Wisenet SSM Client SDK v2.10.7

Programmer's Guide

for C# programming

WISENET_SSM_CLIENT_SDK_PG_EN

2.10.7

2020-05-15

Contact Information

HANWHA TECHWIN Co., LTD.

Hanwha techwin R&D Center, 6, Pangyo-ro 319beon-gil, Bundang-gu, Seongnam-si, Gyeonggi-do, Korea, 463-400

TEL: +82-70-7147-8740~60 FAX: +82-31-8018-3745

https://step.hanwha-security.com

Hanwha Techwin America Inc.

100 Challenger Road Ridgefield Park, New Jersey, 07660 U.S.A.

Hanwha Techwin Europe Ltd.

2nd Floor, No. 5 The Heights, Brooklands, Weybridge, Surrey, KT13 0NY, U.K.

Park Chertsey, Surrey, UNITED KINGDOM KT16 OPS

Preface

Objectives

This document describes how to develop applications using Hanwha Techwin's Wisenet SSM Client SDK.

Reader

This document is intended for those who use the SSM OpenConsole SDK to develop applications (CMS, viewers, and applications).

.

Scope

This document describes how to develop applications using Wisenet SSM Client SDK, how to implement sample programs, and more.

Document Organization

This document is organized as follows.

- CHAPTER 1. Wisenet SSM Client SDK
- CHAPTER 2. Setting up the development environment
- CHAPTER 3. Introduction of sample program
- CHAPTER 4. LogIn Sample
- CHAPTER 5. PasswordChange Sample
- CHAPTER 6. Search Sample
- CHAPTER 7. Backup Sample
- CHAPTER 8. SingleLive Sample
- CHAPTER 9. PTZ Sample
- CHAPTER 10. MultiMonitoring Sample
- CHAPTER 11. Playback Sample
- CHAPTER 12. InsertLog Sample
- CHAPTER 13. LiveSnapshot Sample
- CHAPTER 14. LiveLocalRecord Sample
- CHAPTER 15. UserManagement Sample
- CHAPTER 16. NTP Sample
- CHAPTER 17. DeviceRecord Sample
- CHAPTER 18. Wisenet DDNS Login Sample
- CHAPTER 19. URL Login Sample

Convention

Indicating button or menu: Button names and menu names are surrounded by brackets([ ]).

Indicating menu selection route: Menu selection routes are displayed by using (>).

Revision History

Refer to the following table for document versions and revision history.

Version	Date	Description
2.0	2017. 08. 31	First draft
2.0	2017. 10. 31	SDK's name has been changed.
2.10.1	2018. 10. 30	The version has been changed.
2.10.1	2018. 11. 16	Some functions have been added.
2.10.3	2019. 04. 15	DeviceRecord sample has been added.
2.10.4	2019. 05. 15	Active X controls have been deleted. The C++/CLI wrapper has been added.
2.10.5	2019. 07. 10	Wisenet DDNS Login, URL Login samples have been added.
2.10.5	2019. 07. 25	Send event with Event Key.
2.10.6	2020. 03. 24	Backup Sample – Track ID description has been added.

Table of Contents

List of figures

List of Tables

CHAPTER 1

Wisenet SSM Client SDK

This chapter describes the overview, composition and suppoted server of Wisenet SSM Client SDK.

Contents

Overview

Wisenet SSM(Wisenet Smart Security Manager) Client SDK is the library created by adapting Microsoft .NET Framework 4.5.

The Wisenet SSM Client SDK supports video/audio streaming and event receiving from the device which registered at Wisenet SSM Core Server.

To develop an application using the Wisenet SSM Client SDK, you need C DLL provided by SDK.

Using the installation file of Wisenet SSM Client SDK, the DLL files are saved in the specified location. Please refer to 'SSM manual' for installation.

Wisenet SSM Client SDK provides functions through network connection with Wisenet SSM Core Server. The Wisenet SSM Core Server program must be installed through the Wisenet SSM Core Server installation file. Please refer to 'SSM manual' for installation.

The main functions provided by Wisenet SSM Client SDK are as follows.

- Core Server login / logout, change password
- Real-time event reception from network equipments
- Live video surveillance and playback of recorded video
- PTZ Control
- Recording search
- Backup recorded video
- User management
- Device record control

SDK Architecture

The structure of Wisenet SSM Client SDK is as follows.

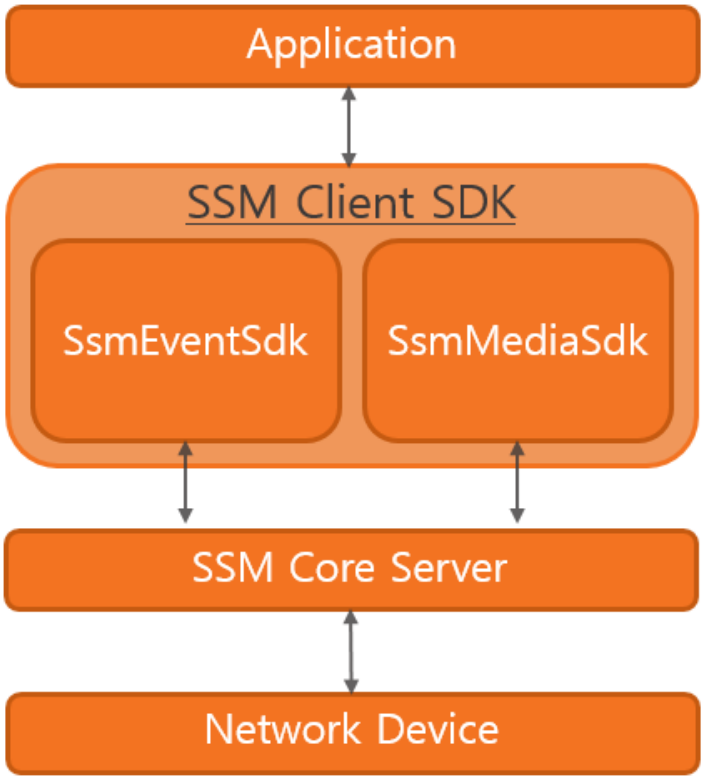


Figure 1 Wisenet SSM Client SDK Architecture

- Application
  - Programs developed using the Wisenet SSM Client SDK
- Wisenet SSM Client SDK
  - SsmEventSdk: A library containing the main functions of SSM Console
  - SsmMediaSdk: A library perform functions related to media control such as receving video/audio/metadata, decoding and rendering.
- Wisenet SSM Core Server
  - Main server of SSM
  - Manage users, devices, messaging, and more.
  - Media Relay Server of SSM
  - Transfer media data to the user program and manage the device
  - Record video and audio data

- Network Device
  - NVR, DVR, etc.
  - Analog/Network camera
  - Encoder

SDK Configuration

The Wisenet SSM Client SDK consists of the following:

- Library
  - DLL files used for application development.
  - Installed in Bin, Bin_x64 folder.
- Document
  - API and programming guide.
  - Saved in {$SDK path}₩Doc folder.
- Sample execution files
  - Programs which have been developed using the SDK.
  - Saved in {$SDK path}₩Bin, {$SDK path}₩Bin_x64 folder.
- Sample source code
  - Source of the sample program.
  - Saved in {$SDK path}₩Sample folder.

System Requirements

The following system requirements exist for using the Wisenet SSM Client SDK.

- Operation system(OS)
  - Microsoft Windows 7 or higher version.
- .NET Framework
  - Microsoft .NET Framework 4.5
- Wisenet SSM Core Server
  - v2.10.6
- CPU
  - Intel i5 or higher.
- Development environment
  - Microsoft Visual Studio 2013 or higher.

Server Supported

The same version between Wisenet SSM Client SDK and Wisenet SSM Core Server is essential for proper operation.

For example, if you are using Wisenet SSM Core Server v2.10.6, you need to use Wisenet SSM Client SDK v2.10.6.

CHAPTER 2

Development Enviroment Setting

This chapter describes how to set up your development environment.

Contents

Prior Knowledge

The Wisenet SSM Client SDK's sample codes are based on C# and .NET Framework 4.5. For more details, refer to the MSDN.

C#

https://msdn.microsoft.com/en-us/library/kx37x362(v=vs.120).aspx

.NET Framework 4.5

https://msdn.microsoft.com/en-US/library/k1s94fta(v=vs.100).aspx

Creating Projects

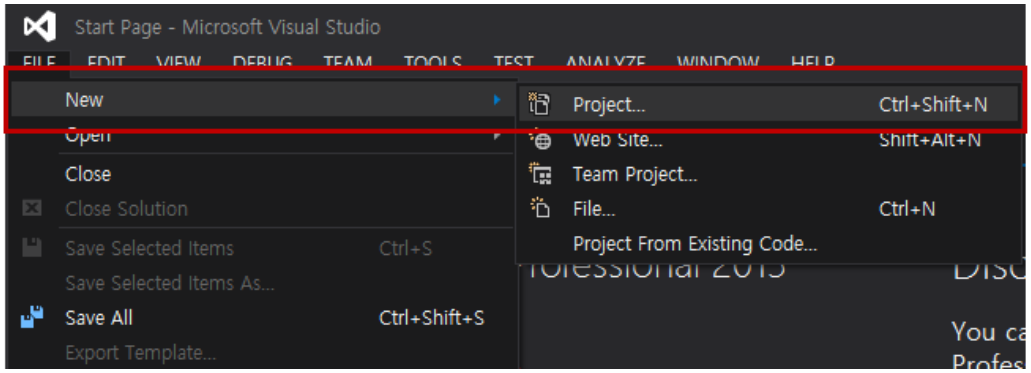
You can create projects necessary for developing applications as follows. (Explanations in this document are based on the Visual Studio 2013 version.)

Visual Studio 2013

Procedures

Step 1. Execute Visual Studio 2013.

Step 2. Select [File] > [New] > [Project].

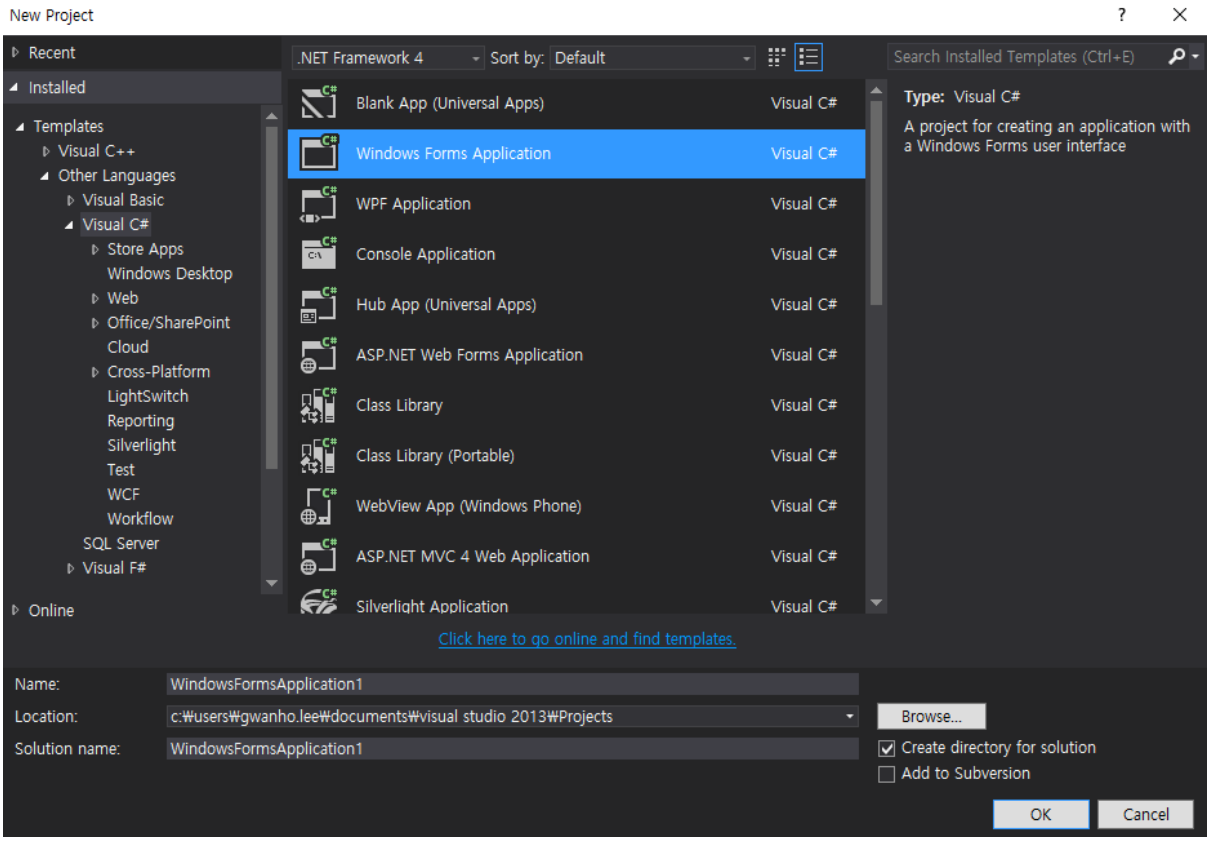


Step 1. In the [New Project] dialogue,

A. Select the "Visual C#", "Windows Forms Application" template.

B. Designate a project name and path.

A. Click on [OK].



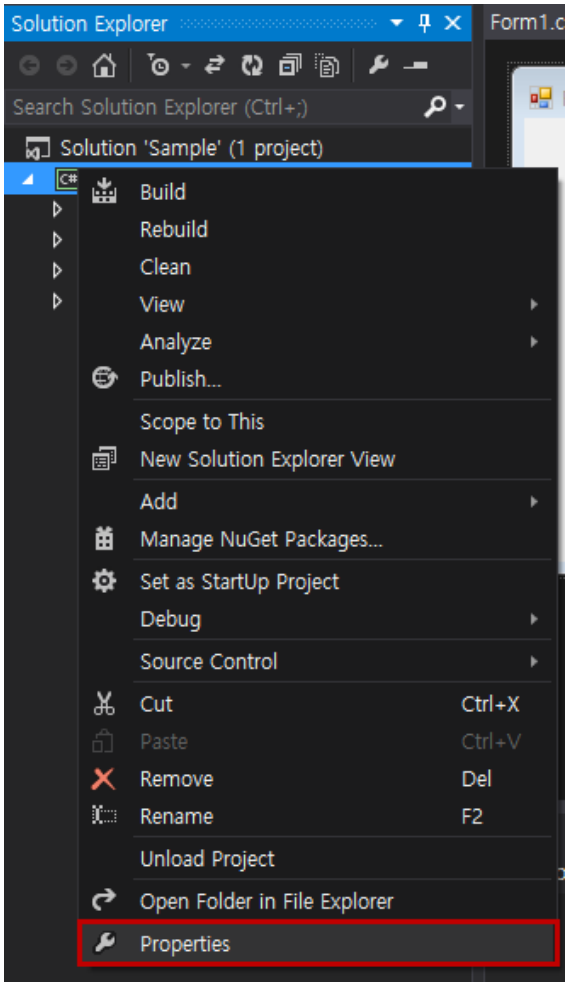
Setting Projects

After preparing an application, you have to first set the project properties to build the application. In Visual Studio 2013, you can set the project properties as follows

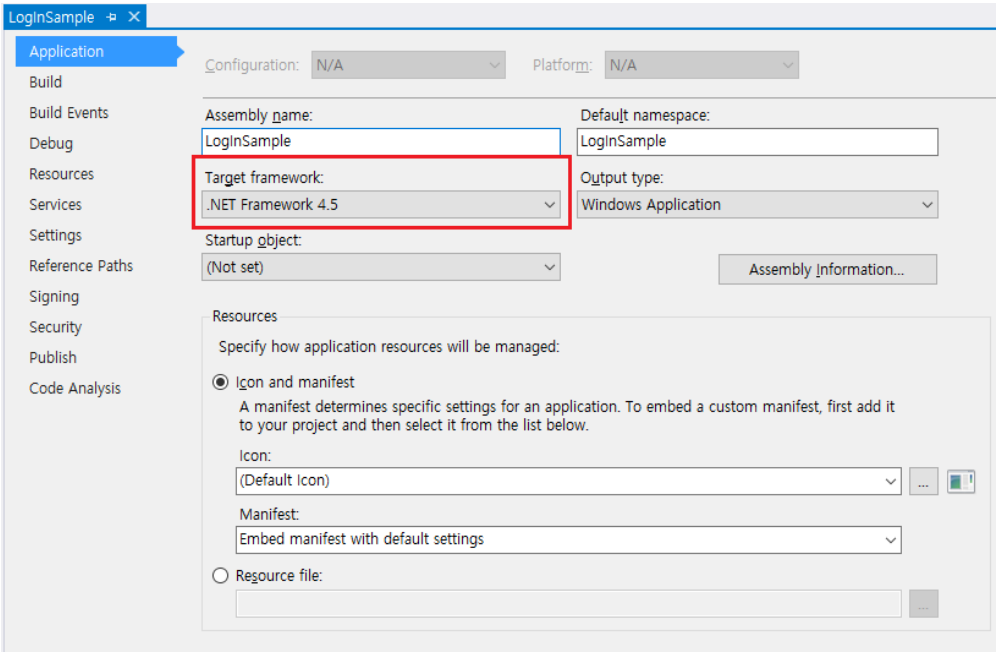
Visual Studio 2013

Procedures

Step 1. In [Solution Explorer], select a project and right click with your mouse button to select the [Properties] menu.



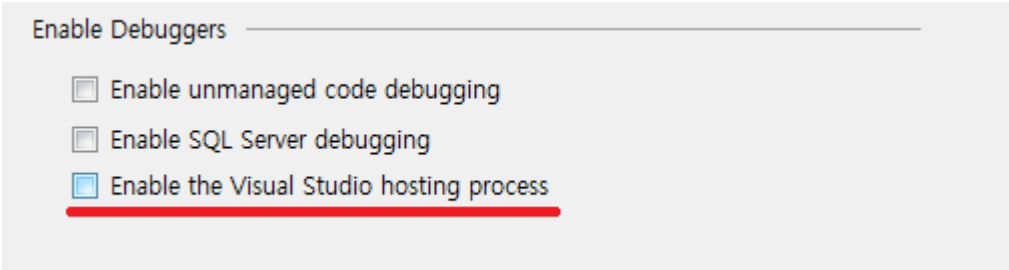
Step 2. Select [General] in the left tree menu, and select [.NET Framework 4.5] in the [Target Framework] field. Click on [OK].



How to trouble shoot C# debugs in XP or higher

Procedures

Step 1. Go to [Properties] – [Debug] and see if "Enable the Visual Studio hosting process" is checked as shown in the below figure. If so, uncheck it.



DEP Troubleshooting Methods

The data execution prevention (DEP) aims to guarantee safe use of the system memory by monitoring a program. It is the security function of your window that can prevent your computer from being damaged by viruses or other security threats. Due to the DEP function setting, your SDK program may not function properly. In this case, you can disable DEP in the development environment as follows.

.

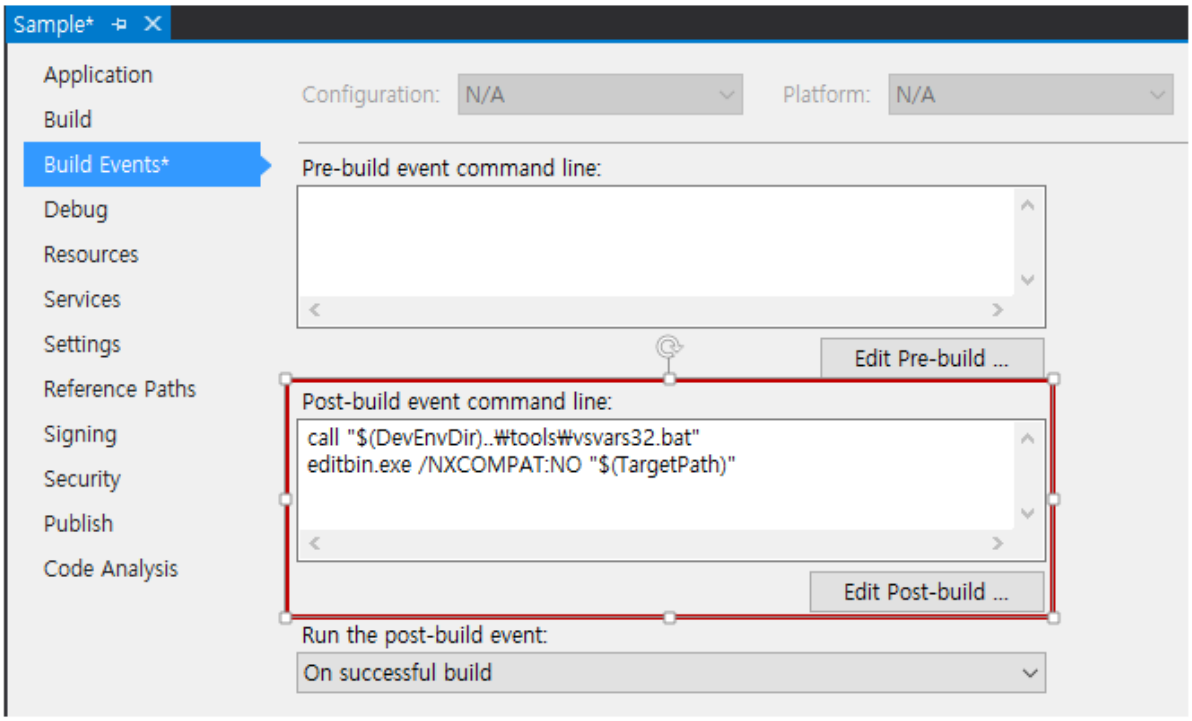
C# Project Setting

Procedures

Step 1. In [Solution Explorer], select a project and then right click with your mouse button to select the [Properties] menu.

Step 2. After selecting the [**Build Events**] tab, enter the following command.

```
call "$(DevEnvDir)..₩tools₩vsvars32.bat"
editbin.exe /NXCOMPAT:NO "$(TargetPath)"
```



CHAPTER 3

Introduction to Sample Programs

This chapter introduces the sample programs provided by the Wisenet SSM Client SDK. The sample program uses the SsmEventSdk and SsmMediaSdk libraries to illustrate examples of programs that you can implement

Contents

Locations

The sample programs can be found in the {$SDK path}₩sample_code folder.

Sample Program List

Table 1 Sample Program List

Sample Program	Description
LogIn	It describes how to login to Wisenet SSM Core Server.
PasswordChange	It describes how to change user password.
Backup	It describes how to backup recorded video.
Search	It describes how to search for information related to recorded video.
SingleLive	It describes how to receive 1 channel live video.
PTZ	It describes how to use Pan, Tilt, Zoom and Preset functions of PTZ camera.
MultiMonitoring	It describes how to receive multiple live video.
Playback	It describes how to receive recorded video from the server.
InsertLog	It describes how to send and log events to the server.
LiveSnapshot	It describes how to get live snapshot.
LiveLocalRecord	It describes how to save .avi file from live video.
UserManagement	It describes how to manage user informations.
NTP	It describes how to modify NTP setting.
Device Record	It describes how to start/stop the device recording.

CHAPTER 4

LogIn Sample Program

This chapter describes how to login to the Wisenet SSM core server.

Contents

Introduction

This sample program is an example of how to login to Wisenet SSM Core Server. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

Step 3. Terminate the program.

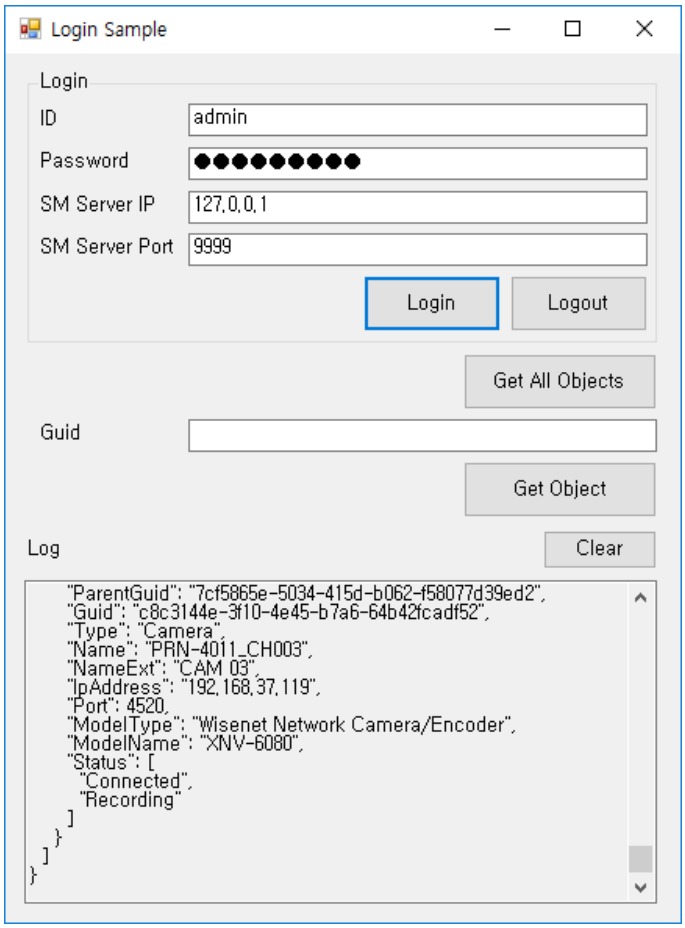


Figure 2 LogIn Sample Program Execution

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.Initialize()

Login

2. ssmSdkWrapper.Login()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Logout

5. ssmSdkWrapper.Logout()

Release

6. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the LogIn sample program in detail.

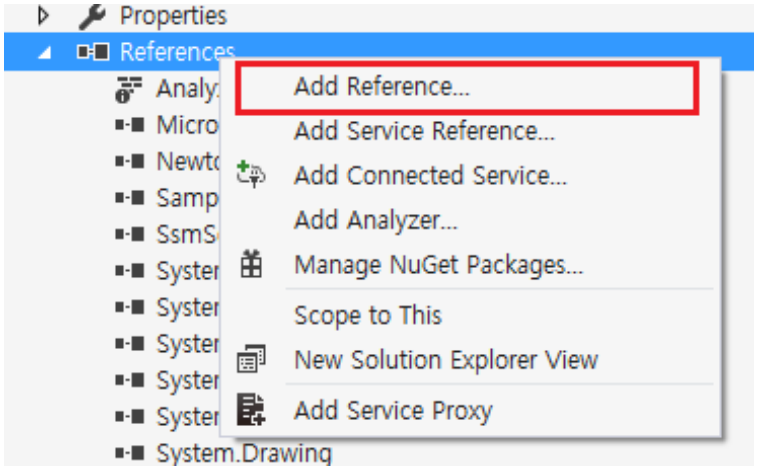
Add SsmSdkWrapper project to the reference

Procedures

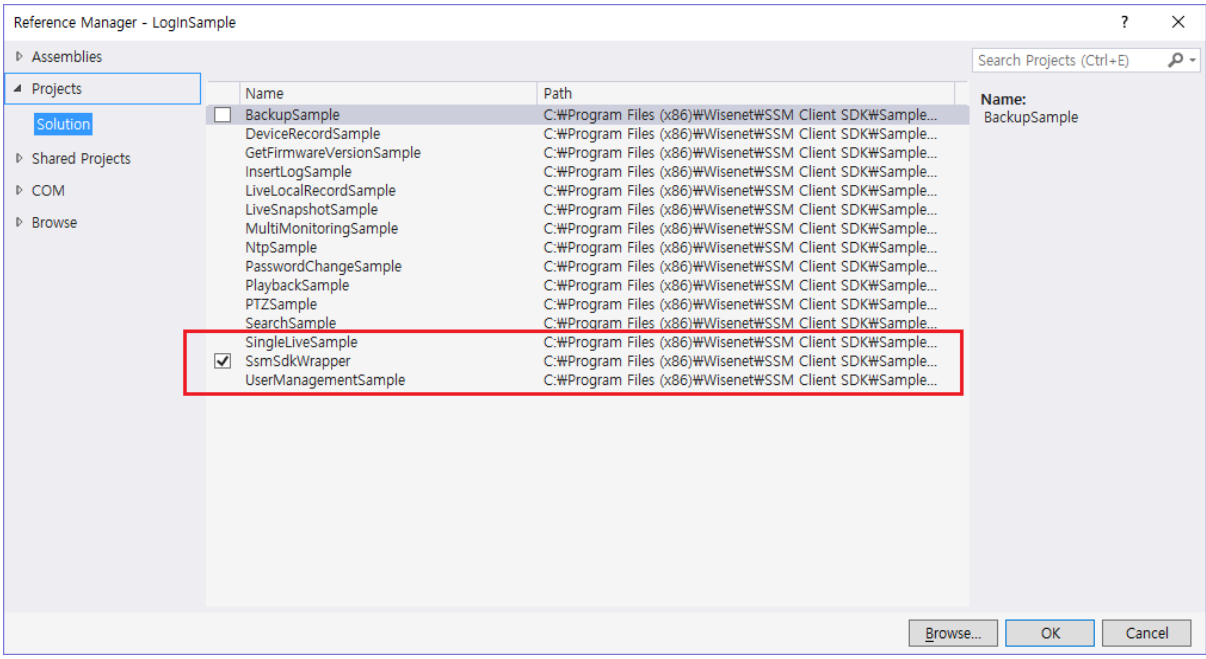
In Visual Studio 2013, create Windows Forms and then add SsmSdkWrapper reference.

Step 1. Create Windows Forms.

Step 2. Click the right mouse button on References. And choose [Add Reference]



Step 3. Add the SsmSdkWrapper on Reference Manager.



Add SsmSdkWrapper Callback functions

In order to receive the response and the event from the server, you need to implement Callback functions.

Implement Callback: OnResponse

Implement the OnResponse in the LogInSample class.

```
// LoginSample.cs
private void OnResponse(UInt32 commandID, UInt32 errorCode, UInt32 sequenceID, string info)
{
        _logger.WLOGD(
         "OnResponse()::" +
         " Command ID=" + commandID +
         " Result=" + errorCode +
         " Sequence ID= " + sequenceID +
         " Info=" + info);
}
```

Implement Callback: OnEvent

Implement the OnEvent in the LogInSample class.

```
// LoginSample.cs
private void OnEvent(UInt32 eventID, String info)
{
        _logger.WLOGD(
                "OnEvent()::" +
                " Event ID=" + eventID +
                " Info=" + info);
}
```

Initialize SsmSdkWrapper

Create an SsmSdkWrapper and initialize it by calling ssmSdkWrapper.Initialize().

Call the InitializeEvent() method within the LogInSample_Load() of the LoginSample class. InitializeEvent() starts using the service of the SsmEventSdk.

```
// LogInSample.cs
private SsmSdkWrapper ssmSdkWrapper = null;

private void LogInSample_Load(object sender, EventArgs e)
{
        ssmSdkWrapper = new SsmSdkWrapper(this.OnResponse, this.OnEvent);
        ssmSdkWrapper.InitializeEvent();
}
```

Login

By creating a login button event handler, you can execute logging in. You need the IP address, port number, ID and password of Wisenet SSM Core Server to login.

```
// LogInSample.cs
private void btnLogIn_Click(object sender, EventArgs e)
{
        uint resCode = ssmSdkWrapper.LogIn(
                txtID.Text,
                txtPassword.Text,
                txtIPAddress.Text,
                Convert.ToUInt32(txtPort.Text));
        _logger.WLOGD("LogIn()::Result=" + resCode);
}
```

Logout

By creating a logout button event handler, you can execute logging out.

```
// LogInSample.cs
private void btnLogOut_Click(object sender, EventArgs e)
{
        uint resCode = ssmSdkWrapper.Logout();
```

```
        _logger.WLOGD("LogOut()::Result=" + resCode);
}
```

Release SsmSdkWrapper

After using SsmEventSdk, you must stop the service and release the resources.

Call the ReleaseEvent() method within the LogInSample_Closed() of the LoginSample class.

```
// LogInSample.cs
private void LogInSample_Closed(object sender, EventArgs e)
{
        ssmSdkWrapper.ReleaseEvent();
}
```

See also

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 5

PasswordChange Sample Program

This chapter describes how to change password with using Wisenet SSM Client SDK.

Contents

Introduction

This sample program is an example of changing the password for the Wisenet SSM user account. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

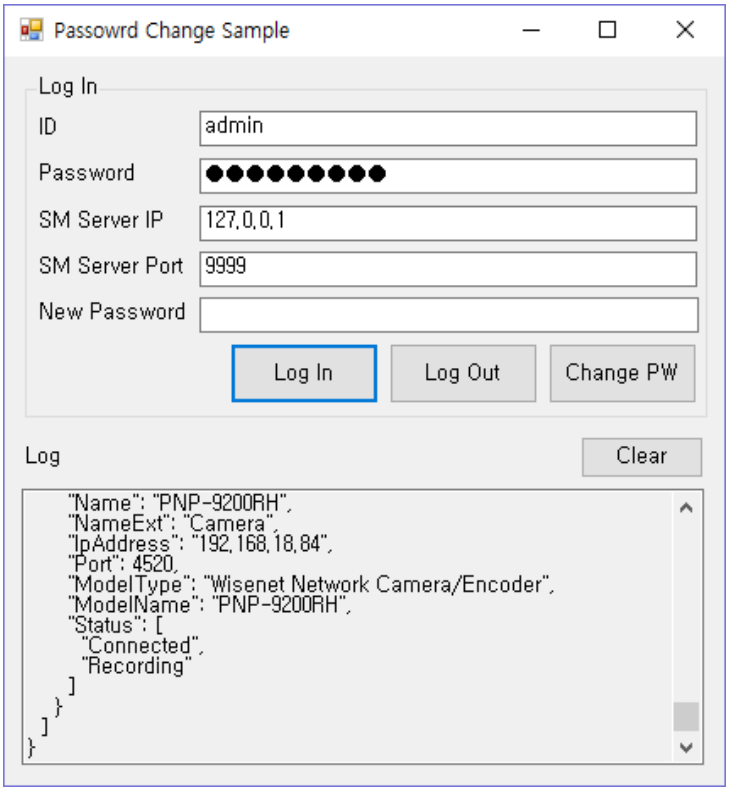


Figure 3 PasswordChange Sample Program Execution

Step 3. Check the ResultCode of the OnResponse to see if login was successful.

Step 4. Enter the new password to the [New Password] textbox.

Step 5. Click the [Change PW] button to request a password change

Step 6. In the [Log] window, check the response received from the server.

Step 7. Click on [Logout] to release a connection to the server

Step 8. Terminate the program

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()

Login

2. ssmSdkWrapper.Login()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

5. ssmSdkWrapper.OnEvent

Password change

6. ssmSdkWrapper.ChangePassword()

Response of Password change

7. ssmSdkWrapper.OnResponse

Logout

8. ssmSdkWrapper.Logout()

Release

9. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the PasswordChange sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing SsmSdkWrapper

Add SsmSdkWrapper project to the reference

> **Note**
> For more information on how to add SsmSdkWrapper project to the reference, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

Add SsmSdkWrapper Callback functions

Implement Callback Function: OnResponse

> **Note**
> For more information on how to implement callback function: OnResponse, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

Implement Callback Function: OnEvent

> **Note**
> For more information on how to implement callback function: OnEvent, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

Initialize SsmSdkWrapper

> **Note**
> For more information on how to initialize the SsmSdkWrapper, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

Login

> **Note**
> For more information on how to implement the Login, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

Change password

Create the event handler for clicking the change password function button and execute the command to send the change password function command. ID, current password, and new password are required as arguments to the method.

```
// PasswordChangeSample.cs
private void btnChangePassword_Click(object sender, EventArgs e)
{
        UInt32 sequenceID = 0;
        uint resCode = ssmSdkWrapper.ChangePassword(
                txtID.Text,
                txtPassword.Text,
                txtNewPassword.Text,
                ref sequenceID);
        _logger.WLOGD("ChangePassword()::Result=" + resCode + ", SequenceID=" +
sequenceID);
}
```

Logout

> **Note**
> For more information on how to implement the logout, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

Release SsmSdkWrapper

> **Note**
> For more information on how to release the SsmSdkWrapper, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

See also

LogIn sample program.

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 6

Search Sample Program

This chapter describes how to obtain information related to recorded video using Wisenet SSM Client SDK.

Contents

Introduction

The Search sample program is an example of obtaining information related to recorded video from a network device. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Login] button to see if you can successfully access the Wisenet SSM Core Server.

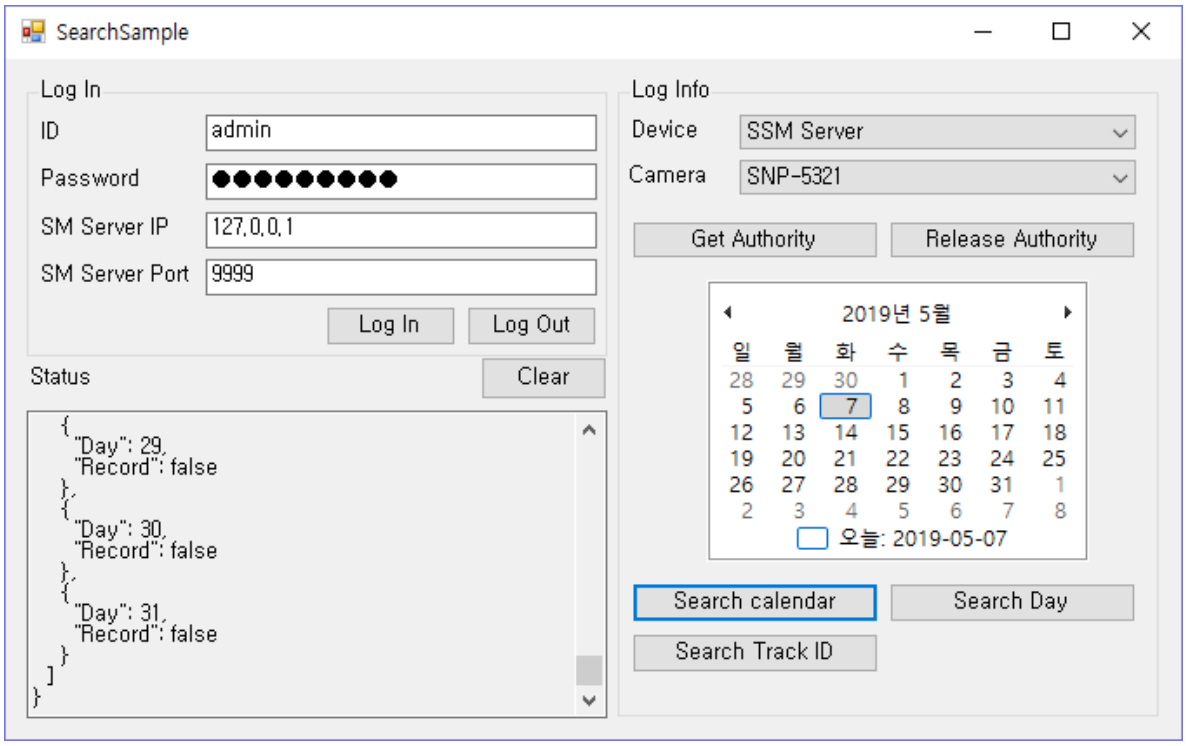


Figure 4 Search Sample Program Execution

Step 3. Check the ResultCode of the OnResponse to see if login was successful.

Step 4. When login is executed successfully, the name of connected devices and cameras appear in the [Device] and [Camera] ComboBox.

Step 5. Select Device and Camera in ComboBox.

Step 6. Click the [Get Authority] button to get the search authority of the selected device.

Step 7. Click the [Search Calendar] button and get the dates when the video was recorded in the selected month.

Step 8. Click the [Search Day] button and get the time section and record type of the video recorded on the selected date.

Step 9. Click the [Search TrackID] button to get track ID information of the selected recording section.

Step 10. Click the [Release Authority] button to release the search authority of the selected device

Step 11. Click on [Logout] to release a connection to the server.

Step 12. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()

Login

2. ssmSdkWrapper.Login()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

5. ssmSdkWrapper.OnEvent

GetSearchAuthority

6. ssmSdkWrapper.GetSearchAuthority

Response of GetSearchAuthority

7. ssmSdkWrapper.OnResponse

Search Calendar

8. ssmSdkWrapper.SearchCalendar()

Response of SearchCalendar

9. ssmSdkWrapper.OnResponse

Search TrackID

10. ssmSdkWrapper.SearchTrackID()

Response of SearchDay

11. ssmSdkWrapper.OnResponse

Search Day

12. ssmSdkWrapper.SearchDay()

Release Authority

13. ssmSdkWrapper.ReleaseAuthority()

Logout

14. ssmSdkWrapper.Logout()

Release

15. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the Search sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing SsmSdkWrapper.

Get search authority

> **Note**
> For more information on how to get search authority, see the 'How to Implement' section of the Backup sample program described in CHAPTER 6 of this document (p.55)

Search calendar

Create [Search calendar] button click event handler and perform a 'SearchCalendar' request through the SearchCalendar() method of SsmSdkWrapper. Whether the recording exists or not on the date in the selected month can be obtained through OnResponse.

```
// SearchSample.cs
private void btnSearchCalendar_Click(object sender, EventArgs e)
{
	int year = monthCalendar.SelectionStart.Year;
	int month = monthCalendar.SelectionStart.Month;
	int trackID = -1;
	UInt32 sequenceID = 0;

	try
	{
		uint result = ssmSdkWrapper.SearchCalendar(
		_deviceUuid,
		_cameraUuid,
		year.ToString(),
		month.ToString(),
		ref sequenceID);
		_logger.WLOGD("SearchCalendar()::Result=" + result + ", SequenceID=" +
sequenceID);
	}
	catch (Exception ex)
	{
		_logger.WLOGD(ex.Message);
	}
}
```

Search recording track

Create [Search track id] button click event handler and perform a 'SearchTrackID' request through the SearchTrackID() method of SsmSdkWrapper. A track id information can be obtained through OnResponse.

```
// BackupSample.cs
private void btnSearchTrackID_Click(object sender, EventArgs e)
{
	var startDate = monthCalendar.SelectionStart.Date;
	var endDate = monthCalendar.SelectionEnd.Date.AddHours(23.59);
	string startTime = startDate.ToLocalTime().ToString(UtcFormat);
	string endTime = endDate.ToLocalTime().ToString(UtcFormat);
```

```
        UInt32 sequenceID = 0;

        try
        {
                uint result = ssmSdkWrapper.SearchTrackID(
                _deviceUuid,
                startTime,
                endTime,
                ref sequenceID);
                _logger.WLOGD("SearchCalendar()::Result=" + result + ", SequenceID=" +
sequenceID);
        }
        catch (Exception ex)
        {
                _logger.WLOGD(ex.Message);
        }
}
```

Search for recording section information

Create [Search day] button click event handler and perform a 'SearchDay' request through the SearchDay() method of SsmSdkWrapper. Recording section information can be obtained through OnResponse.

```
// SearchSample.cs
private void btnSearchDay_Click(object sender, EventArgs e)
{
        var startDate = monthCalendar.SelectionStart.Date;
        var endDate = monthCalendar.SelectionEnd.Date.AddHours(23.59);
        string startTime = startDate.ToLocalTime().ToString(UtcFormat);
        string endTime = endDate.ToLocalTime().ToString(UtcFormat);
        UInt32 sequenceID = 0;

        try
        {
                uint result = ssmSdkWrapper.SearchDay(
                                                _deviceUuid,
                                                _cameraUuid,
```

```
                                                                    trackId,
                                                                    startTime,
                                                                    false,        // DST
                                                                    endTime,
                                                                    false,        // DST
                                                                    (uint)RecordType.ALL,
                                                                    (uint)IVEventType.ALL,
                                                                    ref sequenceID);
                    _logger.WLOGD("SearchDay()::Result=" + result + ", SequenceID=" +
sequenceID);
        }
        catch (Exception ex)
        {
                    _logger.WLOGD(ex.Message);
        }
}
```

Release search authority

> **Note**
> For more information on how to release search authority, see the 'How to Implement' section of the Backup sample program described in CHAPTER 6 of this document (p.55).

See also

LogIn Sample Program

Backup Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 7

Backup Sample Program

This chapter describes how to backup recorded video using the Wisenet SSM Client SDK. The example uses AVI format.

Contents

Introduction

The Backup sample program is an example of receiving 1 channel recorded video from a network device and backing it up as a file. It only includes an AVI file creation example. The program only includes creating file with AVI format. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Login] button to see if you can successfully access the Wisenet SSM Core Server.

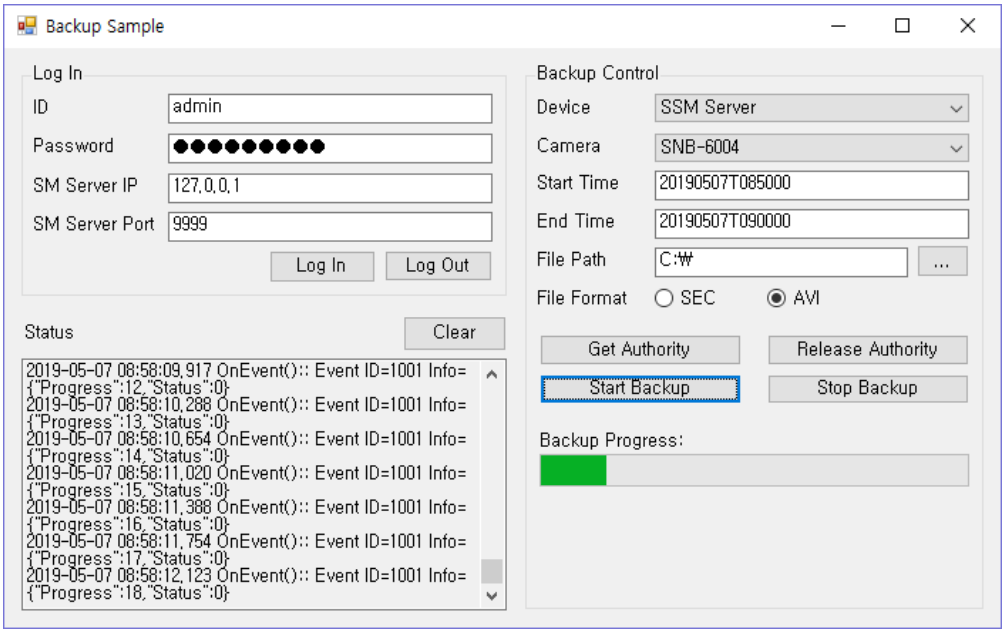


Figure 5 Backup Sample Program Execution

Step 3. Check the ResultCode of the OnResponse to see if login was successful.

Step 4. When login is executed successfully, the name of connected devices and cameras appear in the [Device] and [Camera] ComboBox.

Step 5. Select Device and Camera in ComboBox.

Step 6. Click the [Get Authority] button to get the search authority of the selected device

Step 7. Enter the backup start time in the [Start Time] text box. The format is "yyyyMMddThhmmss".

Step 8. Enter the backup end time in the [End Time] text box. The format is "yyyyMMddThhmmss"

Step 9. Click the [File Path] control to specify the backup file storage path.

Step 10. Click the [Start Backup] button to start the backup

Step 11. Through the [Status] text box, confirm the value which is passed to the BackupStatusChanged event.

Step 12. Check progress of backup through [Progress Bar].

Step 13. Confirm that the 'BackupStopped' event occurs through the [Status] text box

Step 14. Click the [Release Authority] button to release the search authority of the selected device

Step 15. Click on [Logout] to release a connection to the server.

Step 16. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()

Login

2. ssmSdkWrapper.Login()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

5. ssmSdkWrapper.OnEvent

GetSearchAuthority

6. ssmSdkWrapper.GetSearchAuthority

Response of GetSearchAuthority

7. ssmSdkWrapper.OnResponse

SearchTrackID

8. ssmSdkWrapper.SearchTrackID()

SearchDay

9. ssmSdkWrapper.SearchDay()

StartBackup

10. ssmSdkWrapper.StartBackup()

Response of StartBackup

11. ssmSdkWrapper.OnResponse

Receive Event: BackupStatusChanged

12. ssmSdkWrapper.OnEvent

Release Authority

13. ssmSdkWrapper.ReleaseSearchAuthority()

Logout

14. ssmSdkWrapper.Logout()

Release

15. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the Backup sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing SsmSdkWrapper

Add SsmSdkWrapper to the reference

> **Note**
> For more information on how to add SsmSdkWrapper to the reference, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

Add SsmSdkWrapper Event Handler

Implement Callback Function: OnResponse

> **Note**
> For more information on how to implement callback function: OnResponse, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

Implement Callback Function: OnEvent

Implement the Login and BackupStatusChanged cases on the OnEvent() of the BackupSample class. When the Login event occurs, it updates the combo box with the list of currently connected devices and cameras. When the BackupStatusChanged event occurs, it updates the backup progress on the ProgressBar.

```
// BackupSample.cs
private void OnEvent(UInt32 eventID, String info)
{
        _logger.WLOGD(
                "OnEvent()::" +
```

```
                    " Event ID=" + eventID +
                    " Info=" + info);

        if (eventID == (uint)EventID.Login)
        {
                    SetObjectList(info);
                    UpdateComboDeives();
                    UpdateComboCameras();
        }
        else if (eventID == (uint)EventID.BackupStatusChanged)
        {
                    UpdateBackupStatusProgressbar(info);
        }
}
```

Initialize SsmSdkWrapper

> **Note**
> For more information on how to initialize the SsmSdkWrapper, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

Login

> **Note**
> For more information on how to implement the Login, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

Get search authority

Create a [Get Authority] button click event handler and perform a request to send a 'Get SearchAuthority' command. Device Uuid and multi-password information are required as parameters of the method. If you don't use multi-passwords, pass an empty string("").

```
// BackupSample.cs
private void btnGetAuthority_Click(object sender, EventArgs e)
{
        try
        {
                    UInt32 sequenceID = 0;
```

```
                uint result = ssmSdkWrapper.GetSearchAuthority(_deviceUuid, "", ref
sequenceID);
                _logger.WLOGD("GetAuthority()::Result=" + result + ", SequenceID=" +
sequenceID);
        }
        catch (Exception ex)
        {
                _logger.WLOGD(ex.Message);
        }
}
```

Start backup

Request backup start through the [StartBackup] button click event.

```
// BackupSample.cs
private void btnStartBackup_Click(object sender, EventArgs e)
{
        uint result = 0;
        int trackID = -1;
        uint fileSizeLimit = 0;
        uint timePeriodLimit = 0;
        uint sequenceID = 0;
        try
        {
                result = ssmSdkWrapper.StartBackup(
                _cameraUuid,
                trackID,
                txtStartTime.Text,
                false,
                txtEndTime.Text,
                false,
                (uint)_fileFormat,
                txtFilePath.Text,
                fileSizeLimit,
                timePeriodLimit,
                ref _mediaID,
                ref sequenceID);
        }
```

```
        catch (Exception ex)
        {
                _logger.WLOGD(ex.Message);
        }
        _logger.WLOGD("StartBackup()::Result=" + result + ", SequenceID=" + sequenceID);
}
```

Stop backup

If you want to abort the operation during the backup, send a 'StopBackup' command via the StopBackup() method.

```
// BackupSample.cs
private void btnStopBackup_Click(object sender, EventArgs e)
{
        uint result = 0;
        uint sequenceID = 0;
        try
        {
                result = ssmSdkWrapper.StopBackup(_cameraUuid, _mediaID, ref sequenceID);
        }
        catch (Exception ex)
        {
                _logger.WLOGD(ex.Message);
        }
        _logger.WLOGD("StartBackup()::Result=" + result + ", SequenceID=" + sequenceID);
}
```

Release search authority

Create a [Release Authority] button click event handler to perform 'Release SearchAuthority' command. Pass the device uuid to release the search authority as a parameter of the method.

```
// BackupSample.cs
private void btnReleaseAuthority_Click(object sender, EventArgs e)
{
        try
```

```
        {
                UInt32 sequenceID = 0;
                uint result = ssmSdkWrapper.ReleaseSearchAuthority(_deviceUuid, ref
sequenceID);
                _logger.WLOGD("ReleaseAuthority()::Result=" + result + ", SequenceID=" +
sequenceID);
        }
        catch (Exception ex)
        {
                _logger.WLOGD(ex.Message);
        }
}
```

Logout

> **Note**
> For more information on how to implement the logout, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

Release SsmSdkWrapper

> **Note**
> For more information on how to release the SsmSdkWrapper, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 4 of this document (p.32).

See also

LogIn Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

What value should be filled in TrackID in StartBackup?

→ In the BackupSample, -1 is set, but the TrackID obtained in the timeline data received in response to SearchTrackID and SearchDay should be set.

CHAPTER 8

SingleLive Sample Program

This chapter describes how to display one channel live video on the window using Wisenet SSM Client SDK.

Contents

Introduction

The SingleLive sample program is an example that receives one channel live video from the network device and displays it to the screen. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

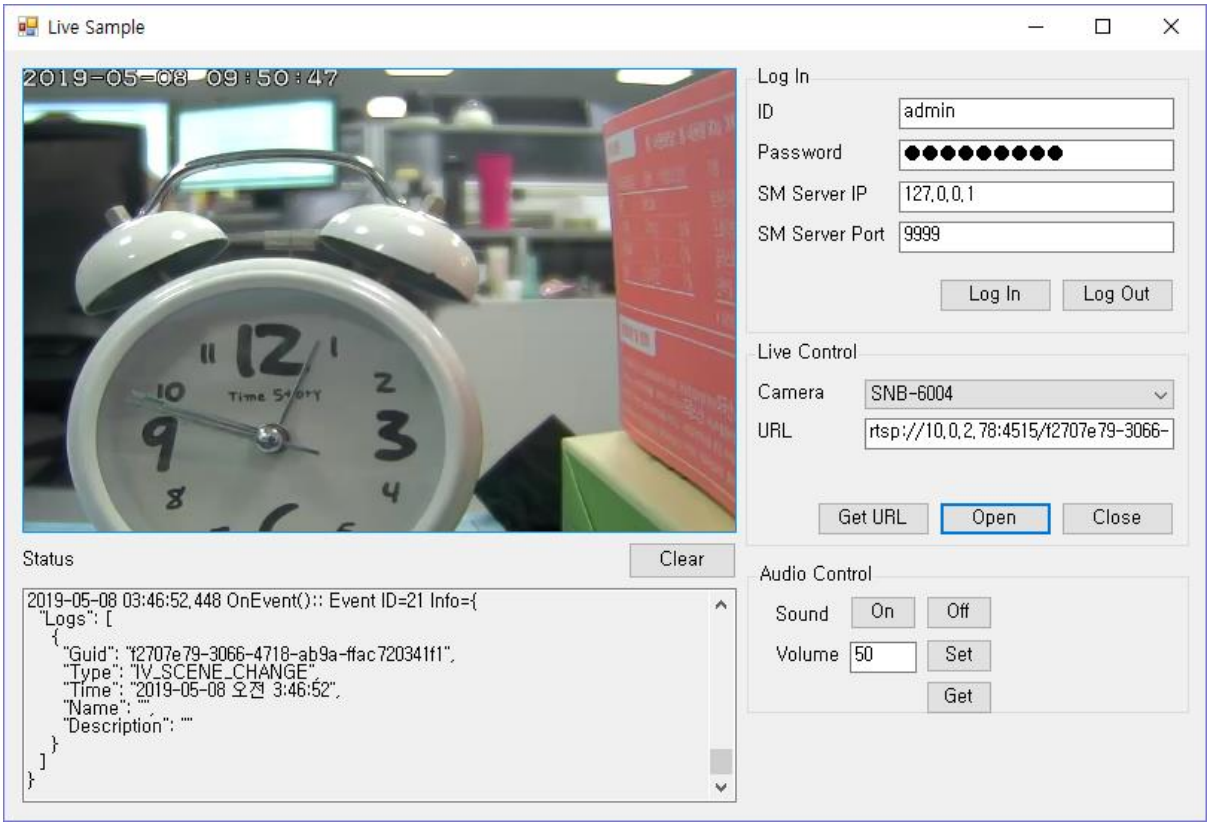


Figure 6 SingleLive Sample Program Execution

Step 3. When login is executed successfully, the name of connected cameras appear in the [Camera] ComboBox.

Step 4. Select the camera in the combo box.

Step 5. Click the [Get Rtsp Url] button to get the URL of the selected camera.

Step 6. Receive a media stream from the selected camera with the URL obtained by clicking the [Open] button.

Step 7. Check that the live video is being displayed on the left window.

Step 8. Click the [Close] button to stop receiving the media stream and release the resource.

Step 9. Click on [Logout] to release a connection to the server.

Step 10. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()
2. ssmSdkWrapper.InitailizeMedia()

Login

3. ssmSdkWrapper.Login()

Response of Login

4. ssmSdkWrapper.OnResponse

Receive Event: LogIn

5. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

6. ssmSdkWrapper.OnEvent

GetRtspUrl

7. ssmSdkWrapper.GetRtspUrl()

Response of GetRtspUrl

8. ssmSdkWrapper.OnResponse

Open Media

9. ssmSdkWrapper.MediaOpen()

Close Media

10. ssmSdkWrapper.MediaClose()

Logout

11. SsmSdkWrapper.Logout()

Release

12. ssmSdkWrapper.ReleaseEvent()
13. ssmSdkWrapper.ReleaseMedia()

How to Implement

This section describes how to implement the SingleLive sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing SsmSdkWrapper.

Initialize SsmSdkWrapper

In the LiveSample class member function LiveSample_Load(), create a SsmSdkWrapper and initialize it by calling ssmSdkWrapper.InitializeEvent(), ssmSdkWrapper.InitializeMedia(). To output the image to the screen, set the first parameter of InitailizeMedia to the window handle and the second parameter to 0. If set to 1 or 2, the OnVideo, OnAudio callback function is called when the image is opened. (See Chapter 4 of the API documentation). On the other hand, the Media ID is a value that is used to identify media, so be sure to save it.

```
// SingleLiveSample.cs
private SsmSdkWrapper ssmSdkWrapper = null;

private void LiveSample_Load(object sender, EventArgs e)
{
        ssmSdkWrapper = new SsmSdkWrapper(this.OnResponse, this.OnEvent, this.OnVideo,
this.OnAudio);
        ssmSdkWrapper.InitializeEvent();

        ssmSdkWrapper.InitializeMedia(pictureBox1.Handle, 0, ref this.mediaId);
        ssmSdkWrapper.MoveWindow((UInt32)this.mediaId, 0, 0, pictureBox1.Width,
pictureBox1.Height);
}
```

Get RTSP URL

Create a button click event handler and execute the GetRtspUrl command. Get the camera's url by using GetRtspUrl() method of SsmSdkWrapper. In this sample, the profile value is HIGH (= 1), the stream type is LIVE (= 2), and the protocol is TCP (= 1). The obtained URL address will be used in the Open() method of the SsmMediaSdk control.

```
// SingleLiveSample.cs
private void btnGetRtspUrl_Click(object sender, EventArgs e)
{
        UInt32 sequenceID = 0;
        uint resCode = ssmSdkWrapper.GetRtspUrl(_cameraUuid, 2, 2, 1, ref sequenceID);   //
MEDIUM, LIVE, TCP
        _logger.WLOGD("GetRtspUrl()::Result=" + resCode + ", SequenceID=" + sequenceID);
}
```

Open Media

Create a [Media Open] button click event handler and invoke the SsmSdkWrapper MediaOpen() method to start receiving media streams.

If you chose 'Display' mode when initializing the InitailizeMedia(), rendered video will be displayed on the window. If you chose 'Relay' mode, received media streams are passed to the OnVideo and OnAudio event. If you choose 'Live' as MediaType, StartTime, TimeZone, and DST are not used. Therefore, you can pass "0", "", and false respectively as parameters.

```
// SingleLiveSample.cs

private void btnOpen_Click(object sender, EventArgs e)
{
    ssmSdkWrapper.MediaOpen(
        txtRtspUrl.Text,
        txtID.Text,
        txtPW.Text,
        _cameraName,
        (uint)MediaType.Live,
        "0",        // start-time
        "",         // Time zone
        false);     // DST
}
```

Close Media

Create a [Media Close] button click event handler and invoke the SsmSdkWrapper MediaClose() method to stop receiving media streams.

```
// SingleLiveSample.cs

private void btnClose_Click(object sender, EventArgs e)
{
    ssmSdkWrapper.MediaClose();
}
```

Release SsmSdkWrapper

After using SsmSdkWrapper, you must stop the service and release the resources.

Call the ReleaseEvent(), ReleaseMedia() methods within the SingleLiveSample_Closing() of the SingleLiveSample class

```
// SingleLiveSample.cs
private void LiveSample_Closing(object sender, FormClosingEventArgs e)
{
        ssmSdkWrapper.ReleaseEvent();

        ssmSdkWrapper.MediaClose(this.mediaId);
        ssmSdkWrapper.ReleaseMedia(this.mediaId);
}
```

See also

LogIn Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 9

PTZ Sample Program

This chapter describes how to use pan/tilt/zoom/preset using Wisenet SSM Client SDK.

Contents

Introduction

PTZ sample program is an example of using Pan, Tilt, Zoom, Preset function of network camera that supports PTZ function. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

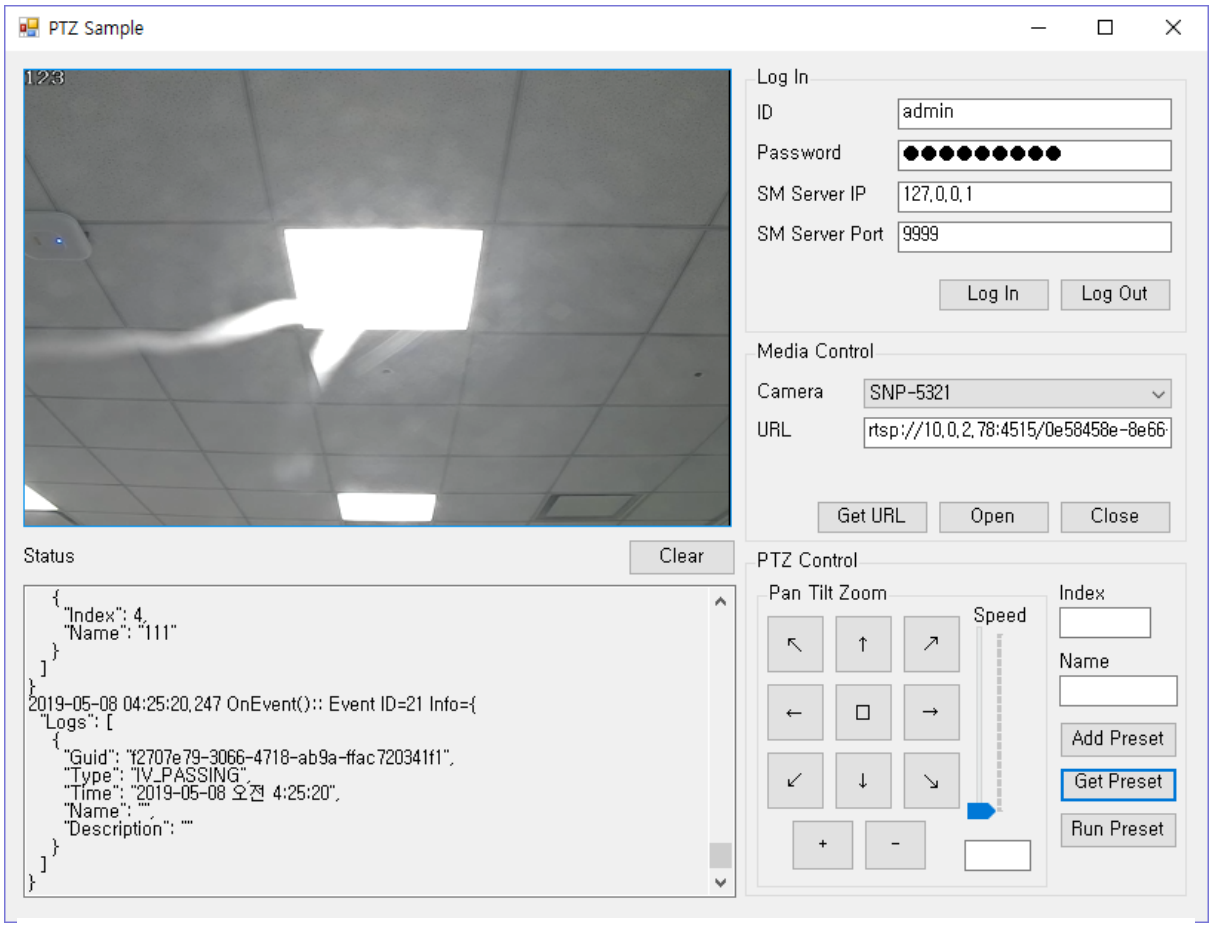


Figure 7 PTZ Sample Program Execution

Step 3. When login is executed successfully, the name of connected cameras appear in the [Camera] ComboBox.

Step 4. Select the camera in the combo box.

Step 5. Click the [Get Rtsp Url] button to get the URL of the selected camera.

Step 6. Receive a media stream from the selected camera with the URL obtained by clicking the [Open] button.

Step 7. Check that the live video is being displayed on the left window.

Step 8. Click the Pan/Tilt/Zoom button to see if the corresponding function works.

Step 9. Click the ▫ button to stop the PTZ command

Step 10. Adjust the speed bar to change PTZ operation speed

Step 11. Click the [Add Preset list] button to add a preset.

Step 12. Click the [Get Preset list] button to get the preset list

Step 13. Enter the preset number and index name and click the [Run Preset] button to load the preset

Step 14. Click the [Close] button to stop receiving the media stream and release the resource.

Step 15. Click on [Logout] to release a connection to the server.

Step 16. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()
2. ssmSdkWrapper.InitailizeMedia()

Login

3. ssmSdkWrapper.Login()

Response of Login

4. ssmSdkWrapper.OnResponse

Receive Event: LogIn

5. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

6. ssmSdkWrapper.OnEvent

GetRtspUrl

7. ssmSdkWrapper.GetRtspUrl()

Response of GetRtspUrl

8. ssmSdkWrapper.OnResponse

Open Media

9. ssmSdkWrapper.MediaOpen()

ControlPtz

10. ssmSdkWrapper.ControlPtz()

AddPreset

11. ssmSdkWrapper.AddPreset()

Response of AddPreset

12. ssmSdkWrapper.OnResponse

GetPresetList

13. ssmSdkWrapper.GetPresetList()

Response of GetPresetList

14. ssmSdkWrapper.OnResponse

RunPreset

15. ssmSdkWrapper.RunPreset()

Reponse of RunPreset

16. ssmSdkWrapper.OnResponse

Close Media

17. ssmSdkWrapper.MediaClose()

Logout

18. SsmSdkWrapper.Logout()

Release

19. ssmSdkWrapper.ReleaseEvent()
20. ssmSdkWrapper.ReleaseMedia()

How to Implement

This section describes how to implement the PTZ sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing SsmSdkWrapper.

Initialize SsmSdkWrapper

> **Note**
> For more information on how to initialize the SsmSdkWrapper, see the 'How to Implement' section of the Live sample program described in CHAPTER 8 of this document (p. 64).

Get RTSP URL

> **Note**
> For more information on how to implement getting rtsp url, see the 'How to Implement' section of the SingleLive sample program described in CHAPTER 8 of this document (p. 64).

Open Media

> **Note**
> For more information on how to implement opening media, see the 'How to Implement' section of the SingleLive sample program described in CHAPTER 8 of this document (p. 64).

Control PTZ

Create a button click event handler and perform the Pan / Tilt / Zoom function with using the ControlPtz() method of SsmSdkWrapper. Set the PTZ detailed commands (Up, Left, Zoom In, Stop, etc.) as the nAction value. Set the control speed to the nSpeed value. After sending a PTZ command, you must send a STOP command to stop the previous PTZ command.

```
// PTZSample.cs


private void btnPTZLeft_Click(object sender, EventArgs e)
```

```
{
    ssmSdkWrapper.ControlPtz(
        _cameraUuid,
        (uint)PtzAction.Left,
        (ushort)trackBarPTZSpeed.Value);
}
```

Add Preset

Create a button click event handler and add the preset to the camera with using the AddPreset() method of SsmSdkWrapper. You must set the preset number and name together.

```
// PTZSample.cs
private void btnAddPreset_Click(object sender, EventArgs e)
{
        try
        {
                UInt32 sequenceID = 0;
                uint resCode = ssmSdkWrapper.AddPreset(
                        _cameraUuid,
                        Convert.ToUInt32(txtPresetIndex.Text),
                        txtPresetName.Text,
                        ref sequenceID);
                _logger.WLOGD("AddPreset()::Result=" + resCode + ", SequenceID=" +
sequenceID);
        }
        catch (Exception ex)
        {
                _logger.WLOGD(ex.Message);
        }
}
```

Get Preset List

Create a button click event handler and get a list of all the presets which have been set on the camera with using the GetPresetList() method of SsmSdkWrapper. The preset list will be received on the OnResponse event of SsmSdkWrapper.

```
// PTZSample.cs
```

```
private void btnGetPresetList_Click(object sender, EventArgs e)
{
	try
	{
		UInt32 sequenceID = 0;
		uint resCode = ssmSdkWrapper.GetPresetList(_cameraUuid, ref sequenceID);
		_logger.WLOGD("GetPresetList()::Result=" + resCode + ", SequenceID=" +
sequenceID);
	}
	catch (Exception ex)
	{
		_logger.WLOGD(ex.Message);
	}
}
```

Run Preset

Create a button click event handler and run the selected preset in the camera with using RunPreset() method of SsmSdkWrapper.

```
// PTZSample.cs
private void btnRunPreset_Click(object sender, EventArgs e)
{
	try
	{
		UInt32 sequenceID = 0;
		uint resCode = ssmSdkWrapper.RunPreset(_cameraUuid,
Convert.ToUInt32(txtPresetIndex.Text), ref sequenceID);
		_logger.WLOGD("RunPreset()::Result=" + resCode + ", SequenceID=" +
sequenceID);
	}
	catch (Exception ex)
	{
		_logger.WLOGD(ex.Message);
	}
}
```

Close Media

> **Note**
> For more information on how to implement closing media, see the 'How to Implement' section of the SingleLive sample program described in CHAPTER 8 of this document (p. 64).

Release SsmSdkWrapper

> **Note**
> For more information on how to release the SsmSdkWrapper, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 8 of this document (p. 64).

See also

LogIn Sample Program

SingleLive Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 10

MultiMonitoring Sample Program

This chapter describes how to display four channel live video to the screen using Wisenet SSM Client SDK

Contents

Introduction

MultiMonitoring sample program is an example of receiving four channel live video from network device and displaying on the window. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

Step 3. When login is executed successfully, the name of connected cameras appear in the [Camera] ComboBox.

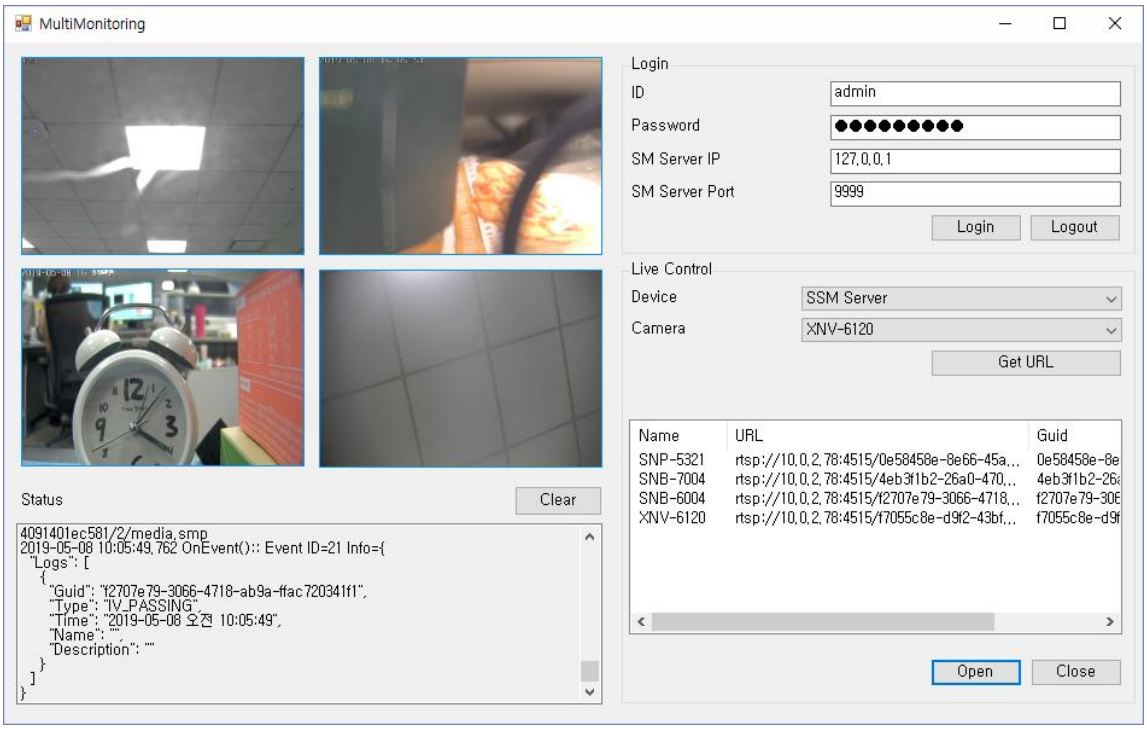


Figure 8 MultiMonitoring Sample Program Execution

Step 4. Select the camera in the combo box

Step 5. Click the [Get Rtsp Url] button to get the URL of the selected camera. URL list will be shown up at the ListView.

Step 6. Select the camera name in the ListView and click the [Open] button to start receiving the media stream.

Step 7. Check that the live video is being displayed on the left window. It will be displayed in the order you selected.

Step 8. Click the [Close] button to stop receiving the media stream and release the resource.

Step 9. Click on [Logout] to release a connection to the server.

Step 10. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()
2. ssmSdkWrapper.InitailizeMedia()

Login

3. ssmSdkWrapper.Login()

Response of Login

4. ssmSdkWrapper.OnResponse

Receive Event: LogIn

5. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

6. ssmSdkWrapper.OnEvent

GetRtspUrl

7. ssmSdkWrapper.GetRtspUrl()

Response of GetRtspUrl

8. ssmSdkWrapper.OnResponse

Open Media

9. ssmSdkWrapper.MediaOpen()

Close Media

10. ssmSdkWrapper.MediaClose()

Logout

11. SsmSdkWrapper.Logout()

Release

12. ssmSdkWrapper.ReleaseEvent()
13. ssmSdkWrapper.ReleaseMedia()

How to Implement

This section describes how to implement the MultiMonitoring sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing SsmSdkWrapper.

Initialize SsmSdkWrapper

> **Note**
> For more information on how to initialize the SsmSdkWrapper, see the 'How to Implement' section of the SingleLive sample program described in CHAPTER 8 of this document (p. 64).

Get RTSP URL

> **Note**
> For more information on how to implement getting RTSP URL, see the 'How to Implement' section of the SingleLive sample program described in CHAPTER 8 of this document (p. 64).

Open Media

> **Note**
> For more information on how to implement opening media, see the 'How to Implement' section of the SingleLive sample program described in CHAPTER 8 of this document (p. 64).

Close Media

> **Note**
> For more information on how to implement closing media, see the 'How to Implement' section of the SingleLive sample program described in CHAPTER 8 of this document (p. 64).

Release SsmSdkWrapper

> **Note**
> For more information on how to release the SsmSdkWrapper, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 8 of this document (p. 64).

See also

LogIn Sample Program

SingleLive Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 11

Playback Sample Program

This chapter describes how to display recorded videos to the screen using Wisenet SSM Client SDK. The example uses an H.264 profile.

Contents

Introduction

The Playback sample program is an example that receives recorded video from a network device and displays it on the screen. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

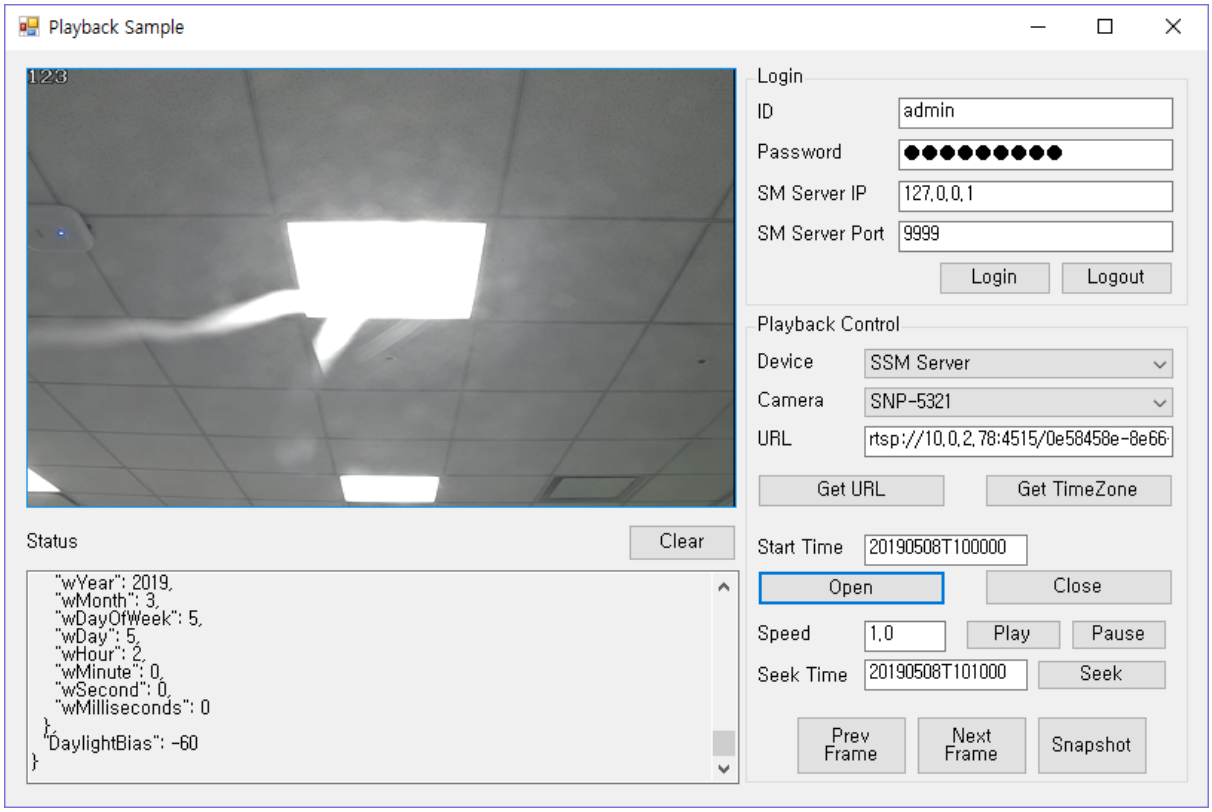


Figure 9 Playback Sample Program Execution

Step 3. When login is executed successfully, the name of connected devices and cameras appear in the [Device] and [Camera] ComboBox.

Step 4. Select device and camera in ComboBox.

Step 5. Click the [Get URL] button to get the URL of the selected camera.

Step 6. Click the [Get TimeZone] button to get the time zone information of the device.

Step 7. Enter the backup start time in the [Start Time] text box. The format is

"yyyyMMddThhmmss".

Step 8. Receive a media stream from the selected camera with the URL obtained by clicking the [Open] button.

Step 9. Check that the playback video is being displayed on the left window..

Step 10. Click the [Close] button to stop receiving the media stream and release the resource.

Step 11. Click on [Logout] to release a connection to the server.

Step 12. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()
2. ssmSdkWrapper.InitailizeMedia()

Login

3. ssmSdkWrapper.Login()

Response of Login

4. ssmSdkWrapper.OnResponse

Receive Event: LogIn

5. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

6. ssmSdkWrapper.OnEvent

GetRtspUrl

7. ssmSdkWrapper.GetRtspUrl()

Response of GetRtspUrl

8. ssmSdkWrapper.OnResponse

Open Media

9. ssmSdkWrapper.MediaOpen()

Close Media

10. ssmSdkWrapper.MediaClose()

Logout

11. SsmSdkWrapper.Logout()

Release

12. ssmSdkWrapper.ReleaseEvent()
13. ssmSdkWrapper.ReleaseMedia()

How to Implement

This section describes how to implement the MultiMonitoring sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing SsmSdkWrapper.

Initialize SsmSdkWrapper

> **Note**
> For more information on how to initialize the SsmSdkWrapper, see the 'How to Implement' section of the SingleLive sample program described in CHAPTER 8 of this document (p. 64).

Get RTSP URL

> **Note**
> For more information on how to implement getting RTSP URL, see the 'How to Implement' section of the SingleLive sample program described in CHAPTER 8 of this document (p. 64).

Get TimeZoneInfo

Create button click event handler and get the time zone information of selected device with using GetTimeZoneInfo() method of SsmSdkWrapper.

```
// PlaybackSample.cs

private void btnGetTimeZone_Click(object sender, EventArgs e)
{
    _timeZoneInfo = ssmSdkWrapper.GetTimeZoneInfo(_deviceUuid);
    _logger.WLOGD("GetTimeZoneInfo()::Result=" + _timeZoneInfo);
}
```

Open Media

> **Note**
> For more information on how to implement opening media, see the 'How to Implement' section of the SingleLive sample program described in CHAPTER 7 of this document (p. 64).

Close Media

> **Note**
> For more information on how to implement closing media, see the 'How to Implement' section of the SingleLive sample program described in CHAPTER 7 of this document (p. 64).

Release SsmSdkWrapper

> **Note**
> For more information on how to release the SsmSdkWrapper, see the 'How to Implement' section of the LogIn sample program described in CHAPTER 8 of this document (p. 64).

See also

LogIn Sample Program

SingleLive Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 12

InsertLog Sample Program

This chapter describes how to send event information to the Wisenet SSM Core Server server using the Wisenet SSM Client SDK.

Contents

Introduction

The InsertLog sample program sends event information to the Wisenet SSM Core Server server. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

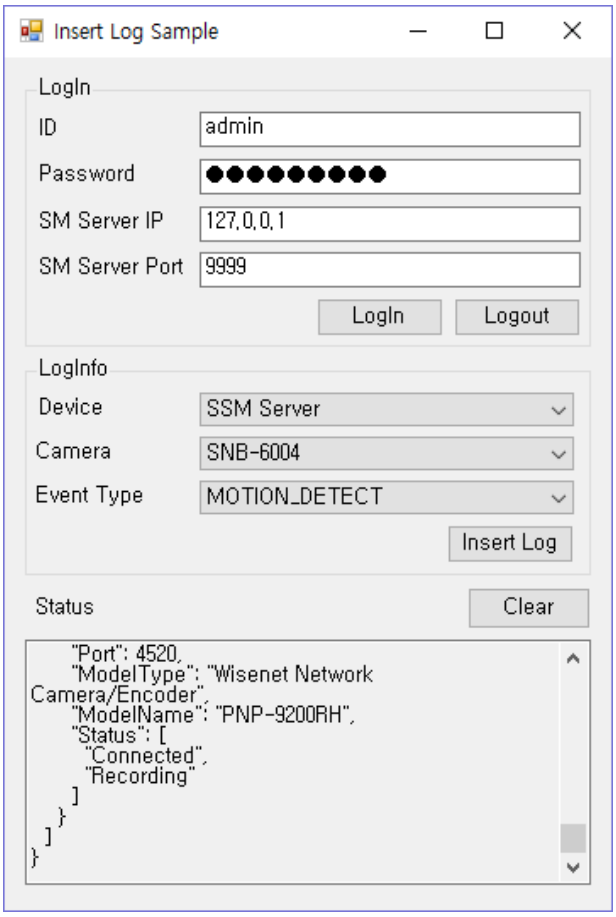


Figure 10 InsertLog Sample Program Execution

Step 3. When login is executed successfully, the name of connected devices and cameras appear in the [Device] and [Camera] ComboBox.

Step 4. Select device and camera in ComboBox.

Step 5. Select the event type.

Step 6. Click the [Insert Log] button to send the event.

Step 7. Check if the event you sent is received.

Step 8. Click on [Logout] to release a connection to the server.

Step 9. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()

Login

2. ssmSdkWrapper.Login()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

5. ssmSdkWrapper.OnEvent
6. ssmSdkWrapper.InsertExternalLog()

Response of InsertExternalLog

7. ssmSdkWrapper.OnResponse

Recevice Event: Log

8. ssmSdkWrapper.OnEvent

Logout

9. ssmSdkWrapper.Logout()

Release

10. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the Search sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing SsmSdkWrapper.

Send the Event Log

Create a button click event handler, and send the event log through the InsertExternalLog () method of the SsmSdkWrapper.

```
// InsertLogSample.cs
private void btnInsertLog_Click(object sender, EventArgs e)
{
	uint resCode = 0;
	UInt32 sequenceID = 0;
	try
	{
		resCode = ssmSdkWrapper.InsertExternalLog(
		_deviceUuid,
		_cameraUuid,
		_logTypeDictionary[_logType],
		ref sequenceID);
	}
	catch (Exception ex)
	{
		_logger.WLOGD(ex.Message);
	}
	_logger.WLOGD("InsertLog()::Result=" + resCode + ", SequenceID=" + sequenceID);
}
```

Send the Event Log with Event Key

Create a button click event handler, and send the event log through the InsertExternalLogEventKey () method of the SsmSdkWrapper.

```
// InsertLogSample.cs
private void btnInsertLogWithEventKey_Click(object sender, EventArgs e)
{
    uint resCode = 0;
    UInt32 sequenceID = 0;

    UInt32 eventKey = 0;

    if (!UInt32.TryParse(this.txtEventKey.Text, out eventKey))
    {
        _logger.WLOGD("[ERROR] Check Event Key");
        return;
    }

    try
    {
        resCode = ssmSdkWrapper.InsertExternalLogEventKey(
        _deviceUuid,
        _cameraUuid,
        eventKey,
        ref sequenceID);
    }
    catch (Exception ex)
    {
        _logger.WLOGD(ex.Message);
    }
    _logger.WLOGD("InsertLog()::Result=" + resCode + ", SequenceID=" + sequenceID);
}
```

See also

LogIn Sample Program

Backup Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 13

LiveSnapshot Sample Program

This chapter describes how to take snapshot of live video using the Wisenet SSM Client SDK.

Contents

Introduction

The LiveSnapshot sample program captures live video and saves it as a file. It can be used as follows

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

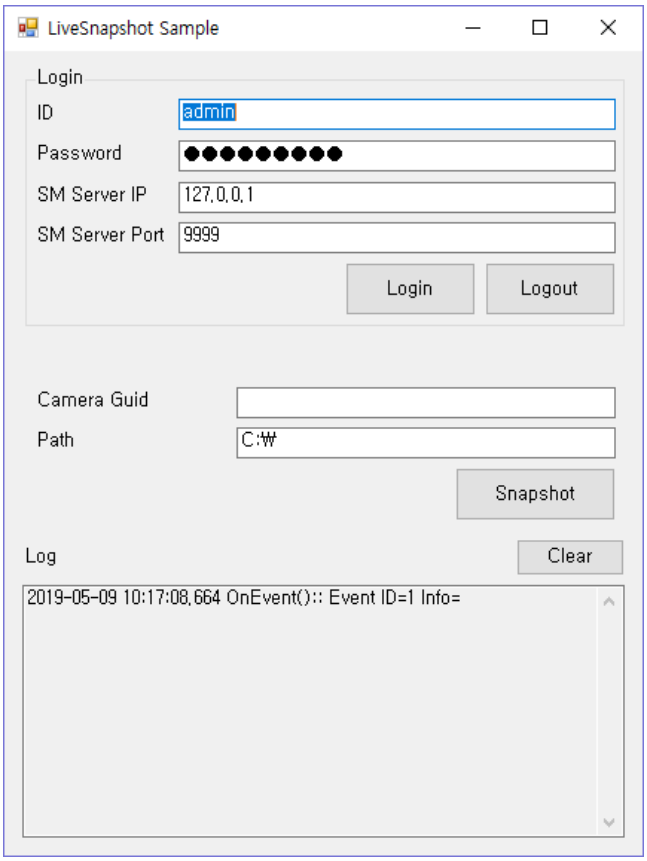


Figure 11 LiveSnapshot Sample Program Execution

Step 3. Enter the camera's Guid and Path.

Step 4. Click the [Snapshot] button.

Step 5. Click on [Logout] to release a connection to the server.

Step 6. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()

Login

2. ssmSdkWrapper.Login()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

5. ssmSdkWrapper.OnEvent

Get snapshot

6. ssmSdkWrapper.GetSnapshot

Logout

7. ssmSdkWrapper.Logout()

Release

8. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the LiveSnapshot sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing SsmSdkWrapper.

Live snapshot request

Create a button click event handler, and send the live snapshot request through the GetSnapshot () method of the SsmSdkWrapper.

```
// LiveSnapshotSample.cs
private void Snapshot_Click(object sender, EventArgs e)
{
	String guidText = textBoxUnitID.Text;
	Guid guid = Guid.Empty;
	UInt32 sequenceID = 0;
	UInt32 resCode = 0;

	if (Guid.TryParse(guidText, out guid))
	{
		SnapshotModel snapshotRequest = new SnapshotModel();

		snapshotRequest.CameraGuid = textBoxUnitID.Text;
		snapshotRequest.ProfileType = 1; // HIGH
		snapshotRequest.FilePath = textBoxFilePath.Text;

		String json = JsonConvert.SerializeObject(snapshotRequest,
Formatting.Indented);
		resCode = ssmSdkWrapper.GetSnapshot(json, ref sequenceID);
		_logger.WLOGD("GetSnapshot()::Result=" + resCode + ", SequenceID=" +
sequenceID);
	}
	else
	{
		MessageBox.Show("Check the guid");
	}
}
```

See also

LogIn Sample Program

Backup Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 14

LiveLocalRecord Sample Program

This chapter describes how to save live videos to .avi file using the Wisenet SSM Client SDK.

Contents

Introduction

The LiveLocalRecord sample program is an example that save live video to .avi file. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

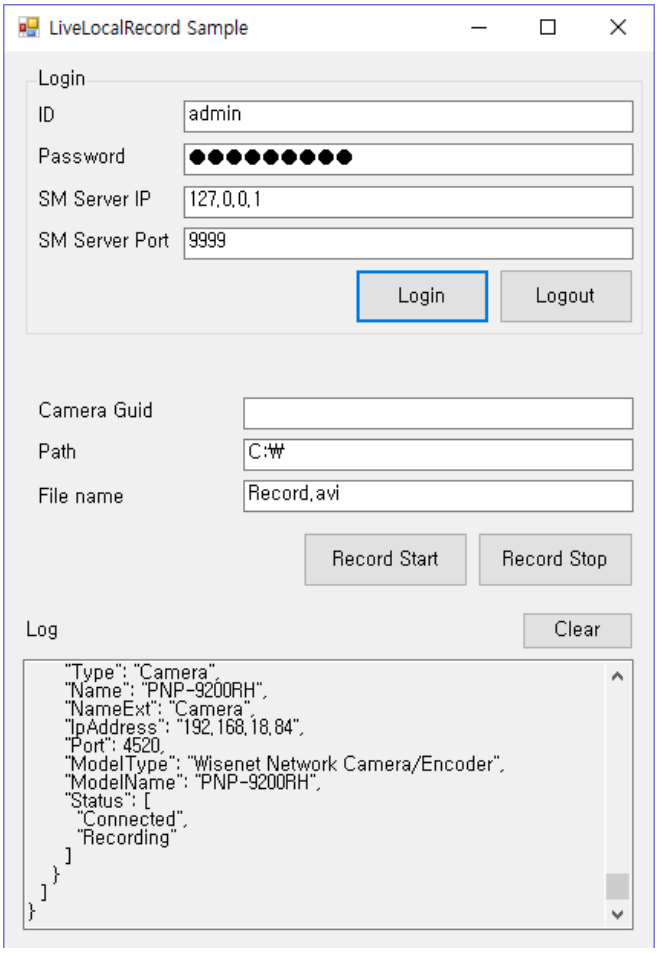


Figure 12 LiveLocalRecord Sample Program Execution

Step 3. Enter the camera's guid, path and file name.

Step 4. Click the [Record start] button.

Step 5. Click the [Record stop] button.

Step 6. Click on [Logout] to release a connection to the server.

Step 7. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()

Login

2. ssmSdkWrapper.Login()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

5. ssmSdkWrapper.OnEvent

Start Live local record

6. ssmSdkWrapper.StartLiveLocalRecording

Stop Live local record

7. ssmSdkWrapper.StopLiveLocalRecording

Logout

8. ssmSdkWrapper.Logout()

Release

9. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the LiveSnapshot sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing SsmSdkWrapper.

Start Local Record

Create button click event handler and send the local record start request using StartLiveLocalRecording () method of SsmSdkWrapper.

```
// LiveLocalRecordSample.cs
private void btnRecordStart_Click(object sender, EventArgs e)
{
        String guidText = textBoxUnitID.Text;
        Guid guid = Guid.Empty;
        uint resCode = 0;
        UInt32 sequenceID = 0;

        if (Guid.TryParse(guidText, out guid))
        {
                LocalRecordModel localRecordModel = new LocalRecordModel();

                localRecordModel.CameraGuid = textBoxUnitID.Text;
                localRecordModel.ProfileType = 1; // HIGH
                localRecordModel.FilePath = textBoxFilePath.Text;
                localRecordModel.FileName = textBoxFileName.Text;

                String json = JsonConvert.SerializeObject(localRecordModel,
Formatting.Indented);

                resCode = ssmSdkWrapper.StartLiveLocalRecording(json, ref sequenceID);
                _logger.WLOGD("StartLiveLocalRecording()::Result=" + resCode + ",
SequenceID=" + sequenceID);
        }
```

```
        else
        {
                MessageBox.Show("Check the guid");
        }
}
```

Stop Local Record

Create button click event handler and send the local record stop request using StartLiveLocalRecording () method of SsmSdkWrapper.

```
// LiveLocalRecordSample.cs
private void btnRecordStop_Click(object sender, EventArgs e)
{
        String guidText = textBoxUnitID.Text;
        Guid guid = Guid.Empty;
        uint resCode = 0;
        UInt32 sequenceID = 0;

        if (Guid.TryParse(guidText, out guid))
        {
                LocalRecordModel localRecordModel = new LocalRecordModel();

                localRecordModel.CameraGuid = textBoxUnitID.Text;
                localRecordModel.ProfileType = 1; // HIGH

                String json = JsonConvert.SerializeObject(localRecordModel,
Formatting.Indented);

                resCode = ssmSdkWrapper.StopLiveLocalRecording(json, ref sequenceID);
                _logger.WLOGD("StopLiveLocalRecording()::Result=" + resCode + ",
SequenceID=" + sequenceID);
        }
        else
        {
                MessageBox.Show("Check the guid");
        }
}
```

See also

LogIn Sample Program

Backup Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 15

UserManagement Sample Program

This chapter describes how to manage user groups and users using the Wisenet SSM Client SDK.

Contents

Introduction

The User management sample program is an example that add, modify, delete user groups and users. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

Step 3. Click the button related to User Group, User.

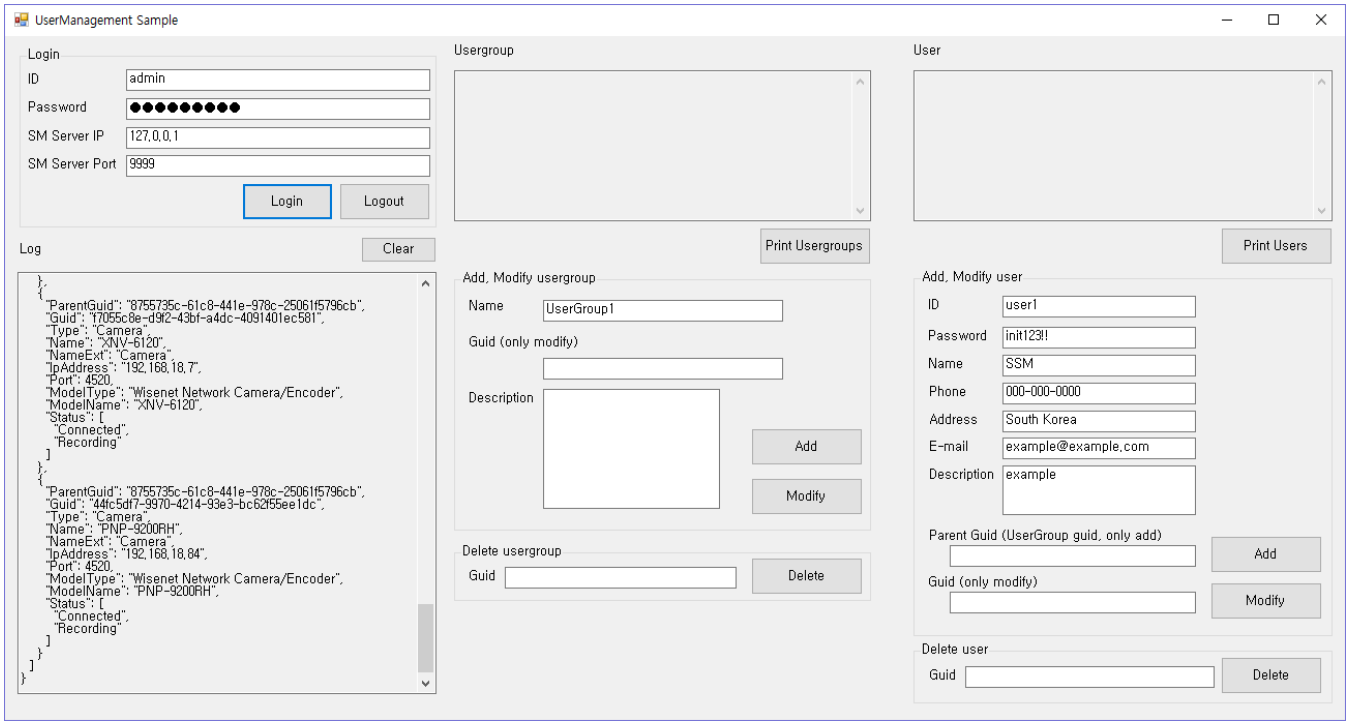


Figure 13 UserManagement Sample Program Execution

Step 4. Click on [Logout] to release a connection to the server.

Step 5. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()

Login

2. ssmSdkWrapper.Login()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

5. ssmSdkWrapper.OnEvent

User group, User get, add, modify, delete

6. ssmSdkWrapper.GetUserGroupInfo
   ssmSdkWrapper.AddUserGroup
   ssmSdkWrapper.ModifyUserGroup
   ssmSdkWrapper.DeleteUserGroup
   ssmSdkWrapper.GetUserInfo
   ssmSdkWrapper.AddUser
   ssmSdkWrapper.ModifyUser
   ssmSdkWrapper.DeleteUser

Logout

7. ssmSdkWrapper.Logout()

Release

8. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the UserManagement sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding

SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing SsmSdkWrapper.

Getting Information of User Groups

Create button click event handler and get the information of user groups using GetUserGroupInfo () method of SsmSdkWrapper.

```
// UserManagementSample.cs
private void PrintUserGroup_Click(object sender, EventArgs e)
{
        String str = ssmSdkWrapper.GetUserGroupInfo();
        this.textBoxUserGroupList.Text = str;
}
```

Add a User Group

Create button click event handler and add a user group using AddUserGroup () method of SsmSdkWrapper.

```
// UserManagementSample.cs
private void AddUsergroupButton_Click(object sender, EventArgs e)
{
        UserGroupModel userGroupModel = new UserGroupModel();

        userGroupModel.Name = textBoxAddUsergroupName.Text;
        userGroupModel.Permission.MonitoringViewer = true;
        userGroupModel.Permission.Live = true;

        String json = JsonConvert.SerializeObject(userGroupModel, Formatting.Indented);
        UInt32 resCode = 0;
        UInt32 sequenceID = 0;

        resCode = ssmSdkWrapper.AddUserGroup(json, ref sequenceID);
        _logger.WLOGD("AddUserGroup()::Result=" + resCode + ", SequenceID=" + sequenceID);
}
```

Modify a User Group

Create button click event handler and modify a user group using ModifyUserGroup () method of SsmSdkWrapper.

```
// UserManagementSample.cs
private void ModifyUsergroupButton_Click(object sender, EventArgs e)
{
        UserGroupModel userGroupModel = new UserGroupModel();

        userGroupModel.Guid = textBoxModifyUsergroupGuid.Text;
        userGroupModel.Name = textBoxAddUsergroupName.Text;
        userGroupModel.Description = textBoxModifyUsergroupDescription.Text;

        userGroupModel.Permission.MonitoringViewer = true;
        userGroupModel.Permission.Live = true;

        String json = JsonConvert.SerializeObject(userGroupModel, Formatting.Indented);
        UInt32 resCode = 0;
        UInt32 sequenceID = 0;

        resCode = ssmSdkWrapper.ModifyUserGroup(json, ref sequenceID);
        _logger.WLOGD("ModifyUserGroup()::Result=" + resCode + ", SequenceID=" +
sequenceID);
}
```

Delete a User Group

Create button click event handler and delete a user group using DeleteUserGroup () method of SsmSdkWrapper.

```
// UserManagementSample.cs
private void DeleteUsergroupButton_Click(object sender, EventArgs e)
{
        UserGroupModel userGroupModel = new UserGroupModel();

        userGroupModel.Guid = textBoxDeleteUsergroupGuid.Text;
        String json = JsonConvert.SerializeObject(userGroupModel, Formatting.Indented);
        UInt32 resCode = 0;
        UInt32 sequenceID = 0;

        resCode = ssmSdkWrapper.DeleteUserGroup(json, ref sequenceID);
```

```
        _logger.WLOGD("DeleteUserGroup()::Result=" + resCode + ", SequenceID=" +
sequenceID);
}
```

Getting Information of Users

Create button click event handler and get the information of user using GetUserInfo () method of SsmSdkWrapper.

```
// UserManagementSample.cs
private void PrintUsers_Click(object sender, EventArgs e)
{
        String str = ssmSdkWrapper.GetUserInfo();

        this.textBoxUserList.Text = str;
}
```

Add a User

Create button click event handler and add a user using AddUser () method of SsmSdkWrapper.

```
// UserManagementSample.cs
private void AddUserButton_Click(object sender, EventArgs e)
{
        if (textBoxAddUserID.Text == String.Empty)
        {
                MessageBox.Show("ID is empty.");
                return;
        }

        Guid usergroupGuid = Guid.Empty;
        if (!Guid.TryParse(textBoxAddUserParentGuid.Text, out usergroupGuid))
        {
                MessageBox.Show("Check parent guid(usergroup guid).");
                return;
        }

        UserModel userModel = new UserModel();
        userModel.UserGroupGuid = usergroupGuid.ToString();
```

```
        userModel.ID = textBoxAddUserID.Text;
        userModel.UserName = textBoxAddUserName.Text;
        userModel.Password = textBoxAddUserPassword.Text;
        userModel.PhoneNumber = textBoxAddUserPhoneNumber.Text;
        userModel.Email = textBoxAddUserEmail.Text;
        userModel.Address = textBoxAddUserAddress.Text;
        userModel.Description = textBoxAddUserDescription.Text;

        String json = JsonConvert.SerializeObject(userModel, Formatting.Indented);
        UInt32 resCode = 0;
        UInt32 sequenceID = 0;

        resCode = ssmSdkWrapper.AddUser(json, ref sequenceID);
        _logger.WLOGD("AddUser()::Result=" + resCode + ", SequenceID=" + sequenceID);
}
```

Modify a User

Create button click event handler and modify a user using ModifyUser () method of SsmSdkWrapper.

```
// UserManagementSample.cs
private void ModifyUserButton_Click(object sender, EventArgs e)
{
        if (textBoxAddUserID.Text == String.Empty)
        {
                MessageBox.Show("ID is empty.");
                return;
        }

        Guid userGuid = Guid.Empty;
        if (!Guid.TryParse(textBoxModifyUserGuid.Text, out userGuid))
        {
                MessageBox.Show("Check guid.");
                return;
        }

        UserModel userModel = new UserModel();
        userModel.Guid = userGuid.ToString();
```

```
        userModel.ID = textBoxAddUserID.Text;
        userModel.UserName = textBoxAddUserName.Text;
        userModel.Password = textBoxAddUserPassword.Text;
        userModel.PhoneNumber = textBoxAddUserPhoneNumber.Text;
        userModel.Email = textBoxAddUserEmail.Text;
        userModel.Address = textBoxAddUserAddress.Text;
        userModel.Description = textBoxAddUserDescription.Text;

        String json = JsonConvert.SerializeObject(userModel, Formatting.Indented);
        UInt32 resCode = 0;
        UInt32 sequenceID = 0;

        resCode = ssmSdkWrapper.ModifyUser(json, ref sequenceID);
        _logger.WLOGD("ModifyUser()::Result=" + resCode + ", SequenceID=" + sequenceID);
}
```

Delete a User

Create button click event handler and delete a user using DeleteUser () method of SsmSdkWrapper.

```
// UserManagementSample.cs
private void DeleteUserButton_Click(object sender, EventArgs e)
{
        UserModel userModel = new UserModel();

        userModel.Guid = textBoxDeleteUserGuid.Text;
        String json = JsonConvert.SerializeObject(userModel, Formatting.Indented);
        UInt32 resCode = 0;
        UInt32 sequenceID = 0;

        resCode = ssmSdkWrapper.DeleteUser(json, ref sequenceID);
        _logger.WLOGD("DeleteUser()::Result=" + resCode + ", SequenceID=" + sequenceID);
}
```

See also

LogIn Sample Program

Backup Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 16

NTP Sample Program

This chapter describes how to load or change NTP settings using the Wisenet SSM Client SDK.

Contents

Introduction

The NTP sample program is an example that load or change the NTP settings. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

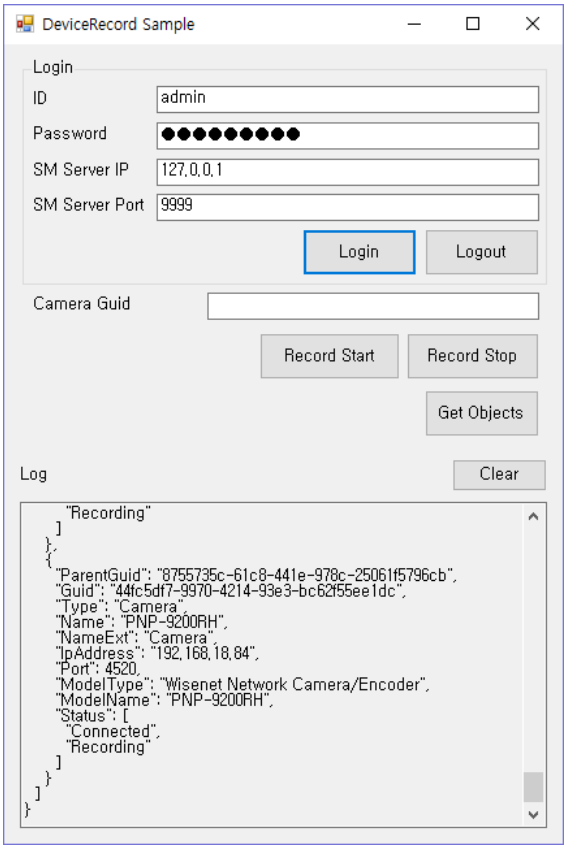


Figure 14 NTP Sample Program Execution

Step 3. Click the [Get] button to get the current NTP settings.

Step 4. Change NTP settings and click [Set] button.

Step 5. Click on [Logout] to release a connection to the server.

Step 6. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()

Login

2. ssmSdkWrapper.Login()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

5. ssmSdkWrapper.OnEvent

Get Ntp

6. ssmSdkWrapper.GetNtp

Set Ntp

7. ssmSdkWrapper.SetNtp

Logout

8. ssmSdkWrapper.Logout()

Release

9. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the NTP sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing

SsmSdkWrapper.

Load the NTP settings

Create button click event handler and get the NTP settings using GetNtp () method of SsmSdkWrapper.

```
// NtpSample.cs
private void NtpGet_Click(object sender, EventArgs e)
{
        UInt32 sequenceID = 0;
        UInt32 resCode = 0;

        resCode = ssmSdkWrapper.GetNtp(ref sequenceID);

        _logger.WLOGD("GetNtp()::Result=" + resCode + ", SequenceID=" + sequenceID);
}
private void OnResponse(UInt32 commandID, UInt32 errorCode, UInt32 sequenceID, string info)
{
        _logger.WLOGD(
         "OnResponse()::" +
         " Command ID=" + commandID +
         " Result=" + errorCode +
         " Sequence ID= " + sequenceID +
         " Info=" + info);

        if (commandID == (uint)CommandID.GetNtp)
        {
                UpdateNtp(info);
        }
}

public delegate void DeleUpdateNtp(String json);
private void UpdateNtp(String json)
{
        NtpInfoModel ntpInfoModel = JsonConvert.DeserializeObject<NtpInfoModel>(json);

        if (this.InvokeRequired)
        {
```

```
                DeleUpdateNtp deUpdateNtp = new DeleUpdateNtp(this.UpdateNtp);
                this.Invoke(deUpdateNtp, new object[] { json });
                return;
        }
        else
        {
                this.ntpServerEnable.Checked = ntpInfoModel.NtpServerEnabled;
                this.ntpHost.Checked = ntpInfoModel.NtpEnabled;
                this.textBoxNtpHost.Text = ntpInfoModel.NtpHostName;
                this.textBoxLastSyncTime.Text = ntpInfoModel.LastSyncTime;
                this.textBoxSyncInterval.Text = ntpInfoModel.NtpSyncIntervalMinute.ToString();
        }
}
```

Change the NTP settings

Create button click event handler and change the NTP settings using SetNtp () method of SsmSdkWrapper.

```
// NtpSample.cs
private void NtpSet_Click(object sender, EventArgs e)
{
        String json = String.Empty;
        UInt32 sequenceID = 0;
        UInt32 resCode = 0;

        NtpInfoModel ntpInfoModel = new NtpInfoModel();

        ntpInfoModel.NtpServerEnabled = this.ntpServerEnable.Checked;
        ntpInfoModel.NtpEnabled = this.ntpHost.Checked;
        ntpInfoModel.NtpHostName = this.textBoxNtpHost.Text;

        Int32 interval = 0;
        Int32.TryParse(textBoxSyncInterval.Text, out interval);
        ntpInfoModel.NtpSyncIntervalMinute = interval;

        json = JsonConvert.SerializeObject(ntpInfoModel, Formatting.Indented);

        resCode = ssmSdkWrapper.SetNtp(json, ref sequenceID);
```

```
        _logger.WLOGD("SetNtp()::Result=" + resCode + ", SequenceID=" + sequenceID);
}
```

See also

LogIn Sample Program

Backup Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 17

Device Record Sample Program

This chapter describes how to start/stop the manual recording of camera using the Wisenet SSM Client SDK.

Contents

Introduction

The Device Record sample program is an example that start/stop manual recording of camera. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Log In] button to see if you can successfully access the Wisenet SSM Core Server.

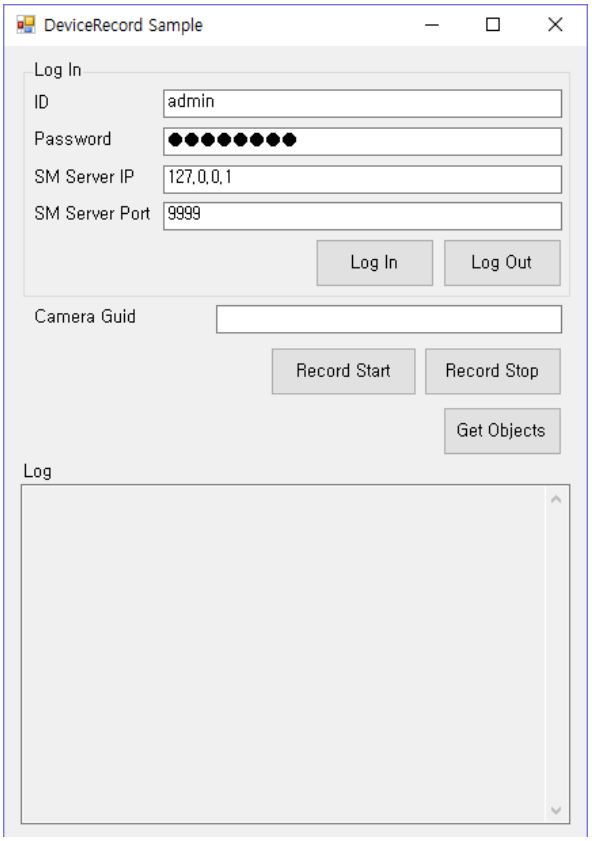


Figure 15 DeviceRecord Sample Program Execution

Step 3. Enter the camera's guid.

Step 4. Click the [Record Start] button to start manual recording.

Step 5. If you click [Get Objects] button, you can see the status of devices on log text box.

Step 6. Click the [Record Stop] button to stop manual recording.

Step 7. Terminate the program.

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.InitializeEvent()

Login

2. ssmSdkWrapper.Login()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

5. ssmSdkWrapper.OnEvent

Start Recording

6. ssmSdkWrapper.StartDeviceRecording

Stop Recording

7. ssmSdkWrapper.StopDeviceRecording

Logout

8. ssmSdkWrapper.Logout()

Release

9. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the DeviceRecord sample program in detail.

Refer the 'How to Implement' of CHAPTER 4 for adding SsmSdkWrapper reference, adding SsmSdkWrapper Callback functions, initializing SsmSdkWrapper, login, logout, releasing

SsmSdkWrapper.

Start manual recording

Create button click event handler and start manual recording using StartDeviceRecording() method of SsmSdkWrapper.

```
// DeviceRecordSample.cs
private void btnRecordStart_Click(object sender, EventArgs e)
{
	String guidText = textBoxUnitID.Text;
	Guid guid = Guid.Empty;
	UInt32 sequenceID = 0;

	if (Guid.TryParse(guidText, out guid))
	{
		uint resCode = ssmSdkWrapper.StartDeviceRecording(textBoxUnitID.Text, ref sequenceID);
		_logger.WLOGD("StartDeviceRecording()::Result=" + resCode + ", SequenceID=" + sequenceID);
	}
	else
	{
		MessageBox.Show("Check the guid.");
	}
}
```

Stop manual recording

Create button click event handler and stop manual recording using StopDeviceRecording() method of SsmSdkWrapper.

```
// DeviceRecordSample.cs
private void btnRecordStop_Click(object sender, EventArgs e)
{
	String guidText = textBoxUnitID.Text;
	Guid guid = Guid.Empty;
	UInt32 sequenceID = 0;
```

```
        if (Guid.TryParse(guidText, out guid))
        {
                uint resCode = ssmSdkWrapper.StopDeviceRecording(textBoxUnitID.Text, ref
sequenceID);
                _logger.WLOGD("StopDeviceRecording()::Result=" + resCode + ",
SequenceID=" + sequenceID);
        }
        else
        {
                MessageBox.Show("Check the guid.");
        }
}
```

See also

LogIn Sample Program

Backup Sample Program

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 18

Wisenet DDNS Login Sample Program

This chapter describes how to login to the Wisenet SSM core server by Wisenet DDNS ID.

Contents

Introduction

This sample program is an example of how to login to Wisenet SSM Core Server by Wisenet DDNS ID. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Login] button to see if you can successfully access the Wisenet SSM Core Server.

Step 3. Terminate the program.

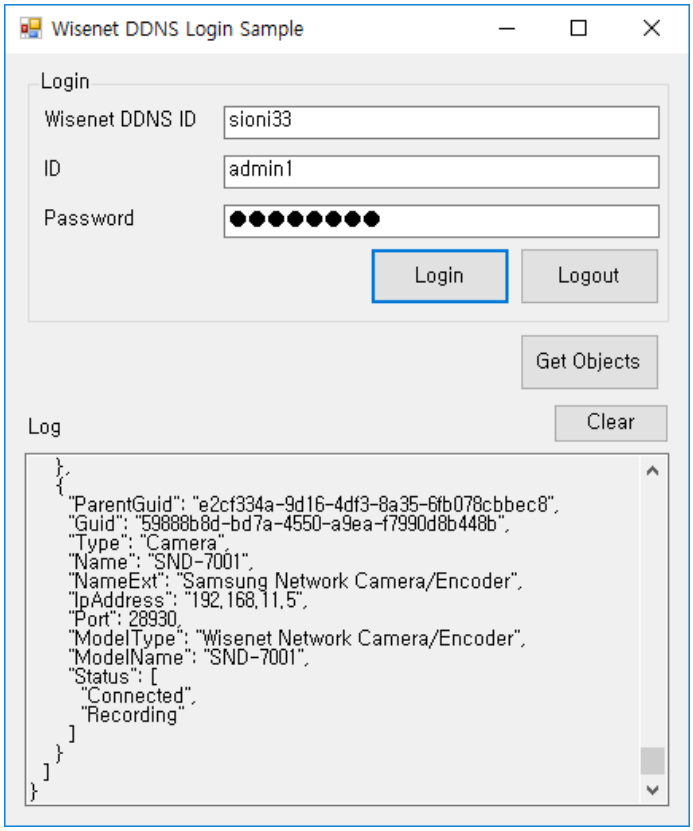


Figure 16 Wisenet DDNS Login Sample Program Execution

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.Initialize()

Login

2. ssmSdkWrapper.DdnsLogin()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

5. ssmSdkWrapper.OnEvent

Logout

6. ssmSdkWrapper.Logout()

Release

7. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the Wisenet DDNS Login sample program in detail.

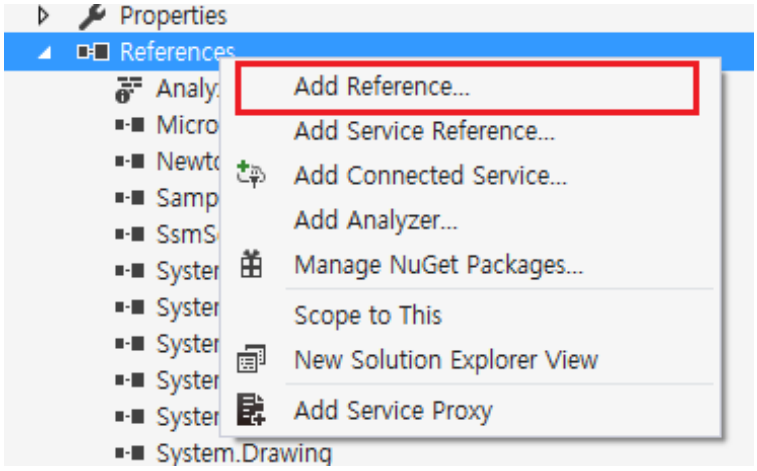
Add SsmSdkWrapper project to the reference

Procedures

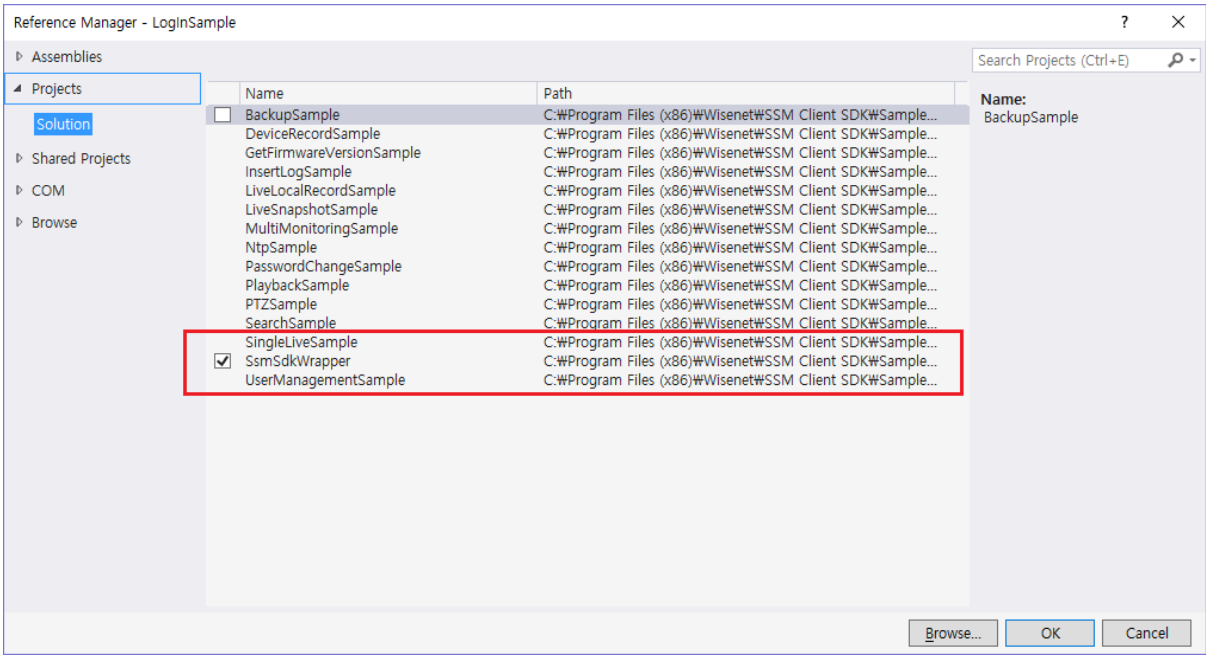
In Visual Studio 2013, create Windows Forms and then add SsmSdkWrapper reference.

Step 1. Create Windows Forms.

Step 2. Click the right mouse button on References. And choose [Add Reference]



Step 3. Add the SsmSdkWrapper on Reference Manager.



Add SsmSdkWrapper Callback functions

In order to receive the response and the event from the server, you need to implement Callback functions.

Implement Callback: OnResponse

Implement the OnResponse in the DdnsLogInSample class.

```
// DdnsLoginSample.cs
private void OnResponse(UInt32 commandID, UInt32 errorCode, UInt32 sequenceID, string info)
{
        _logger.WLOGD(
         "OnResponse()::" +
         " Command ID=" + commandID +
         " Result=" + errorCode +
         " Sequence ID= " + sequenceID +
         " Info=" + info);
}
```

Implement Callback: OnEvent

Implement the OnEvent in the DdnsLogInSample class.

```
// DdnsLoginSample.cs
private void OnEvent(UInt32 eventID, String info)
{
        _logger.WLOGD(
                "OnEvent()::" +
                " Event ID=" + eventID +
                " Info=" + info);
}
```

Initialize SsmSdkWrapper

Create an SsmSdkWrapper and initialize it by calling ssmSdkWrapper.Initialize().

Call the InitializeEvent() method within the DdnsLogInSample_Load() of the DdnsLoginSample class. InitializeEvent() starts using the service of the SsmEventSdk.

```
// DdnsLoginSample.cs
private SsmSdkWrapper ssmSdkWrapper = null;

private void DdnsLogInSample_Load(object sender, EventArgs e)
{
        ssmSdkWrapper = new SsmSdkWrapper(this.OnResponse, this.OnEvent);
        ssmSdkWrapper.InitializeEvent();
}
```

Login

By creating a login button event handler, you can execute logging in. You need the Wisenet DDNS ID, ID and password of Wisenet SSM Core Server to login.

```
// DdnsLoginSample.cs
private void btnLogIn_Click(object sender, EventArgs e)
{
        uint resCode = ssmSdkWrapper.DdnsLogIn(
                txtID.Text,
                txtPassword.Text,
                txtDdnsId.Text);
        _logger.WLOGD("LogIn()::Result=" + resCode);
}
```

Logout

By creating a logout button event handler, you can execute logging out.

```
// DdnsLoginSample.cs
private void btnLogOut_Click(object sender, EventArgs e)
{
        uint resCode = ssmSdkWrapper.Logout();
        _logger.WLOGD("LogOut()::Result=" + resCode);
```

```
}
```

Release SsmSdkWrapper

After using SsmEventSdk, you must stop the service and release the resources.

Call the ReleaseEvent() method within the DdnsLogInSample_Closed() of the DdnsLoginSample class.

```
// DdnsLogInSample.cs
private void DdnsLogInSample_Closed(object sender, EventArgs e)
{
        ssmSdkWrapper.ReleaseEvent();
}
```

See also

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

CHAPTER 19

URL Login Sample Program

This chapter describes how to login to the Wisenet SSM core server's URL.

Contents

Introduction

This sample program is an example of how to login to Wisenet SSM Core Server' URL. It can be used as follows.

Step 1. After building the sample program, see if they run properly.

Step 2. Click on the [Login] button to see if you can successfully access the Wisenet SSM Core Server.

Step 3. Terminate the program.

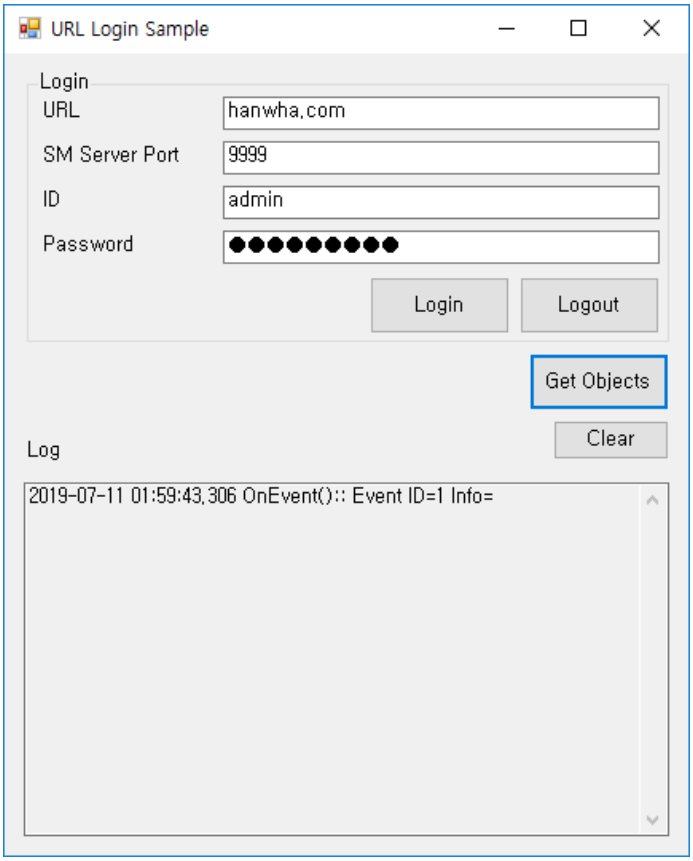


Figure 17 URL Login Sample Program Execution

API Call Procedures

The order of API calls is as follows:

Preprocessing

1. ssmSdkWrapper.Initialize()

Login

2. ssmSdkWrapper.UrlLogin()

Response of Login

3. ssmSdkWrapper.OnResponse

Receive Event: LogIn

4. ssmSdkWrapper.OnEvent

Receive Event: ObjectConnected

5. ssmSdkWrapper.OnEvent

Logout

6. ssmSdkWrapper.Logout()

Release

7. ssmSdkWrapper.ReleaseEvent()

How to Implement

This section describes how to implement the URL Login sample program in detail.

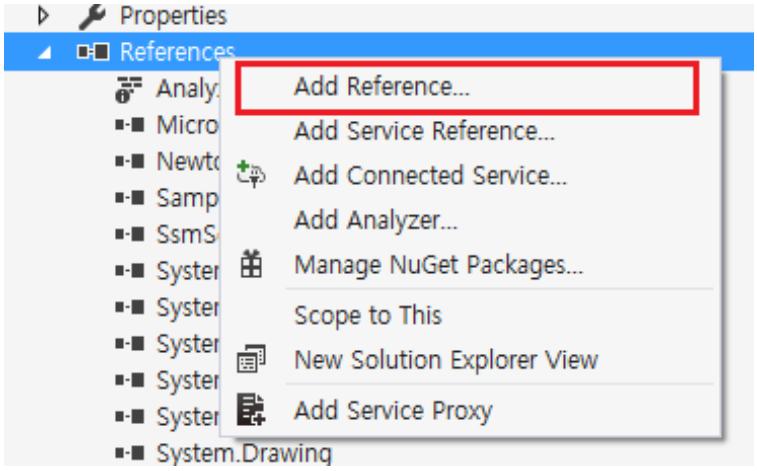
Add SsmSdkWrapper project to the reference

Procedures

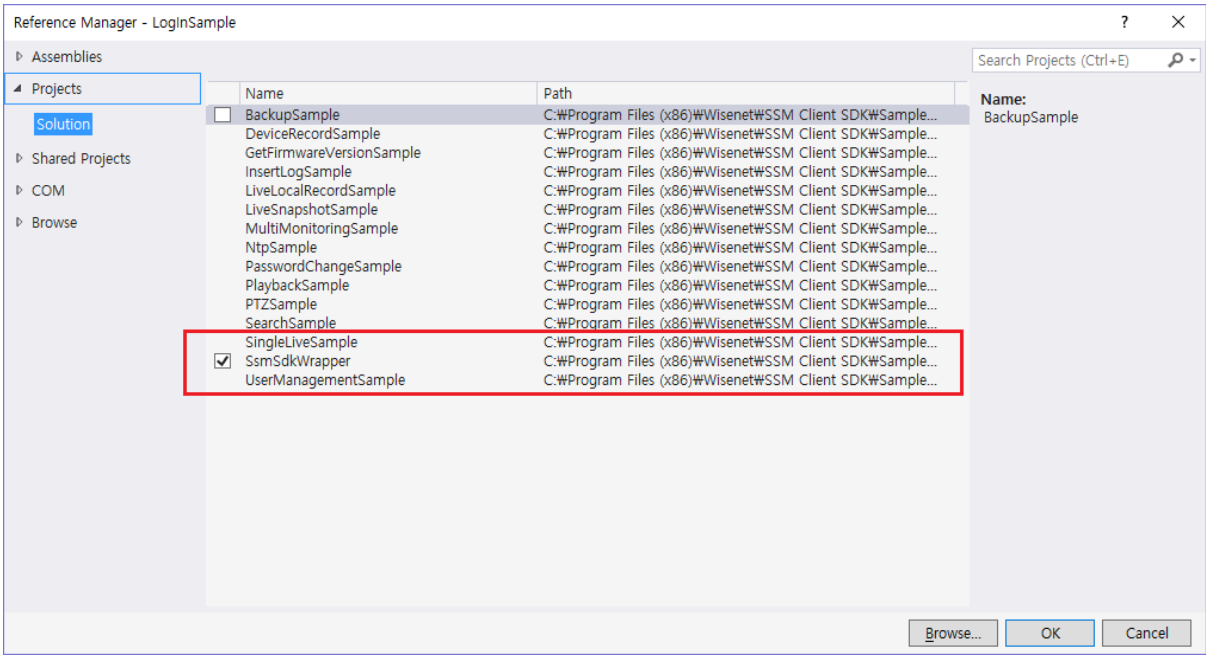
In Visual Studio 2013, create Windows Forms and then add SsmSdkWrapper reference.

Step 1. Create Windows Forms.

Step 2. Click the right mouse button on References. And choose [Add Reference]



Step 3. Add the SsmSdkWrapper on Reference Manager.



Add SsmSdkWrapper Callback functions

In order to receive the response and the event from the server, you need to implement Callback functions.

Implement Callback: OnResponse

Implement the OnResponse in the UrlLogInSample class.

```
// UrlLoginSample.cs
private void OnResponse(UInt32 commandID, UInt32 errorCode, UInt32 sequenceID, string info)
{
        _logger.WLOGD(
         "OnResponse()::" +
         " Command ID=" + commandID +
         " Result=" + errorCode +
         " Sequence ID= " + sequenceID +
         " Info=" + info);
}
```

Implement Callback: OnEvent

Implement the OnEvent in the UrlLogInSample class.

```
// UrlLoginSample.cs
private void OnEvent(UInt32 eventID, String info)
{
        _logger.WLOGD(
                "OnEvent()::" +
                " Event ID=" + eventID +
                " Info=" + info);
}
```

Initialize SsmSdkWrapper

Create an SsmSdkWrapper and initialize it by calling ssmSdkWrapper.Initialize().

Call the InitializeEvent() method within the UrlLogInSample_Load() of the UrlLoginSample class. InitializeEvent() starts using the service of the SsmEventSdk.

```
// UrlLoginSample.cs
private SsmSdkWrapper ssmSdkWrapper = null;

private void UrlLogInSample_Load(object sender, EventArgs e)
{
        ssmSdkWrapper = new SsmSdkWrapper(this.OnResponse, this.OnEvent);
        ssmSdkWrapper.InitializeEvent();
}
```

Login

By creating a login button event handler, you can execute logging in. You need the URL, server port, ID and password of Wisenet SSM Core Server to login.

```
// UrlLoginSample.cs
private void btnLogIn_Click(object sender, EventArgs e)
{
        uint resCode = ssmSdkWrapper.UrlLogIn(
                        txtID.Text,
                        txtPassword.Text,
                        txtUrl.Text,
                        Convert.ToUInt32(txtPort.Text));
        _logger.WLOGD("LogIn()::Result=" + resCode);
}
```

Logout

By creating a logout button event handler, you can execute logging out.

```
// UrlLoginSample.cs
private void btnLogOut_Click(object sender, EventArgs e)
{
        uint resCode = ssmSdkWrapper.Logout();
```

```
        _logger.WLOGD("LogOut()::Result=" + resCode);
}
```

Release SsmSdkWrapper

After using SsmEventSdk, you must stop the service and release the resources.

Call the ReleaseEvent() method within the UrlLogInSample_Closed() of the UrlLoginSample class.

```
// UrlLoginSample.cs
private void UrlLogInSample_Closed(object sender, EventArgs e)
{
        ssmSdkWrapper.ReleaseEvent();
}
```

See also

Wisenet SSM Client SDK API Reference (v2.10.6)

FAQ

This section summarizes frequently asked questions on the sample programs.

This part is left empty since there has not been any inquiry registered so far.

Abbreviations

D

DEP

Data Execution Prevention

DLL

Dynamic Linking Library

DVR

Digital Video Recorder

NVR

Network Video Recorder

P

PTZ

Pan/Tilt/Zoom

S

SDK

Software Development Kit

SSM

Smart Security Manager