

VCA MetaRender API Manual

Contents

References	5
Introduction	6
When to use the Video Meta Renderer	7
VCA MetaRender API	8
VMR_InitVCAMetaRender	9
VMR_DeInitVCAMetaRender	10
VMR_SetMode	11
VMR_SetRenderingFlags	12
VMR_SetColor	13
VMR_SetFont	14
VMR_SetFontSize	15
VMR_OnMessage	16
VMR_SetMetaData	18
VMR_ResetMetaData	19
VMR_Render	20
VMR_RegisterCallBack	22
VMR_UnRegisterCallBack	24
VMR_SetZone	25
VMR_RemoveZone	27
VMR_GetZone	28
VMR_SetCounter	30
VMR_RemoveCounter	32
VMR_GetCounter	33
VMR_GetObject	35
VMR_SetObject	37
VMR_SetCalibParams	38
VMR_GetCalibParams	39
VMR_SetModel	40
VMR_RemoveModel	42
VMR_CentreModels	43
Data structures	44
VMR_ERROR_E	44
VMR_MODE_E	46
VMR_RENDER_FLAGS_E	46
VMR_RENDER_COLORS_E	49
VMR_ZONE_TYPE_E	49
VMR_ZONE_STYLE_E	50
VMR_OBJECT_STATUS_E	50
VMR_CALLBACK_FLAGS_E	50
VMR_CALLBACK_STATUS_E	51
VMR_POINT_T	52
VMR_ZONE_DISPLAY_FLAGS	53
VMR_ZONE_T	53

VMR_COUNTER_T	55
VMR_ZNCT_STATUS_E	56
VMR_ZNCT_STATUS_T	56
VMR_CALIB_INFO_T	57
VMR_MODEL_INFO_T	57
VMR_CALLBACK_INFO_T	58
VMR_CB	58
Revision history	59

Glossary

DLL	Dynamic-link Library
DDS	DirectDraw Surface
TBD	To Be Determined
VMR	VCA MetaRender

References

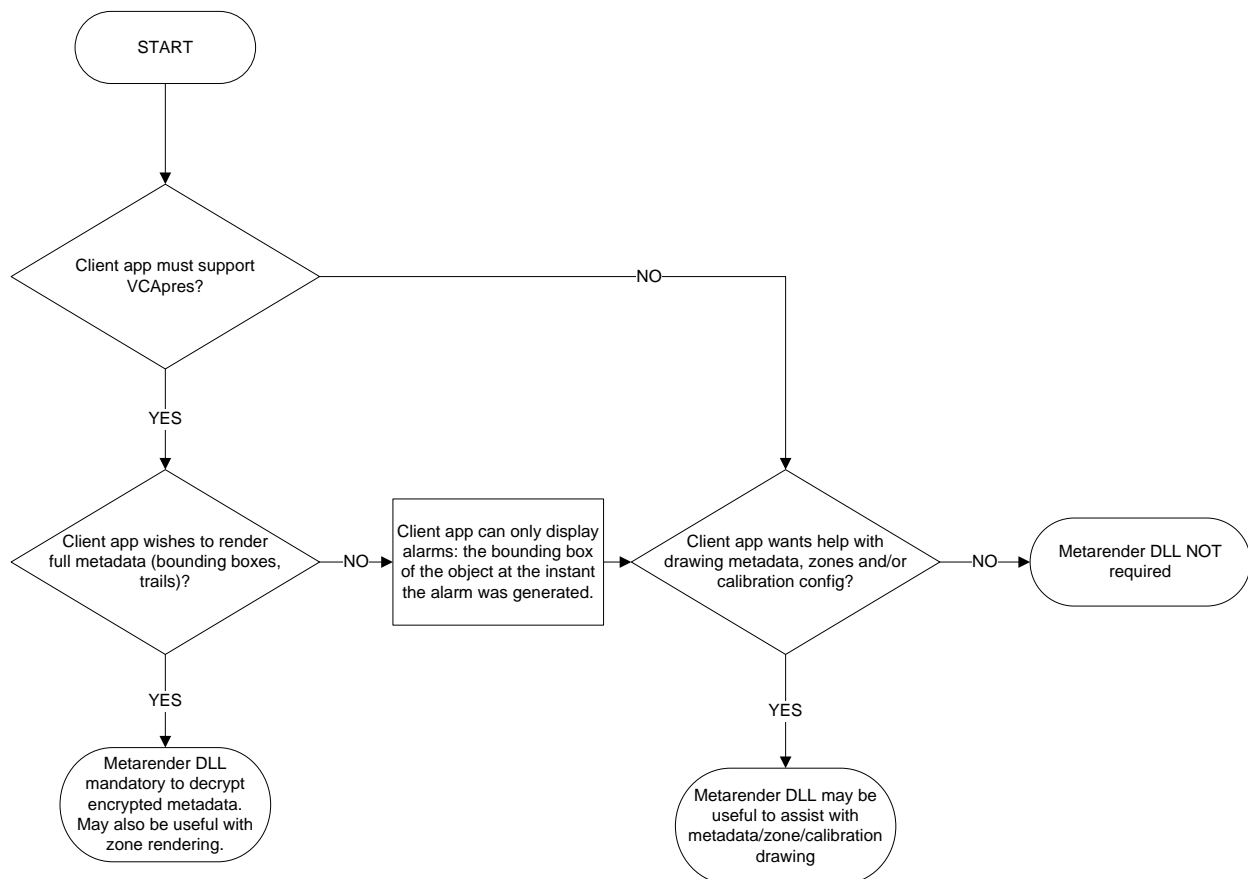
Introduction

This document describes the VCAMetaRender.dll – an API for rendering and simple configuration of VCA information (Metadata/Zones/Counters/Calibration) onto a DirectDraw Surface (DDS). It explains how to integrate the output of the VCA tracking engine into a third party product.

The VCAMetaRender.dll supports running with multiple instances (up to 16), which makes it possible to render up to 16 channel VCA information simultaneously. However, the implementation is very simple. A unique handle is generated for each instance of the VMR library. This handle uniquely identifies the calling context in subsequent library function calls.

When to use the Video Meta Renderer

In order to clarify when the VMR should be used, study the following decision tree to see if the Video Meta Renderer DLL is right for your application:



VCA MetaRender API

VCA MetaRender APIs are mainly designed for handling the drawing of VCA metadata together with 2 different configuration modes: zones/counters configuration and 3D calibration configuration. All API functions have “VMR_” prefix. The following table shows the supported functions for different modes:

VMR API \ Configuration Mode	Zones/Counters Configuration	3D Calibration Configuration
VMR_InitVCAMetaRender	Yes	Yes
VMR_DeInitVCAMetaRender	Yes	Yes
VMR_SetMode	Yes	Yes
VMR_OnMessage	Yes	Yes
VMR_SetRenderingFlags	Yes	Yes
VMR_SetColor	Yes	Yes
VMR_SetAlarmTimer	Yes	Yes
VMR_SetFont	Yes	Yes
VMR_SetFontSize	Yes	Yes
VMR_SetMetaData	Yes	Yes
VMR_ResetMetaData	Yes	Yes
VMR_Render	Yes	Yes
VMR_RegisterCallBack	Yes	Yes
VMR_UnRegisterCallBack	Yes	Yes
VMR_SetZone	Yes	No
VMR_RemoveZone	Yes	No
VMR_GetZone	Yes	No
VMR_SetCounter	Yes	No
VMR_RemoveCounter	Yes	No
VMR_GetCounter	Yes	No
VMR_GetObject	Yes	No
VMR_SetObject	Yes	No
VMR_SetCalibParams	No	Yes
VMR_GetCalibParams	No	Yes
VMR_SetModel (NOT IMPLEMENTED)	No	Yes
VMR_RemoveModel (NOT IMPLEMENTED)	No	Yes
VMR_CentreModels (NOT IMPLEMENTED)	No	Yes

VMR_InitVCAMetaRender

VMR_ERROR_E VMR_InitVCAMetaRender(VMR_HANDLE *hhVMRHandle)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hhVMRHandle

The pointer to the VMR handle with a valid VMR_HANDLE pointer if the call succeeds

Remarks

Initialize the VMR, this function needs to be called before any other VMR function. For multiple-channel application, this function should be called multiple times to retrieve multiple handles.

Example

```
#include "VCAMetaRenderAPI.h" // include API header file

VMR_HANDLE  hVMR;
VMR_ERROR_E eError;
eError = VMR_InitVCAMetaRender( &hVMR );
if ( eError != VMRRERR_SUCCESS )
{
    // Failed?
    printf("Failed to initialize VMR, error code=%d", (int) eError );
}
```

VMR_DeInitVCAMetaRender

VMR_ERROR_E VMR_DeInitVCAMetaRender(VMR_HANDLE *hhVMRHandle)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hhVMRHandle

The pointer to the VMR handle

Remarks

De-initialize the VMR.

Example

```
#include "VCAMetaRenderAPI.h" // include API header file

VMR_HANDLE hVMR;
VMR_ERROR_E eError;

// Initialize
eError = VMR_InitVCAMetaRender( &hVMR );
ASSERT( eError == VMRERR_SUCCESS );
// Do something here:
VMR_SetMode(VMR_ZNCTCFG);      // Set to zone/counter configuration mode

.
.
.

// De-initialize
eError = VMR_DeInitVCAMetaRender( &hVMR );
ASSERT( eError == VMRERR_SUCCESS );
```

VMR_SetMode

VMR_ERROR_E VMR_SetMode(VMR_HANDLE hVMRHandle, VMR_MODE_E eMode)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

eMode

The rendering mode of the VMR engine: VMR_ZNCTCFG for zone/counter configuration mode; VMR_CALIBCFG for 3D calibration mode. Refer to [VMR_MODE_E](#) for details.

Remarks

Set the rendering mode for VMR. The mode needs to be set before using any other APIs except VMR_InitVCAMetaRender. After this, the mode can be changed without re-initializing the VMR.

Example

```
#include "VCAMetaRenderAPI.h" // include API header file

VMR_HANDLE  hVMR;
VMR_ERROR_E eError;

// Initialize
eError = VMR_InitVCAMetaRender( &hVMR );
ASSERT( eError == VMRERR_SUCCESS );

// Set to zone/counter configuration mode
VMR_SetMode( hVMR, VMR_ZNCTCFG );
```

VMR_SetRenderingFlags

VMR_ERROR_E VMR_SetRenderingFlags(VMR_HANDLE hVMRHandle, int nRenderFlags)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

nRenderFlags

The render flags for VMR, refer to [VMR_RENDER_FLAGS_E](#) for details.

Remarks

Set the rendering flags for VMR.

Example

```
VMR_HANDLE hVMR;
VMR_ERROR_E eError;

.
.

// Set to zone/counter configuration mode
eError = VMR_SetMode( hVMR, VMR_ZNCTCFG );
ASSERT( eError == VMRERR_SUCCESS );

// Render zones, counters and blobs
eError = VMR_SetRenderingFlags( hVMR, VRF_ZONES | VRF_COUNTERS | VRF_BLOBS );
ASSERT( eError == VMRERR_SUCCESS );
```

VMR_SetColor

VMR_ERROR_E VMR_SetColor(VMR_HANDLE hVMRHandle, VMR_RENDER_COLORS_E eColorIdx, COLORREF crColor)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

eColorIdx

The index of color to be set, refer to [VMR_RENDER_COLORS_E](#) for details

crColor

The RGB color value in xxRRGGBB format: **(R<<16) + (G<<8) + B**.

Remarks

Set the rendering colors for VMR.

Example

```
VMR_HANDLE hVMR;
VMR_ERROR_E eError;

.
.
.

// Render zones, counters and blobs
eError = VMR_SetRenderingFlags( hVMR, VRF_ZONES | VRF_COUNTERS | VRF_BLOBS )
ASSERT( eError == VMRERR_SUCCESS );

// Set cyan to be the color for blobs
#define RGB2(r,g,b)    (r<<16)+(g<<8)+b
COLORREF crBlobs = RGB2(0,255,255); // cyan

eError = VMR_SetColor( hVMR, VRC_BLOB, crBlobs );
ASSERT( eError == VMRERR_SUCCESS );
```

VMR_SetFont

VMR_ERROR_E VMR_SetFont (VMR_HANDLE hVMRHandle, HFONT hFont)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

hFont

The font handle

Remarks

Set the font handle for annotation within the VMR.

Example

```
VMR_HANDLE  hVMR;
VMR_ERROR_E eError;

.
.
.

// Set to zone/counter configuration mode
eError = VMR_SetMode( hVMR, VMR_ZNCTCFG );
ASSERT( eError == VMRERR_SUCCESS );

hFont = ::CreateFont( 26, 0, 0, 0, FW_NORMAL, FALSE,
    FALSE, FALSE, DEFAULT_CHARSET, OUT_DEFAULT_PRECIS, CLIP_DEFAULT_PRECIS,
    ANTIALIASED_QUALITY, VARIABLE_PITCH, _T("Arial") );

// Set the font size to be 32 pixels
eError = VMR_SetFont( hVMR, hFont );
```

VMR_SetFontSize

VMR_ERROR_E VMR_SetFontSize(VMR_HANDLE hVMRHandle, int nFontSize)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

nFontSize

The font size in pixels

Remarks

Set the font size in pixels for annotation within the VMR.

Example

```
VMR_HANDLE hVMR;  
VMR_ERROR_E eError;  
  
.  
.  
.  
  
// Set to zone/counter configuration mode  
eError = VMR_SetMode( hVMR, VMR_ZNCTCFG );  
ASSERT( eError == VMRERR_SUCCESS );  
  
// Set the font size to be 32 pixels  
eError = VMR_SetFontSize( hVMR, 32 );
```

VMR_OnMessage

VMR_ERROR_E VMR_OnMessage(VMR_HANDLE hVMRHandle, HWND hWnd, unsigned int nCode, WPARAM wParam, LPARAM lParam)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

hWnd

Identifies the parent window.

nCode

Specifies the message.

wParam

Specifies additional message-specific information.

lParam

Specifies additional message-specific information.

Remarks

Send the standard windows message using this function if you wish the VMR to handle the mouse/keyboard messages for configuration. For zone/counter configuration mode, it handles both the zone movement and node adjustment. For camera calibration configuration mode, it handles both camera height and tilt angle adjustment.

Example

```
// This is the callback function
LRESULT CALLBACK CStreamWindow::_lpWndProc( HWND _hwnd, UINT _msg, WPARAM _wParam,
LPARAM _lParam )
{
    // Pass the message to VMR
    VMR_OnMessage( m_hVMR, _hwnd, _msg, _wParam, _lParam );

    // Do some other message handling stuff
    switch( _msg )
    {
```



```
case WM_CREATE:
    {
        // Do something here:
    }
    .
    .
    .
default:
    return DefWindowProc(_hwnd, _msg, _wParam, _lParam);
}
```

VMR_SetMetaData

VMR_ERROR_E VMR_SetMetaData(VMR_HANDLE hVMRHandle, unsigned char *pMetadata, int nLength)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

pMetadata

The pointer to the raw VCA metadata, which might be encrypted depending on the VCA license.

nLength

The length of the metadata buffer in bytes.

Remarks

Pass the raw VCA metadata to VMR for rendering.

Example

```
unsigned char *pMetaData;  
int nLength = 0;  
  
// Allocate 128Kbytes as metadata buffer  
pMetaData = new char[128*1024];  
  
// Get the metadata using HTTP  
GetMetaFromHTTP( pMetaData, &nLength );  
  
// Send the metadata to VMR  
eError = VMR_SetMetaData( hVMR, pMetaData, nLength );  
ASSERT( eError == VMRERR_SUCCESS );  
  
delete[] pMetaData;
```

VMR_ResetMetaData

VMR_ERROR_E VMR_ResetMetaData(VMR_HANDLE hVMRHandle)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

Remarks

Clear the VCA metadata.

Example

```
// Reset the metadata
eError = VMR_ResetMetaData( hVMR );
ASSERT( eError == VMRERR_SUCCESS );
```

VMR_Render

VMR_ERROR_E VMR_Render(VMR_HANDLE hVMRHandle, LPDIRECTDRAWSURFACE7 lpDDDstSurface, LPRECT lpDstRect)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

lpDDDstSurface

The DirectDraw Surface (DDS) for rendering

lpDstRect

The destination rectangle on the DDS where the annotation should be rendered.

Remarks

Render all VCA annotation (zones/rules/metadata/calibration) onto the destination area of DirectDraw Surface. Please note this surface **must have DDSCAPS_3DDEVICE** set when creating the surface.

Example

```
LPDIRECTDRAW7      pDD = NULL;
LPDIRECTDRAWSURFACE7 pddsTexture;
DDSURFACEDESC2     ddsd;
DWORD              dwWidth = 1024;           // width of DirectDraw surface
DWORD              dwHeight = 1024;          // height of DirectDraw surface
VMR_HANDLE         hVMR;

_INIT_DIRECTDRAW_STRUCT( ddsd );

// Create DirectDraw here
.
.
.

// Set all the surface descriptions
ddsd.dwFlags      = DDSD_WIDTH | DDSD_HEIGHT | DDSD_CAPS;
ddsd.dwWidth      = dwWidth;
ddsd.dwHeight     = dwHeight;
```

```
ddsd.ddsCaps.dwCaps= DDSCAPS_VIDEMEMORY|DDSCAPS_3DDEVICE |DDSCAPS_LOCALVIDMEM ;  
// must have DDSCAPS_3DDEVICE set  
  
// Create DirectDraw Surface  
hr = _pDD->CreateSurface( &ddsd, &pddsTexture, NULL );  
  
// Set the rect for rendering  
RECT rcDst;  
SetRect( &rcDst , 0, 0, dwWidth, dwHeight);  
  
// Render all  
eError = VMR_Render( hVMR, pDDS,&rcDst);  
ASSERT( eError == VMRERR_SUCCESS );
```

VMR_RegisterCallBack

VMR_ERROR_E VMR_RegisterCallBack(VMR_HANDLE hVMRHandle, VMR_CALLBACK_INFO_T *pCallBackInfo)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

pCallBackInfo

The pointer to the call back information to be set, refer to VMR_CALLBACK_INFO_T for details.

Remarks

Register the call-back function for VMR, refer to VMR_CB for the call-back function syntax. Please note that the call-back function can be only registered once unless the current call-back function is unregistered by calling VMR_UnRegisterCallBack beforehand.

Example

```
//static call-back function
void static __stdcall VMRCBFunc( VMR_HANDLE hVMRHandle, unsigned int uiCBStatus, void *pData,
void *pUserData )
{
    // Zone triggered call-back?
    if ( uiCBStatus & VMR_ZONE_MSK )
    {
        VMR_ZONE_T *pZone = (VMR_ZONE_T *) pData;
        if ( pData )
        {
            printf("Zoneid=%d,Selected=%d,Hovered=%d\n",pZone->usZoneId,pZone->uiStatus&VMR_OBJECT_SELECTED,pZone->uiStatus&VMR_OBJECT_HOVERED);
        }
    }
    // Counter triggered call-back?
    if ( uiCBStatus & VMR_COUNTER_MSK )
    {
        VMR_COUNTER_T *pCounter = (VMR_COUNTER_T *) pData;
        if ( pData )
    }
```

```
        {
            printf("Counterid=%d,Selected=%d,Hovered=%d\n",pCounter-
>usCounterId,pCounter->uiStatus&VMR_OBJECT_SELECTED,pCounter-
>uiStatus&VMR_OBJECT_HOVERED);
        }
    }
}

void main()
{
    VMR_HANDLE  hVMR;
    VMR_ERROR_E eError;
    VMR_CALLBACK_INFO_T tCallBackInfo;
    // Init VMR
    eError = VMR_InitVCAMetaRender( &hVMR );
    ASSERT( eError == VMRERR_SUCCESS );

    // Set up call-back function for VMR
    tCallBackInfo.pCB = VMRCBFunc;
    tCallBackInfo.uiFlags = VMR_CALLBACK_ZONE | VMR_CALLBACK_COUNTER;
    tCallBackInfo.pUserData = 0;
    eError = VMR_RegisterCallBack( hVMR, &tCallBackInfo );
    ASSERT( eError == VMRERR_SUCCESS );
}
```

VMR_UnRegisterCallBack

VMR_ERROR_E | VMR_UnRegisterCallBack(VMR_HANDLE hVMRHandle)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

Remarks

Un-register the call-back function for VMR.

Example

```
VMR_HANDLE hVMR;
VMR_ERROR_E eError;
VMR_CALLBACK_INFO_T tCallBackInfo;
// Init VMR
eError = VMR_InitVCAMetaRender( &hVMR );
ASSERT( eError == VMRERR_SUCCESS );

// Set up call-back function for VMR
tCallBackInfo.pCB = VMRCBFunc;
tCallBackInfo.uiFlags = VMR_CALLBACK_ZONE | VMR_CALLBACK_COUNTER;
tCallBackInfo.pUserData = 0;
eError = VMR_RegisterCallBack( hVMR, &tCallBackInfo );
ASSERT( eError == VMRERR_SUCCESS );

.
.
.
eError = VMR_UnRegisterCallBack( hVMR );
ASSERT( eError == VMRERR_SUCCESS );
```


VMR_SetZone

VMR_ERROR_E VMR_SetZone(VMR_HANDLE hVMRHandle, VMR_ZONE_T *pZone)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

pZone

The pointer to the zone to be set, refer to [VMR_ZONE_T](#) for details.

Remarks

Set the properties of a zone within the VMR. If a zone with same id exists in VMR, the zone settings will be replaced.

Example

```
#include "VCAMetaRenderAPI.h" // include API header file

VMR_HANDLE  hVMR;
VMR_ERROR_E eError;

// Initialize
eError = VMR_InitVCAMetaRender( &hVMR );
if ( eError != VMRERR_SUCCESS )
{
    // Failed?
    printf("Failed to initialize VMR, error code=%d", (int) eError );
}

// Set to zone/counter configuration mode
VMR_SetMode( hVMR, VMR_ZNCTCFG );

// Render zones, counters and blobs
eError = VMR_SetRenderingFlags( hVMR, VRF_ZONES | VRF_COUNTERS | VRF_BLOBS );
ASSERT( eError == VMRERR_SUCCESS );

#define RGB2(r,g,b)    (r<<16)+(g<<8)+b
// Set a max zone with red color
VMR_ZONE_T tZone;
```

```
tZone.usZoneId = 0;
tZone.eZoneType = VMR_ALARM_ZONE;
tZone.eZoneStyle = VMR_ZONE_POLYGON;
strcpy( tZone.strZoneName, "Zone Test" );
tZone.uiColor = RGB2(255,0,0); // red color
tZone.ucDisplay = 1;
// All the points are normalized to 0-65535, 0 -> 0%, 65535 -> 100%
tZone.uiTotalPoints = 5;
tZone.pPoints[0].x = 0;          tZone.pPoints[0].y = 0;
tZone.pPoints[1].x = 65535;      tZone.pPoints[1].y = 0;
tZone.pPoints[2].x = 65535;      tZone.pPoints[2].y = 65535;
tZone.pPoints[3].x = 0;          tZone.pPoints[3].y = 65535/2;
tZone.pPoints[4].x = 0;          tZone.pPoints[4].y = 0;
eError = VMR_SetZone( hVMR, &tZone );
if ( eError != VMRERR_SUCCESS )
    printf( "Failed to set zone, error code = %d\n", (int) eError );
else
    printf( "Set zone [%d] successfully\n", tZone.usZoneId );
```

VMR_RemoveZone

VMR_ERROR_E VMR_RemoveZone(VMR_HANDLE hVMRHandle, unsigned short usZoneId)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

usZoneId

The id of the zone to be removed

Remarks

Remove a zone from the VMR. If the zone does not exist, this function call will fail and return VMRERR_INVALID_ZONEID.

Example

```
VMR_HANDLE  hVMR;
VMR_ERROR_E eError;
unsigned short usZoneId = 1;

.
.
.

// Remove the zone with id 1
eError = VMR_RemoveZone( hVMR, usZoneId );
if ( eError != VMRERR_SUCCESS )
{
    If ( eError == VMRERR_INVALID_ZONEID )
        printf( "Failed to remove zone, invalid zone id [%d]\n", usZoneId );
    else
        printf( "Failed to remove zone, error code = %d\n", (int) eError );
}
else
    printf( "Remove zone [%d] successfully\n", usZoneId );
```

VMR_GetZone

VMR_ERROR_E VMR_GetZone(VMR_HANDLE hVMRHandle, VMR_ZONE_T *pZone)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

pZone

The pointer to the zone. pZone->usZoneId needs to be set before calling this function

Remarks

Get a zone from VMR. If the zone doesn't exist, this function call will fail and return VMRERR_INVALID_ZONEID.

Example

```
VMR_HANDLE hVMR;
VMR_ERROR_E eError;
VMR_ZONE_T tZone;

.
.
.

// Clear
memset( &tZone, 0, sizeof( VMR_ZONE_T ) );

// Set id to be 2, therefore, we are going to load the zone with id 2
tZone.usZoneId = 2;

// Get zone
eError = VMR_GetZone( hVMR, &tZone );
if ( eError != VMRERR_SUCCESS )
{
    If ( eError == VMRERR_INVALID_ZONEID )
        printf( "Failed to get zone, invalid zone id [%d]\n", tZone.usZoneId );
    else
        printf( "Failed to get zone, error code = %d\n", (int) eError );
}
```

```
}  
else  
    printf( "Got zone [%d] successfully\n", tZone.usZoneId );
```

VMR_SetCounter

VMR_ERROR_E VMR_SetCounter(VMR_HANDLE hVMRHandle, VMR_COUNTER_T *pCounter)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

pCounter

The pointer to the counter to be set, refer to the [VMR_COUNTER_T](#) for details

Remarks

Set to the properties of a counter in the VMR. If a counter with same id exists, the properties will be replaced with those specified.

Example

```
#include "VCAMetaRenderAPI.h" // include API header file

VMR_HANDLE  hVMR;
VMR_ERROR_E eError;

// Initialize
eError = VMR_InitVCAMetaRender( &hVMR );
if ( eError != VMRERR_SUCCESS )
{
    // Failed?
    printf("Failed to initialize VMR, error code=%d", (int) eError );
}

// Set to zone/counter configuration mode
VMR_SetMode( hVMR, VMR_ZNCTCFG );

// Render zones, counters and blobs
eError = VMR_SetRenderingFlags( hVMR, VRF_ZONES | VRF_COUNTERS | VRF_BLOBS );
ASSERT( eError == VMRERR_SUCCESS );

#define RGB2(r,g,b)    (r<<16)+(g<<8)+b
// Set a counter with red color
VMR_COUNTER_T tCounter;
```

```
tCounter.usCounterId = 0;
strcpy(tCounter.strCounterName, "Counter Test" );
tCounter.uiColor = RGB2(255,0,0);      // red color
tCounter.ucDisplay = 1;
// All the points are normalized to 0-65535, 0 -> 0%, 65535 -> 100%
// Set up the rectangle area to display counter
tCounter.rcRegion.left = 0;
tCounter.rcRegion.top = 0;
tCounter.rcRegion.right = tCounter.rcRegion.left + 65535/10;
tCounter.rcRegion.bottom = tCounter.rcRegion.top + 65535/10;

eError = VMR_SetCounter( hVMR, &tCounter );
if ( eError != VMRERR_SUCCESS )
    printf( "Failed to set counter, error code = %d\n", (int) eError );
else
    printf( "Set counter [%d] successfully\n", tCounter.usCounterId );
```

VMR_RemoveCounter

VMR_ERROR_E VMR_RemoveCounter(VMR_HANDLE hVMRHandle, unsigned short usCounterId)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

usCounterId

The id of the counter to be removed

Remarks

Removes a counter from VMR. If the counter does not exist, this function call will fail and return VMRERR_INVALID_COUNTERID.

Example

```
VMR_HANDLE  hVMR;
VMR_ERROR_E eError;
unsigned short  usCounterId = 1;

.
.
.

// Remove the counter with id 1
eError = VMR_RemoveCounter( hVMR, usCounterId );
if ( eError != VMRERR_SUCCESS )
{
    If ( eError == VMRERR_INVALID_COUNTERID )
        printf( "Failed to remove counter, invalid counte id [%d]\n", usCounterId);
    else
        printf( "Failed to remove counter, error code = %d\n", (int) eError );
}
else
    printf( "Remove counter [%d] successfully\n", usCounterId );
```


VMR_GetCounter

VMR_ERROR_E VMR_GetCounter(VMR_HANDLE hVMRHandle, VMR_COUNTER_T *pCounter)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

pCounter

The pointer to the counter. pCounter->usCounterId needs to be set before calling this function.

Remarks

Get a counter from the VMR. If the counter doesn't exist, this function call will fail and return VMRERR_INVALID_COUNTERID.

Example

```
VMR_HANDLE hVMR;
VMR_ERROR_E eError;
VMR_COUNTER_T tCounter;

.
.
.

// Clear
memset( &tCounter, 0, sizeof( VMR_COUNTER_T ) );

// Set id to be 3, therefore, we are going to load the counter with id d
tCounter.usCounterId = 3;

// Get counter
eError = VMR_GetCounter( hVMR, &tCounter );
if ( eError != VMRERR_SUCCESS )
{
    If ( eError == VMRERR_INVALID_COUNTERID )
        printf( "Failed to get counter, invalid zone id [%d]\n", tCounter.usCounterId );
    else
        printf( "Failed to get counter, error code = %d\n", (int) eError );
}
```

```
}  
else  
    printf( "Get counter [%d] successfully\n", tCounter.usCounterId);
```

VMR_GetObject

VMR_ERROR_E VMR_GetObject(VMR_HANDLE hVMRHandle, VMR_OBJECT_STATUS_E eObject, VMR_ZNCT_STATUS_T *pObject)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

eObject

The type of the object, either selected or hovered, refer to VMR_OBJECT_STATUS_E for more details.

pObject

A pointer to the selected or hovered object, refer to [VMR_ZNCT_STATUS_T](#) for more details.

Remarks

Get the current selected or hovered object from VMR if the windows messages are sent using VMR_OnMessage. The object might be zone, counter or nothing.

Example

```
VMR_HANDLE  hVMR;
VMR_ERROR_E eError;
VMR_ZNCT_STATUS_T tSelected;

// Clear
memset( &tSelected, 0, sizeof(VMR_ZNCT_STATUS_T) );

// Get the current selected object
eError = VMR_GetObject ( hVMR, VMR_OBJECT_SELECTED, &tSelected );

if ( eError != VMRERR_SUCCESS )
    printf( "Failed to get selected object, error code = %d\n", (int) eError );
else
{
    printf( "Get selected object successfully\n" );
    switch ( tSelected. eType)
    {
```

```
case VOZ_ZONE:
    // A zone is currently selected
    // We know the id of selected zone, get this zone from VMR
    VMR_ZONE_T tZone;
    tZone.usZoneId = tSelected.iObjectIdx;
    VMR_GetZone(&tZone);
    printf( " Zone with name %s is selected!\n", tZone.strZoneName );
    break;

case VOZ_COUNTER:
    // A counter is currently selected
    // We know the id of selected counter, get this counter from VMR
    VMR_COUNTER_T tCounter;
    tCounter.usCounterId = tSelected.iObjectIdx;
    VMR_GetCounter(&tCounter);
    printf( " Counter with name %s is selected!\n", tCounter.strCounterName );
    break;

case VOZ_NONE:
    printf( "Nothing is selected!\n" );
    break;
}
}
```

VMR_SetObject

VMR_ERROR_E VMR_SetObject(VMR_HANDLE hVMRHandle, VMR_OBJECT_STATUS_E eObject, VMR_ZNCT_STATUS_T *pObject)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

eObject

The type of the object, either selected or hovered, refer to VMR_OBJECT_STATUS_E for more details.

pObject

A pointer to the object to be set, refer to [VMR_ZNCT_STATUS_T](#) for more details.

Remarks

Set the current selected or hovered object inside VMR. The new object might be zone, counter or nothing.

Example

```
VMR_HANDLE hVMR;
VMR_ERROR_E eError;
VMR_ZNCT_STATUS_T tHovered;

memset( &tHovered, 0, sizeof(VMR_ZNCT_STATUS_T) );
// Set the current hovered object to be nothing
tHovered.eType = VOZ_NONE;
tHovered.iObjectIdx = -1;
tHovered.iNodeIdx = -1;

// Set the current hovered object
eError = VMR_SetObject ( hVMR, VMR_OBJECT_HOVERED, &tHovered );
ASSERT( eError == VMRERR_SUCCESS);
```

VMR_SetCalibParams

VMR_ERROR_E VMR_SetCalibParams(VMR_HANDLE hVMRHandle, VMR_CALIB_INFO_T *pCalibInfo)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

pCalibInfo

A pointer to the calibration info to be set, refer to the [VMR_CALIB_INFO_T](#) for details.

Remarks

Set the calibration parameters for VMR.

Example

```
#include "VCAMetaRenderAPI.h" // include API header file

VMR_HANDLE  hVMR;
VMR_ERROR_E eError;

// Initialize
eError = VMR_InitVCAMetaRender( &hVMR );
ASSERT( eError == VMRERR_SUCCESS );

// Set to calibration mode
VMR_SetMode( hVMR, VMR_CALIBCFG );

// Set calibration params
VMR_CALIB_INFO_T tCalibInfo;
tCalibInfo.fHeight      = 10.0f; // 10 metres
tCalibInfo.fTilt        = 60.0f; // 60 degrees
tCalibInfo.fFOV         = 40.0f; // 40 degrees
eError = VMR_SetCalibParams( hVMR, &tCalibInfo );
if ( eError != VMRERR_SUCCESS )
    printf( "Failed to set calibration parameters, error code = %d\n", (int) eError );
else
    printf( "Set calibration parameters successfully\n");
```

VMR_GetCalibParams

VMR_ERROR_E VMR_GetCalibParams(VMR_HANDLE hVMRHandle, VMR_CALIB_INFO_T *pCalibInfo)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

pCalibInfo

A pointer to the calibration info, refer to [VMR_CALIB_INFO_T](#) for details.

Remarks

Get the calibration parameters from the VMR.

Example

```
VMR_HANDLE hVMR;
VMR_ERROR_E eError;

.
.
.

// Get the current calibration params
VMR_CALIB_INFO_T tCalibInfo;

eError = VMR_GetCalibParams( hVMR, &tCalibInfo );
if ( eError != VMRERR_SUCCESS )
    printf( "Failed to set calibration parameters, error code = %d\n", (int) eError );
else
    printf( "Set calibration parameters successfully, Camera Height = %.4f metres, Tilt Angle
= %.4f degree, Vertical Field of View = %.4f degree\n", tCalibInfo.fHeight, tCalibInfo.fTilt,
tCalibInfo.fFOV);
```

VMR_SetModel

VMR_ERROR_E VMR_SetModel(VMR_HANDLE hVMRHandle, VMR_MODEL_INFO_T *pModelInfo)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

pModelInfo

The pointer to the calibration model to be set, refer to [VMR_MODEL_INFO_T](#) for details.

Remarks

Set a model for calibration within the VMR. If a model with the same id exists in the VMR, it will be replaced with the model specified.

Example

```
#include "VCAMetaRenderAPI.h" // include API header file

VMR_HANDLE  hVMR;
VMR_ERROR_E eError;

// Initialize
eError = VMR_InitVCAMetaRender( &hVMR );
ASSERT( eError == VMRERR_SUCCESS );

// Set to calibration mode
VMR_SetMode( hVMR, VMR_CALIBCFG );

#define RGB2(r,g,b)    (r<<16)+(g<<8)+b
// Set a max zone with red color
VMR_MODEL_INFO_T tModel;
tModel.usModelId = 0;
tModel.uiColor = RGB2(0,0,255);          // blue color
// All the points are normalized to 0-65535, 0 -> 0%, 65535 -> 100%
tModel.tPosition.x = 65535/2;
tModel.tPosition.y = 65535/2;

eError = VMR_SetModel( hVMR, &tModel );
if ( eError != VMRERR_SUCCESS )
```



```
        printf( "Failed to set model, error code = %d\n", (int) eError );  
else  
    printf( "Set model [%d] successfully\n", tModel.usModelId);
```

VMR_RemoveModel

VMR_ERROR_E VMR_RemoveModel(VMR_HANDLE hVMRHandle, unsigned short usModelId)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

usModelId

The id of the model to be removed

Remarks

Remove a model from the VMR. If the specified model does not exist, this function call will fail and return VMRERR_INVALID_MODELID.

Example

```
VMR_HANDLE  hVMR;
VMR_ERROR_E eError;
unsigned short  usModelId = 1;

.
.
.

// Remove the model with id 1
eError = VMR_RemoveModel( hVMR, usModelId );
if ( eError != VMRERR_SUCCESS )
{
    If ( eError == VMRERR_INVALID_MODELID )
        printf( "Failed to remove model, invalid model id [%d]\n", usModelId);
    else
        printf( "Failed to remove model, error code = %d\n", (int) eError );
}
else
    printf( "Remove model [%d] successfully\n", usModelId );
```

VMR_CentreModels

VMR_ERROR_E VMR_CentreModels(VMR_HANDLE hVMRHandle)

Return Value

VMR error code, please refer to [VMR_ERROR_E](#) for details.

Parameters

hVMRHandle

The VMR instance handle

Remarks

Centres the models for calibration inside VMR.

Example

```
VMR_HANDLE  hVMR;  
VMR_ERROR_E eError;  
  
.  
.  
.  
  
eError = VMR_CentreModel( hVMR );  
ASSERT ( eError == VMRERR_SUCCESS );
```

Data structures

VMR_ERROR_E

```
typedef enum
{
    VMRERR_SUCCESS = 0x00000000,
    VMRERR_EXCEED_VMRHANDLE_LIMITS,
    VMRERR_INVALID_VMRHANDLE,
    VMRERR_ALLOCATE_MEMORY,
    VMRERR_INVALID_MODE,
    VMRERR_INVALID_ZONEID,
    VMRERR_INVALID_COUNTERID,
    VMRERR_INVALID_MODELID,
    VMRERR_INVALID_CALLBACK_PTR,
    VMRERR_INVALID_DIRECTIONARROW,
    VMRERR_CALLBACK_NOT_REGISTERED,
    VMRERR_CALLBACK_ALREADY_REGISTERED,
    VMRERR_INTERNAL_ERROR,
    VMRERR_INVALID_CAPS,
    VMRERR_INVALID_ARGUMENT
}VMR_ERROR_E;
```

VMRERR_SUCCESS

Success

VMRERR_EXCEED_VMRHANDLE_LIMITS

Too many VMR handles, maximum 16 handles

VMRERR_INVALID_VMRHANDLE

Invalid VMR handle

VMRERR_ALLOCATE_MEMORY

Failed when allocating memory inside VMR

VMRERR_INVALID_MODE

No valid mode has been set for VMR, please call VMR_SetMode first.

VMRERR_INVALID_ZONEID

The id of zone cannot be found inside VMR

VMRERR_INVALID_COUNTERID

The id of counter cannot be found inside VMR

VMRERR_INVALID_MODELID

The id of calibration model cannot be found inside VMR.

VMRERR_INVALID_CALLBACK_PTR

The pointer to the call-back function is invalid.

VMRERR_INVALID_DIRECTIONARROW

The direction arrow settings of zone are invalid. Note: the following conditions must be satisfied: $fFinishAngle > fStartAngle$ and $(fFinishAngle - fStartAngle) < 360.00$

VMRERR_CALLBACK_NOT_REGISTERED

The call-back function has not been registered, VMR_UnRegisterCallBack is called before VMR_RegisterCallBack.

VMRERR_CALLBACK_ALREADY_REGISTERED

The call-back function has already been registered; VMR_RegisterCallBack is called multiple times before VMR_UnRegisterCallBack is called.

VMRERR_INTERNAL_ERROR

An error is occurred unexpectedly in the function called.

VMRERR_INVALID_CAPS

Initializing font in the function has failed due to DDERR_INVALID_CAPS of DirectX 7.

VMRERR_INVALID_ARGUMENT

Some of arguments of the call of the function are invalid.

VMR_MODE_E

```
typedef enum
{
    VMR_ZNCTCFG          = 0x0001,
    VMR_CALIBCFG,
    VMR_CFG_RESERVED
}VMR_MODE_E;
```

VMR_ZNCTCFG

Zone/Counter configuration mode

VMR_CALIBCFG

Calibration configuration mode

VMR_CFG_RESERVED

Reserved, do not use this mode

VMR_RENDER_FLAGS_E

This is the bitmask flag. Multiple flags can be used together.

For example, “VRF_ZONES|VRF_COUNTER” means rendering both zones and counters.

```
typedef enum
{
    VRF_NOTHING          = 0x00000000,
    VRF_ZONES             = 0x00000001,
    VRF_COUNTERS         = 0x00000002,
    VRF_BLOBS            = 0x00000004,
    VRF_OBJECTS          = 0x00000008,
    VRF_NON_ALARMS       = 0x00000010,
    VRF_OBJECT_SPEED     = 0x00000020,
    VRF_OBJECT_HEIGHT    = 0x00000040,
    VRF_OBJECT_AREA      = 0x00000080,
    VRF_OBJECT_CLASS     = 0x00000100,
    VRF_OBJECT_WAIT_TIME = 0x00000200,
    VRF_OBJECT_COLSIG    = 0x00000400,
    VRF_OBJECT_MULTICOLOR = 0x00000800,
    VRF_CALIB_HORIZON    = 0x00001000,
    VRF_CALIB_SOLIDGRID  = 0x00002000,
```

```
VRF_SYSMMSG          = 0x10000000,  
VRF_ALL              = 0xFFFFFFFF  
}VMR_RENDER_FLAGS_E;
```

VRF_NOTHING

Nothing will be rendered on the DDS

VRF_ZONES

Render zones on the DDS

VRF_COUNTERS

Render counters on the DDS

VRF_BLOBS

Render blobs on the DDS

VRF_OBJECTS

Render the objects from VCA metadata on the DDS

VRF_NON_ALARMS

Render the non-alarmed objects from VCA metadata on the DDS

VRF_OBJECT_SPEED

Render the object speed from VCA metadata on the DDS

VRF_OBJECT_HEIGHT

Render the object height from VCA metadata on the DDS

VRF_OBJECT_AREA

Render the object area from VCA metadata on the DDS

VRF_OBJECT_CLASS

Render the object classification from VCA metadata on the DDS

VRF_OBJECT_WAIT_TIME

Render the object waiting time from VCA metadata on the DDS

VRF_OBJECT_COLSIG

Render the object colour signature from VCA metadata on the DDS

VRF_OBJECT_MULTICOLOR

Render the objects in multiple colors on the DDS (objects will be rendered in two colors if not set)

VRF_CALIB_HORIZON

Render the horizon line on the DDS for calibration configuration mode

VRF_CALIB_SOLIDGRID

Render the solid grid on the DDS for calibration configuration mode (a transparent grid will be rendered instead if not set)

VRF_SYSMSG

Render all VCA system messages on the DDS, including learning scene, camera tamper detection and etc.

VRF_ALL

Render all VCA information on the DDS

VMR_RENDER_COLORS_E

```
typedef enum
{
    VRC_BLOB                = 0x00000000,
    VRC_ALARM,
    VRC_NONALARM,
    VRC_NUM_RENDERCOLORS
}VMR_RENDER_COLORS_E;
```

VRC_BLOB

The color index for blobs

VRC_ALARM

The color index for alarmed objects

VRC_NONALARM

The color index for non-alarmed objects

VMR_ZONE_TYPE_E

```
typedef enum
{
    VMR_ALARM_ZONE          = 0x0001,
    VMR_PREALARM_ZONE,
    VMR_NONDETECTION_ZONE
}VMR_ZONE_TYPE_E;
```

VMR_ALARM_ZONE

Alarm zone, rendered by a solid brush on DDS

VMR_NONDETECTION_ZONE

Non-detection zone, rendered by a hatched brush on DDS

VMR_PREALARM_ZONE,

NOT IMPLEMENTED

VMR_ZONE_STYLE_E

```
typedef enum
{
    VMR_ZONE_LINE           = 0x0001,
    VMR_ZONE_POLYGON        = 0x0002,
}VMR_ZONE_STYLE_E;
```

VMR_ZONE_LINE

Multi-node detection line

VMR_ZONE_POLYGON

Multi-node detection polygon

VMR_OBJECT_STATUS_E

```
typedef enum
{
    VMR_OBJECT_SELECTED      = 0x0001,
    VMR_OBJECT_HOVERED      = 0x0002
}VMR_OBJECT_STATUS_E;
```

VMR_OBJECT_SELECTED

Object is selected

VMR_OBJECT_HOVERED

Object is hovered

VMR_CALLBACK_FLAGS_E

This is the bitmask flag. Multiple flags can be used together. For example, “VMR_CALLBACK_ZONE|VMR_CALLBACK_COUNTER” means the call-back function will be triggered when either zone or counter is added/changed/deleted/updated.

```
typedef enum
{
    VMR_CALLBACK_NONE        = 0x0000,
    VMR_CALLBACK_ZONE        = 0x0001,
    VMR_CALLBACK_COUNTER     = 0x0002,
```

```

VMR_CALLBACK_CALIBMODEL = 0x0100,
VMR_CALLBACK_CALIBPARAM = 0x0200,
VMR_CALLBACK_ALL        = 0xFFFF
}VMR_CALLBACK_FLAGS_E;

```

VMR_CALLBACK_NONE

Call-back function will not be triggered

VMR_CALLBACK_ZONE

Call-back function will be triggered when a zone is added/changed/deleted/updated.

VMR_CALLBACK_COUNTER

Call-back function will be triggered when a counter is added/changed/deleted/updated.

VMR_CALLBACK_CALIBMODEL

Call-back function will be triggered when a calibration model is added/changed/deleted/updated.

VMR_CALLBACK_CALIBPARAM

Call-back function will be triggered when a calibration parameter is added/changed/deleted/updated.

VMR_CALLBACK_ALL

Call-back function will be triggered for all the situations above.

VMR_CALLBACK_STATUS_E

This is the bitmask flag. Multiple flags can be used together. For example, “VMR_ZONE_ADD|VMR_ZONE_UPD” means the specific zone is just added first and changed later on.

```

typedef enum
{
    VMR_ZONE_ADD          = 0x00000001,
    VMR_ZONE_CHG          = 0x00000002,
    VMR_ZONE_DEL          = 0x00000004,
    VMR_ZONE_UPD          = 0x00000008,
    VMR_ZONE_MSK          = 0x0000000F,

    VMR_COUNTER_ADD       = 0x00000100,

```

```

VMR_COUNTER_CHG      = 0x00000200,
VMR_COUNTER_DEL      = 0x00000400,
VMR_COUNTER_UPD      = 0x00000800,
VMR_COUNTER_MSK      = 0x00000F00,

VMR_CALIBMODEL_ADD   = 0x01000000,
VMR_CALIBMODEL_CHG   = 0x02000000,
VMR_CALIBMODEL_DEL   = 0x04000000,
VMR_CALIBMODEL_UPD   = 0x08000000,
VMR_CALIBMODEL_MSK   = 0x0F000000,

VMR_CALIBPARAM_ADD   = 0x10000000,
VMR_CALIBPARAM_CHG   = 0x20000000,
VMR_CALIBPARAM_DEL   = 0x40000000,
VMR_CALIBPARAM_UPD   = 0x80000000,
VMR_CALIBPARAM_MSK   = 0xF0000000
}VMR_CALLBACK_STATUS_E;

```

VMR_ZONE_ADD/VMR_COUNTER_ADD/VMR_CALIBMODEL_ADD/VMR_CALIBPARAM_ADD

The specific zone/counter/calibration model/calibration parameter is added

VMR_ZONE_CHG/VMR_COUNTER_CHG/VMR_CALIBMODEL_CHG/VMR_CALIBPARAM_CHG

The specific zone/counter/calibration model/calibration parameter is changed

VMR_ZONE_DEL/VMR_COUNTER_DEL/VMR_CALIBMODEL_DEL/VMR_CALIBPARAM_DEL

The specific zone/counter/calibration model/calibration parameter is deleted

VMR_ZONE_UPD/VMR_COUNTER_UPD/VMR_CALIBMODEL_UPD/VMR_CALIBPARAM_UPD

The specific zone/counter/calibration model/calibration parameter is updated

VMR_ZONE_MSK/VMR_COUNTER_MSK/VMR_CALIBMODEL_MSK/VMR_CALIBPARAM_MSK

The bit mask for zone/counter/calibration model/calibration status

VMR_POINT_T

All point values are normalized to 16bits, which is 0-65535. 0 corresponds to 0% and 65535 to 100%.

```

typedef struct
{
    unsigned short    x;           // 16-bit

```

```

        unsigned short        y;           // 16-bit
    }VMR_POINT_T;

```

VMR_ZONE_DISPLAY_FLAGS

```

typedef enum
{
    VMR_ZONE_DISPLAY_DIRECTION_ARROWS    = 0x00000001,
    VMR_ZONE_DISPLAY_LINECOUNTER_A      = 0x00000002,
    VMR_ZONE_DISPLAY_LINECOUNTER_B      = 0x00000004,
}VMR_ZONE_DISPLAY_FLAGS;

```

VMR_ZONE_DISPLAY_DIRECTION_ARROWS

Display direction arrow on the zone

VMR_ZONE_DISPLAY_LINECOUNTER_A

Display linecounter A on the zone

VMR_ZONE_DISPLAY_LINECOUNTER_B

Display linecounter B on the zone

VMR_ZONE_T

```

typedef struct
{
    unsigned short        usZoneId;
    VMR_ZONE_TYPE_E       eZoneType;
    VMR_ZONE_STYLE_E      eZoneStyle;
    char                  strZoneName[VMR_MAX_NUM_NAME];
    unsigned int           uiColor;
    unsigned char          ucDisplay;
    unsigned int           uiTotalPoints;
    VMR_POINT_T           pPoints[VMR_MAX_NUM_ZONE_POINTS];
    float                 fStartAngle;
    float                 fFinishAngle;
    unsigned int           uiCalibrationWidth;
    unsigned int           uiDisplayFlags;
    unsigned int           uiStatus;
}VMR_ZONE_T;

```

usZoneId

The unique id of the zone

eZoneType

The type of the zone, refer to [VMR_ZONE_TYPE_E](#) for details

eZoneStyle

The style of the zone, refer to [VMR_ZONE_STYLE_E](#) for details

strZoneName

Zone name, maximum 32 characters including terminating NULL

uiColor

Zone color in xxRRGGBB format

ucDisplay

Zone visible or not

uiTotalPoints

Total zone points, maximum 40

pPoints

Array of zone vertices, refer to [VMR_POINT_T](#) for details

uiDisplayFlags

Display flags for zones, refer to [VMR_ZONE_DISPLAY_FLAGS](#) for details

fStartAngle

The start acceptance angle in degrees, the current accuracy is 0.01 degree.

fFinishAngle

The finish acceptance angle in degrees, the accuracy is same as start acceptance angle. The following conditions must be satisfied: $fFinishAngle > fStartAngle$ and $(fFinishAngle - fStartAngle) < 360.00$

uiCalibrationWidth

Calibration width for linecounter A&B, normalized according to the screen width: 0 corresponds to 0% and 65535 to 100%

uiStatus

The zone status: selected or hovered, refer to VMR_OBJECT_STATUS_E for details.

VMR_COUNTER_T

```
typedef struct
{
    unsigned short    usCounterId;
    char              strCounterName[VMR_MAX_NUM_NAME];
    unsigned int      uiColor;
    unsigned char     ucDisplay;
    RECT              rcRegion;
    int               iCounterValue;
    unsigned int      uiStatus;
}VMR_COUNTER_T;
```

usCounterId

The unique id of the counter

strCounterName

Counter name, maximum 32 characters including terminating NULL.

uiColor

Counter color in xxRRGGBB format

ucDisplay

Counter visible or not

rcRegion

Rectangle for displaying the counter, including top, bottom, left and right

iCounterValue

The value of the counter

uiStatus

The counter status: selected or hovered, refer to VMR_OBJECT_STATUS_E for details.

VMR_ZNCT_STATUS_E

```
typedef enum
{
    VOZ_ZONE                = 0x0001,
    VOZ_COUNTER,
    VOZ_NONE
}VMR_ZNCT_STATUS_E;
```

VOZ_ZONE

Zone is selected

VOZ_COUNTER

Counter is selected

VOZ_NONE

Nothing is selected

VMR_ZNCT_STATUS_T

```
typedef struct
{
    VMR_ZNCT_STATUS_E    eType;
    int                   iObjectIdx;
    int                   iNodeIdx;
}VMR_ZNCT_STATUS_T;
```

eType

The type of object selected, refer to [VMR_ZNCT_STATUS_E](#) for details

iObjectIdx

The id of the object selected: (1) it is the zone id when eType = VOZ_ZONE; (2) it is counter id when eType = VON_COUNTER; (3) it is -1 when eType = VOZ_NONE

iNodeIdx

The id of selected node, -1 when no node is selected

VMR_CALIB_INFO_T

```
typedef struct
{
    float fHeight;
    float fTilt;
    float fRoll;
    float fFOV;
}VMR_CALIB_INFO_T;
```

fHeight

Camera height in meters

fTilt

Camera tilt angle in degrees

fRoll

Camera roll angle in degrees, **NOT IMPLEMENTED**

fFOV

Camera vertical field of view in degrees

VMR_MODEL_INFO_T

```
typedef struct
{
    unsigned short    usModelId;
    unsigned int      uiColor;
    VMR_POINT_T       tPosition;
}VMR_MODEL_INFO_T;
```

usModelId

The unique id of a calibration model

uiColor

Model color in xxRRGGBB format

tPosition

The position of the calibration model, refer to [VMR_POINT_T](#) for details

VMR_CALLBACK_INFO_T

```
typedef struct
{
    unsigned int    uiFlags;
    void*           pUserData;
    VMR_CB          pCB;
}VMR_CALLBACK_INFO_T;
```

uiFlags

The triggering flags for call-back function, refer to VMR_CALLBACK_STATUS_E for details

pUserData

The pointer to the user data

pCB

The pointer to the call-back function

VMR_CB

```
typedef void (__stdcall *VMR_CB)( VMR_HANDLE hVMRHandle, unsigned int uiCBStatus, void
*pData, void *pUserData );
```

hVMRHandle

The handle to the VMR instance which triggered the call-back function

uiCBStatus

The status of call-back function, refer to VMR_CALLBACK_STATUS_E for details

pData

The pointer to the call-back data, cast the pData with VMR_ZONE_T* when (uiCBStatus&VMR_ZONE_MSK) > 0, cast the pData with VMR_COUNTER_T* when (uiCBStatus&VMR_COUNTER_MSK) > 0.

pUserData

The preset pointer to the user data when registering the call-back function

Revision history

Version	Change	By	Date	Pages Affected
1.0	Initial Draft	BC	15/06/09	All
1.1	Added decision tree about when VMR should be used	BW	27/08/09	6
1.2	<p>Added VMR_RegisterCallBack and VMR_UnRegisterCallBack for handling VMR call-back function.</p> <p>Added VMR_GetObject and VMR_SetObject for getting/setting current selected/hovered object.</p> <p>Removed VMR_GetSelectedObject.</p> <p>Added uiDisplayFlags, uiStartAngle, uiFinishAngle, uiStatus inside VMR_ZONE_T; Added uiStatus inside VMR_COUNTER_T.</p>	BC	02/10/09	All
1.3	<p>Changed uiStartAngle, uiFinishAngle to fStartAngle, fFinishAngle inside VMR_ZONE_T.</p> <p>Added VMRERR_INVALID_DIRECTIONARRAY inside VMR_ERROR_E</p>	BC	21/10/09	42,43,50,51
1.4	<p>Added VMR_ResetMetaDat for clearing the metadata</p> <p>Added VRF_CALIB_HORIZON and VRF_CALIB_SOLIDGRID for rendering flags</p>	BC	21/01/10	18,46
1.5	<p>Added VRF_OBJECT_MULTICOLOR and VRF_SYSMSG for VMR_RENDER_FLAGS_E</p> <p>Added automatic obtaining measurement units from metadata</p>	BC	25/06/10	45

1.6	Added VRF_OBJECT_WAIT_TIME and VRF_OBJECT_COLSIG for VMR_RENDER_FLAGS_E Added VMR_ZONE_DISPLAY_FLAGS Added uiCalibrationWidth for VMR_ZONE_T	BC	26/03/12	45,46,52,53
1.7	Minor layout correction.	Trevor	27/02/13	45, 57
02-2014-A	manual name changed (VCA MetaRender API Manual Developer's Guide -> VCA MetaRender API Manual)	Kate	27/02/14	1
03-2017-A	Added VMR_SetFont for using custom font. Added VMRERR_INTERNAL_ERROR, VMRERR_INVALID_CAPS and VMRERR_INVALID_ARGUMENT inside VMR_ERROR_E	Charles	28/02/17	8,14,44,45