VIVOTEK NETWORK DEVELOPMENT PLATFORM

Discover Register Mechanism Module

Version 2.0.0.5

2009/8/31

VIVOTEK may make changes to specifications and product descriptions at any time, without notice.

The following is trademarks of VIVOTEK Inc., and may be used to identify VIVOTEK products only: VIVOTEK. Other product and company names contained herein may be trademarks of their respective owners.

TABLE OF CONTENTS

1. Overview

1.1. Introduction

This document describes the properties and methods supported by the VIVOTEK Discover Register Mechanism (DRM) module.

1.2. Getting Started with DRM module

The main function of DRM module is to discover all available network cameras or video servers of vivoteks in the LAN.

File Structure

FILE	DESCRIPTION
doc\VNDP_DRMContro_API.pdf	This manual
lib\d_DRMControl.lib	The dynamic linking library
lib\DRMControl.dll	The dynamic runtime library
inc\DRMControl.h	Header file

2. PROGRAMMER'S GUIDE

2.1. Using DRM Module

- Start DRM module
- Discover
- Get information of available network cameras or video servers in the LAN via callback function

3. SAMPLE CODE

3.1. DiscoverDevice

DESCRIPTION

Discove the devices in the LAN at first, and than discover a certain device by MAC address.

SAMPLE CODE

STEP 1: Prepare a callback function for DRM control to do notification

```
DWORD __stdcall DRMControlCallback(
HANDLE hParentObject, EDRMControl_CallbackType eCallbackType, void* pvCallbackData)
{
    tagTDRMServiceInfo* ptDRMServiceInfo = NULL;
    switch (eCallbackType)
    {
    case DRMControl_Callback_DiscoveryResult :
            ptDRMServiceInfo= (tagTDRMServiceInfo*)pvCallbackData;
            printf("IP address: %-15s ",ptDRMServiceInfo->acIP);
            printf("Server Type: %s ",ptDRMServiceInfo->acMachineType);
            printf("CameraName:%s\n",ptDRMServiceInfo->acCameraName);
            printf("MAC address:");
            for(int i = 0; i < 6; i++)
                    printf("%02x",ptDRMServiceInfo->abyMac[i]);
            printf(" HTTP port: %-4d ",ptDRMServiceInfo->sHTTPPort);
            printf("FTP port: %-4d ",ptDRMServiceInfo->sFTPPort);
            printf("Language: %s\n\n",ptDRMServiceInfo->acLanguage);
            break;
    ...
    }
}
```

STEP 2: Startup windows socket module

```
WSADATA wsaData;
if (WSAStartup(0x202, &wsaData) == SOCKET_ERROR)
{
    fprintf(stderr,"WSAStartup error %d\n",WSAGetLastError());
    return -1;
}
else
{
    printf("WSAStartup successed.\n");
}
```

STEP 3: Create DRMControl

```
HANDLE hDRMControl = NULL;
TDRMControlCreateOptions tDRMOption;
memset(&tDRMOption, 0, sizeof(tDRMOption));
tDRMOption.dwMaxDiscoveryItem = 100;        // Set maximum discovery item
tDRMOption.dwAliasIP = TRUE;                // Search with 169.254.x.x interface too.
tDRMOption.usDRMControlPort = 9000;         // Assign a free port
DRMControl_Create(&hDRMControl, &tDRMOption);
```

STEP 4: Set callback function

```
DRMControl_SetCallback(hDRMControl, (DWORD) NULL, DRMControlCallback);
```

STEP 5: Start the DRMControl

```
DRMControl_Start(hDRMControl);
```

STEP 6: Start to discover devices in the LAN.

```
printf("Start to discover the devices in the LAN\n");
DRMControl_Discovery(hDRMControl);    // discover all device in the LAN
Sleep(10000);
```

```
printf(“Discover device by MAC\n”);
DRMControl_DiscoveryByMAC(hDRMControl,”0002D100089C”); // discover this MAC only
Sleep(10000);
```

STEP 7: Release the module

```
DRMControl_Stop(hDRMControl);
DRMControl_Close(&hDRMControl);
WSACleanup();
```

TIPS

4. API Reference

This chapter contains the API function calls for the DRM

4.1. Enumeration

The enumeration used is depicted here.

EDRMControl_CallbackType

The enumeration presents FDRMControlCallback callback type.

```
typedef enum{
DRMControl_Callback_DiscoveryResult     = 1

DRMControl_Callback_Stopped             = 4

DRMControl_Callback_Error               = 5
} EDRMControl_CallbackType
```

Values

DRMControl_Callback_DiscoveryResult

Discover device and information acquired

DRMControl_Callback_Stopped

DRM module stopped

DRMControl_Callback_Error

Search device error

Remarks

Requirements

DRMControl.h

4.2. Data Structure

The data structure is depicted here.

TDRMControlCreateOptions

This structure collects the settings of the DRM object. When creating DRM object, fill this structure to setup it.

```
typedef struct {
DWORD                   dwMaxDiscoveryItem
unsigned short          usDRMControlPort;
DWORD                   dwAliasIP;
} TDRMControlCreateOptions;
```

Members

dwMaxDiscoveryItem

The maximal items that the controller can handle.

usDRMControlPort

Port number which DRM object is going to use for discover devices in the LAN.

dwAliasIP

Discover devices with the 169.254.x.x interface, too. The value should be TRUE or FALSE.

Remarks

Requirements

DRMControl.h

tagTDRMServiceInfo

This structure defines of the information of the device searched from LAN. The information will callback in the callback function

```
typedef struct {
char                    acMachineType[44]
BYTE                    abyMac[6]
char                    acIP[20]
char                    acServiceName[44]
short                   sHTTPPort
short                   sFTPPort;
char                    acLanguage[6]
DWORD                   dwEZversion
char                    acCameraName[97]
} tagTDRMServiceInfo;
```

Members

acMachineType[44]

The firmware version of the device

abyMac[6]

MAC address of the device

acIP[20]

IP address of the device

acServiceName[44]

Not used

sHTTPPort

Port number of the web server of the device

sFTPPort

FTP port of the ftp server of the device

acLanguage[6]

lauguange of the deivce

dwEZversion

This parameter is for EZ Installation. Please reference the related document.

acCameraName

The camera name, only some newer camera support this field.

Remarks

Requirements

DRMControl.h

4.3. Callback Function

The Callback function is depicted here.

FDRMControlCallback

This define the type of the callback function

Syntex

typedef DWORD (*FDRMControlCallback)	
HANDLE	hInstance
EDRMControl_CallbackType	eDRMControlCallbackType
void	*pvParam

Members

hInstance

[in] the bypass external 32-bit data to callback function which is the object handle (the first parameter) used in the DRMControl_SetCallback( )

EDRMControl_CallbackType

[in]The callback type of the callback function

***pvParam**

If the callback type is DRMControl_Callback_DiscoveryResult, the pvParam points to the structure TDRMServiceInfo that contains the information of found device.In other callback types, pvParam points to NULL

Remarks

Requirements

DRMControl.h

4.4. API Definition

The API definition is depicted here.

DRMControl_Create

This function create the DRM Module. You must call this function before using this module.

Syntax

```
SCODE DRMControl_Create (   HANDLE *phObject,
                            TDRMControlCreateOptions
                            *pDRMControlInitOptions
                            );
```

Parameters

***phPacketMaker**

[out] the pointer to receive the handle of DRM object.

***pDRMControlInitOptions**

[in] pointer to the structure that contain the setting to create DRM object.

Return Values

S_OK

Create this module ok.

S_FAIL

Create this module failed.

Remarks

Requirements

DRMControl.h

See Also

DRMControl_Close

Call this function to release the DRM object.

Syntax

```
SCODE DRMControl_Close (        HANDLE * phObject              );
```

Parameters

*** phObject**

[in] the address of the pointer to the DRM object, returned by DRMControl_Create( ).

Return Values

S_OK

Release the object successfully.

S_FAIL

Failed to release the object .

Remarks

Requirements

DRMControl.h

See Also

typedef DWORD (*FDRMControlCallback)(DWORD dwInstance, EDRMControl_CallbackType eDRMControlCallbackType,void *pvParam);

DRMControl_SetCallback

Create a channel to display video or play sound.

Syntax

```
SCODE DRMControl_SetCallback (  HANDLE hObject
                                DWORD dwInstance,
                                FDRMControlCallback fDRMControlCallback  );
```

Parameters

hObject

[in] the handle of the DRM object created by DRMControl_Create

dwInstance

[in] the instance pass to the DRM module. Usually the object might used in the callback function.

fDRMControlCallback

[in] pointer to the callback function. See

Return Values

S_OK

Set callback function OK.

S_FAIL

Set callback function failed.

Remarks

Requirements

DRMControl.h

See Also

DRMControl_Start

Start DRM module

Syntax

```
SCODE DRMControl_Start(          HANDLE  hDRMObject          );
```

Parameters

hDRMObject

[in] the handle of DRM Object created by DRMControl_Create( )

Return Values

S_OK

Start the DRM module successfully.

S_FAIL

Fail to start DRM module.

Remarks

Requirements

DRMControl.h

See Also

DRMControl_Stop

Stop DRM module

Syntax

```
SCODE DRMControl_Stop (      HANDLE hDRMObject                    );
```

Parameters

hDRMObject

[in] the handle of the DRM Object, which created by DRMControl_Create( ).

Return Values

S_OK

Stop the DRM module successfully.

S_FAIL

Fail to stop DRM module.

Remarks

Requirements

PacketMaker.h

See Also

DRMControl_Discovery

Call this function to search all the devices in the LAN.

Syntax

```
SCODE DRMControl_Discovery (      HANDLE hDRMObject,                );
```

Parameters

hDRMObject

[in] the handle of the DRM Object, which created by DRMControl_Create( ).

Return Values

S_OK

The discover message send out successfully.

S_FAIL

Fail to send out discover message..

Remarks

Once this API is called, the module will ignore any previous search and only callback the result of current search. The best way to use this API is to call it once, wait the result in the callback function about 2~5 seconds.

Requirements

PacketMaker.h

See Also

DRMControl_DiscoveryByMAC

Call this function to search the device of the specific MAC address in the LAN.

Syntax

```
SCODE DRMControl_Discovery (    HANDLE hDRMObject, char* acMACAddress    );
```

Parameters

hDRMObject

[in] the handle of the DRM Object, which created by DRMControl_Create( ).

acMACAddress

[in] the pointer to the string of MAC address. The MAC address must be a 12 byte string

Return Values

S_OK

The discover message send out successfully.

S_FAIL

Fail to send out discover message..

Remarks

.

Requirements

PacketMaker.h

See Also