Surround Video API Reference

1Overview

The Surround Video API is an interface that allows application programmers to control one or more viewports onto a Surround Video Image. This image is encapsulated in a Surround Video Document, which represents some portion of a spherical surface which is viewed from the center of a sphere.

2Surround Video Document

A Surround Video Document along with it's runtime component, represents some portion of a spherical image, this image could be made up from actual image data within the document, or by rendering engines that create the exposed surface at run time. How the image is finally rendered onto the application's surface is independent of the API and is solely the responsibility of the runtime component that supports the document.
The initial release of the SDK will ship with authoring tools for creating Surround Video Documents that support images taken by 360° panoramic cameras.

3Coordinate Space

The coordinate space in a Surround Video Document is expressed in latitude and longitude which enables the application programmer to create a viewport anywhere within the sphere.

4Interfaces

The Surround Video API is based on Component Object Model (COM) interfaces.

4.1ISurround

This interface provides detailed control of the Surround Video Document.

4.1.1ISurround::ForceValid()

ISurround::ForceValid

HRESULT ForceValid(SPHERE_RECT ***pExtent*);**

Parameters

pExtent	Points to a **SPHERE_RECT** which is to be forced valid.

Comments

When an ISurround image is opened, it is not read into memory until it is needed. This is normally handled automatically by the API. ***ISurround::ForceValid*** can be used to force a section of image into memory before it would normally be needed. This "pre-loading" of the image can substantially improve performance, especially on CD-ROM based applications.

Return Values

Value	Meaning
S_OK	Success.
SV_E_INVALID_PARAMETER	*pExtent* is an invalid pointer.
SV_E_RANGE	*pExtent* does not lie within the image.

4.1.2ISurround::GetBits()

ISurround::GetBits

HRESULT GetBits(SPHERE_POINT ****pPt*, SIZE ****pSize***, BITMAPINFOHEADER ****pbmi,* **VOID ****pvBits* **);**

Parameters

pPt	Points to an **SPHERE_POINT** which defines the image location at the center of the viewport.
pSize	Points to a **SIZE** that defines the size of the viewport in pixels.
pbmi	Address of structure containing bitmap size, format, and color data.
pvBits	Pointer to the destination bits.

Comments

The function returns a rectangular bitmap of raw bits that make up the image at the specified location.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INCOMPATIBLE_SURFACE	Bit depth does not match in pbmi.
SV_E_INVALID_PARAMETER	Invalid parameter.
SV_E_RANGE	*pPt* lies outside the image.

Example

The following code example uses ***ISurround::GetBits()***, ***ISurround::GetDepth()*** and ***ISurround::GetColors()*** to create a DIB from a section of the image. It should be noted that at the present time, 256 color images (8 bit) are assumed to have a full color table of 256 entries.

```
// CreateDibFromImage() - Create a DIB from a section of an ISurround image
BITMAPINFO* CreateDibFromImage( ISurround* pISurround, SPHERE_POINT* pLocation, SIZE*
pSize )
{
    BITMAPINFOHEADER* pDib;
    DWORD memSize;
    UINT depth;
    HRESULT hr;

    // Sanity check
    if( pISurround == NULL )
        return NULL;

    // Get the image depth
    depth = pISurround->GetDepth();

    // Calculate memory size needed the DIB
    memSize = sizeof( BITMAPINFOHEADER );

    // ...add in the size of the color table
    if( depth == 8 )
        memSize += 256 * sizeof( RGBQUAD );

    // ...and the size of the image
    memSize +=  WIDTHBYTES( pSize->cx * depth ) * pSize->cy;

    // Allocate memory for Header + Colors(if any) + bits
    // Note: GMEM_DDESHARE makes it suitable for passing to the clipboard
    pDib = (BITMAPINFOHEADER *)GlobalAllocPtr( GMEM_MOVEABLE|GMEM_DDESHARE, memSize );

    // Create the header
    pDib->biSize = sizeof( BITMAPINFOHEADER );
    pDib->biWidth = pSize->cy;
    pDib->biHeight = pSize->cx;
    pDib->biPlanes = 1;
    pDib->biBitCount = depth;
    pDib->biCompression = BI_RGB;
    pDib->biSizeImage = WIDTHBYTES( pSize->cx * depth ) * pSize->cy;
    pDib->biXPelsPerMeter = 0;
    pDib->biYPelsPerMeter = 0;
    pDib->biClrUsed = ( depth == 8 ) ? 256 : 0;
    pDib->biClrImportant = pDib->biClrUsed;

    // Get color table
    if( depth == 8 )
    {
        hr = pISurround->GetColors( 0, 256, DibColors(pDib) );
        if( FAILED(hr) )
        {
            GlobalFreePtr( pDib );
            return NULL;
        }
    }

    // Get the bits
    hr = pISurround->GetBits( pLocation, pSize, pDib, DibPtr(pDib) );
    if( FAILED(hr) )
    {
        GlobalFreePtr( pDib );
        return NULL;
    }

    return (BITMAPINFO*)pDib;

}
```

4.1.3ISurround::GetColors()

ISurround::GetColors

HRESULT GetColors(UINT *iFirstEntry*, **UINT** *iNumEntries*, **RGBQUAD FAR*** *pColors* **);**

Parameters

iFirstEntry	Index of first color entry to return.
iNumEntries	Number of color entries to return.
pColors	Pointer to an array of RGBQUAD color entries.

Comments

The function retrieves color entries for the image.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INCOMPATIBLE_SURFACE	Image does not have a palette.
SV_E_INVALID_PARAMETER	Invalid parameter.
SV_E_RANGE	*iNumEntries* specifies more colors than are present in the image.

Example

See the example for ***ISurround::GetBits()*** on page 4

4.1.4ISurround::GetDepth()

ISurround::GetDepth

UINT GetDepth();

Return Value

The bit depth of the image in bits per pixel.

Example

See the example for ***ISurround::GetBits()*** on page 4.

4.1.5ISurround::GetExtents()

ISurround::GetExtents

HRESULT GetExtents(SPHERE_RECT ***pExtent* **);

Parameters

pExtent	Points to a **SPHERE_RECT** which will receive the viewable extents of the document.

Comments

The function returns the extents (maximum and minimum longitude and latitude) of the image.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INVALID_PARAMETER	*pExtent* is an invalid pointer.

4.1.6ISurround::GetHorizon()

ISurround::GetHorizon

HRESULT GetHorizon(int ****piHorizon* **);

Parameters

piHorizon	Points to an int that will receive the Horizon of the image.

Comments

The function returns the horizon of the image.

Return Values

Value	**Meaning**
S_OK	Success.

4.1.7ISurround::GetMaxViewSize()

ISurround::GetMaxViewSize

HRESULT GetMaxViewSize(float *fZoom*, **SIZE** **pSize* **);**

Parameters

fZoom	Viewport zoom factor. (See ISurround::GetView() on page 11.
pSize	Points to a **SIZE** structure that will receive the size of the largest view that can be created with the given zoom factor. The units are in pixels.

Comments

The function is used to calculate the largest view that can be created for a given zoom factor.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INVALID_PARAMETER	*pSize* is an invalid pointers

Example

See the example for ***ISurround::GetView()*** on page 11.

4.1.8ISurround::GetView()

ISurround::GetView

HRESULT GetView(SIZE ****pSize*, **float *fZoom*, **int** *iViewQuality*, **DWORD** *dwFlags,*
ISurroundView FAR* FAR* *ppView* **);Parameters**

pSize	Points to a **SIZE** that defines the size of the viewport in pixels.
fZoom	The Zoom factor of the viewport.
iViewQuality	A number 0 to 100 that represents the desired image quality of the viewport.
dwFlags	Correction flags. These set the method of image correction used when rendering the image using ***ISurroundView::Draw()***.

SV_NO_CORRECTION	**No image correction.**
SV_HORIZONTAL_CORRECTION	**Image correction along the horizontal axis only**
SV_VERTICAL_CORRECTION	**Image correction along the vertical axis only**
SV_TOTAL_CORRECTION	**Image correction along both horizontal and vertical axes.**

ppView	Points to where to return the pointer to the requested interface. Must be NULL on error.

Comments

This function creates an instance of ISurroundView and returns a pointer to it. *ppv* is NULL on error.

Return Values

Value	Meaning
S_OK	An instance of the specified object class was successfully created.
SV_E_INCOMPATIBLE_SURFACE	Desired view is incompatible with the document image.
SV_E_INVALID_PARAMETER	Invalid parameter. *pSize* and/or *ppv* are invalid pointers
SV_E_RANGE	One or more arguments are out of range.

Any values returned by ***IUnknown::QueryInterface()*** or ***IClassFactory::CreateInstance()*** can also be returned.

Example

The following example code shows the use of ***ISurround::GetMaxViewSize()***, ***ISurround::GetView()***, ***ISurroundView::GetViewRange()*** and ***ISurroundView::GetDepth()***. It is derived from the SVViewer sample code and shows how to initialize an ISurrondView. This code could be used

Please note that variables starting with "m_" are data members of the CView class.

```
// InitSurroundView() - Initialize surround view
HRESULT CView::GetSurroundView()
{
    SIZE maxSize;
    SIZE viewSize;
    HDC hdc;
    CRect clientRect;
    HRESULT hr;

    if( pISurround == NULL )
        return FALSE;

    // Get client rect
    this->GetClientRect( clientRect );

    // Get the maximum view size we can get (zoom = 1:1)
    m_pISurround->GetMaxViewSize( 1.0f, &maxSize );
    // Calculate the view size we will create
    viewSize.cx = min( clientRect.Size().cx, maxSize.cx );
    viewSize.cy = min( clientRect.Size().cy, maxSize.cy );

    // Create the view
    hr = m_pISurround->GetView( &viewSize, 1.0f, 100, &m_pISurroundView );

    if( FAILED(hr) )
        return hr;

    // Get the view range
    m_pISurroundView->GetViewRange( &m_viewExtents, &m_location.latitude );

    // see if it's a 360 degree image...
    m_b360Image = ( m_viewExtents.left == 0  &&
                    m_viewExtents.right == MAX_ARCSECONDS ) ? TRUE : FALSE;

    // ...if not, then make sure we are in a valid location
    if( !m_b360Image )
    {
        if( m_location.longitude < m_viewExtents.left )
            m_location.longitude = m_viewExtents.left;
        else if( m_location.longitude > m_viewExtents.right )
            m_location.longitude = m_viewExtents.right;
    }

    // Delete any previous bitmap
    if( m_hOffscreenBitmap != NULL )
        DeleteObject( m_hOffscreenBitmap );

    // Create off screen bitmap and palette
    m_offscreenInfo.header.biBitCount = m_pISurroundView->GetDepth();

    hdc = ::GetDC( m_hWnd );

    if( m_offscreenInfo.header.biBitCount == 8 )
        m_hPalette = CreateIdentityPalette( 256 );

    m_offscreenInfo.header.biHeight = viewSize.cy;
    m_offscreenInfo.header.biWidth = viewSize.cx;
    m_hOffscreenBitmap = CreateDIBSection( hdc, (BITMAPINFO*)&m_offscreenInfo,
                                       DIB_RGB_COLORS, &m_pOffscreenBits, NULL, 0 );
    ::ReleaseDC( NULL, hdc );

    return S_OK;
}
```

4.1.9ISurround::ReadMore()

ISurround::ReadMore

BOOL ReadMore(UINT *nAmount* **);**

Parameters

nAmount	Specifies the amount of processing to do before returning. This value should range from 0 to 100. A value of 100 will cause the entire image to be read in.

Comments

This function is used to allow background preprocessing of the image. For the time being, it is used to read the image in from disk in the background. This function should be called from a thread in Win32 or from a message loop in a Win32s application.

Return Values

TRUE if there is more background loading to do; otherwise FALSE.

Note

At the present time, this function always returns TRUE

4.1.10ISurround::SetBits()

ISurround::SetBits

HRESULT SetBits(SPHERE_POINT **pPt*, **SIZE** **pSize*, **BITMAPINFOHEADER** **pbmi,* **VOID** **pvBits* **);**

Parameters

pPt	Points to an **SPHERE_POINT** which defines the image location at the center of the viewport.
pSize	Points to a **SIZE** that defines the size of the viewport in pixels.
pbmi	Address of structure containing bitmap size, format, and color data.
pvBits	Pointer to the source bits.

Comments

The function writes raw bits from a source bitmap to the document at the specified location. It is the opposite of ***ISurround::GetBits()***.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INCOMPATIBLE_SURFACE	*pbmi* is incompatible with the document image.
SV_E_INVALID_PARAMETER	Invalid parameter.
SV_E_RANGE	One or more arguments are out of range.

4.1.11ISurround::SetHorizon()

ISurround::SetHorizon

HRESULT SetHorizon(int *iHorizon* **);**

Parameters

iHorizon	Horizon of the image.

Comments

The function set the horizon of the image.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_RANGE	iHorizon is out of range.

4.1.12ISurround::UpdateStreamLength()

ISurround::UpdateStreamLength

HRESULT UpdateStreamLength(DWORD *dwValidBytes*, **BOOL FAR *** *pbUpdate* **);**

Parameters

dwValidBytes	Number of valid bytes in the stream.
pbUpdate	Offset to first byte of Image data. This is normally 0..

Comments

The function updates the stream length and reads and decompresses any new image data and adds it to the current image. This functions is usually called from the ***IBindStatusCallback::OnDataAvailable()*** method as new data becomes available. If the image has been compressed using the Progressive JPEG methid, this function should be called repeatedly until SV_E_COMPLETE is returned. This allows you to find out when the image has been fully decompressed.

Return Values

Value	Meaning
S_OK	Success.
SV_E_INVALID_STREAM	The stream is invalid or NULL.
SV_E_COMPLETE	The JPEG image has been fully decompressed.

4.2ISurroundView

4.2.1ISurroundView::Draw()

ISurroundView::Draw

HRESULT Draw(SPHERE_POINT ***pPt*, BITMAPINFOHEADER ****pbmi,* **VOID ****pvBits,* **RECT** ****pRect*, **int** *iDrawQuality* **);**

Parameters

pPt	Points to an **SPHERE_POINT** which defines the image location at the center of the viewport.
pbmi	Address of structure containing bitmap size, format, and color data of the surface.
pvBits	Pointer to the destination bits.
pRect	Points to a **RECT** that defines the area of the bitmap to be drawn. If NULL the entire surface is assumed.
iDrawQuality	Draw quality. This value ranges from 0 (lowest) to 100 (highest) and sets the quality of the image that is returned. This value is normally set to 100 but since lower quality images can be rendered faster, this value can be changed to optimize drawing time when high quality images are not needed (e.g., during panning).

Comments

The function draws the image centered on the location specified by *pPt* on surface defined by *pbmi*,*pvBits*. The size of the surface defined in *pbmi* must match the current size of the viewport.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INCOMPATIBLE_SURFACE	*pbmi* is incompatible with the document image.
SV_E_INVALID_PARAMETER	Invalid parameter.
SV_E_RANGE	One or more arguments are out of range.

Example

The following example shows how ***ISurroundView::Draw()*** is used to draw the image into an offscreen bitmap and then to the screen.

```
void CView::OnDraw( CDC* pDC )
{
    HDC hdc;
    CRect imageRect;
    CRect clientRect;
    HRESULT hr;

    // Only do this if we have a view
    if( m_pISurroundView == NULL )
        return;

    // Get the HDC
    hdc = pDC->GetSafeHdc();

    // Select the palette
    HPALETTE hOldPalette = ::SelectPalette( hdc, m_hPalette, FALSE );
    ::RealizePalette(hdc);

    // imageRect is the size of the offscreen bitmap
    imageRect.SetRect( 0, 0,
                      m_offscreenInfo.header.biWidth,
                      m_offscreenInfo.header.biHeight );

    // Tell the view to draw into our offscreen bitmap
    hr = m_pISurroundView->Draw( &m_location, &m_offscreenInfo.header,
                                 m_pOffscreenBits, &imageRect );
    if( FAILED(hr) )
        return;

    // Get the size of the client rect
    GetClientRect( &clientRect );

    // NOTE: in practice, imageRect and clientRect should be the same.

    // Paint it to the screen
    StretchDIBits( hdc,
                   imageRect.left, imageRect.top,
                   imageRect.Width(), imageRect.Height(),
                   clientRect.left, clientRect.top,
                   clientRect.Width(), clientRect.Height(),
                   m_pOffscreenBits, (BITMAPINFO *)&m_offscreenInfo,
                   DIB_RGB_COLORS, SRCCOPY );
}
```

4.2.2ISurroundView::GetColors()

ISurroundView::GetColors

HRESULT GetColors(UINT *iFirstEntry*, **UINT** *iNumEntries*, **RGBQUAD FAR*** *pColors* **);**

Parameters

iFirstEntry	Index of first color entry to return.
iNumEntries	The number of color entries to return.
pColors	Pointer to an array of RGBQUAD color entries.

Comments

The function retrieves color entries for the current viewport.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INCOMPATIBLE_SURFACE	Image does not have a palette.
SV_E_INVALID_PARAMETER	Invalid parameter.
SV_E_RANGE	*iNumEntries* specifies more colors than are present in the image.

4.2.3ISurroundView::GetDepth()

ISurroundView::GetDepth

UINT GetDepth();

Return Value

The bit depth of the view in bits per pixel.

Example

See the example for ***ISurround::GetView()*** on page 11.

4.2.4ISurroundView::GetSize()

ISurroundView::GetSize

HRESULT GetSize(SIZE ****pSize* **);

Parameters

pSize	Address of a SIZE structure to receive the size of the view.

Comments

The function returns the size of the view in pixels. This is the size the view was created with in ***ISurround::GetView()***.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INVALID_PARAMETER	*pSize* is not a valid pointer.

4.2.5ISurroundView::GetViewRange()

ISurroundView::GetViewRange

HRESULT GetViewRange(SPHERE_RECT ****pExtent*, ARCSECONDS FAR*** *pLatitudeCenter* **);**

Parameters

pExtent	Points to a **SPHERE_RECT** that will receive the extents of the view.
pLatitudeCenter	Points to an ARCSEONDS that will receive the latitude of the center of the image.

Comments

The function is used to get the range of valid location values that can be passed to ***ISurroundView::Draw()*** as well as the latitude of the center of the image. If either *pExtent* or *pLatitudeCenter* is 0, then that parameter will not be returned.

Return Values

Value	**Meaning**
S_OK	Success.

Example

See the example for ***ISurround::GetView()*** on page 11.

4.2.6ISurroundView::GetZoom()

ISurroundView::GetZoom

float GetZoom( );

Return Value

The Zoom factor of the viewport.

Comments

The function returns the zoom factor of the viewport.

4.2.7ISurroundView::SphereToView()

ISurroundView::SphereToView

HRESULT SphereToView(SPHERE_POINT ****pSpt*, **SPHERE_POINT **pSpTangent*, **POINT ****pPt* **);**

Parameters

pSpt	Points to a **SPHERE_POINT** that specifies a location in the document.
pSpTangent	Points to a **SPHERE_POINT** that defines the viewport's point of tangency on the sphere. This will usually be point that defines the center of the viewport.
pPt	Points to a **POINT** that will receive the converted coordinate.

Comments

The function converts spherical coordinates of the document to rectangular coordinates of the viewport.

Return Values

Value	Meaning
S_OK	Success.
SV_E_INVALID_PARAMETER	Invalid parameter.
SV_E_RANGE	One or more parameters are out of range.

4.2.8ISurroundView::ViewToSphere()

ISurroundView::ViewToSphere

HRESULT ViewToSphere(POINT ***pPt*, SPHERE_POINT** **pSpTangent*, **SPHERE_POINT** **pSpt* **);**

Parameters

pPt	Points to a **POINT** that specifies a location relative to the viewport.
pSpTangent	Points to a **SPHERE_POINT** that defines the viewport's point of tangency on the sphere. This will usually be point that defines the center of the viewport.
pSpt	Points to a **SPHERE_POINT** that will receive the converted coordinate.

Comments

The function converts rectangular coordinates of the viewport to spherical coordinates of the document.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INVALID_PARAMETER	Invalid parameter.
SV_E_RANGE	One or more parameters are out of range.

5Surround.lib, the C library

To instantiate an ISurround object you can use one three C functions contained in the file SURROUND.LIB.

5.1.1PanoramicSurroundFromDib()

PanoramicSurroundFromDib

HRESULT PanoramicSurroundFromDIB(LPBITMAPINFOHEADER *lpbmi*, **LPRGBQUAD** *lpColors*, **LPVOID** *lpBits*, **int** *iHorizon*, **ARCSECONDS** *extent*, **ISurround**** *ppISurround* **);**

Parameters

lpbmi	Pointer to BITMAPINFOHEADER.
lpColors	Pointer to the images color table which is an array of RGBQUAD colors. This parameter can be NULL for images that do not have a color table. (ex. 24 bit images)
lpBits	Pointer to image data.
iHorizon	Horizon of image. The Horizon of the image controls the internal image correction for cylindrical images and is normally the center of the image (1/2 the image height) but can be different due to cropping of the original image
extent	Horizontal extent of the image in arcseconds. This should be set to MAX_ARCSECONDS (defined in surround.h) for 360° images.
ppISurround	Pointer to returned ISurround interface

Comments

The function creates an ISurround object and uses the image data to initialize it.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INCOMPATIBLE_SURFACE	Image format is not supported.

Any HRESULT returned by QueryInterface() or CoCreateInstance().

5.1.2PanoramicSurroundFromFile()

PanoramicSurroundFromFile

HRESULT PanoramicSurroundFromFile(LPCTSTR *lpszFilename*, **UINT** *iDepthRequested*, **ISurround**** *ppISurround* **);**

Parameters

lpszFilename	Filename (with path) of a Surround Video Image file (.SVI file).
iDepthRequested	Since certain Surround Video Images can support both 8 and 24 bit ISurrounds, this parameter is used to determine what mode the SVI file or stream is opened with. If the requested bit depth is not available, **SV_E_INCOMPATIBLE_SURFACE** is returned.
ppISurround	Pointer to returned ISurround interface

Comments

This function creates an ISurround based on a file. This file is assumed to have been created by SVEdit or another application that adheres to the Storage/Stream naming convention used by SVEdit .

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INCOMPATIBLE_SURFACE	Image format is not supported.

Any HRESULT returned by StgOpenStorage(), QueryInterface() or CoCreateInstance().

5.1.3PanoramicSurroundFromStream()

PanoramicSurroundFromStream

HRESULT PanoramicSurroundFromStream(IStream __RPC_FAR **pStream*, **UINT** *iDepthRequested*, **ISurround**** *ppISurround* **);**

Parameters

pStream	Stream pointer as returned from IStorage::OpenStream(). The stream is assumed to contain the Surround Video Image data.
iDepthRequested	Since certain Surround Video Images can support both 8 and 24 bit ISurrounds, this parameter is used to determine what mode the SVI file or stream is opened with. If the requested bit depth is not available, **SV_E_INCOMPATIBLE_SURFACE** is returned.
ppISurround	Pointer to returned ISurround interface

Comments

This function creates an ISurround based on a stream.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INCOMPATIBLE_SURFACE	Image format is not supported.

Any HRESULT returned by QueryInterface() or CoCreateInstance().

5.1.4PanoramicSurroundFromPartialStream()

PanoramicSurroundFromPartialStream

HRESULT PanoramicSurroundFromPartialStream(IStream __RPC_FAR **pStream*, **UINT** *iDepthRequested*, **ISurround**** *ppISurround,* **DWORD** *dwValidBytes,* **DWORD** *dwOrigin* **);**

Parameters

pStream	Stream pointer as returned from IStorage::OpenStream(). The stream is assumed to contain the Surround Video Image data.
iDepthRequested	Since certain Surround Video Images can support both 8 and 24 bit ISurrounds, this parameter is used to determine what mode the SVI file or stream is opened with. If the requested bit depth is not available, **SV_E_INCOMPATIBLE_SURFACE** is returned.
ppISurround	Pointer to returned ISurround interface
dwValidBytes	Number of valid bytes in the stream.
dwOrigin	Offset to the first byte of Image data. This is normally 0.

Comments

This function creates an ISurround based on a partial stream. This function is used when progressively downloading an image when the image needs to be viewed as it is being downloaded Once an ISurround is created, the stream length is updated by calls to ***ISurround::UpdateStreamLength()***.

Return Values

Value	**Meaning**
S_OK	Success.
SV_E_INCOMPATIBLE_SURFACE	Image format is not supported.
SV_E_IMPROPERMODE	Only compressed, striped images are supported.
SV_E_INSUFFICIENTBYTES	Insufficient number of bytes in the data stream to instanciate an ISurround. User should call again when more data is available.

Any HRESULT returned by QueryInterface() or CoCreateInstance().

6Data Structures and Types

6.1ARCSECONDS

ARCSECONDS is a measurement of “Seconds of Arc” in polar coordinates. There are 360 degrees in a circle. Each degree is 60 minutes and each minute has 60 seconds. For example, the measurement of 20˚16´50˝ would be equal to (20*60*60) + (60*16) + 50 = 73010 seconds.

```
typedef long ARCSECONDS;
```

The coordinate system for the document is as follows:

Latitude ranges from -90˚0´0˝(looking up) to +90˚0´0˝(looking down) or from -324000 to +324000 in ARCSECONDS.
Longitude goes from 0˚0´0˝ to 360˚0´0˝ or from a value of ARCSECONDS of 0 to 1296000.

6.2SPHERE_POINT

A SPHERE_POINT is a collection of ARCSECONDS for Latitude and Longitude. It defines a point in spherical coordinates that lie in the inside surface of the document.

```
typedef struct _tagSPHERE_POINT
{
    ARCSECONDS latitude;
    ARCSECONDS longitude;
} SPHERE_POINT;
```

6.3SPHERE_RECT

A SPHERE_RECT is a definition of a size of a viewport in Seconds of Arc.

```
typedef struct _tagSPHERE_RECT
{
    ARCSECONDS left;
    ARCSECONDS top;
    ARCSECONDS right;
    ARCSECONDS bottom;
} SPHERE_RECT;
```

6.4HRESULT return codes

HRESULT error codes returned by ISurround and ISurroundView Interfaces:

Value	**Explanation**
SV_E_INVALID_MESSAGE	Invalid message parameter.
SV_E_INVALID_FLAG	Invalid flag specified.
SV_E_INVALID_ZOOM	Invalid zoom specified.
SV_E_INVALID_LOCATION	The specified location was not contained within the document.
SV_E_INVALID_PARAMETER	Parameter specified was invalid.
SV_E_RANGE	Parameter specified was out of range.
SV_E_INCOMPATIBLE_SURFACE	The surface supplied was incompatible with the desired operation.
SV_E_BUG	Internal BUG.
SV_E_OUTOFMEMORY	Insufficient memory to do the desired operation.

SV_E_NODECOMPRESSOR	No suitable decompressor can be found to decompress the image .
SV_E_INSUFFICIENTBYTES	Not enough bytes in stream to create an ISurround.
SV_E_INVALID_STREAM	Stream is invalid.
SV_E_IMPROPERMODE	Improper Mode.