VIVOTEK NETWORK DEVELPMENT PLATFORM

ServerManager

Version 3.3.2.0

2017/01/20

TABLE OF CONTENTS

1. Overview

1.1 Introduction

This document describes how to use the ServerManager module to get/set system parameters or status of your server.

In old type of server, we use the ServerUtl. But in new type of servers, we have a brand new parameter retrieving system, ServerManager.

Furthermore, the method to communicate with servers is improved. It becomes more efficient and more resource-saving.

The ServerManager is a module to integrate new modules and ServerUtl. It wraps two different style of server control interface as one. The users do not need to know which module supports which server. The simplified interface provides an easy way to control the server.

The ServerManager provides get/set system parameters, get/set DIDO, send PTZ commands and reading/writing serial port. To use the ServerManager component, user should include “ControlCommon.h“ and “ServerManager.h” in your application.

1.2 File Structure

Table 1-1 File Structure

File	Description
doc\VNDP_ServerManager_API.pdf	This manual document
lib\d_ServerManager.lib	The dynamic linking library
lib\NetScheduler.dlll	The dynamic runtime library
lib\ServerController.dll	The dynamic runtime library
lib\ServerControllerLoader.dll	The dynamic runtime library
lib\ServerManager.dll	The dynamic runtime library
lib\ServerUtilityLoader.dll	The dynamic runtime library
lib\ServerUtl.dll	The dynamic runtime library
lib\SrvDepResource.dll	The dynamic runtime library

inc\ControlCommon.h	Common definition file
inc\ServerManager.h	ServerManager header file
inc\ServerManagerError.h	ServerManager error code definition

2. Programmer's Guide

2.1 Using the ServerManager Module

2.1.1 Create an Access Permission to the Camera Server

Before controlling the camera server, you must gain the access. Call **ServerManager_Initial** to get a ServerManager control handle and specify the maximum device number that it can control.

After get the control handle, you can use this handle to create instance of devices by calling **ServerManager_CreateDevice**. Then, you can set the properties to the created instance of device by call **ServerManager_SetDeviceProperty**().

Next, calling **ServerManager_OpenDevice**() or **ServerManager_OpenDeviceBlock**() to gain the permission of access to the camera server.

Finally, you can start to get/set system parameters, get/set DIDO, send PTZ commands and read/write serial ports.

3. Sample Code

3.1 Get/Set Network Information

Description

Creates a device and gets the network information.

Sample Code

Step 1. Initialize ServerManager and Create a Device

```
// Set ServerManager.dll version
tVerInfo.wMajor = SRVMNGR_VERSION_MAJOR;
tVerInfo.wMinor = SRVMNGR_VERSION_MINOR;
tVerInfo.wBuild = SRVMNGR_VERSION_BUILD;
tVerInfo.wRevision = SRVMNGR_VERSION_REVISION;
// Initialize a server manager handle
// param CONST_MAX_DEVICE The maximum number of devices are allowed to open.
ServerManager_Initial(&hSrvMgr, CONST_MAX_DEVICE, 0, &tVerInfo);
// Create a device handle
ServerManager_CreateDevice(hSrvMgr, &hDevice);
```

Step 2. Set Device Property

```
TSRVMNGR_DEV_PROPERTY tDevProperty = {0};
strcpy(tDevProperty.tServerProperty.szHost, "192.168.1.100");
strcpy(tDevProperty.tServerProperty.szUserName, "root");
strcpy(tDevProperty.tServerProperty.szPassword, "123");
tDevProperty.tServerProperty.dwHttpPort = 80;
tDevProperty.tServerProperty.dwTimeout = 30000;
// Update device property
ServerManager_SetDeviceProperty(hDevice, &tDevProperty);
```

Step 3. Gain an Access Right to the Server

```
// Open device (block mode)
// param lpszModelName The char pointer to receive the retrieved model name
// param dwMaxSize The maximum size of model name
char szModel[MAX_PATH + 1] = {0};
ServerManager_OpenDeviceBlock(hDevice, szModel, MAX_PATH);
```

Step 4. Get Network Information, ex. RTSP Port

```
char szValue[MAX_PATH + 1] = {0};
DWORD dwBufSize = MAX_PATH;
// param pszName The pointer to the name of the key you want to get the value.
// param pszValue The pointer to the buffer to receive the value.
// param pdwValueSize The pointer to the DWORD to indicate the size of buffer.
// After the function returned, it will be filled with the length of value.
ServerManager_GetValueByName(hDevice, "network_rtsp_port", szValue, &dwBufSize);
```

Step 5. Set Network Information, ex. RTSP Port

```
// Set Network Information : ex. Set Server Rtsp port
// param pszName The pointer to the name of the key you want to update the value.
// param pszValue The pointer to the buffer which contain the value.
// param bMustExist The key must exist or not.
ServerManager_SetValueByName(hDevice, "network_rtsp_port", "554", TRUE);
```

Step 6. Update the Local Changes to the Remote Server

```
ServerManager_UpdateRemoteConfigBlock(hDevice);
```

Step 7. Delete Device and Release ServerManager

```
ServerManager_DeleteDevice(&hDevice);
ServerManager_Release(&hSrvMgr);
```

Tips

1. ServerManager_GetValueByName()/ServerManager_SetValueByName() functions are modifying the information stores in local that was retrieved by ServerManager_OpenDeviceBlock.If you want to update the local changes to the camera server, call ServerManager_UpdateRemoteConfigBlock() or ServerManager_UpdateRemoteConfig().
2. The key name can be retrieved from the server by send HTTP request "/cgi-bin/admin/getparam.cgi", or you can call ServerManager_GetNameList() and input empty string("") to the second parameter and it will return all key names.
3. You can refer to the SDK package for full sample code.

3.2 Send PTZ Command

Description

Creates a device and sends a PTZ command to server.

Sample Code

Step 1. Initialize ServerManager and Create a Device

This code is similar as the previous sample.

Step 2. Set Device Property

This code is similar as the previous sample.

Step 3. Gain an Access to the Server

This code is similar as the previous sample.

Step 4. Send a PTZ Command

```
// Sends a PTZ command to server (blocking)
// Param dwCamera In a multiple camera model, it indicates the order from the right with
// starting as 1. Otherwise it will be ignored.
// Param ePtzCommand The ESRVCTRL_PTZ_COMMAND.
// Param pszExtraCmd The extra information for certain PTZ command.
// Param bWaitRes The boolean value to wait for the server's response or not.
ServerManager_SendPTZCommandBlock(hDevice, 1, eSrvCtrlPTZMoveDown, "", TRUE);
```

Step 5. Delete Device and Release ServerManager

This code is similar as the previous sample.

Tips

You can refer to the SDK package for full sample code.

3.3 Get/Set DO

Description

Sample Code

Step 1. Initialize ServerManager and Create a Device

This code is similar as the previous sample.

Step 2. Set Device Property

This code is similar as the previous sample.

Step 3. Gain an Access to the Server

This code is similar as the previous sample.

Step 4. Get DO Information from Server

```
ESrvCtrlDiDoLevel aeDOStatus[CONST_MAX_DIDO_NUM];
DWORD dwDoNum = CONST_MAX_DIDO_NUM;
// Get DO status
// Param peDOStatus The pointer to a ESrvCtrlDiDoLevel array.
// Param pdwMaxDONum The pointer to a DWORD to represent the size value of DO
// status array buffer. And the module will replace it with the actual size value
// of DO status array. If the actual size is more than *pdwMaxDINum,
// the module will return error. And the give buffer will not be filled.
ServerManager_GetDOStatusBlock(hDevice, aeDOStatus, &dwDoNum);
```

Step 5. Change DO Array and Set to the Server.

```
for(iIndex = 0; iIndex < (int) dwDoNum; iIndex++)
{
if (aeDOStatus[iIndex] == eSrvCtrlDiDoLow)
{
// set DO to eSrvCtrlDiDoHigh;
aeDOStatus[iIndex] = eSrvCtrlDiDoHigh;
}
else
{
```

```
// set DO to eSrvCtrlDiDoLow
aeDOStatus[iIndex] = eSrvCtrlDiDoLow;
}
}
// Set DO status
// Param peDOStatus The pointer to a ESrvCtrlDiDoLevel array.
// Param dwMaxDONum The size value of DO status array
// Param bWaitRes The Boolean value to wait for the server's response or not.
ServerManager_SetDOStatusBlock(hDevice, aeDOStatus, dwDoNum, FALSE);
```

Step 6. Delete Device and Release ServerManager

This code is similar as the previous sample.

Tips

You can refer to the SDK package for full sample code.

3.4 Send HTTP Command to Server (Non-Blocking Mode)

Description

Sample Code

Step 1. Prepare the callback functions

```
SCODE __stdcall ProcNotify(TSVRCTRL_NOTIFY_CONT *ptContext)
```

Step 2. Initialize ServerManager and Create a Device

This code is similar as the previous sample

Step 3. Set Device Property

This code is similar as the previous sample

Step 4. Gain an Access Right to the Server (Non-Blocking)

```
// Non-block setting, Callback infomation
tCBContent.hEvent = hWaitEvent;
tNotify.pvContext = &tCBContent;
tNotify.pfCB = ProcNotify;
// Open device (non-block mode)
ServerManager_OpenDevice(hDevice, &tNotify);
```

Step 5. Send HTTP Command

```
// Non-block setting, Callback infomation
tCBContent.hEvent = hWaitEvent;
tNotify.pvContext = &tCBContent;
tNotify.pfCB = ProcNotify;
```

// Step 6: Semd HTTP command (non-block) to server

```
// Param pszCommand The string pointer to an http command.
// Param bPost True is for POST, false for GET
// Param ptNotify The pointer TSVRCTRL_NOTIFY contains the callback function and
// related information. It cannot be null and the content should be valid.
// Param phProc The pointer to a http operation handle. The http operation handle
```

```
// is used to read more data. The module will pass a handle to the caller
// if there is more data to be read. If none, the phOper will be null.
// The users do not need to free the http operation handle.
ServerManager_SendHttpCommand(hDevice,
"/cgi-bin/admin/setparam.cgi?system_hostname=Network Camera", TRUE, &tNotify, NULL);
```

Tips

1. The ServerManager_OpenDevice() function and ServerManager_SendHttpCommand() function are both asynchronous functions, the result can be obtained in SVRCTRL_STATUS_CALLBACK callback function.
2. You can refer to the SDK package for full sample code.

4. API Reference

This chapter describes the API function calls for the ServerManager.

4.1 Enumeration

The enumeration used is depicted here.

4.1.1 EJOYSTICK_ZOOMING_COMMAND

This enumeration indicates the simulation of joystick zooming commands.

```
typedef enum {
                                        eJoystickZoomStop,
                                        eJoystickZoomWide,
                                        eJoystickZoomTele,
} EJOYSTICK_ZOOMING_COMMAND;
```

Values

eJoystickZoomStop

Stop the zooming operation.

eJoystickZoomWide

Start zooming wide.

eJoystickZoomTele

Start zooming tele.

Remarks

Requirements

ControlCommon.h

4.1.2 ESrvCtrlDiDoLevel

This enumeration indicates the DIDO status.

```
typedef enum {
                                        eSrvCtrlDiDoNone,
                                        eSrvCtrlDiDoLow,
                                        eSrvCtrlDiDoHigh
} ESrvCtrlDiDoLevel;
```

Values

eSrvCtrlDiDoNone

The server does not have digital output.

eSrvCtrlDiDoLow

The digital output status of the server is at low level.

eSrvCtrlDiDoHigh

The digital output status of the server is at high level.

Remarks

Requirements

ControlCommon.h

4.1.3 ESrvCtrlPTZType

This enumeration indicates the PTZ command type.

```
typedef enum {
                                        eSrvCtrlPPTZ,
                                        eSrvCtrlEPTZ
} ESrvCtrlPTZType;
```

Values

eSrvCtrlPPTZ

Use physical PTZ command.

eSrvCtrlEPTZ

Use digital PTZ command.

Remarks

Make sure the camera server supports digital PTZ command if you want to control the camera by digital PTZ commands.

Requirements

ControlCommon.h

4.1.4 ESRVCTRL_PTZ_COMMAND

This enumeration indicates the PTZ commands.

```
typedef enum {
                                        eSrvCtrlPTZMoveUp,
                                        eSrvCtrlPTZMoveDown,
                                        eSrvCtrlPTZMoveLeft,
                                        eSrvCtrlPTZMoveRight,
                                        eSrvCtrlPTZMoveHome,
                                        eSrvCtrlPTZZoomIn,
                                        eSrvCtrlPTZZoomOut,
                                        eSrvCtrlPTZFocusNear,
                                        eSrvCtrlPTZFocusFar,
                                        eSrvCtrlPTZFocusAuto,
                                        eSrvCtrlPTZPresetAdd,
                                        eSrvCtrlPTZPresetDel,
                                        eSrvCtrlPTZPresetRecall,
                                        eSrvCtrlPTZPanSpeed,
                                        eSrvCtrlPTZTiltSpeed,
                                        eSrvCtrlPTZStartPan,
                                        eSrvCtrlPTZStopPan,
                                        eSrvCtrlPTZStartPatrol,
                                        eSrvCtrlPTZIrisOpen,
                                        eSrvCtrlPTZIrisClose,
                                        eSrvCtrlPTZIrisAuto,
                                        eSrvCtrlPTZZoomSpeed,
                                        eSrvCtrlPTZSetHome,
                                        eSrvCtrlPTZDefaultHome,
                                        eSrvCtrlPTZFocusSpeed,
                                        eSrvCtrlPTZCustom
} ESRVCTRL_PTZ_COMMAND;
```

Values

eSrvCtrlPTZMoveUp

Move the camera one step up. The step is decided on the tilt speed.

eSrvCtrlPTZMoveDown

Move the camera one step down. The step is decided on the tilt speed.

eSrvCtrlPTZMoveLeft

Move the camera one step left. The step is decided on the pan speed.

eSrvCtrlPTZMoveRight

Move the camera one step right. The step is decided on the pan speed.

eSrvCtrlPTZMoveHome

Move the camera back to its home position. Usually, this is the position when camera boots up.

eSrvCtrlPTZZoomIn

Zoom in the camera to see more details. Note that this command only works for cameras that support zooming. The return code will be OK even if the camera does not support zooming. The OK means the command is sent to the server successfully.

eSrvCtrlPTZZoomOut

Zoom out the camera to see wider view. Note that this command only works for cameras that support zooming. The return code will be OK even if the camera does not support zooming. The OK means the command is sent to the server successfully.

eSrvCtrlPTZFocusNear

Set the focus of the camera to shorter focal distance. Note that this command only works for cameras that support focus adjustment. The return code will be OK even if the camera does not support focus adjustment. The OK means the command is sent to the server successfully.

eSrvCtrlPTZFocusFar

Set the focus of the camera to longer focal distance. Note that this command only works for cameras that support focus adjustment. The return code will be OK even if the camera does not support focus adjustment. The OK means the command is sent to the server successfully.

eSrvCtrlPTZFocusAuto

Let the camera adjusts its focal distance automatically. After this command is set, the camera will try to locate the proper focal distance when camera moves every time. Note that this command only works for cameras that support focus adjustment. The return code will be OK even if the camera does not support focus adjustment. The OK means the command is sent to the server successfully.

eSrvCtrlPTZPresetAdd

Add a new preset location. The preset location is a string that indicates a saved coordinate in camera.

eSrvCtrlPTZPresetDel

Remove a preset location in camera.

eSrvCtrlPTZPresetRecall

Move the camera to a preset location.

eSrvCtrlPTZPanSpeed

Adjust the camera's pan speed.

eSrvCtrlPTZTiltSpeed

Adjust the camera's tilt speed.

eSrvCtrlPTZStartPan

Let PT or PTZ camera (both are built in IP camera) start to pan for a round.

eSrvCtrlPTZStopPan

Stop the auto pan or patrol in PT or PTZ camera (both are built in IP camera).

eSrvCtrlPTZStartPatrol

Let PT or PTZ camera (both are built in IP camera) patrol the predefined route.

User must setup the patrol route before this command takes effect.

eSrvCtrlPTZIrisOpen

Increase the IRIS level.

eSrvCtrlPTZIrisClose

Decrease the IRIS level.

eSrvCtrlPTZIrisAuto

Set the IRIS to auto mode.

eSrvCtrlPTZZoomSpeed

Adjust the camera's zoom speed.

eSrvCtrlPTZSetHome

Set the new home position.

eSrvCtrlPTZDefaultHome

Set back to the default home position.

eSrvCtrlPTZFocusSpeed

Adjust the camera's focus speed.

eSrvCtrlPTZCustom

Call a custom command. This is useful for video server that might connect to analog camera that provides more commands except the standard pan and tilt commands.

Remarks

Requirements

ControlCommon.h

4.2 Callback Function

The Callback function is depicted here.

4.2.1 SVRCTRL_STATUS_CALLBACK

This status callback is used when non-blocking mode is specified. Non-blocking mode lets application to submit a work and go ahead to handle other tasks. This module will call application back when work done through this callback.

Syntax

```
typedef SCODE (__stdcall SVRCTRL_STATUS_CALLBACK)
                                                    TSVRCTRL_NOTIFY_CONT    *ptContent
);
```

Parameters

ptContent

[in] The pointer to **TSVRCTRL_NOTIFY_CONT** that contains the callback information.

Return Values

Ignored.

Remarks

Requirements

ControlCommon.h

4.3 Data Structure

The data structure is depicted here.

4.3.1 TSRVCTRL_DEV_PROPERTY

This structure indicates the properties of a device. When opening a device, you must set the properties for the device.

```
typedef struct {
                                    char        szHost [SRVCTRL_MAX_HOST_LEN+1];
                                    char        szUserName [SRVCTRL_MAX_USERNAME_LEN+1];
                                    char        szPassword [SRVCTRL_MAX_USERPASS_LEN+1];
                                    DWORD       dwHttpPort;
                                    DWORD       dwFtpPort;
                                    DWORD       dwTimeout;
                                    BOOL        bUseSSL;
} TSRVCTRL_DEV_PROPERTY,
*PTSRVCTRL_DEV_PROPERTY;
```

Members

szHost

The host name or IP of server to be connected to.

szUserName

User name for the server.

szPassword

Password for the server.

dwHttpPort

Http port for the server.

dwFtpPort

Ftp port for the server.

dwTimeout

The timeout for a operation in milliseconds.

bUseSSL

Connect to server by using SSL protocol.

Remarks

Requirements

ControlCommon.h

4.3.2 TSRVCTRL_DEV_PROPERTYW

This structure indicates the properties of a device. When opening a device, you must set the properties for the device. This is a wide-character version of **TSRVCTRL_DEV_PROPERTY**.

```
typedef struct {
                                wchar_t         szHost [SRVCTRL_MAX_HOST_LEN+1] t;
                                wchar_t         szUserName [SRVCTRL_MAX_USERNAME_LEN+1];
                                wchar_t         szPassword [SRVCTRL_MAX_USERPASS_LEN+1];
                                DWORD           dwHttpPort;
                                DWORD           dwFtpPort;
                                DWORD           dwTimeout;
                                BOOL            bUseSSL;
} TSRVCTRL_DEV_PROPERTYW,
*PTSRVCTRL_DEV_PROPERTYW;
```

Members

szHost

The host name or IP of server to be connected to.

szUserName

User name for the server.

szPassword

Password for the server.

dwHttpPort

Http port for the server.

dwFtpPort

Ftp port for the server.

dwTimeout

The timeout for the operation in milliseconds.

bUseSSL

Connect to server by using SSL protocol.

Remarks

Requirements

ControlCommon.h

4.3.3 TSRVCTRL_MODETECT_INFO

This structure contains the information of a motion detection window.

```
typedef struct {
                                char        szName[SRVCTRL_MAX_WINDOWNAME+1];
                                int         nX;
                                int         nY;
                                int         nW;
                                int         nH;
                                int         nPercent;
                                int         nSensitivity;
                                BOOL        bWindowEnabled;
} TSRVCTRL_MODETECT_INFO;
```

Members

szName

The name for the motion detection window.

nX

The x coordinate for the window related to the left-upper corner of the video. The value is for normal size image. If the image is shown in other size, translation is needed.

nY

The x coordinate for the window related to the left-upper corner of the video. The value is for normal size image. If the image is shown in other size, translation is needed.

nW

The width of the window. The value is for normal size image. If the image is shown in other size, translation is needed.

nH

The height of the window. The value is for normal size image. If the image is shown in other size, translation is needed.

nPercent

The percentage of the pixels in the specified window. The difference of two adjacent frames is larger than the threshold (as specified in nSensitivity) will be judged as triggered for motion detection.

nSensitivity

The threshold of motion detection sensitivity. A pixel in the window is called “moved” if the difference of the pixel between two adjacent frames is larger than this value.

bWindowEnabled

Specify whether the window is enable or not.

Remarks

Requirements

ControlCommon.h

4.3.4 TSRVCTRL_PRIVATEMASK_INFO

This structure contains the information of a private mask window.

```
typedef struct {
                                char        szName[SRVCTRL_MAX_WINDOWNAME+1];
                                int         nX;
                                int         nY;
                                int         nW;
                                int         nH;
                                BOOL        bWindowEnabled;
} TSRVCTRL_PRIVATEMASK_INFO;
```

Members

szName

The name for the motion detection window.

nX

The x coordinate for the window related to the left-upper corner of the video. The value is for normal size image. If the image is shown in other size, translation is needed.

nY

The x coordinate for the window related to the left-upper corner of the video. The value is for normal size image. If the image is shown in other size, translation is needed.

nW

The width of the window. The value is for normal size image. If the image is shown in other size, translation is needed.

nH

The height of the window. The value is for normal size image. If the image is shown in other size, translation is needed.

bWindowEnabled

Specify whether the window is enable or not.

Remarks

Requirements

ControlCommon.h

4.3.5 TSVRCTRL_NOTIFY

This structure is used to specify the notification interface and parameters for asynchronous operation.

```
typedef struct {
                                SVRCTRL_STATUS_CALLBACK    pfCB;
                                void                       *pvContext;
} TSVRCTRL_NOTIFY;
```

Members

pfCB

The SVRCTRL_STATUS_CALLBACK that specify the callback function to be used to notify.

pvContext

Specify the context value that is to be sent back when callback function gets called.

Remarks

Requirements

ControlCommon.h

4.3.6 TSVRCTRL_NOTIFY_CONT

The structure is used to hold the notified data when callback is called.

```
typedef struct {
                                    HANDLE      hDevice;
                                    void        *pvContext;
                                    SCODE       scStatusCode;
                                    void        *pvParam1;
                                    void        *pvParam2;
} TSVRCTRL_NOTIFY_CONT;
```

Members

hDevice

The device that is associated with this callback.

pvContext

Specify the context value that is to be sent back when callback function gets called.

scStatusCode

The status code for the callback.

pvParam1

Extra parameter 1 for each status code.

pvParam2

Extra parameter 2 for each status code.

Remarks

Requirements

ControlCommon.h

4.3.7 TSRVMNGR_DEV_PROPERTY

The structure is used to hold the property of target host and the priority interface to connect the server.

```
typedef struct {
                                TSRVCTRL_DEV_PROPERTY   tServerProperty;
                                BOOL                    bServerUtlPriority;
} TSRVMNGR_DEV_PROPERTY;
```

Members

tServerProperty

The **TSRVCTRL_DEV_PROPERTY** contain the property of target host.

bServerUtlPriority

It indicates that we use ServerUtl module first or not. If TRUE, we will use ServerUtl module to connect first. If failed, we will automatically roll to ServerController to do it. If users don't know which module is used first, just set it false. Then we will use ServerManager to connect it first.

Remarks

Requirements

ServerManager.h

4.3.8 TSRVMNGR_DEV_PROPERTYW

The structure is used to hold the property of target host and the priority interface to connect the server. This is a wide-character version of **TSRVMNGR_DEV_PROPERTY**.

```
typedef struct {
    TSRVCTRL_DEV_PROPERTYW    tServerProperty;
    BOOL                      bServerUtlPriority;
} TSRVMNGR_DEV_PROPERTYW;
```

Members

tServerProperty

The **TSRVCTRL_DEV_PROPERTYW** contain the property of target host.

bServerUtlPriority

It indicates that we use ServerUtl module first or not. If TRUE, we will use ServerUtl module to connect first. If failed, we will automatically roll to ServerController to do it. If users don't know which module is used first, just set it false. Then we will use ServerManager to connect it first.

Remarks

Requirements

ServerManager.h

4.3.9 TVersionInfo

This structure contains version information.

```
typedef struct {
                                WORD        wMajor;
                                WORD        wMinor;
                                WORD        wBuild;
                                WORD        wRevision;
} TVersionInfo;
```

Members

wMajor

Major version number.

wMinor

Minor version number

wBuild

The build number.

wRevision

The revision number.

Remarks

Requirements

ControlCommon.h

4.4 API Definition

The API definition is depicted here.

4.4.1 ServerManager_Initial

Create a ServerManager object instance and initialize it.

Syntax

```
SCODE ServerManager_Initial (
                                        HANDLE          *phObject,
                                        DWORD           dwMaxDevice,
                                        DWORD           dwFlag,
                                        TVersionInfo    *ptVersion
);
```

Parameters

phObject

[out] Pointer to the handle of ServerManager object.

dwMaxDevice

[in] The maximum number of devices are allowed to open.

dwFlag

[in] The flag used in the ServerManager module.

ptVersion

[in] Pointer to a TVersionInfo that contains the version of the ServerManager module.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

Requirements

ServerManager.h

See Also

ServerManager_Release

4.4.2 ServerManager_Release

Delete a ServerManager object instance. This function must be called to prevent the resource leaking whenever the ServerManager handle is no longer used.

Syntax

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

Parameters

phObject

[in/out] Pointer to the handle of ServerManager object. It is returned from **ServerManager_Initial**.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

Requirements

ServerManager.h

See Also

ServerManager_Initial

4.4.3 ServerManager_GetVersionInfo

This function is used to retrieve the version information of the current module.

Syntax

```
SCODE ServerManager_GetVersionInfo (
                                        TVersionInfo        *ptVersion
);
```

Parameters

ptVersion

[in/out] Pointer to the struct of the module's version information.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

Requirements

ServerManager.h

See Also

ServerManager_Initial

4.4.4 ServerManager_CreateDevice

Create a device object instance. The device is used to control a server. We must create a device handle first. And we can manage the device based on this handle.

Syntax

```
SCODE ServerManager_CreateDevice (
                                                    HANDLE          hObject,
                                                    HANDLE          *phDevice
);
```

Parameters

hObject

[in] The handle of the ServerManager object instance. It is returned from **ServerManager_Initial**.

phDevice

[out] Pointer to a handle to receive the created device object instance.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

Requirements

ServerManager.h

See Also

ServerManager_DeleteDevice

4.4.5 ServerManager_DeleteDevice

Delete the specified device object instance.

Syntax

```
SCODE ServerManager_DeleteDevice (
                                                    HANDLE          *phDevice
);
```

Parameters

phDevice

[in/out] Pointer to the handle of the device object. It is returned from **ServerManager_CreateDevice**.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice

4.4.6 ServerManager_SetDeviceProperty

Set the property of the specified device object.

Syntax

```
SCODE ServerManager_SetDeviceProperty (
                                        HANDLE                  hDevice,
                                        TSRVMNGR_DEV_PROPERTY   *ptDevProperty
);
```

Parameters

hDevice

[in] The handle of the device object. It is returned from ServerManager_CreateDevice.

ptDevProperty

[in] Pointer to TSRVMNGR_DEV_PROPERTY that stores the property of target host.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

If you want to setup a used device handle, the module may call ServerManager_StopRequest to stop the unfinished or pending request. And it will set bBlocking as true to avoid unexpected time delay response problem. So it may take times to wait for the function return and then it could set up the device property.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_StopRequest

4.4.7 ServerManager_SetDevicePropertyW

Set the property of the specified device object.

This is a wide-character version of **ServerManager_SetDeviceProperty**.

Syntax

```
SCODE ServerManager_SetDevicePropertyW (
                                        HANDLE                  hDevice,
                                        TSRVMNGR_DEV_PROPERTYW  *ptDevPropertyW
);
```

Parameters

hDevice

[in] The handle of the device object. It is returned from **ServerManager_CreateDevice**.

ptDevPropertyW

[in] Pointer to **TSRVMNGR_DEV_PROPERTYW** that stores the property of target host.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

If you want to setup a used device handle, the module may call **ServerManager_StopRequest** to stop the unfinished or pending request. And it will set bBlocking as true to avoid unexpected time delay response problem. So it may take times to wait for the function return and then it could set up the device property.

Requirements

ServerManager.h

See Also

ServerManager_SetDeviceProperty

4.4.8 ServerManager_GetNameList

Retrieve the value entries under the specified name.

Only the first layer names under root key are retrieved. The name will be cascaded with root name to get full name. If the buffer size is not enough, an error will be returned, and *pdwValueSize will give the needed buffer size.

Syntax

```
SCODE ServerManager_GetNameList (
                                        HANDLE          hDevice,
                                        const char      *pszName,
                                        char            *pszNameList,
                                        DWORD           *pdwValueSize
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

pszName

[in] Name of the key you want to get the value.

pszNameList

[out] Pointer to the buffer to receive the value.

pdwValueSize

[in/out] The size of buffer. After the function returned, it will be filled with the length of value.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

In ServerUtl applied case, this function may not find the appropriate one to do it. The module will return ERR_NOT_IMPLEMENT.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty

4.4.9 ServerManager_GetValueByName

Get a value from local buffer.

Syntax

```
SCODE ServerManager_GetValueByName (
                                        HANDLE          hDevice,
                                        const char      *pszName,
                                        char            *pszValue,
                                        DWORD           *pdwValueSize
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

pszName

[in] The name of the key you want to get the value.

pszValue

[out] The pointer to the buffer to receive the value.

pdwValueSize

[in/out] The size of buffer. After the function returned, it will be filled with the length of value.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty,

4.4.10 ServerManager_SetValueByName

Set a value to local buffer.

Syntax

```
SCODE ServerManager_SetValueByName (
                                        HANDLE          hDevice,
                                        const char      *pszName,
                                        const char      *pszValue,
                                        BOOL            bMustExist
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

pszName

[in] The pointer to the name of the key you want to update the value.

pszValue

[in] The pointer to the buffer which contain the value.

bMustExist

[in] The key must exist or not.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty

4.4.11 ServerManager_FreeOperator

Return the operator to hDevice.

Syntax

```
SCODE ServerManager_FreeOperator (
                                        HANDLE          hDevice,
                                        HANDLE          hOper
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from **ServerManager_CreateDevice**.

hOper

[in] The http operation handle. It is returned from **ServerManager_SendHttpCommand** or **ServerManager_SendHttpCommandReadBlock**.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

Requirements

ServerManager.h

See Also

ServerManager_SendHttpCommand, **ServerManager_SendHttpCommandReadBlock**

4.4.12 ServerManager_SetPTZType

Set PTZ command type, physical or digital.

Syntax

```
SCODE ServerManager_SetPTZType (
                                        HANDLE          hDevice,
                                        ESrvCtrlPTZType ePTZType
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

ePTZType

[in] The type of PTZ command send by ServerManager_SendPTZCommand, ServerManager_MoveCamera, ServerManager_SendPTZCommandBlock, andServerManager_MoveCameraBlock.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

Requirements

ServerManager.h

See Also

ServerManager_SendPTZCommand, ServerManager_MoveCamera, ServerManager_SendPTZCommandBlock, ServerManager_MoveCameraBlock

4.4.13 ServerManager_StopRequest

This function is used to stop all pending or unfinished requests for a device handle. If you are calling this before calling **ServerManager_DeleteDevice**, set the bBlocking flag to FALSE.

Syntax

```
SCODE ServerManager_StopRequest (
                                    HANDLE      hDevice,
                                    BOOL        bBlocking
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from **ServerManager_CreateDevice**.

bBlocking

[in] The boolean value to stop request in block mode. If yes, it will block until all connections done. You had better set block mode when you want to reuse this device to operate another different host.

Remarks

Requirements

ServerManager.h

See Also

4.5 Non-blocking mode API Definition

The non-blocking API definition is depicted here.

The non-block mode functions are all asynchronous. Calling any non-block function will not block the process and it will notify the user with the callback function after the operation is done.

Every non-block function has a parameter which type is pointer of **TSVRCTRL_NOTIFY**. It contains the information of the callback function and user's context value. The user can install the callback function and user's data while call non-block function. Once the operation has been done and the callback function is raised, it will pass that value to the user to specify whatever data user wants.

Non-block function also has a parameter which type is pointer of handle. It is used to receive the instance of the http operation process. The user can control the process based on this handle. If not use this operation before the callback function is notified, the user can call **ServerManager_AbortProcess** to abort the operation.

4.5.1 ServerManager_GetDIStatus

Get current DI information of the server.

Syntax

```
SCODE ServerManager_GetDIStatus (
                                        HANDLE          hDevice,
                                        TSVRCTRL_NOTIFY *ptNotify,
                                        HANDLE          *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phOper will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done. In the successful callback function, the pvParam1 contains the number of DI the server has. And pvParam2 would be the ESrvCtrlDiDoLevel array that contains each DI status. The array is invalid after callback returns, so the caller must copy the value if he needs that value later.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.2 ServerManager_GetDOStatus

Get current DO information of the server.

Syntax

```
SCODE ServerManager_GetDOStatus (
                                        HANDLE          hDevice,
                                        TSVRCTRL_NOTIFY *ptNotify,
                                        HANDLE          *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phOper will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done. In the success notification the pvParam1 contains the number of DO the server has. And pvParam2 would be the ESrvCtrlDiDoLevel array that contains each DO status. The array is invalid after callback returns, so the caller must copy the value if he needs that value later.

In ServerUtl applied case, this function may not find the appropriate one to do it. The module will return ERR_NOT_IMPLEMENT.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.3 ServerManager_GetMotionDetectionInfo

Get current motion detection setting information of the server.

Syntax

```
SCODE ServerManager_GetMotionDetectionInfo (
                                        HANDLE          hDevice,
                                        DWORD           dwCamera,
                                        TSVRCTRL_NOTIFY *ptNotify,
                                        HANDLE          *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwCamera

[in] In the multiple camera model, it indicates the order from the right with starting as 1. Otherwise the module will ignore it.

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phOper will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done. In the success notification, pvParam1 is the TSRVCTRL_MODETECT_INFO array that contains motion window

information. The array is invalid after callback returns, so the caller must copy the value if he needs that value later.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.4 ServerManager_GetPrivateMaskInfo

Get the current private mask setting information of the server.

Syntax

```
SCODE ServerManager_GetPrivateMaskInfo (
                                        HANDLE          hDevice,
                                        DWORD           dwCamera,
                                        TSVRCTRL_NOTIFY *ptNotify,
                                        HANDLE          *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwCamera

[in] In the multiple camera model, it indicates the order from the right with starting as 1. Otherwise the module will ignore it.

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phOper will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.5 ServerManager_MoveCamera

Move the camera by the specified coordinate.

Syntax

```
SCODE ServerManager_MoveCamera (
                                    HANDLE          hDevice,
                                    DWORD           dwCamera,
                                    DWORD           dwStream,
                                    DWORD           dwX,
                                    DWORD           dwY,
                                    DWORD           dwDisplayWidth,
                                    DWORD           dwDisplayHeight,
                                    TSVRCTRL_NOTIFY *ptNotify,
                                    HANDLE          *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwCamera

[in] In the multiple camera model, it indicates the order from the right with starting as 1. Otherwise it will be ignored.

dwStream

[in] In the dual-stream camera model, it indicates resolution of which stream will be get for computing the target position.

dwX

[in] The x coordinate for the new position. This position is related to the left-upper corner of current camera view. The camera will treat this value as the new center x coordinate. This value should be the normal size of camera video size.

dwY

[in] The y coordinate for the new position. This position is related to the left-upper corner of current camera view. The camera will treat this value as the new center y coordinate. This value should be the normal size of camera video size.

dwDisplayWidth

[in] The weight of the display video size.

dwDisplayHeight

[in] The height of the display video size.

ptNotify

[in] The pointer **TSVRCTRL_NOTIFY** contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phOper will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

For supporting digital move commands, you must set the device to EPTZ mode by call the **ServerManager_SetPTZType**.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, **ServerManager_SetDeviceProperty**, **ServerManager_OpenDevice**, **ServerManager_OpenDeviceBlock**, **ServerManager_AbortProcess**

4.5.6 ServerManager_OpenDevice

Open the specified device to access some information. It will connect to the server and retrieve the server model.

Syntax

```
SCODE ServerManager_OpenDevice (
                                                    HANDLE              hDevice,
                                                    TSVRCTRL_NOTIFY     *ptNotify
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

ptNotify

[in] Pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

If you want to control the device or get some information from the device, you must open the device first.

The function will return soon. And callback function will be raised when the operation is really done. In the success notification, the pvParam1 contains the string for the model value. User could also use the ServerManager_GetValueByName to get the server model value.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice

4.5.7 ServerManager_ReadData

This function is used to read the data from the http connection. User must ensure that the connection is already connected, or error will be returned.

Syntax

```
SCODE ServerManager_ReadData (
                                        HANDLE          hDevice,
                                        HANDLE          hOper,
                                        BYTE            *pbyBuffer,
                                        DWORD           *pdwBufLen,
                                        TSVRCTRL_NOTIFY *ptNotify
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

hOper

[in] The http operation handle. It is reutrned from ServerManager_SendHttpCommand or ServerManager_SendHttpCommandReadBlock.

pbyBuffer

[out] The pointer to buffer to receive the http response data.

pdwBufLen

[in/out] The length of the buffer. After the function return, it will be replaced with the actual length value.

ptNotify

[in] Pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

In ServerUtl applied case, this function may not find the appropriate one to do it. The module will return ERR_NOT_IMPLEMENT.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.8 ServerManager_SendHttpCommand

This function is used to send an http command to server. For POST, the parameter is also put after the question mark as GET does. The result of the command is ignored by this function.

If you want to read the response data, please use **ServerManager_SendHttpCommandReadBlock**.

Syntax

```
SCODE ServerManager_SendHttpCommand (
                                        HANDLE          hDevice,
                                        const char      *pszCommand,
                                        BOOL            bPost,
                                        TSVRCTRL_NOTIFY *ptNotify,
                                        HANDLE          *phOper
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from **ServerManager_CreateDevice**.

pszCommand

[in] The string pointer to an http command.

bPost

[in] Indicate the way you send the command. True is to send the command by POST, and false is by GET.

ptNotify

[in] The pointer **TSVRCTRL_NOTIFY** contains the callback function and related information. It cannot be null and the content should be valid.

phOper

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phOper will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.9 ServerManager_SendPTZCommand

Send the PTZ command to the server.

Syntax

```
SCODE ServerManager_SendPTZCommand (
                                        HANDLE                  hDevice,
                                        DWORD                   dwCamera,
                                        ESRVCTRL_PTZ_COMMAND    ePtzCommand,
                                        const char              *pszExtraCmd,
                                        TTSVRCTRL_NOTIFY        *ptNotify,
                                        HANDLE                  *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwCamera

[in] In the multiple camera model, it indicates the order from the right with starting as 1. Otherwise it will be ignored.

ePtzCommand

[in] The ESRVCTRL_PTZ_COMMAND to represent a command will be sent to the target device. Not all commands are accepted in every camera, so you may check the user's manual to make sure it.

pszExtraCmd

[in] The extra information for certain PTZ command. Here is a list for the command:

- eSrvCtrlPTZPresetAdd: the pointer to the name of the new preset location
- eSrvCtrlPTZPresetDel: the pointer to the name of preset location to be deleted
- eSrvCtrlPTZPresetRecall: the pointer to the name of the preset location to be recalled
- eSrvCtrlPTZPanSpeed: the pointer to the pan speed in string (number presented in string format)
- eSrvCtrlPTZTiltSpeed: the pointer to the tilt speed in string (number presented in string format)

- eSrvCtrlPTZFocusSpeed: the pointer to the tilt speed in string (number presented in string format)
- eSrvCtrlPTZCustom: the pointer to the customed command ID in string (number presented in string format)

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phProc will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

For supporting digital PTZ commands, you must set the device to EPTZ mode by call the ServerManager_SetPTZType, otherwise the device will use physical PTZ command.

In EPTZ mode, you may need to input stream index by set the high word of dwCamera, the low word is camera index, the stream index is start from 1.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.10 ServerManager_SetDOStatus

Set current DO information of the server. If DO is not to be set, put eSrvCtrlDiDoNone in the corresponding position of the array.

Syntax

```
SCODE ServerManager_SetDOStatus (
                                        HANDLE              hDevice,
                                        ESrvCtrlDiDoLevel   *peDOStatus,
                                        DWORD               dwMaxDOStatus,
                                        TSVRCTRL_NOTIFY     *ptNotify,
                                        HANDLE              *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

peDOStatus

[in] The pointer to an ESrvCtrlDiDoLevel to update the DO status of the server. It is the pointer of the first element of the DO status array.

dwMaxDOStatus

[in] The size value of DO status array.

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phProc will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.11 ServerManager_SetMotionDetectionInfo

Set current motion detection setting information of the server.

Syntax

```
SCODE ServerManager_SetMotionDetectionInfo (
                                        HANDLE              hDevice,
                                        DWORD               dwCamera,
                                        const TSRVCTRL_MODETECT_INFO  *ptMDInfo,
                                        TSVRCTRL_NOTIFY     *ptNotify,
                                        HANDLE              *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwCamera

[in] In the multiple camera model, it indicate the order from the right with starting as 1. Otherwise the module will ignore it.

ptMDInfo

[in] The pointer to a TSRVCTRL_MODETECT_INFO to store the motion detection to update to the server.

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phProc will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.12 ServerManager_SetPrivateMaskInfo

Set current private mask setting information of the server.

Syntax

```
SCODE ServerManager_SetPrivateMaskInfo (
                                        HANDLE                          hDevice,
                                        DWORD                           dwCamera,
                                        const TSRVCTRL_PRIVATEMASK_INFO *ptPMInfo,
                                        TSVRCTRL_NOTIFY                 *ptNotify,
                                        HANDLE                          *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwCamera

[in] In the multiple camera model, it indicate the order from the right with starting as 1. Otherwise the module will ignore it.

ptPMInfo

[out] The pointer to a TSRVCTRL_PRIVATEMASK_INFO to receive the private mask setting information to be update to the serve.

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phProc will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.13 ServerManager_UartRead

Read responded data from device's serial port. Note that the server will not cache the UART data. So if the timing is incorrect, the data may lose.

Syntax

```
SCODE ServerManager_UartRead (
                                    HANDLE          hDevice,
                                    DWORD           dwPort,
                                    DWORD           dwReadLen,
                                    BOOL            bFlush,
                                    DWORD           dwWaitTime,
                                    TSVRCTRL_NOTIFY *ptNotify,
                                    HANDLE          *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwPort

[in] The com port.

dwReadLen

[in/out] The pointer to a DWORD to represent the length of the buffer. After the function return, it will be replaced with the actual length value.

bFlush

[in] The boolean value to the server should flush the Uart or not before it begin to read data.

dwWaitTime

[in] The milliseconds unit value for the server to wait for the specified amount of data.

ptNotify

[in] The pointer **TSVRCTRL_NOTIFY** contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phProc will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, **ServerManager_SetDeviceProperty**, **ServerManager_OpenDevice**, **ServerManager_OpenDeviceBlock**, **ServerManager_AbortProcess**

4.5.14 ServerManager_UartReadAFWrite

Read responded data from device's serial port after send the command to the server. Note that the server will not cache the UART data. So if the timing is incorrect, the data may lose.

Syntax

```
SCODE ServerManager_UartReadAFWrite (
                                        HANDLE          hDevice,
                                        DWORD           dwPort,
                                        const BYTE      *pbyCommand,
                                        DWORD           dwLen,
                                        DWORD           dwReadLen,
                                        BOOL            bFlush,
                                        DWORD           dwWaitTime,
                                        TSVRCTRL_NOTIFY *ptNotify,
                                        HANDLE          *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwPort

[in] The com port.

pbyCommand

[in] The pointer to the command string.

dwLen

[in] The length value of the command.

dwReadLen

[in/out] The pointer to a DWORD to represent the length of the buffer. After the function return, it will be replaced with the actual length value.

bFlush

[in] The boolean value to the server should flush the Uart or not before it begin to read data.

dwWaitTime

[in] The milliseconds unit value for the server to wait for the specified amount of data.

ptNotify

[in] The pointer **TSVRCTRL_NOTIFY** contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phProc will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, **ServerManager_SetDeviceProperty**, **ServerManager_OpenDevice**, **ServerManager_OpenDeviceBlock**, **ServerManager_AbortProcess**

4.5.15 ServerManager_UartWrite

Send data to device's serial port.

Syntax

```
SCODE ServerManager_UartWrite (
                                    HANDLE          hDevice,
                                    DWORD           dwPort,
                                    const BYTE      *pbyCommand,
                                    DWORD           dwLen,
                                    TSVRCTRL_NOTIFY *ptNotify,
                                    HANDLE          *phProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwPort

[in] The com port.

pbyCommand

[in] The pointer to the command string.

dwLen

[in] The length value of the command.

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

phProc

[out] Pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phProc will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.16 ServerManager_UpdateLocalConfig

Update the local system parameter. It means to retrieve the configuration file from server again. Any change made locally is overwritten.

Syntax

```
SCODE ServerManager_UpdateLocalConfig (
                                        HANDLE              hDevice,
                                        TSVRCTRL_NOTIFY     *ptNotify
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess

4.5.17 ServerManager_UpdateRemoteConfig

Update the remote system parameters. It means upload the local configuration file to remote server. The server will reboot after the file is uploaded by default. To prevent server rebooting, please set the eSystemResetSystem item to “No”. But some change of the configuration will take effect only if server reboot, such as the new IP address.

Syntax

```
SCODE ServerManager_UpdateRemoteConfig (
                                        HANDLE              hDevice,
                                        TSVRCTRL_NOTIFY     *ptNotify
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess, ServerManager_UpdateRemoteConfigEx

4.5.18 ServerManager_UpdateRemoteConfigEx

This function is similar to ServerManager_UpdateRemoteConfig except that it will not cause the server reboot.

Syntax

```
SCODE ServerManager_UpdateRemoteConfigEx (
                                        HANDLE              hDevice,
                                        TSVRCTRL_NOTIFY     *ptNotify
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

ptNotify

[in] The pointer TSVRCTRL_NOTIFY contains the callback function and related information. It cannot be null and the content should be valid.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will return soon. And callback function will be raised when the operation is really done.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess, ServerManager_UpdateRemoteConfig

4.5.19 ServerManager_AbortProcess

Abort the process of the non-block mode operation.

Syntax

```
SCODE ServerManager_AbortProcess (
                                        HANDLE          hDevice,
                                        HANDLE          hProc
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from **ServerManager_CreateDevice**.

hProc

[in] The handle to the non-block mode operation handle. The handle is returned when calling non-blocking operation.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

In ServerUtl applied case, this function will return ERR_NOT_IMPLEMENT.

Requirements

ServerManager.h

See Also

4.6 Block mode API Definition

The blocking mode API definition is depicted here.

4.6.1 ServerManager_GetDIStatusBlock

Get current DI information of the server. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_GetDIStatusBlock (
                                    HANDLE          hDevice,
                                    ESrvCtrlDiDoLevel   *peDIStatus,
                                    DWORD           *pdwMaxDINum
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

peDIStatus

[out] The pointer to a ESrvCtrlDiDoLevel array buffer to receive the retrieved DI status from the server.

pdwMaxDINum

[in/out] The pointer to a DWORD to represent the size value of DI status array buffer. And the module will replace it with the actual size value of DI status array. If the actual size is more than *pdwMaxDINum, the module will return error. And the give buffer will not be filled.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.2 ServerManager_GetDOStatusBlock

Get current DO information of the server. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_GetDOStatusBlock (
                                        HANDLE          hDevice,
                                        ESrvCtrlDiDoLevel   *peDOStatus,
                                        DWORD           *pdwMaxDONum
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

peDOStatus

[out] The pointer to a ESrvCtrlDiDoLevel array buffer to receive the retrieved DO status from the server.

pdwMaxDONum

[in/out] The pointer to a DWORD to represent the size value of DO status array buffer. And the module will replace it with the actual size value of DO status array. If the actual size is more than *pdwMaxDINum, the module will return error. And the give buffer will not be filled.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

In ServerUtl applied case, this function may not find the appropriate one to do it. The module will return ERR_NOT_IMPLEMENT.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.3 ServerManager_GetMotionDetectionInfoBlock

Get current motion detection setting information of the server. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_GetMotionDetectionInfoBlock
(
                                    HANDLE                  hDevice,
                                    DWORD                   dwCamera,
                                    TSRVCTRL_MODETECT_INFO  *ptMDInfo
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwCamera

[in] In the multiple camera model, it indicates the order from the right with starting as 1. Otherwise the module will ignore it.

ptMDInfo

[out] The pointer to a TSRVCTRL_MODETECT_INFO to receive the motion detection setting information.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.4 ServerManager_GetPrivateMaskInfoBlock

Get current private mask setting information of the server. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_GetPrivateMaskInfoBlock (
                                        HANDLE                  hDevice,
                                        DWORD                   dwCamera,
                                        TSRVCTRL_PRIVATEMASK_INFO *ptPMInfo
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwCamera

[in] In the multiple camera model, it indicates the order from the right with starting as 1. Otherwise the module will ignore it.

ptPMInfo

[out] The pointer to a TSRVCTRL_PRIVATEMASK_INFO to receive the private mask setting information .

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.5 ServerManager_MoveCameraBlock

Move the camera by the specified coordinate. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_MoveCameraBlock (
                                        HANDLE      hDevice,
                                        DWORD       dwCamera,
                                        DWORD       dwStream,
                                        DWORD       dwX,
                                        DWORD       dwY,
                                        DWORD       dwDisplayWidth,
                                        DWORD       dwDisplayHeight,
                                        BOOL        bWaitRes
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from **ServerManager_CreateDevice**.

dwCamera

[in] In the multiple camera model, it indicates the order from the right with starting as 1. Otherwise it will be ignored.

dwStream

[in] In the dual-stream camera model, it indicates resolution of which stream will be get for computing the target position.

dwX

[in] The x coordinate for the new position. This position is related to the left-upper corner of current camera view. The camera will treat this value as the new center x coordinate. This value should be the normal size of camera video size.

dwY

[in] The y coordinate for the new position. This position is related to the left-upper corner of current camera view. The camera will treat this value as the new center y coordinate. This value should be the normal size of camera video size.

dwDisplayWidth

[in] The weight of the display video size.

dwDisplayHeight

[in] The height of the display video size.

bWaitRes

[in] The boolean value to wait for the server's response or not.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

If you want to control the device or get some information from the device, you must open the device first.

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

For supporting digital move command, you must set the device to EPTZ mode by call the ServerManager_SetPTZType.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.6 ServerManager_OpenDeviceBlock

Open the specified device to access some information. It will connect to the server and retrieve the server model and configuration file. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_OpenDeviceBlock (
                                        HANDLE      hDevice,
                                        char        *pszModelName,
                                        DWORD       dwMaxSize
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from **ServerManager_CreateDevice**.

pszModelName

[in] The retrieved model name.

dwMaxSize

[in] The model name of the specified device.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

If you want to control the device or get some information from the device, you must open the device first.

The function will not return the result until the whole operation has been done. But you can use **ServerManager_StopRequest** to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.7 ServerManager_OpenDeviceBlockW

Open the specified device to access some information. It will connect to the server and retrieve the server model and configuration file. This is a synchronous function and it will operate in block way. This is wide-character version of **ServerManager_OpenDeviceBlock**.

Syntax

```
SCODE ServerManager_OpenDeviceBlockW (
                                        HANDLE      hDevice,
                                        wchar_t     *pszModelName,
                                        DWORD       dwMaxSize
);
```

Parameters

hDevice

[in] The handle of the specified device. It is returned from **ServerManager_CreateDevice**.

pszModelName

[in] The retrieved model name.

dwMaxSize

[in] The model name of the specified device.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

If you want to control the device or get some information from the device, you must open the device first.

The function will not return the result until the whole operation has been done. But you can use **ServerManager_StopRequest** to abort it.

Requirements

ServerManager.h

See Also

ServerManager_OpenDeviceBlock

4.6.8 ServerManager_OpenDeviceLiteBlock

Open the specified device to access some information. It will connect to the server and retrieve the server model, but it will not to retrieve configuration file. You can retrieve configuration file later by using ServerManager_UpdateLocalConfig or ServerManager_UpdateLocalConfigBlock. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_OpenDeviceLiteBlock (
                                        HANDLE      hDevice,
                                        char        *pszModelName,
                                        DWORD       dwMaxSize
);
```

Parameters

hDevice

[in] The handle of the specified device. It is returned from ServerManager_CreateDevice.

pszModelName

[in] The retrieved model name.

dwMaxSize

[in] The model name of the specified device.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

If you want to control the device or get some information from the device, you must open the device first.

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

The function is faster than ServerManager_OpenDeviceBlock, because it will not retrieve configuration file. But it will make some functions about processing configuration file failed until you call ServerManager_UpdateLocalConfig or ServerManager_UpdateLocalConfigBlock.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.9 ServerManager_OpenDeviceLiteBlockW

Open the specified device to access some information. It will connect to the server and retrieve the server model, but it will not to retrieve configuration file. You can retrieve configuration file later by using ServerManager_UpdateLocalConfig or ServerManager_UpdateLocalConfigBlock. This is a synchronous function and it will operate in block way.

This is wide-character version of ServerManager_OpenDeviceLiteBlock.

Syntax

```
SCODE ServerManager_OpenDeviceLiteBlockW (
                                                HANDLE      hDevice,
                                                wchar_t     *pszModelName,
                                                DWORD       dwMaxSize
);
```

Parameters

hDevice

[in] The handle of the specified device. It is returned from ServerManager_CreateDevice.

pszModelName

[in] The retrieved model name.

dwMaxSize

[in] The model name of the specified device.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

If you want to control the device or get some information from the device, you must open the device first.

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

The function is faster than ServerManager_OpenDeviceBlock, because it will not retrieve configuration file. But it will make some functions about processing configuration file failed until you call ServerManager_UpdateLocalConfig or ServerManager_UpdateLocalConfigBlock.

Requirements

ServerManager.h

See Also

ServerManager_OpenDeviceLiteBlock

4.6.10 ServerManager_ReadDataBlock

This function is used to read the data from a http connection. User must ensure that the connection is already connected. Or error will be returned. If the data is still not enough, the same success code is returned. Use this function again to read further data.

The user must make sure the http connection is already established. Or it will return error.

Syntax

```
SCODE ServerManager_ReadDataBlock (
                                        HANDLE      hDevice,
                                        HANDLE      hOper,
                                        BYTE        *pbyBuffer,
                                        DWORD       *pdwBufLen
);
```

Parameters

hDevice

[in] The handle of the specified device. It is returned from **ServerManager_CreateDevice**.

hOper

[in] The http operation handle. It is returned from **ServerManager_SendHttpCommand** or **ServerManager_SendHttpCommandReadBlock**.

pbyBuffer

[out] The pointer to buffer to receive the http response data.

pdwBufLen

[in/out] The pointer to a DWORD to represent the length of the buffer. After the function return, it will be replaced with the actual length value.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

In ServerUtl applied case, this function may not find the appropriate one to do it. The module will return ERR_NOT_IMPLEMENT.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.11 ServerManager_SendHttpCommandBlock

This function is used to send an http command to server. For POST, the parameter is also put after the question mark as GET does. The result of the command is ignored by this function.

If need to read response data, please use **ServerManager_SendHttpCommandReadBlock**.

Syntax

```
SCODE ServerManager_SendHttpCommandBlock (
                                                    HANDLE          hDevice,
                                                    const char      *pszCommand,
                                                    BOOL            bPost
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from **ServerManager_CreateDevice**.

pszCommand

[in] The string pointer to an http command.

bPost

[in] Indicate the way you send the command. True is to send the command by POST, and false is by GET.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use **ServerManager_StopRequest** to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.12 ServerManager_SendHttpCommandBlockW

This function is used to send an http command to server. For POST, the parameter is also put after the question mark as GET does. The result of the command is ignored by this function.

If need to read response data, please use ServerManager_SendHttpCommandReadBlock or ServerManager_SendHttpCommandReadBlockW for wide-character version.

This is a synchronous function and it will operate in block way. This is wide-character version of ServerManager_SendHttpCommandBlock.

Syntax

```
SCODE ServerManager_SendHttpCommandBlockW (
                                                    HANDLE          hDevice,
                                                    const wchar_t   *pszCommand,
                                                    BOOL            bPost
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

pszCommand

[in] The string pointer to an http command.

bPost

[in] Indicate the way you send the command. True is to send the command by POST, and false is by GET.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

Requirements

ServerManager.h

See Also

ServerManager_SendHttpCommandBlock

4.6.13 ServerManager_SendHttpCommandReadBlock

This function is used to send an http command to server. For POST, the parameter is also put after the question mark as GET does. The result is read into pbyData until reaches the buffer length or connection is closed. If there are more data, special success code is returned and the phOper could hold the operation handle. Use **ServerManager_ReadDataBlock** to read further data.

This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_SendHttpCommandReadBlock (
                                                HANDLE          hDevice,
                                                const char      *pszCommand,
                                                BOOL            bPost,
                                                BYTE            *pbyData,
                                                DWORD           *pdwBuflen,
                                                HANDLE          *phOper
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from **ServerManager_CreateDevice**.

pszCommand

[in] The string pointer to an http command.

bPost

[in] Indicate the way you send the command. True is to send the command by POST, and false is by GET.

pbyData

[out] The pointer to buffer to receive the http response data.

pdwBuflen

[in/out] The pointer to a DWORD to represent the length value of buffer. It will be replaced with the length value of the response data.

phOper

[out] The pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phOper will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use **ServerManager_StopRequest** to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, **ServerManager_SetDeviceProperty**, **ServerManager_OpenDevice**, **ServerManager_OpenDeviceBlock**

4.6.14 ServerManager_SendHttpCommandReadBlockW

This function is used to send an http command to server. For POST, the parameter is also put after the question mark as GET does. The result is read into pbyData until reaches the buffer length or connection is closed. If there are more data, special success code is returned and the phOper could hold the operation handle.

This is a synchronous function and it will operate in block way. This is wide-character version of ServerManager_SendHttpCommandReadBlock.

Syntax

```
SCODE ServerManager_SendHttpCommandReadBlockW (
                                                HANDLE          hDevice,
                                                const wchar_t   *pszCommand,
                                                BOOL            bPost,
                                                BYTE            *pbyData,
                                                DWORD           *pdwBuflen,
                                                HANDLE          *phOper
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

pszCommand

[in] The string pointer to an http command.

bPost

[in] Indicate the way you send the command. True is to send the command by POST, and false is by GET.

pbyData

[out] The pointer to buffer to receive the http response data.

pdwBuflen

[in/out] The pointer to a DWORD to represent the length value of buffer. It will be replaced with the length value of the response data.

phOper

[out] The pointer to a http operation handle. The http operation handle is used to read more data. The module will pass a handle to the caller if there is more data to be read. If none, the phOper will be null. The users do not need to free the http operation handle. It is kept internally.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use **ServerManager_StopRequest** to abort it.

Requirements

ServerManager.h

See Also

ServerManager_SendHttpCommandReadBlock

4.6.15 ServerManager_SendPTZCommandBlock

Send the PTZ command to the server. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_SendPTZCommandBlock (
                                        HANDLE              hDevice,
                                        DWORD               dwCamera,
                                        ESRVCTRL_PTZ_COMMAND ePtzCommand,
                                        const char          *pszExtraCmd,
                                        BOOL                bWaitRes
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwCamera

[in] In the multiple camera model, it indicates the order from the right with starting as 1. Otherwise it will be ignored.

ePtzCommand

[in] The ESRVCTRL_PTZ_COMMAND to represent a command will be sent to the target device. Not all commands are accepted in every camera, so you may check the user's manual to make sure it.

pszExtraCmd

[in] The extra information for certain PTZ command. Here is a list for the command:

- eSrvCtrlPTZPresetAdd: the pointer to the name of the new preset location
- eSrvCtrlPTZPresetDel: the pointer to the name of preset location to be deleted
- eSrvCtrlPTZPresetRecall: the pointer to the name of the preset location to be recalled
- eSrvCtrlPTZPanSpeed: the pointer to the pan speed in string (number presented in string format)
- eSrvCtrlPTZTiltSpeed: the pointer to the tilt speed in string (number presented in string format)

- eSrvCtrlPTZFocusSpeed: the pointer to the tilt speed in string (number presented in string format)
- eSrvCtrlPTZCustom: the pointer to the customer command ID in string (number presented in string format)

bWaitRes

[in] The boolean value to wait for the server's response or not.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

For supporting digital PTZ command, you must set the device to EPTZ mode by call the ServerManager_SetPTZType, otherwise the device will use physical PTZ command.

In EPTZ mode, you may need to input stream index by set the high word of dwCamera, the low word is camera index, the stream index is start from 1.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.16 ServerManager_SetDOStatusBlock

Sets current DO information of the server. This is a synchronous function and it will in block the working thread.

Syntax

```
SCODE ServerManager_SetDOStatusBlock (
                                        HANDLE          hDevice,
                                        ESrvCtrlDiDoLevel *peDOStatus,
                                        DWORD           dwMaxDOStatus,
                                        BOOL            bWaitRes
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

peDOStatus

[in] The pointer to a ESrvCtrlDiDoLevel to update the DO status of the server. It is the pointer of the first element of the DO status array.

dwMaxDOStatus

[in] The size value of DO status array.

bWaitRes

[in] The boolean value to wait for the server's response or not.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.17 ServerManager_SetMotionDetectionInfoBlock

Set current motion detection setting information of the server. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_SetMotionDetectionInfoBlock (
                                                    HANDLE                          hDevice,
                                                    DWORD                           dwCamera,
                                                    const TSRVCTRL_MODETECT_INFO    *ptMDInfo,
                                                    BOOL                            bWaitRes
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwCamera

[in] In the multiple camera model, it indicate the order from the right with starting as 1. Otherwise the module will ignore it.

ptMDInfo

[in] The pointer to a TSRVCTRL_MODETECT_INFO to store the motion detection to update to the server.

bWaitRes

[in] The boolean value to wait for the server's response or not.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.18 ServerManager_SetPrivateMaskInfoBlock

Set current private mask setting information of the server. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_SetPrivateMaskInfoBlock
(
    HANDLE                          hDevice,
    DWORD                           dwCamera,
    const TSRVCTRL_PRIVATEMASK_INFO *ptPMInfo,
    BOOL                            bWaitRes
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwCamera

[in] In the multiple camera model, it indicate the order from the right with starting as 1. Otherwise the module will ignore it.

ptPMInfo

[out] The pointer to a TSRVCTRL_PRIVATEMASK_INFO to receive the private mask setting information to be update to the serve.

bWaitRes

[in] The boolean value to wait for the server's response or not.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.19 ServerManager_UartReadBlock

Read responded data from device's serial port. Note that the server will not cache the UART data. So if the timing is incorrect, the data may lose. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_UartReadBlock (
                                    HANDLE      hDevice,
                                    DWORD       dwPort,
                                    BYTE        *pszResponse,
                                    DWORD       *pdwReadLen,
                                    BOOL        bFlush,
                                    DWORD       dwWaitTime
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwPort

[in] The com port.

pszResponse

[out] The pointer to the string to receive the response data.

pdwReadLen

[in/out] The pointer to a DWORD to represent the length of the buffer. After the function return, it will be replaced with the actual length value.

bFlush

[in] The boolean value to the server should flush the Uart or not before it begin to read data.

dwWaitTime

[in] The milliseconds unit value for the server to wait for the specified amount of data.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use **ServerManager_StopRequest** to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, **ServerManager_SetDeviceProperty**, **ServerManager_OpenDevice**, **ServerManager_OpenDeviceBlock**

4.6.20 ServerManager_UartReadAFWriteBlock

Read responded data from device's serial port after send the command to the server. Note that the server will not cache the UART data. So if the timing is incorrect, the data may lose. This is a synchronous function and it will operate in block way

Syntax

```
SCODE ServerManager_UartReadAFWriteBlock (
                                        HANDLE          hDevice,
                                        DWORD           dwPort,
                                        const BYTE      *pbyCommand,
                                        DWORD           dwLen,
                                        BYTE            *pszResponse,
                                        DWORD           *pdwReadLen,
                                        BOOL            bFlush,
                                        DWORD           dwWaitTime
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwPort

[in] The com port.

pbyCommand

[in] The pointer to the command string.

dwLen

[in] The length value of the command.

pszResponse

[out] The pointer to the string to receive the response data.

pdwReadLen

[in/out] The pointer to a DWORD to represent the length of the buffer. After the function return, it will be replaced with the actual length value.

bFlush

[in] The boolean value to the server should flush the Uart or not before it begin to read data.

dwWaitTime

[in] The milliseconds unit value for the server to wait for the specified amount of data.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use **ServerManager_StopRequest** to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, **ServerManager_SetDeviceProperty**, **ServerManager_OpenDevice**, **ServerManager_OpenDeviceBlock**

4.6.21 ServerManager_UartWriteBlock

Send data to device's serial port. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_UartWriteBlock (
                                    HANDLE          hDevice,
                                    DWORD           dwPort,
                                    const BYTE      *pbyCommand,
                                    DWORD           dwLen,
                                    BOOL            bWaitRes
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

dwPort

[in] The com port.

pbyCommand

[in] The pointer to the command string.

dwLen

[in] The length value of the command.

bWaitRes

[in] The boolean value to wait for the server's response or not.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock

4.6.22 ServerManager_UpdateLocalConfigBlock

Update the local system parameter. It means to retrieve the configuration file from server again. Any change made locally is overwritten. This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_UpdateLocalConfigBlock (
                                                HANDLE                          hDevice
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from **ServerManager_CreateDevice**.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use **ServerManager_StopRequest** to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, **ServerManager_SetDeviceProperty**, **ServerManager_OpenDevice**, **ServerManager_OpenDeviceBlock**

4.6.23 ServerManager_UpdateRemoteConfigBlock

Update the remote system parameters. It means upload the local configuration file to remote server. The server will reboot after the file is uploaded by default. To prevent server rebooting, please set the eSystemResetSystem item to “No”. But some change of the configuration will take effect only if server reboot, such as the new IP address.

This is a synchronous function and it will operate in block way.

Syntax

```
SCODE ServerManager_UpdateRemoteConfigBlock(
                                                HANDLE                  hDevice
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess, ServerManager_UpdateRemoteConfigBlockEx

4.6.24 ServerManager_UpdateRemoteConfigBlockEx

This function is similar to ServerManager_UpdateRemoteConfigBlock, except that calling this function does not cause the server reboot.

Syntax

```
SCODE ServerManager_UpdateRemoteConfigBlockEx (
                                                    HANDLE                    hDevice
);
```

Parameters

hDevice

[in] The handle of the targeted device. It is returned from ServerManager_CreateDevice.

Return Values

Returns S_OK if successful, or an error value otherwise.

Remarks

The function will not return the result until the whole operation has been done. But you can use ServerManager_StopRequest to abort it.

Requirements

ServerManager.h

See Also

ServerManager_CreateDevice, ServerManager_SetDeviceProperty, ServerManager_OpenDevice, ServerManager_OpenDeviceBlock, ServerManager_AbortProcess, ServerManager_UpdateRemoteConfigBlock

5. Appendix

5.1 Error Code

- **SVRMNGR_E_TOOMANY_DEVICE – 0x80090000**

  It has created too many devices in ServerManager

- **SVRMNGR_E_NOTLOAD_SRVCTRLLIB - 0x80090001**

  ServerController library is not loaded successfully.

- **SVRMNGR_E_NOTLOAD_SRVUTILLIB - 0x80090002**

  ServerUtility library is not loaded successfully.

- **SVRMNGR_E_DEVICE_NOT_OPEN - 0x80090003**

  ServerManager is not open device yet (not call ServerManager_OpenDeviceBlock or ServerManager_OpenDevice).

- **SVRMNGR_E_PARTIAL_NOT_IMPLEMENT - 0x80090004**

  It may not find the appropriate function to work in some conditions.

- **SVRMNGR_E_NOT_MAPPING - 0x80090005**

  It cannot find corresponding ServerUtility config item index of parameter item in RX (new model).

- **SVRMNGR_E_KEY_NOT_FOUND - 0x80090006**

  It cannot find the item in functions ServerManager_GetValueByName.

- **SVRMNGR_W_MUST_UPDATEREMOTE_FIRST - 0x80090007**

  This is warning that you may do ServerManager_UpdateRemoteConfigBlock or ServerManager_UpdateRemoteConfig first.

- **SVRMNGR_E_TIMEOUT - 0x80090008**

  ServerManager occur timeout.

- **SVRMNGR_E_AUTH - 0x80090009**

  It occurs authentication failed when trying to connect to server.

- **SVRMNGR_E_CONNECT_FAILED - 0x8009000a**

ServerManager connect to server failed.

- **SVRMNGR_E_PAGE_NOT_FOUND - 0x8009000b**

  It cannot find the page in Http request.

- **SVRMNGR_E_ERROR_MODEL - 0x8009000c**

  Use wrong library between ServerController and ServerUtility.

- **SVRMNGR_E_CONVERT_UNITOMB - 0x8009000d**

  It occurs error when converting Unicode to multi-byte string.

- **SVRMNGR_E_READ_DATA_ERROR - 0x8009000e**

  The data that read from server is something wrong.

- **SVRMNGR_E_FTP_GETFILE - 0x8009000f**

  Get data from ftp server failed.

- **SVRMNGR_E_FTP_PETFILE - 0x80090010**

  Put data to ftp server failed.

- **SVRMNGR_E_OPENFILE - 0x80090011**

  Open file failed.

- **SVRMNGR_E_ABORTING - 0x80090012**

  The request is be aborted.

- **SVRMNGR_E_LOAD_LIBRARY - 0x80090013**

  Both libraries are not loaded.

- **SVRMNGR_E_TOOMANY_NONBLOCK_OPER - 0x80090014**

  Too many non-blocking operations are proceeding.