

 设置是否允许通过鼠标左键双击来对图像进行缩放操作

参数说明：

BOOL bEnable TRUE:允许 FALSE:不允许

 说明：

 当启用此功能时，如果图像不在左键双击缩放状态，双击时将缩放图像到SetDoubleClickZoomValue函数这是的放大参数。如果图像处于双击放大状态，双击时将缩放图像到双击缩放前的比例。

 请参考：IsEnableDoubleClickZoom

 SetDoubleClickZoomValue

 GetDoubleClickZoomValue

\*/

 virtual void EnableDoubleClickZoom(BOOL bEnable) = 0;

/\*

获取像当前是否启用鼠标左键双击缩放功能。

返回值：

 TRUE: 启用

 FALSE: 禁用

\*/

 virtual BOOL IsEnableDoubleClickZoom() = 0;

/\*

 设置图像双击时的倍数

参数说明：

float flZoom 设置双击缩放时的缩放比例

 说明：

 当设置此参数后，如果图像不在左键双击缩放状态，双击时将缩放图像到此缩放参数。如果图像处于双击放大状态，双击时将缩放图像到双击缩放前的比例。

 请参考：IsEnableDoubleClickZoom

 EnableDoubleClickZoom

 GetDoubleClickZoomValue

\*/

 virtual void SetDoubleClickZoomValue(float flZoom) = 0;

/\*

获取像当前鼠标左键双击缩放比例。

参数说明：

 float& flZoom 返回设置的鼠标左键双击缩放比例

返回值：

 无

请参考:

IsEnableDoubleClickZoom

EnableDoubleClickZoom

SetDoubleClickZoomValue

\*/

 virtual void GetDoubleClickZoomValue(float& flZoom) = 0;

/\*

设置是否显示一个控件的标题信息。

参数说明：

 BOOL bEnable 设置是否显示

返回值：

 无

说明：

 如果设置显示控件标题信息，将会根据设置在控件的顶部或底部，显示一个标题栏，标题栏可设置背景颜色，文字颜色，文字大小，等信息。

\*/

 virtual void SetControlInfoStatus(BOOL bEnable) = 0;

/\*

获取是否显示控件标题信息

返回值：

 TRUE 显示

 FALSE 不显示

\*/

 virtual BOOL GetControlInfoStatus() = 0;

/\*

设置控件标题显示的文字信息

参数值：

std::string pszControlInfo 要显示在标题栏中的文字信息

返回值：

 无

\*/

 virtual void SetControlInfo(std::string pszControlInfo) = 0;

/\*

获取控件标题显示的文字信息

参数值：

std::string& pszControlInfo 返回显示在标题栏中的文字信息

返回值：

 无

\*/

 virtual void GetControlInfo(std::string& pszControlInfo) = 0;

/\*

设置控件标题显示的位置

参数值：

BOOL bTop TRUE:显示在控件顶部，FALSE显示在控件底部

返回值：

 无

\*/

 virtual void SetControlInfoShowAtTop(BOOL bTop = TRUE) = 0;

/\*

获取控件标题是否显示在顶部

参数值：

 无

返回值：

TRUE 控件显示在顶部

FALSE 控件显示在底部

\*/

 virtual BOOL ContrlInfoPosIsTop() = 0;

/\*

设置控件标题的高度

参数值：

int nHeight 标题栏高度使用多少像素

返回值：

 无

\*/

 //设置控件信息的显示高度;

 virtual void SetControlInfoHeight(int nHeight = 20) = 0;

/\*

获取控件标题的高度

参数值：

无

返回值：

 int nHeight 标题栏高度使用多少像素

\*/

 virtual int GetControlInfoHeight() = 0;

/\*

设置控件信息的显示背景色

参数值：

COLORREF 标题栏使用的背景色

返回值：

 无

\*/

 virtual void SetControlInfoBackColor(COLORREF crBack) = 0;

/\*

获取控件标题的显示背景色

参数值：

COLORREF& crColor 返回标题栏背景颜色的RGB值

返回值：

 无

\*/

 virtual void GetControlInfoBackColor(COLORREF& crColor) = 0;

/\*

设置控件信息的显示文本色

参数值：

COLORREF 标题栏使用的文字颜色

返回值：

 无

\*/

 virtual void SetControlInfoTextColor(COLORREF crBack) = 0;

/\*

获取控件标题的文字颜色

参数值：

COLORREF& crColor 返回标题栏文字颜色的RGB值

返回值：

 无

\*/

 virtual void GetControlInfoTextColor(COLORREF& crColor) = 0;

/\*

设置控件标题的显示字体的名字

参数值：

std::string pszFontName 字体的名字，请参考WIN32 FONT结构体

返回值：

 无

说明：

必须是当前系统安装了的字体。如：宋体，微软雅黑。

参考MFC CFontDialog

\*/

 virtual void SetControlInfoFont(std::string pszFontName) = 0;

/\*

获取控件标题的文字字体名字

参数值：

std::string& pszFontName 字体的名字，请参考WIN32 FONT结构体

返回值：

 无

\*/

 virtual void GetControlInfoFont(std::string& pszFontName) = 0;

 /\*

设置控件标题的显示字体的大小

参数值：

int nFontSize 字体的大小，请参考WIN32 FONT结构体

返回值：

 无

说明：

参考MFC CFontDialog

请参考WIN32 FONT结构体

\*/

 virtual void SetControlInfoFontSize(int nFontSize) = 0;

/\*

获取控件标题的文字字体大小

参数值：

int& nFontSize 字体的大小，请参考WIN32 FONT结构体

返回值：

 无

\*/

 virtual void GetControlInfoFontSize(int& nFontSize) = 0;

/\*

设置是否显示控件信息，控件信息包含 “文本信息显示的可选项”中包含的各可选项。

参数说明：

 DWORD dwPicInfoText 图像信息将要显示的可选项的组合值。

返回值：

 无

说明：

 请参考

* #define PIC\_INFO\_TEXT\_IMAGE\_IDX 0x01 //图像序号

#define PIC\_INFO\_TEXT\_IMAGE\_SIZE 0x02 //图像尺寸

#define PIC\_INFO\_TEXT\_ZOOM\_VALUE 0x04 //缩放比例

#define PIC\_INFO\_TEXT\_MOUSE\_PT 0x08 //鼠标位置。（屏幕位置）

#define PIC\_INFO\_TEXT\_PIC\_POS 0x10 //图像当前位置

#define PIC\_INFO\_TEXT\_IMAGE\_EXT 0X20 //图像扩展信息

#define PIC\_INFO\_TEXT\_IMAGE\_TIME 0X40 //图像时间信息

#define PIC\_INFO\_TEXT\_IMAGE\_KM 0X80 //图像里程信息

#define PIC\_INFO\_TEXT\_CURRENT\_MODE 0X100 //是否显示当前模式；

 设置为0将不显示任何信息

\*/

 virtual void SetImageBaseInfoStatus(DWORD dwPicInfoText\_) = 0;

 /\*

获取当前控件显示的信息项，控件信息包含 “文本信息显示的可选项”中包含的各可选项。

参数说明：

 无

返回值：

 DWORD \_result返回显示设置项的组合值。

 请使用

 If((\_result& PIC\_INFO\_TEXT\_IMAGE\_IDX )== PIC\_INFO\_TEXT\_IMAGE\_IDX)

\*/

 virtual DWORD GetImageBaseInfoStatus() = 0;

 /\*

设置控件信息的显示文本色

参数值：

COLORREF crInfoColor 控件信息使用的文字颜色

返回值：

 无

\*/

 virtual void SetImageBaseInfoColor(COLORREF crInfoColor) = 0;

/\*

获取控件信息的显示文本色

参数值：

COLORREF& crInfoColor 返回控件信息使用的文字颜色

返回值：

 无

\*/

 virtual void GetImageBaseInfoColor(COLORREF&) = 0;

 /\*

设置控件信息的显示字体的名字

参数值：

std::string pszFontName 字体的名字，请参考WIN32 FONT结构体

返回值：

 无

说明：

必须是当前系统安装了的字体。如：宋体，微软雅黑。

参考MFC CFontDialog

\*/

 virtual void SetImageBaseInfoFont(std::string pszFontName) = 0;

/\*

获取控件信息的文字字体名字

参数值：

std::string& pszFontName 字体的名字，请参考WIN32 FONT结构体

返回值：

 无

\*/

 virtual void GetImageBaseInfoFont(std::string& pszFontName) = 0;

/\*

设置控件信息的显示字体的大小

参数值：

int nFontSize 字体的大小，请参考WIN32 FONT结构体

返回值：

 无

说明：

参考MFC CFontDialog

请参考WIN32 FONT结构体

\*/

 virtual void SetImageBaseInfoFontSize(int) = 0;

/\*

获取控件标题的文字字体大小

参数值：

int& nFontSize 字体的大小，请参考WIN32 FONT结构体

返回值：

 无

\*/

 virtual void GetImageBaseInfoFontSize(int&) = 0;

/\*

设置控件信息的显示位置

参数值：

emADVInfoPos 显示位置枚举

返回值：

 无

\*/

virtual void SetImageBaseInfoPos(emADVInfoPos) = 0;

/\*

获取控件信息的显示位置

参数值：

无

返回值：

 emADVInfoPos 显示位置枚举

\*/

 virtual emADVInfoPos GetImageBaseInfoPos() = 0;

/\*

是否启用缩略图显示;

参数值：

 BOOL bShow TRUE ：启用缩略图显示，FALSE不使用缩略图

返回值：

 无

说明：

 启用缩略图，可以在设置的显示位置，显示一个当前图像的显示缩略图。根据需要可以通过缩略图对他图像定位，显示当前可见区域的图像显示位置等信息。

\*/

 virtual void SetThumbnailStatus(BOOL bShow) = 0;

 /\*

获取缩略图显示状态;

参数值：

 无

返回值：

 TRUE ：启用缩略图显示，FALSE不使用缩略图

\*/

 virtual BOOL GetThumbnailStatus() = 0

;

/\*

设置缩略图显示位置;

参数值：

 emADVInfoPos 显示位置

返回值：

 无

说明：

注意和文字信息的位置不要冲突；如果设置为同一显示位置，将覆盖显示

\*/

 virtual void SetThumbnailPositon(emADVInfoPos) = 0;

/\*

获取缩略图显示位置;

参数值：

 emADVInfoPos& 显示位置

返回值：

 无

\*/

 virtual void GetThumbnailPositon(emADVInfoPos&) = 0;

 //设置缩略图边框颜色

 virtual void SetThumbnailBorderColor(COLORREF) = 0;

 //获取缩略图边框颜色

 virtual void GetThumbnailBorderColor(COLORREF&) = 0;

 //设置缩略显示区域边框颜色

 virtual void SetThumbnailRectColor(COLORREF) = 0;

//获取缩略显示区域边框颜色

virtual void GetThumbnailRectColor(COLORREF&) = 0;

/\*

设置缩略图的像素宽度;

参数值：

 int nWidth 缩略图的宽度：单位像素

返回值：

 无

说明：

控件将会把设置值限定在20-1000的值范围内（仅当设置的值不在方位内）

缩略图的高度将根据设置的宽度自动保持图像比例计算出来，并显示。

\*/

 virtual void SetThumbnailWidth(int nWidth) = 0;

//获取缩略图的像素宽度

 virtual int GetThumbnailWidth() = 0;

/\*

设置是否允许缩略图定位;

参数值：

 BOOL bEnable TRUE 允许,FALSE禁用

返回值：

 无

说明：

当允许缩略图定位启用，当鼠标左键在缩略图内点击并移动的时候，显示图像将根据缩略图的位置同步定位。

\*/

virtual void SetThumbnailLocation(BOOL bEnable) = 0;

//获取是否允许缩略图定位

 virtual void GetThumbnailLocation(BOOL& bEnable) = 0;

/\*

设置是否使用图像来显示略图;

参数值：

 BOOL bEnable TRUE 允许,FALSE禁用

返回值：

 无

说明：

当允许缩略图使用图像来显示时，将在缩略图显示区域使用显示图像缩放到缩略区域来显示缩略图，同时使用一个颜色框来显示当前显示区域。

否则仅使用SetThumbnailBorderColor来绘制一个完整缩略框。再使用SetThumbnailRectColor来绘制一个当前显示区域的位置。

\*/

virtual void SetThumnailUseImage(BOOL bEnable) = 0;

//获取是否使用图像来作为缩略图。

 virtual void GetThumnailUseImage(BOOL& bEnable) = 0;

//设置当前图像的显示亮度;

 virtual void AdjustBrightness(int nValue) = 0;

//设置当前图像的显示对比度

 virtual void AdjustContrast(int nValue) = 0;

//设置当前图像的显示清晰度

 virtual void AdjustDefinition(int nValue) = 0;

//获取当前图像的显示亮度;

 virtual void GetBrightness(int& nValue) = 0;

//获取当前图像的显示对比度

 virtual void GetContrast(int& nValue) = 0;

//获取当前图像的显示清晰度

 virtual void GetDefinition(int& nValue) = 0;

//设置当前图像的显示方向;

 virtual void SetImageDirection(emImageDirection \_dir) = 0;

//获取当前图像的显示方向

 virtual void GetImageDirection(emImageDirection& \_dir) = 0;

/\*

设置是否允许使用鼠标左键框选图像进行放大;

参数值：

 BOOL bEnable TRUE 允许,FALSE禁用

返回值：

 无

说明：

当允许鼠标左键框选图像进行放大时，使用鼠标左键在图像显示有效区域框选一个区域，图像将以当前框选的中心点进行图像放大。

鼠标框选放大，由于和图像的其他操作有冲突，所以必须要使用键盘CTRL OR SHIRFT来组合使用

参考：SetMouseZoomRectCorpKeyValue

\*/

virtual void EnableMouseRectZoom(BOOL bEnable) = 0;

//获取当前图像是否允许框选放大

 virtual BOOL IsEnableMouseRectZoom() = 0;

/\*

设置鼠标左键框选图像进行放大时使用的组合键

参数值：

 UINT vk\_key WIN32的键盘虚码。

返回值：

 无

说明：

当允许鼠标左键框选图像进行放大时，使用鼠标左键在图像显示有效区域框选一个区域，图像将以当前框选的中心点进行图像放大。

鼠标框选放大，由于和图像的其他操作有冲突，所以必须要使用键盘CTRL OR SHIRFT来组合使用。

参考：VK\_CONTROL

 VK\_SHIFT

\*/

 virtual void SetMouseZoomRectCorpKeyValue(UINT vk\_key) = 0;

 //获取鼠标左键框选图像进行放大时使用的组合键

 virtual void GetMouseZoomRectCorpKeyValue(UINT& vk\_key) = 0;

//设置绘制鼠标框选放大框的线条的宽度

 virtual void SetMouseRectZoomLineWidth(int nWidth) = 0;

//获取绘制鼠标框选放大框的线条的宽度

 virtual void GetMouseRectZoomLineWidth(int& nWidth) = 0;

//设置绘制鼠标框选放大框的线条的颜色

 virtual void SetMouseRectZoomLineColor(COLORREF cr) = 0;

//设置绘制鼠标框选放大框的线条的颜色

 virtual void GetMouseRectZoomLineColor(COLORREF& cr) = 0;

/\*

设置是否启用标记模式

参数值：

 BOOL bEnable TRUE 启用 FALSE 禁用

返回值：

 无

说明：

启用编辑模式，将允许通过 鼠标在图像上进行标记框操作。

 请参考: IAveImageNotify:: OnControlInsertMark

\*/

 virtual void SetMarkMode(BOOL bEnable) = 0;

//获取是否启用标记模式

 virtual BOOL GetMarkMode() = 0;

/\*

设置是否显示标记

参数值：

 BOOL bEnable TRUE 显示 FALSE 不显示

返回值：

 无

说明：

在标记模式下，恒定显示标记信息。当处于浏览模式的时候，可以通过此接口来指定是否显示标记信息。

\*/

 virtual void SetMarkShowStatus(BOOL bShow) = 0;

//获取是否显示标记

 virtual BOOL GetMarkShowStatus() = 0;

//设置标记的状态颜色颜色

 virtual void SetMarkStatusColor(emMarkStatus,COLORREF) = 0;

//获取标记的状态颜色颜色

 virtual void GetMarkStatusColor(emMarkStatus,COLORREF&) = 0;

/\*

设置是否启用测量模式

参数值：

 BOOL bEnable TRUE 启用 FALSE 禁用

返回值：

 无

说明：

启用编辑模式，将允许通过 鼠标在图像上进行画线操作。

请参考: IAveImageNotify:: OnControlMeasure

通过OnControlMeasure回调会返回当前划线在原始图像中的距离，回调内部可以根据算法，计算出新的数据，然后通过一个字符串返回，控件将显示这个字符串信息。

\*/

 virtual void SetMeasureModeStatus(BOOL bEnable) = 0;

 //获取当前是否是测量模式

 virtual BOOL GetMeasureModeStatus() = 0;

 //设置测量绘制线的颜色

 virtual void SetMeasureLineColor(COLORREF crLine) = 0;

//获取测量绘制线的颜色

 virtual void GetMeasureLineColor(COLORREF& crLine) = 0;

 //设置测量模式线的粗细 (1-N)

 virtual void SetMeasureLineWidth(int nWidth = 1) = 0;

 //获取测量绘制线的粗细

 virtual void GetMeasureLineWidth(int& nWidth) = 0;

 //销毁控件。

 virtual void Release() = 0;