

Samsung XNS ActiveX SDK

Programmer's Guide

for C++programming

XNS_ACTIVEPG_KR

Ed.1.44.00

2016-07-29

Copyright

©2011-2016 Hanwha Techwin Co., Ltd. All rights reserved.

Trademark

XNS ActiveX는 한화테크원(주)의 등록상표입니다. 이밖에 이 문서에 언급된 상표는 해당 회사 고유의 등록상표입니다.

Restriction

이 문서의 저작권은 한화테크원(주)에게 있습니다. 한화테크원(주)의 공식 승인 없이 이 문서의 내용을 복제, 배포 및 재생산 할 수 없고, 임의로 변경할 수 없습니다.

Disclaimer

한화테크원(주)는 이 문서에 수록된 정보의 완결성과 정확성을 검증하기 위해 최대한 노력하였으나 이에 대해 보증하지는 않습니다. 문서의 사용 결과에 따른 책임은 전적으로 사용자에게 있습니다.

한화테크원(주)은 사전 예고 없이 이 문서의 내용을 변경 할 수 있습니다.

Contact Information

HANWHA TECHWIN Co., LTD.

Hanwhatechwin 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

<http://www.samsungsecuritypartner.com>

SAMSUNG TECHWIN AMERICA Inc.

1480 Charles Willard St, Carson, CA 90746, UNITED STATES

Tel Free: +1-877-213-1222 FAX: +1-310-632-2195

SAMSUNGTECHWIN EUROPE LTD.

Samsung House, 1000 Hillswood Drive, Hillswood

Business

Park Chertsey, Surrey, UNITED KINGDOM KT16 OPS

TEL: +44-1932-45-5300 FAX: +44-1932-45-5325

들어가며

목적

이 문서는 한화테크원(주)의 XNS ActiveX SDK를 사용하여 애플리케이션을 작성하는 방법을 설명하기 위해 작성되었습니다. XNS ActiveX SDK와 샘플 프로그램을 이용하면 네트워크 보안 장비와 연동하는 콘솔 프로그램을 손쉽게 작성할 수 있습니다.

독자

이 문서는 XNS ActiveX SDK를 이용하여 애플리케이션 (CMS, 뷰어(viewer) 애플리케이션 등)을 구현하는 개발자를 독자로 합니다.

범위

이 문서는 XNS ActiveX SDK의 설치 방법, 애플리케이션 구현 방법, 샘플 프로그램 구현 방법 등에 대해 설명합니다.

문서 구성

이 문서는 다음과 같이 구성되어 있습니다.

CHAPTER 1. XNS ActiveX SDK: XNS ActiveX SDK의 구조와 설치 방법에 대해 설명합니다.

CHAPTER 2. 개발 환경 설정: 애플리케이션 개발을 위한 프로젝트 설정 방법에 대해 설명합니다.

CHAPTER 3. 샘플 프로그램 소개: XNS ActiveX SDK가 제공하는 샘플 프로그램을 소개합니다.

CHAPTER 4. Start Up 샘플 프로그램: Start Up 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 5. Single Live 샘플 프로그램: Single Live 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 6. Multi Channel 샘플 프로그램: Multi Channel 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 7. SingleLive Stream 샘플 프로그램: Single Live Stream 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 8. Multiple Connect 샘플 프로그램: Multiple Connect 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 9. Playback 샘플 프로그램: Playback 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 10. PTZ 샘플 프로그램: PTZ 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 11. Alarm & Event 샘플 프로그램: Alarm & Event 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 12. Local Recording 샘플 프로그램: Local Recording 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 13. Video Raw Data 샘플 프로그램: Video Raw Data 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 14. Backup 샘플 프로그램: Backup 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 15. Listen & Talk 샘플 프로그램: Listen & Talk 샘플 프로그램 구현 방법을 설명합니다.

CHAPTER 16. SDK Test 샘플 프로그램: SDK Test 샘플 프로그램 사용 방법을 설명합니다.

APPENDIX A. UTC Time: UTC Time에 대해 설명합니다.

버튼 또는 메뉴 표기: 버튼명과 메뉴명은 대괄호([])를 사용해서 표기하였습니다.

메뉴 선택 경로 표기: 메뉴 선택 경로는 산괄호(>)를 사용해서 표기하였습니다.

문서 이력

이 문서의 출판 이력은 다음과 같습니다.

버전	개정일자	개정내역
0.9	2010. 08. 16	최초 작성
1.0	2010. 11. 25	API Reference 완료 (SDK v1.23)
1.1	2011. 05. 03	추가 설명 및 오류 수정. 디바이스 추가(SDK v1.30)
1.2	2011. 05. 25	Window 7에서 프로젝트 환경설정 GMT 시간 설정 추가
1.3	2011. 06. 16	Device Control 이용 관련 추가 설명 (SDK v1.30.06) API 추가 <ul style="list-style-type: none">• AreaZoom(), Zoom1X() 추가• OnLvEvent() 추가• GetPtzPos(), SetPtzPos() 추가• OnGetPtzPos() 추가
1.3.1	2011.06.24	SDK 아키텍처 변경 내용 반영 Stop() 예제 코드 수정 Start() 설명 수정
1.4.0	2011.08.05	샘플 프로그램 9 개에 대한 구현 방법 설명 추가 SDK 설치 절차 업데이트
1.4.1	2011.08.16	C#, Visual Basic 으로 개발시 고려사항 내용 추가
1.4.2	2011.09.29	SingleLive Stream 샘플 프로그램 구현 방법 설명 추가
1.34.00	2011.11.01	문서 버전 정보를 SDK 버전 정보와 일치시킴.
1.39.00	2012.08.13	지원 제품 정보 업데이트
1.41.00	2014.03.28	Multi Channel 샘플 프로그램 구현 방법 설명 추가
1.41.10	2014.07.24	Supported device list 수정
1.41.20	2014.08.27	Contact Information 기술 지원 사이트 변경
1.42.90	2015.11.12	WPF 샘플 프로그램 구현 방법 추가 – ActiveX 컨트롤 삽입 표 1 Supported Device List 수정

목차

들어가며.....	3
문서 이력.....	5
목차.....	6
그림 목차.....	13
표 목차.....	15
CHAPTER 1 XNS ActiveX SDK.....	16
개요.....	17
SDK Architecture.....	18
SDK 구성.....	20
시스템 요구 사양.....	21
지원 장비.....	21
SDK 설치.....	21
SDK 삭제.....	25
CHAPTER 2 개발 환경 설정.....	28
사전 지식.....	28
프로젝트 생성.....	29
프로젝트 설정.....	35
DEP 문제 해결 방법.....	40
CHAPTER 3 샘플 프로그램 소개.....	43
위치.....	44
샘플 프로그램 목록.....	44
CHAPTER 4 Start Up 샘플 프로그램.....	45

샘플 프로그램 소개	46
API 호출 순서	46
구현 방법 상세 설명	47
ActiveX 컨트롤 삽입	47
헤더 파일 추가	54
컨트롤 초기화	54
See also	56
FAQ	56
CHAPTER 5 Single Live 샘플 프로그램	57
샘플 프로그램 소개	58
API 호출 순서	59
구현 방법 상세 설명	60
ActiveX 컨트롤 생성	60
디바이스 핸들러 생성	60
디바이스 연결 설정	61
이벤트 핸들러 구현	62
미디어 오픈	65
버튼 이벤트 핸들러 작성	66
See also	68
FAQ	68
CHAPTER 6 Multi channel 샘플 프로그램	69
샘플 프로그램 소개	70
API 호출 순서	71
구현 방법 상세 설명	72
ActiveX 컨트롤 생성	72
디바이스 핸들러 생성	72
디바이스 연결 설정	73
이벤트 핸들러 구현	73
미디어 오픈	73

버튼 이벤트 핸들러 작성	74
See also	75
FAQ	75
CHAPTER 7 SingleLive Stream샘플 프로그램	76
샘플 프로그램 소개	77
API 호출 순서	78
구현 방법 상세 설명	79
ActiveX 컨트롤 생성	79
디바이스 핸들러 생성	79
디바이스 연결 설정	80
이벤트 핸들러 구현	80
스트림 오픈	80
See also	82
FAQ	82
CHAPTER 8 Multiple Connect 샘플 프로그램	83
샘플 프로그램 소개	84
API 호출 순서	85
구현 방법 상세 설명	86
ActiveX 컨트롤 삽입 및 초기화	86
DEVICEINFO 구조체 및 배열 선언	86
List Control 구현	87
디바이스 핸들러 생성	88
디바이스 연결 설정	88
미디어 오픈	89
See also	91
FAQ	91
CHAPTER 9 Playback 샘플 프로그램	92
샘플 프로그램 소개	93

API 호출 순서	94
구현 방법 상세 설명	95
ActiveX 컨트롤 삽입 및 초기화	96
디바이스 핸들러 생성	96
디바이스 연결 설정	96
Calendar 검색	96
녹화 영상 불러오기	98
미디어 플레이	102
See also	103
FAQ	104

CHAPTER 10 PTZ 샘플 프로그램	105
샘플 프로그램 소개	106
API 호출 순서	107
구현 방법 상세 설명	109
ActiveX 컨트롤 삽입 및 초기화	110
디바이스 핸들러 생성	110
디바이스 연결 설정	110
OSD 메뉴 제어	110
카메라 동작 제어	111
Preset List 이벤트	112
Preset List 추가 및 삭제	113
마우스 이벤트 처리	113
AreaZoom 설정하기	114
PTZ 절대 좌표 값 얻어오기	114
PTZ 제어하기	115
See also	117
FAQ	117
CHAPTER 11 Alarm&Event 샘플 프로그램	118
샘플 프로그램 소개	119

API 호출 순서	120
구현 방법 상세 설명	121
ActiveX 컨트롤 삽입 및 초기화	122
디바이스 핸들러 생성	122
디바이스 연결 설정	122
미디어 플레이	122
이벤트 등록	122
알람 출력 설정	125
See also	126
FAQ	127
CHAPTER 12 Local Recording 샘플 프로그램	128
샘플 프로그램 소개	129
API 호출 순서	130
구현 방법 상세 설명	132
ActiveX 컨트롤 생성	132
디바이스 핸들러 생성	132
디바이스 연결 설정	132
미디어 오픈	135
로컬 레코딩 시작	136
로컬 레코딩 종료	137
로컬 레코딩 파일 재생	137
See also	138
FAQ	138
CHAPTER 13 VideoRawData샘플 프로그램	139
샘플 프로그램 소개	140
API 호출 순서	141
구현 방법 상세 설명	142
ActiveX 컨트롤 삽입 및 초기화	142
디바이스 핸들러 생성	142

디바이스 연결 설정	143
미디어 플레이	143
See also	145
FAQ	145
CHAPTER 14 Backup 샘플 프로그램	146
샘플 프로그램 소개	147
API 호출 순서	148
구현 방법 상세 설명	150
ActiveX 컨트롤 생성	150
디바이스 핸들러 생성	150
백업 시간 설정	150
백업 시작	151
백업 종료	152
백업 파일 재생	152
See also	154
FAQ	154
CHAPTER 15 Listen&Talk 샘플 프로그램	155
샘플 프로그램 소개	156
API 호출 순서	157
구현 방법 상세 설명	158
ActiveX 컨트롤 삽입 및 초기화	158
디바이스 핸들러 생성	159
디바이스 연결 설정	159
미디어 플레이	159
음성 수신 설정	159
수신 음성 재생 설정	160
음성 전송 설정	160
음성 입력	160
음성 데이터 송출	163

See also	163
FAQ	163
CHAPTER 16 16. 샘플 프로그램	164
샘플 프로그램 소개	165
See also	174
FAQ	174
APPENDIX A UTC Time	176
약어	178

그림 목차

그림 1 XNS ActiveX Architecture.....	19
그림 2 Start Up 샘플 프로그램 실행 화면.....	46
그림 3 Start Up 샘플 프로그램 API 호출 순서	47
그림 4 ActiveX 컨트롤 삽입 절차(1)	48
그림 5 ActiveX 컨트롤 삽입 절차(2)	49
그림 6 ActiveX 컨트롤 삽입 결과.....	49
그림 7 컨트롤 변수 추가 절차(1).....	50
그림 8 컨트롤 변수 추가 절차(2).....	51
그림 9 Single Live 샘플 프로그램 실행 화면	58
그림 10 Single Live 샘플 프로그램 API 호출 순서	60
그림 11 Multi Channel 샘플 프로그램 실행 화면.....	70
그림 12 Multi Channel 샘플 프로그램 API 호출 순서	72
그림 13 SingleLiveStream 샘플 프로그램 실행 화면.....	77
그림 14 SingleLiveStream 샘플 프로그램 API 호출 순서	79
그림 15 Multiple Connect 샘플 프로그램 실행 화면.....	84
그림 16 Multiple Connect 샘플 프로그램 API 호출 순서	86
그림 17 Playback 샘플 프로그램 실행 화면	93
그림 18 Playback 샘플 프로그램 API 호출 순서	95
그림 19 PTZ 샘플 프로그램 실행 화면	106
그림 20 Preset 다이얼로그	107
그림 21 PTZ 샘플 프로그램 API 호출 순서	109
그림 22 Alarm&Event 샘플 프로그램 실행 화면.....	119
그림 23 Alarm&Event 샘플 프로그램 API 호출 순서	121
그림 24 이벤트 등록 방법	123
그림 25 Local Recording 샘플 프로그램 실행 화면.....	129
그림 26 Local Recording 샘플 프로그램 API 호출 순서	131
그림 27 VideoRawData 샘플 프로그램 실행 화면.....	140
그림 28 VideoRawData 샘플 프로그램 API 호출 순서	142
그림 29 Backup 샘플 프로그램 실행 화면(1).....	147
그림 30 Backup 샘플 프로그램 실행 화면(2).....	148

그림 31 Backup 샘플 프로그램 API 호출 순서	149
그림 32 Listen&Talk 샘플 프로그램 실행 화면	156
그림 33 Listen&Talk 샘플 프로그램 API 호출 순서	158
그림 34 SdkTest 샘플 프로그램 실행 화면	165
그림 35 초기화 및 디바이스 연결 함수 테스트 화면	165
그림 36 Function Log 확인	166
그림 37 Event Log 확인	166
그림 38 미디어 스트림 제어 함수 테스트 화면	167
그림 39 라이브 영상 제어 함수 테스트 화면	168
그림 40 오디오 및 알람 제어 함수 테스트 화면	168
그림 41 기타 XnsSdkDevice 제어 함수 테스트 화면	169
그림 42 기타 XnsSdkWindow 제어 함수 테스트 화면	170
그림 43 PTZ, OSD 제어 기능 테스트 화면	170
그림 44 디지털 줌 제어 테스트 화면	171
그림 45 영상 백업 기능 테스트 화면	171
그림 46 영상 출력 테스트 화면	172
그림 47 Playback 기능 테스트 화면	172
그림 48 Playback 제어 테스트 화면	173
그림 49 로컬 레코딩 기능 테스트 화면	173
그림 50 영상 버퍼 제어 기능 테스트 화면	173
그림 51 음성 제어 기능 테스트 화면	173
그림 52 스냅샷 기능 테스트 화면	174

표 목차

표 1 Supported device list.....	21
표 2 샘플 프로그램 목록	44
표 3 Menu Control Command.....	111
표 4 Camera Control Command.....	112
표 5 PTZ Command	116
표 6 이벤트 목록	122

CHAPTER 1

XNS ActiveX SDK

이 장은 XNS ActiveX SDK의 개요, 구성 및 설치/삭제 방법 등에 대해 설명합니다.

Contents

- 개요
- SDK Architecture
- SDK 구성
- 시스템 요구 사양
- 지원 장비
- SDK 설치
- SDK 삭제

개요

XNS ActiveX SDK 는 XNS 라이브러리를 기반으로 Microsoft의 ActiveX 기술을 채택하여 만든 라이브러리입니다. ActiveX 기술을 채택함으로써 XNS 라이브러리를 C#, C++, Visual Basic, Internet Explorer용 웹 페이지(Java script) 등 다양한 플랫폼과 언어에서 개발하는 것이 가능해졌습니다.

XNS 라이브러리는 저수준(low-level)의 스트리밍과 이벤트 수신을 지원하는 반면, XNS ActiveX 라이브러리는 영상을 표현하고 음성 데이터를 재생할 수 있습니다. XNS ActiveX 라이브러리의 개발 목적은 개발자가 삼성의 네트워크 장비(예, 카메라, DVR)를 쉽게 통합할 수 있게 하기 위함입니다.

XNS ActiveX 라이브러리를 이용하여 애플리케이션을 구현하기 위해서는 XNS ActiveX SDK에서 제공하는 DLL 파일과 ActiveX 파일이 필요하며, 특히 ActiveX 파일 (즉, ocx 파일)은 레지스트리에 등록해야 합니다. XNS ActiveX SDK의 설치 파일을 이용하면 DLL파일과 ActiveX 파일이 지정된 위치에 저장됩니다. 설치 방법은 '

Device Type	Model Name
N/W Camera Encoder	현재 시판중인 네트워크 카메라, 엔코더 전모델 *홈 네트워크 카메라 제외
DVR	현재 시판중인 DVR 전모델 *PC DVR(SPR-XXXX 제외)
NVR	현재 시판중인 NVR 전모델 *SRN-872 제외

SDK 설치'절을 참고하시기 바랍니다.

XNS ActiveX 라이브러리에서 제공하는 기능은 다음과 같습니다.

- 실시간 영상 감시
- 네트워크 장비에 저장된 영상 재생 및 정보 획득
- 로컬 PC에 영상 저장 및 로컬 PC에 저장된 영상 재생
- 네트워크 장비에 저장된 영상을 로컬 PC로 백업(back-up)
- 네트워크 장비의 펌웨어(firmware) 업그레이드
- 네트워크 장비(카메라)의 PTZ 기능 제어
- 네트워크 장비에서 발생한 이벤트 정보 획득

SDK Architecture

XNS ActiveX의 시스템 구조는 다음과 같습니다.

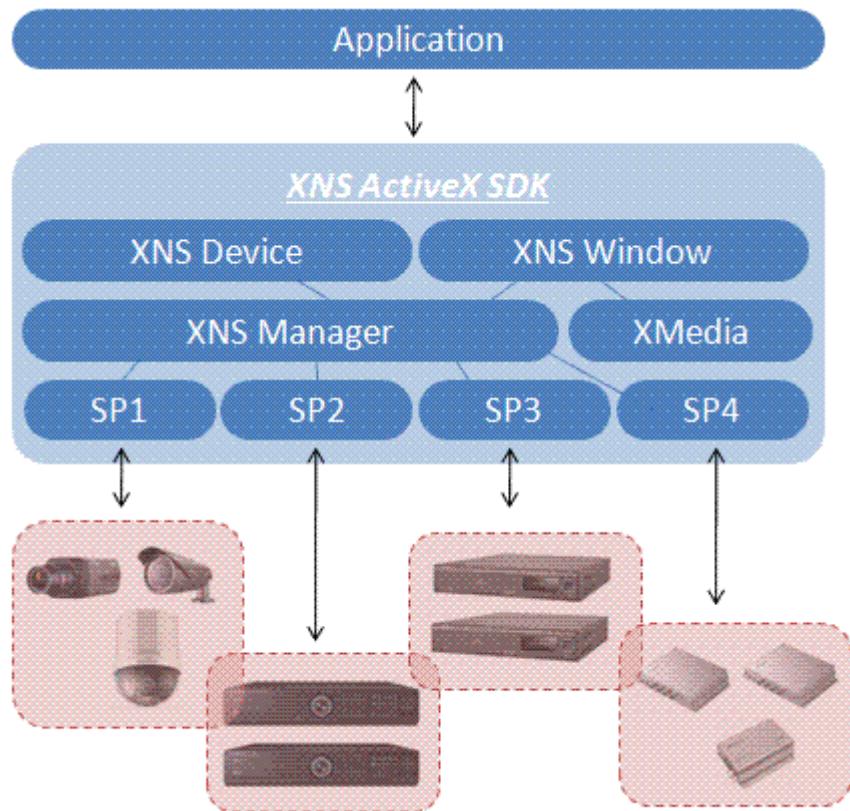


그림 1 XNS ActiveX Architecture

- Application
 - DVR이나 네트워크 카메라 같은 네트워크 장비에 있는 영상 데이터를 감시, 재생, 검색하는 사용자 프로그램.
- XNS ActiveX SDK
 - XNS Device: 네트워크 장비와 XNS 라이브러리를 관리하는 ActiveX control.
 - XNS Window: 비디오/오디오 데이터를 재생하는 ActiveX control.
 - XNS Manager: XNS 라이브러리와 이벤트를 관리하는 라이브러리(DLL).
 - XMedia: 수신된 영상/오디오 데이터를 디코딩&렌더링함
 - SP 라이브러리: 네트워크 장비로부터 받은 영상 데이터를 XNS Manager로 전달하고 사용자 명령에 따라 장비를 제어함.
- 네트워크 장비(DVR, Network camera)
 - 특정 영역에서 수집한 영상 데이터를 다른 장비로 전송함.

SDK 구성

XNS ActiveX SDK는 다음과 같이 구성되어 있습니다.

- 라이브러리 및 ActiveX control
 - 애플리케이션 개발 시 사용되는 DLL 파일과 OCX 파일.
 - Bin 폴더에 설치됨.
- 설정 파일 (Configuration file)
 - 시스템 설정을 위한 XML 파일은 {\$SDK path}\Bin\Config 폴더에 설치됨.
 - 특히, Xns.xml 파일은 각 프로토콜에 대한 DLL 파일의 정보를 저장하고 있음.
- 프로토콜 구현 파일 (DLL)
 - {\$SDK path}\Bin\SP 폴더에 설치됨.
 - 이 파일의 위치가 변경되는 경우, Xns.xml 문서의 내용도 수정되어야 함.
- Simple backup file player
 - {\$SDK path}\Bin\Viewer 폴더에 설치됨.
 - 애플리케이션이 미디어 스트림을 'SEC' 포맷으로 저장/백업한 경우, 저장된 파일은 이 플레이어를 이용하여 재생할 수 있음. 이때, 뷰어의 파일명은 라이브러리에서 백업 파일과 동일한 이름을 부여함. 뷰어를 실행하면, 같은 이름을 가진 SEC 파일이 자동으로 로딩/loading)됨.
- 헤더(header) 파일
 - {\$SDK path}\sample_code\include 폴더에 설치됨.
 - 애플리케이션이 XNS를 사용하기 위해서는 이 헤더 파일들을 인클루드(include)해야 함.
- 문서
 - {\$SDK path}\Doc 폴더에 저장됨.
- 샘플 코드
 - {\$SDK path}\sample_code 폴더에 저장됨.
 - 영상 데이터를 감시/재생하는 샘플과 PTZ를 제어하는 샘플을 제공함.
 - 샘플 코드의 실행 파일은 {\$SDK path}\Bin 폴더에 설치됨.

시스템 요구 사양

XNS ActiveX SDK를 사용하기 위한 시스템의 사양은 다음과 같습니다.

- 운영체제(OS)
 - Microsoft Windows XP 또는 그 상위 버전
- CPU
 - 펜티엄 이상
- 개발 환경
 - Microsoft Visual Studio 2008 이상 권장

지원 장비

XNS ActiveX SDK를 지원하는 장비 리스트는 다음과 같습니다.

표 1 Supported Device List

Device Type	Model Name
N/W Camera Encoder	현재 시판중인 네트워크 카메라, 엔코더 전모델 *홈 네트워크 카메라 제외
DVR	현재 시판중인 DVR 전모델 *PC DVR(SPR-XXXX 제외)
NVR	현재 시판중인 NVR 전모델 *SRN-872 제외

SDK 설치

XNS ActiveX SDK를 설치하는 방법은 다음과 같습니다.

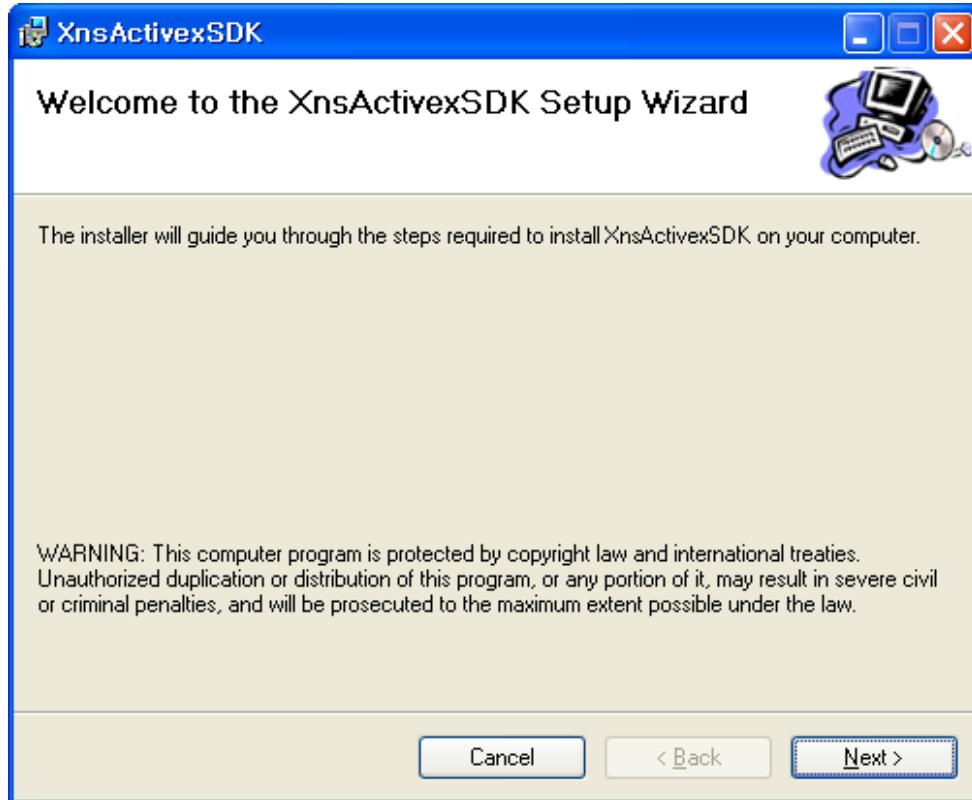
참고

XNS ActiveX SDK는 STEP 웹사이트 (<https://www.samsungsecuritypartner.com>) 를 통해서
얻을 수 있습니다.

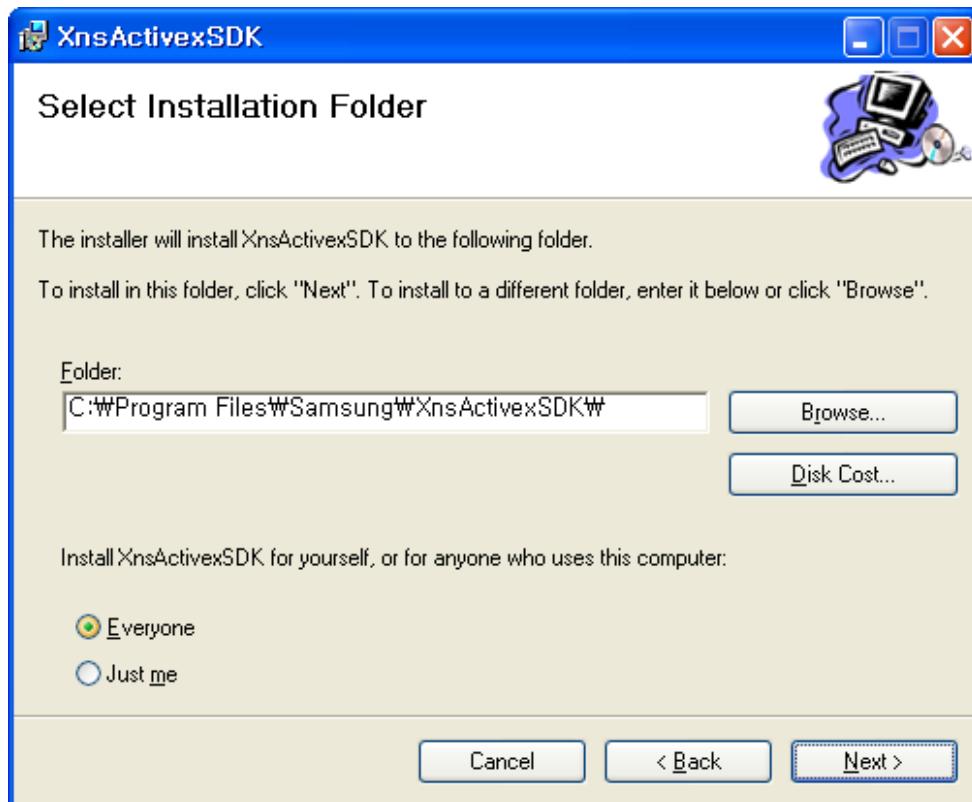
절차

- Step 1.** 설치 파일이 있는 폴더 또는 CD-ROM이 설치된 드라이브로 이동합니다.
- Step 2.** 다운로드 한 설치 파일의 압축을 해제합니다.
- Step 3.** 압축 해제 폴더에서 setup.exe 파일을 실행합니다.

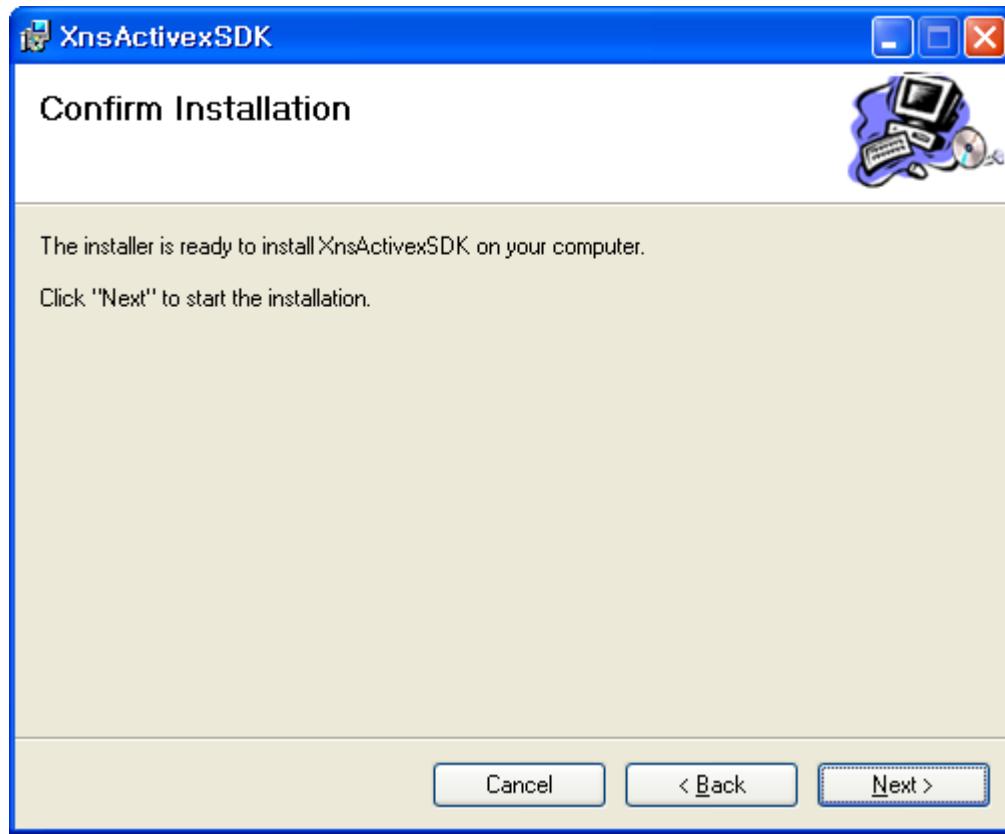
Step 4. 다음과 같은 화면이 나오면 [Next]를 클릭합니다.



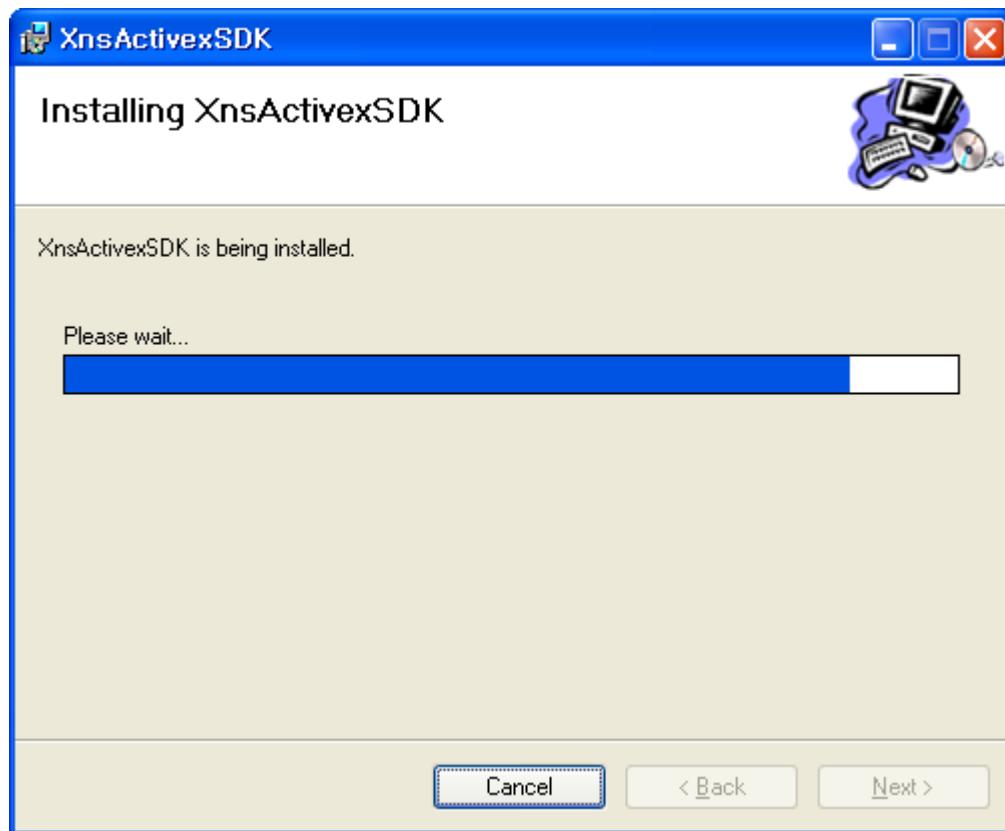
Step 5. XNS ActiveX SDK가 설치되는 기본 경로는 다음과 같습니다. 설치 경로를 변경하려면 [Browse...]를 클릭하여 경로를 선택합니다. 경로를 지정한 후에는 [Next]를 클릭합니다.

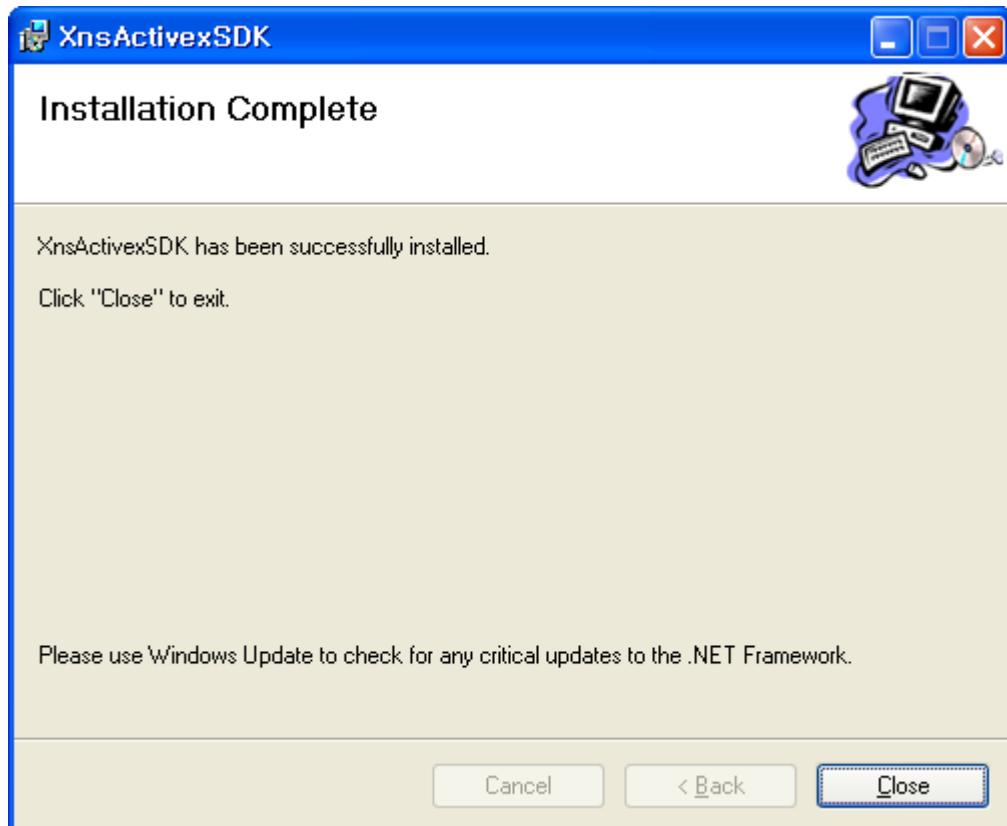


Step 6. [Next]를 클릭하면 SDK 설치가 시작됩니다.



Step 7. 설치가 완료되면 [Close]를 클릭하여 설치 프로그램을 종료합니다.





설치가 완료된 후에는 [시작] > [프로그램] > [Samsung] > [XnsActivexSDK]에서 문서와 샘플 코드를 확인할 수 있습니다.

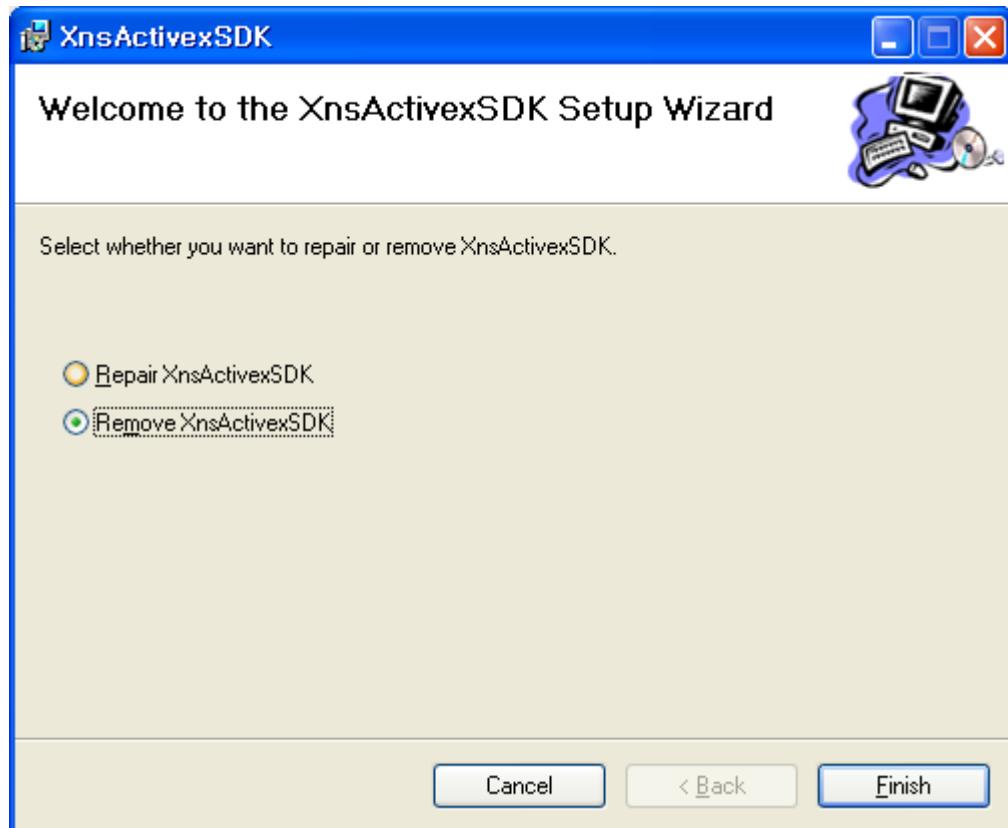
SDK 삭제

1) 실행파일을 통한 삭제

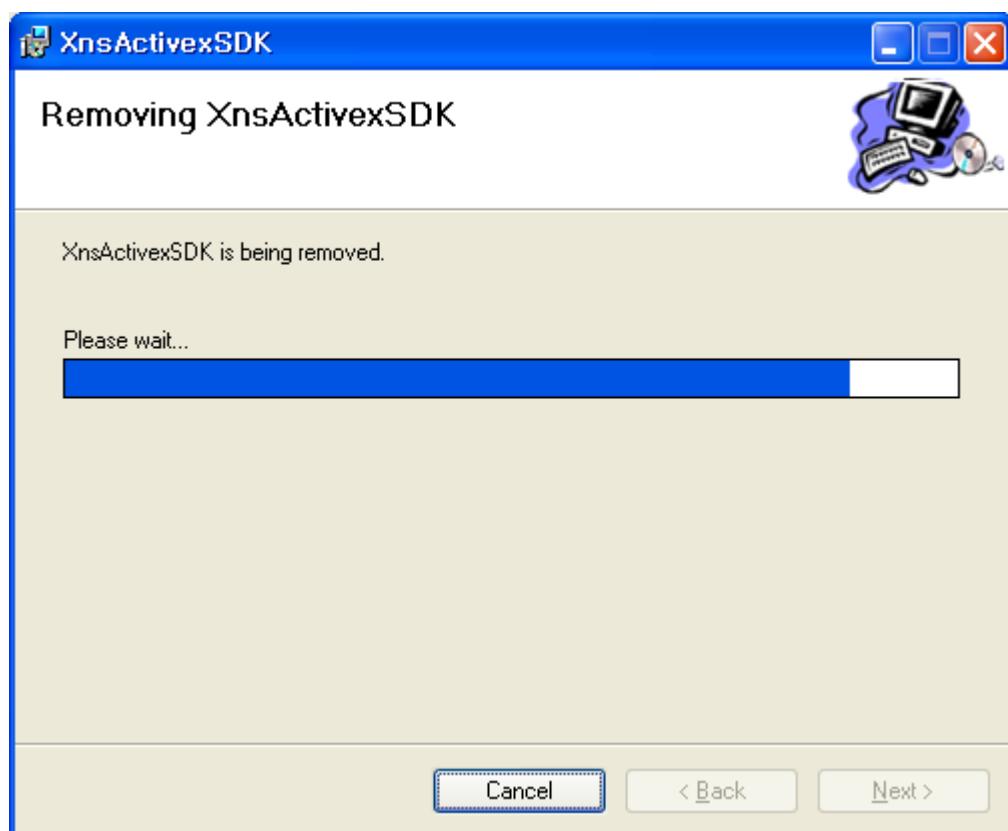
실행파일을 통해 XNS ActiveX SDK를 삭제하는 방법은 다음과 같습니다.

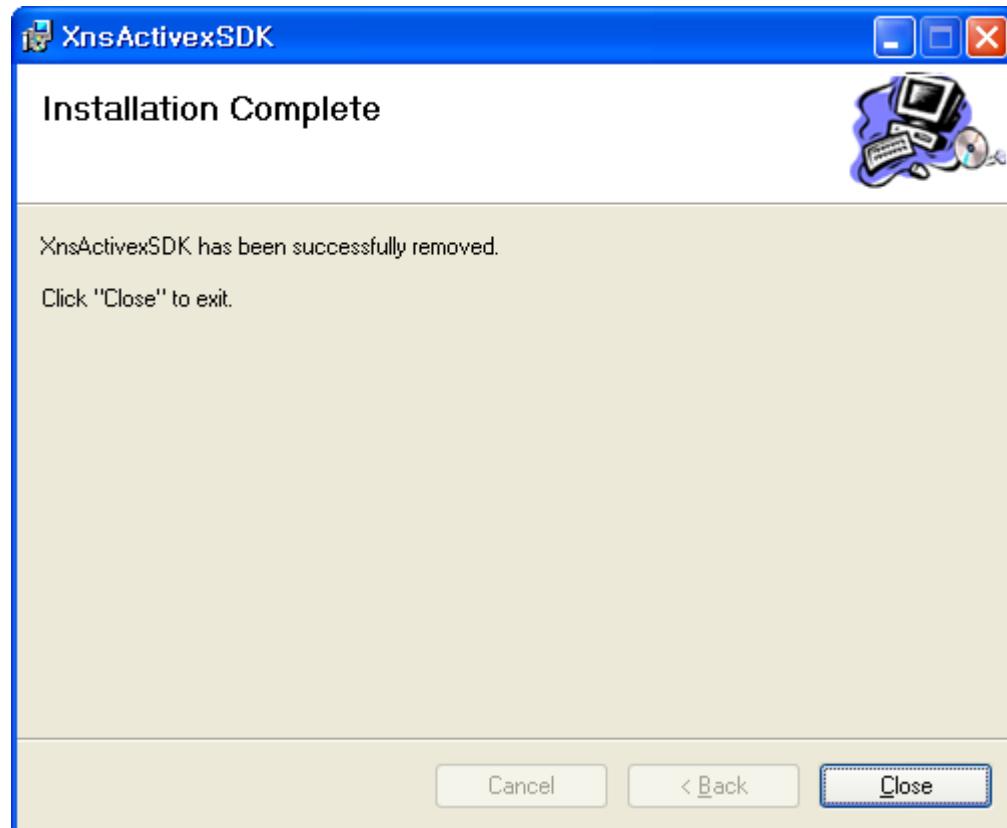
절차

- Step 1. 설치 파일이 있는 폴더 또는 CD-ROM이 설치된 드라이브로 이동합니다.
- Step 2. 다운로드 한 설치 파일의 압축을 해제합니다.
- Step 3. 다음과 같은 화면이 나오면 [Remove XnsActivexSDK_setup]을 선택하고 [Finish]를 클릭합니다.



Step 4. SDK 삭제가 완료되면 [Close]를 클릭하여 설치 프로그램을 종료합니다.





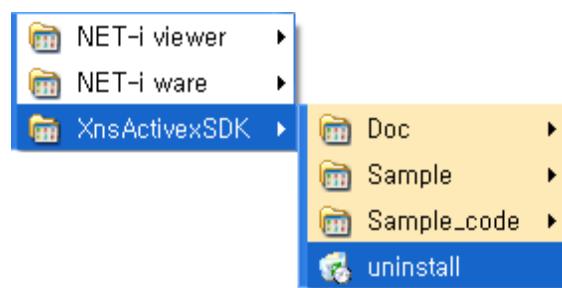
2) 컨텍스트 메뉴를 통한 삭제

프로그램의 컨텍스트 메뉴를 통해 XNS ActiveX SDK를 삭제하는 방법은 다음과 같습니다..

절차

Step 1. 시작 -> 프로그램 -> Samsung -> XnsActivexSDK을 클릭합니다.

Step 2. XnsActivexSDK 의 Uninstall 메뉴를 클릭합니다.



CHAPTER 2

개발 환경 설정

이 장에서는 SDK에서 제공하는 샘플 코드를 가지고 개발 환경을 설정하는 방법에 대해 설명합니다. 샘플 코드는 {\$SDK path}\sample_code 폴더에서 확인하실 수 있습니다.

Contents

- 사전 지식
- 프로젝트 생성
- 프로젝트 설정

사전 지식

XNS ActiveX의 샘플코드는 MFC와 ActiveX를 기반으로 하고 있습니다. 이에 대한 관련 정보는 MSDN을 참조하시기 바랍니다.

MFC (Microsoft Foundation Class)

[http://msdn.microsoft.com/en-us/library/d06h2x6e\(VS.80\).aspx](http://msdn.microsoft.com/en-us/library/d06h2x6e(VS.80).aspx)

ActiveX

[http://msdn.microsoft.com/en-us/library/aa751972\(VS.85\).aspx](http://msdn.microsoft.com/en-us/library/aa751972(VS.85).aspx)

프로젝트 생성

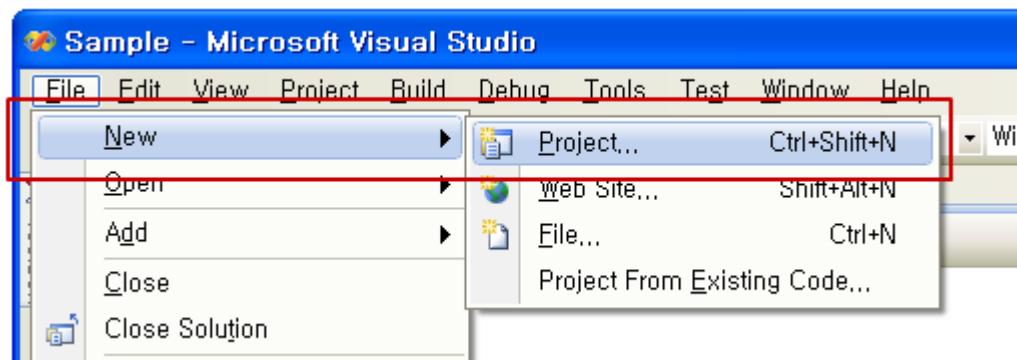
애플리케이션을 작성하는데 필요한 프로젝트는 다음과 같은 방법으로 생성합니다. (본 문서는 VS2008과 VS 6.0 영문 버전을 기준으로 설명합니다.)

Visual Studio 2008

절차

Step 1. Visual Studio 2008을 실행합니다.

Step 2. [File] > [New] > [Project]를 선택합니다.

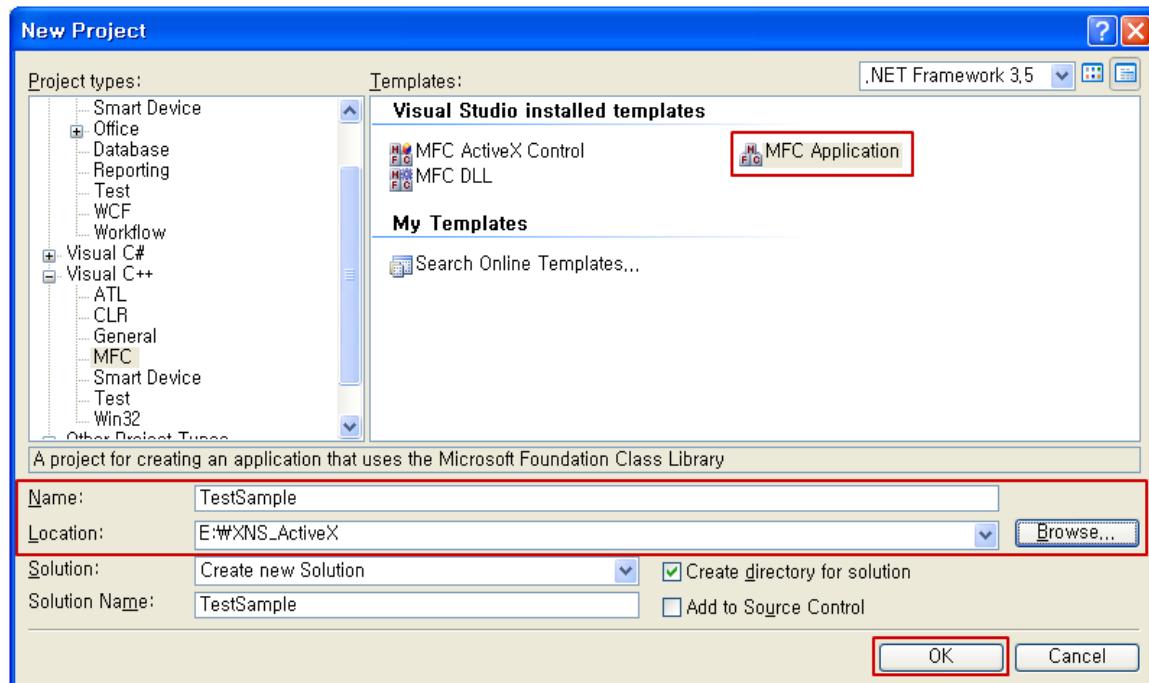


Step 3. [New Project]ダイアログ에서,

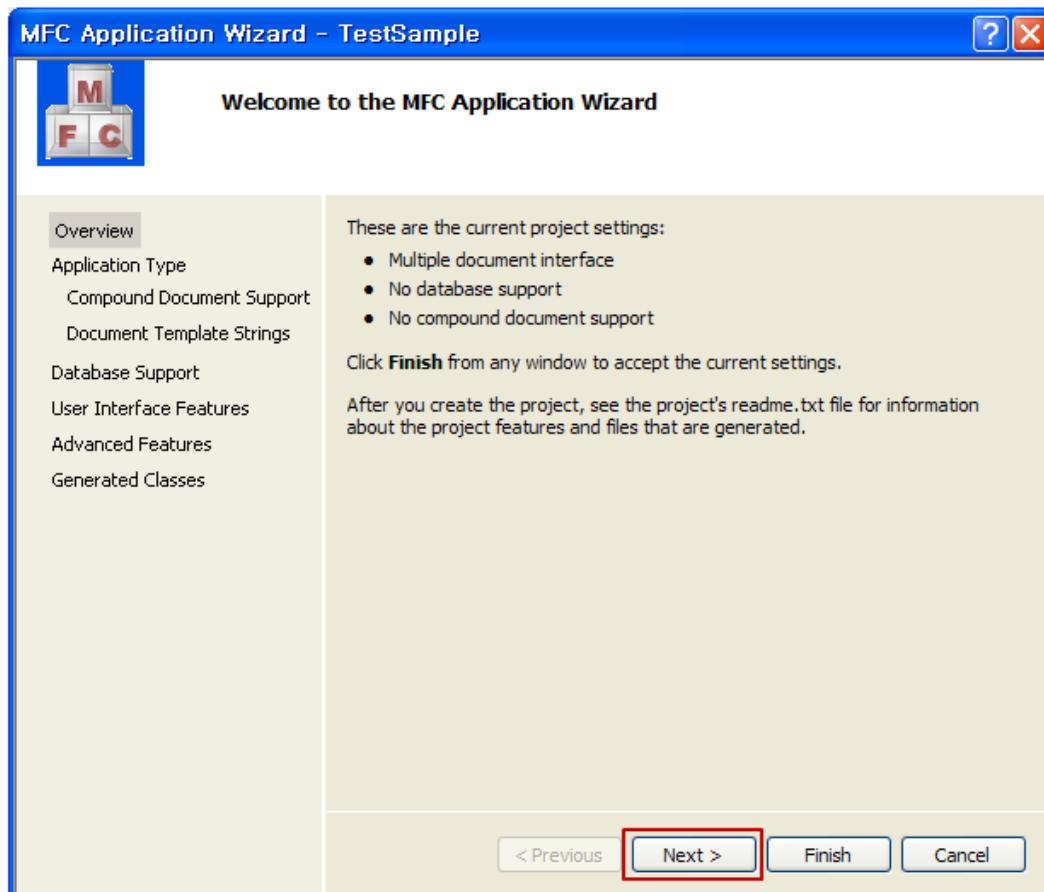
A. "MFC Application" 템플릿을 선택합니다.

B. 프로젝트의 이름과 경로를 지정합니다.

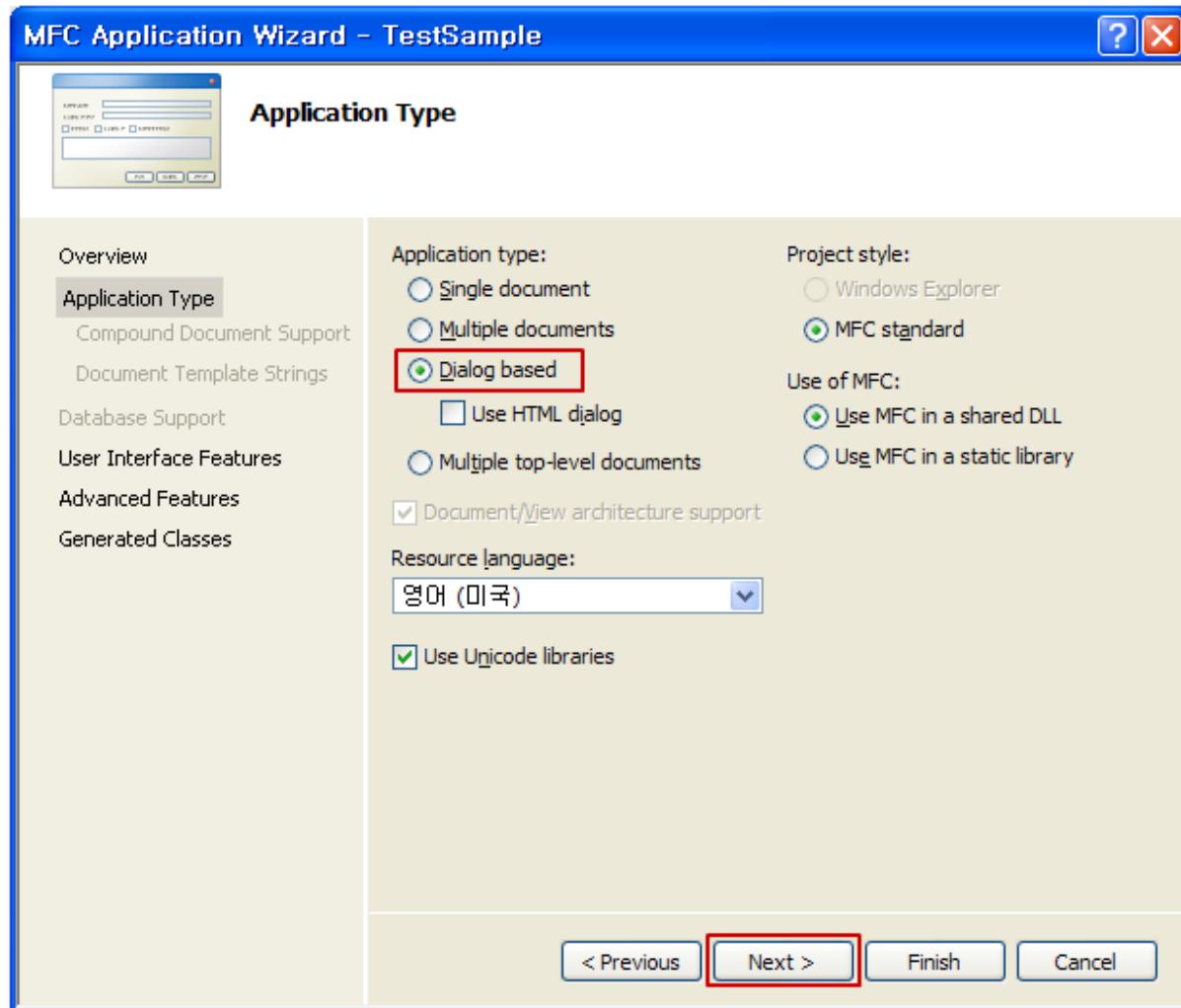
C. [OK]를 클릭합니다.



Step 4. [Next]를 클릭합니다.

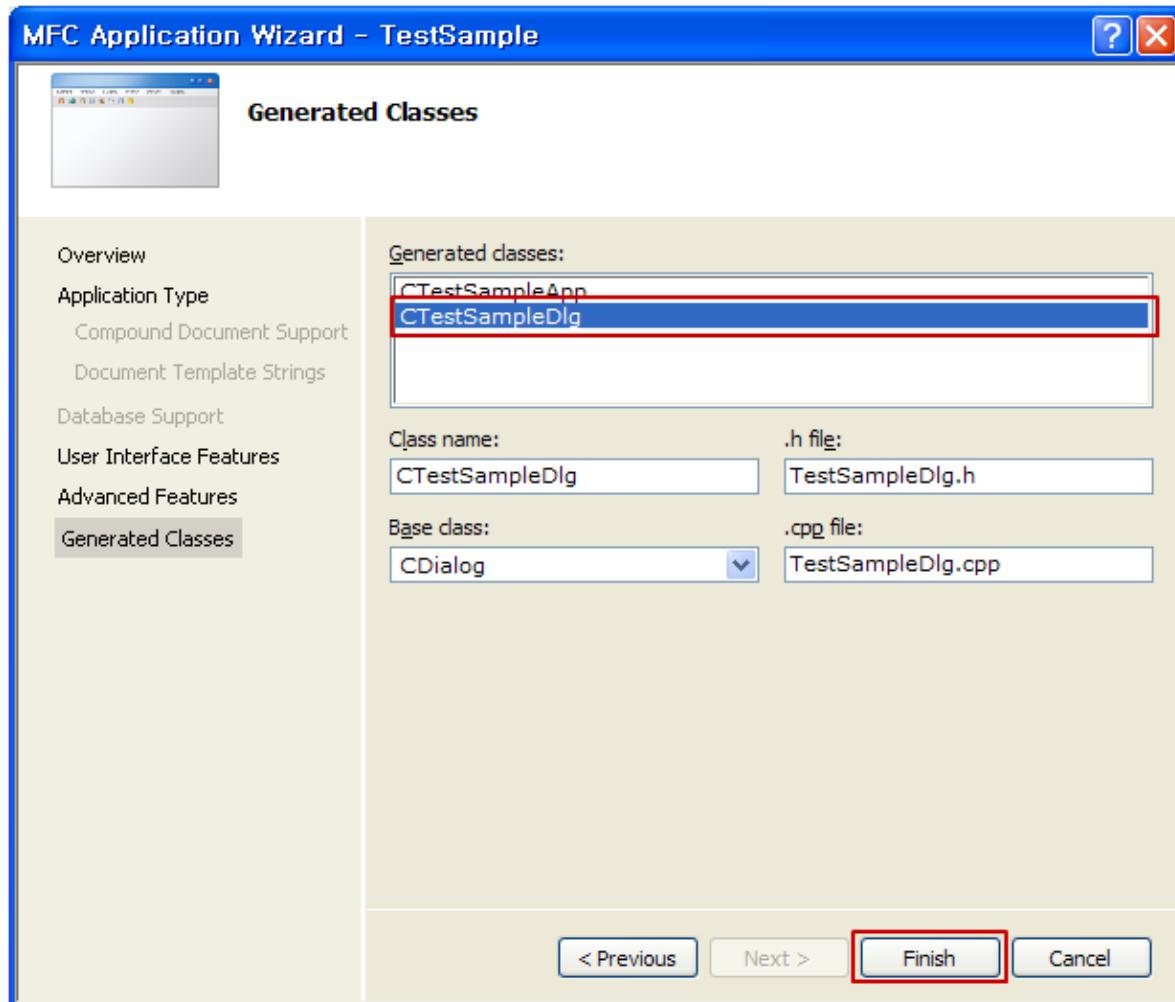


Step 5. 애플리케이션 타입으로 "Dialog based"를 선택하고 [Next]를 클릭합니다.



Step 6. 좌측 메뉴에서 [Generated Classes] 메뉴를 선택합니다.

Step 7. [Generated Classes] 리스트에서 Dialog 클래스를 선택하고 [Finish]를 클릭합니다.

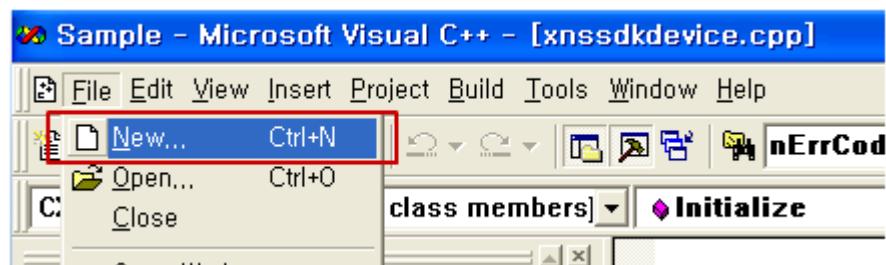


Visual Studio 6.0

절차

Step 1. Visual Studio 6.0을 실행합니다.

Step 2. [File] > [New...] 메뉴를 선택합니다.

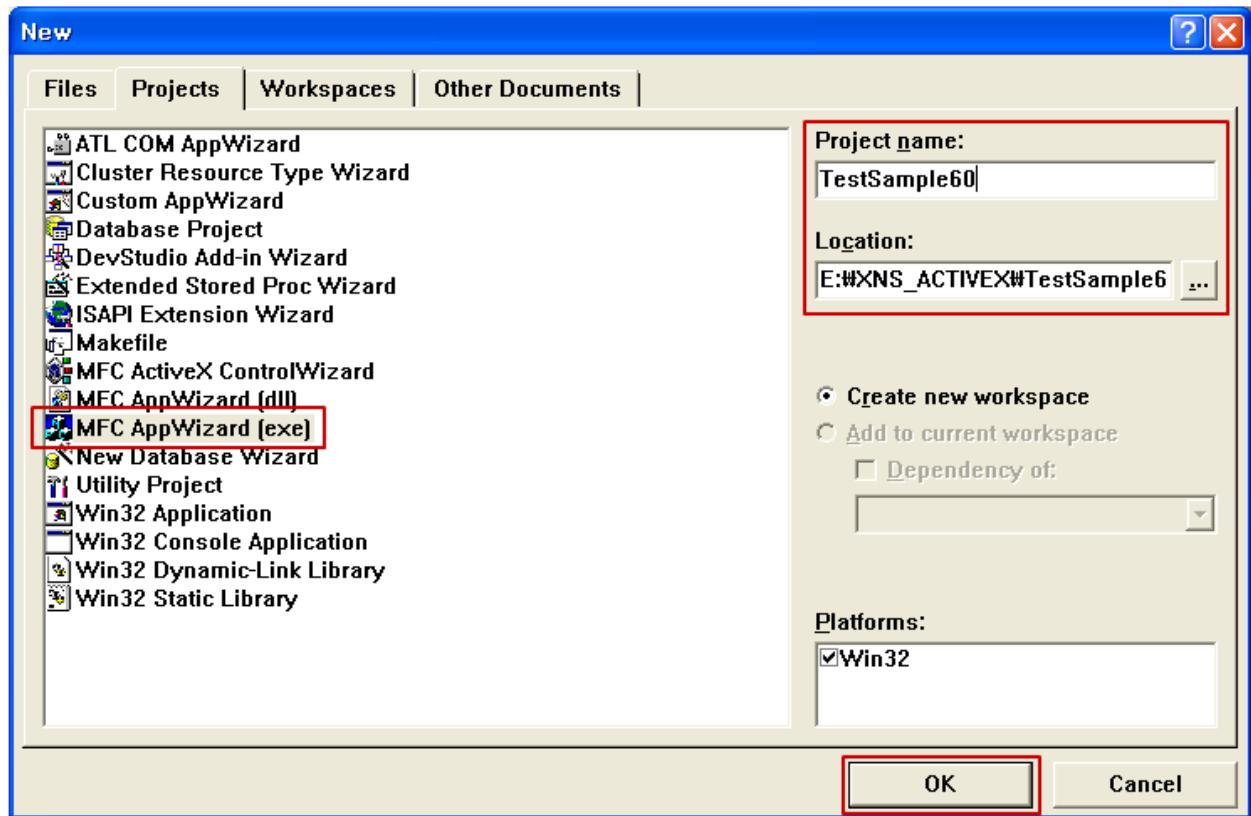


Step 3. [New]ダイアログ에서,

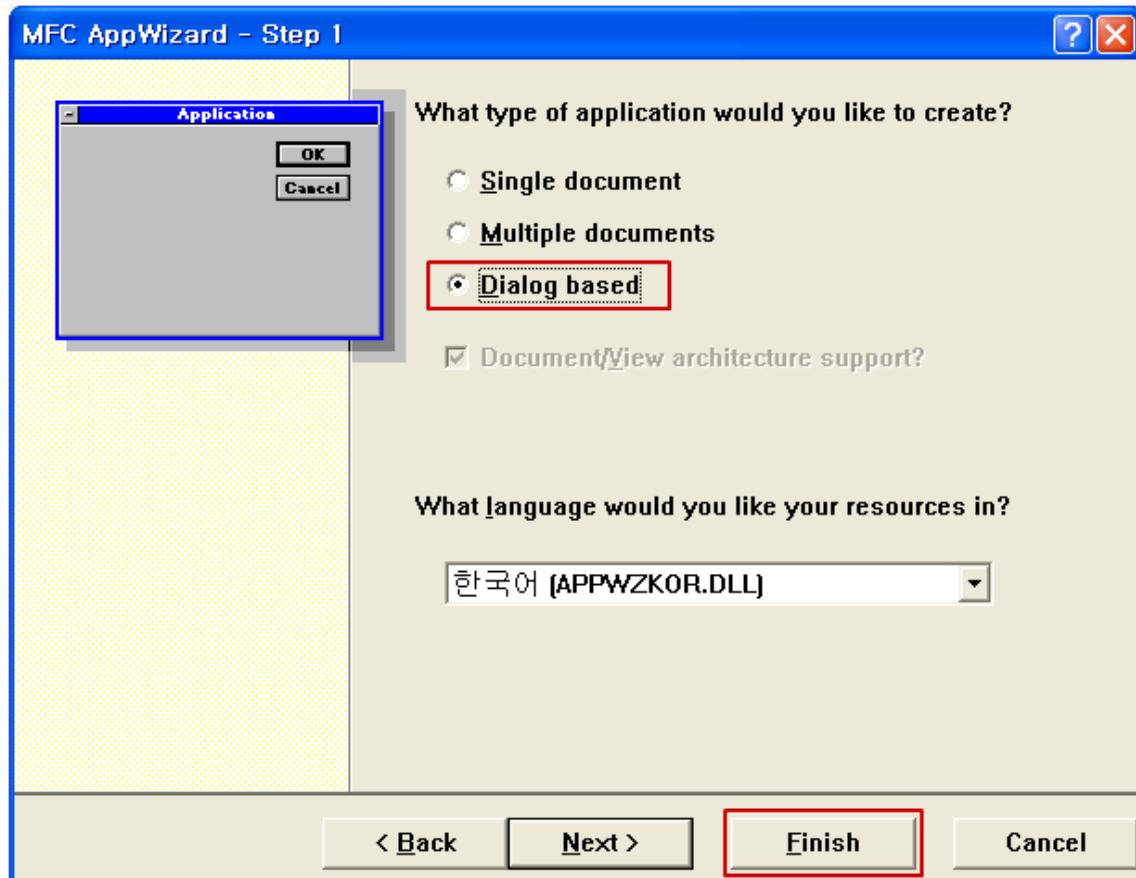
A. 애플리케이션 템플릿으로 [Project] 탭의 "MFC AppWizard (exe)"를 선택합니다.

B. 프로젝트 이름과 경로를 지정합니다.

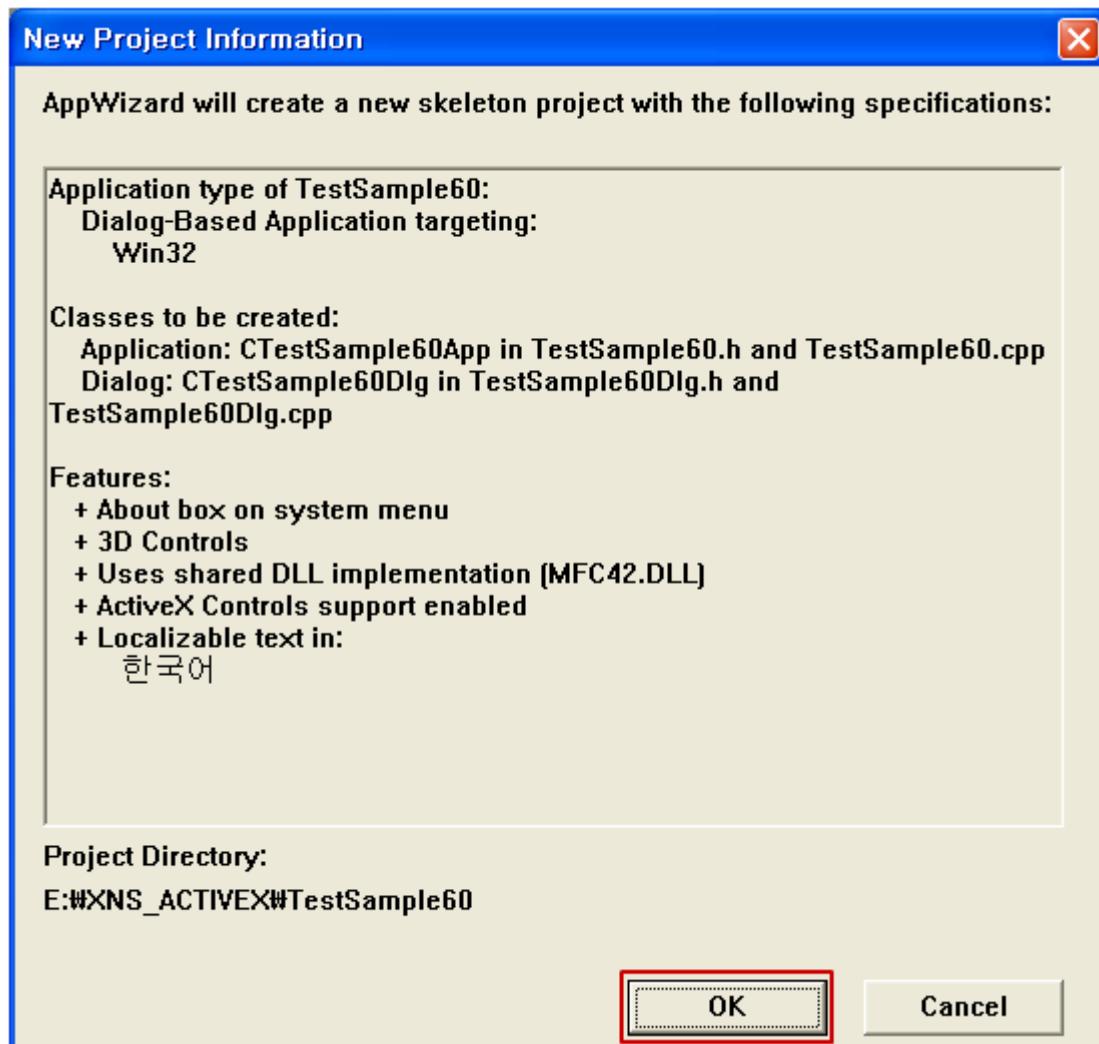
C. [OK]를 클릭합니다.



Step 4. 애플리케이션 타입으로 "Dialog based"를 선택하고 [Finish]를 클릭합니다.



Step 5. [New Project Information] 다이얼로그에서 애플리케이션 설정 정보를 확인하고 [OK] 버튼을 클릭합니다.



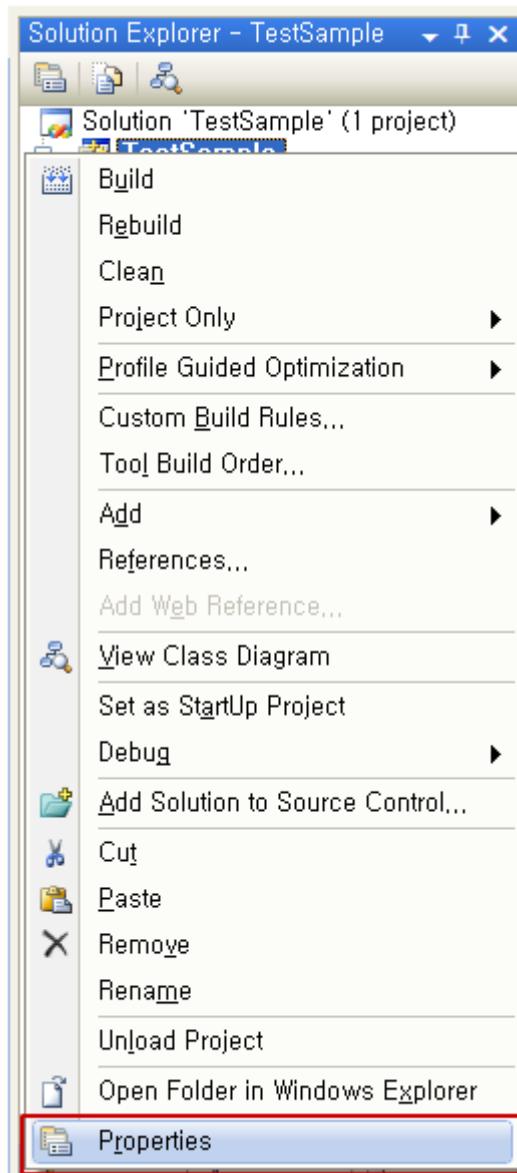
프로젝트 설정

애플리케이션을 작성한 후, 빌드 하기 위해서는 프로젝트의 프로퍼티(property)를 설정해야 합니다. Visual Studio 2008과 Visual Studio 6.0에서 프로젝트 프로퍼티를 설정하는 방법은 아래와 같습니다.

Visual Studio 2008

절차

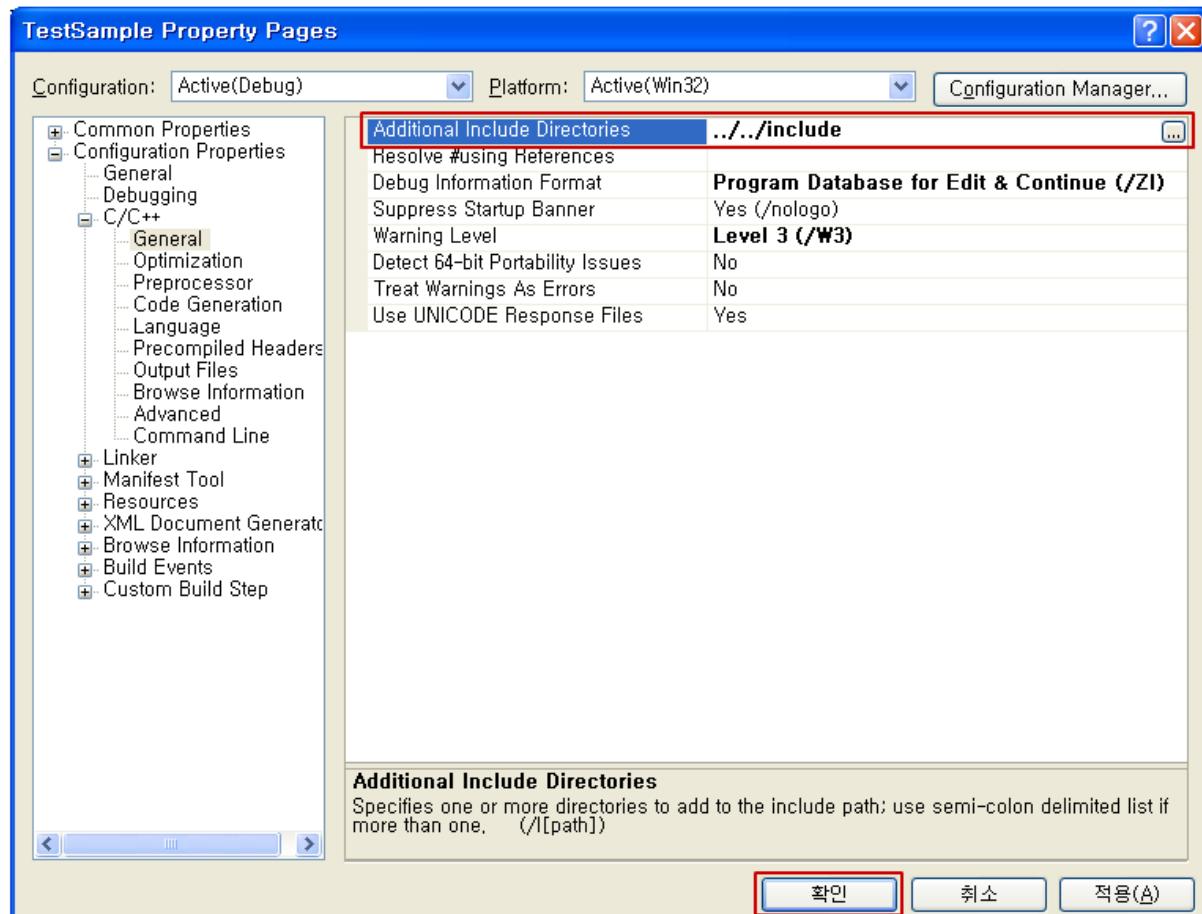
- Step 1. [Solution Explorer]에서 프로젝트를 선택한 후, 마우스 오른쪽 버튼을 클릭하여 [Properties] 메뉴를 선택합니다.



- Step 2. 왼쪽 트리 메뉴에서 [C/C++] > [General]을 선택한 후, [Additional Include Directories] 필드에 헤더 파일의 폴더 경로를 추가합니다. [확인]을 클릭합니다.

참고

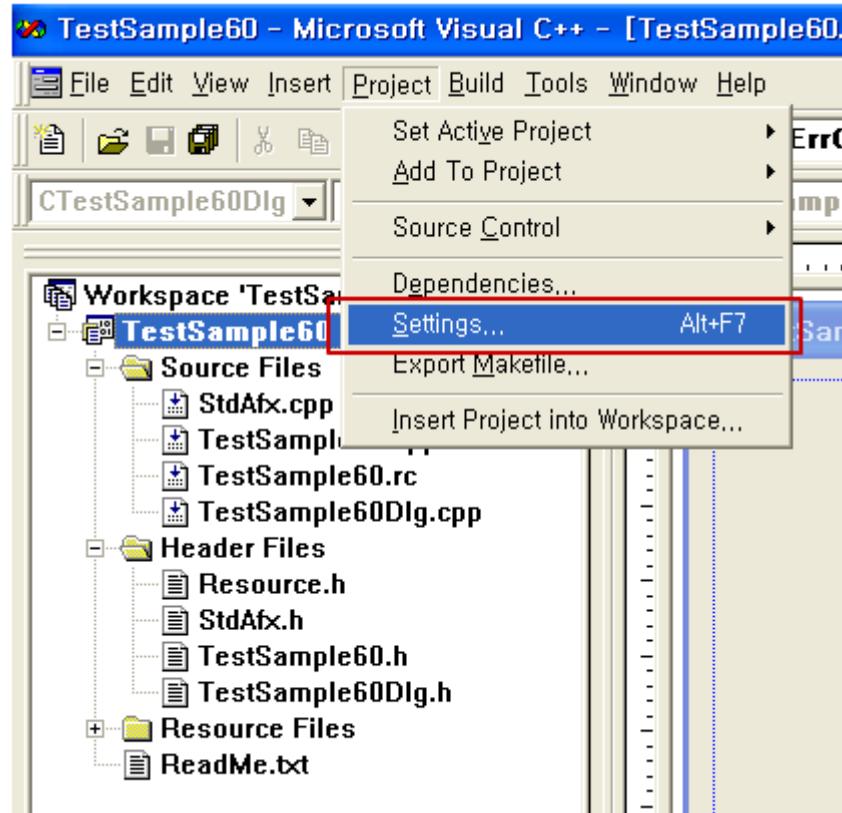
헤더 파일 경로를 변경하지 않고 기본 경로로 설치한 경우, 헤더 파일의 위치는 C:\Program Files\SAMSUNG\XnsActivexSDK\sample_code\include 입니다.



Visual Studio 6.0

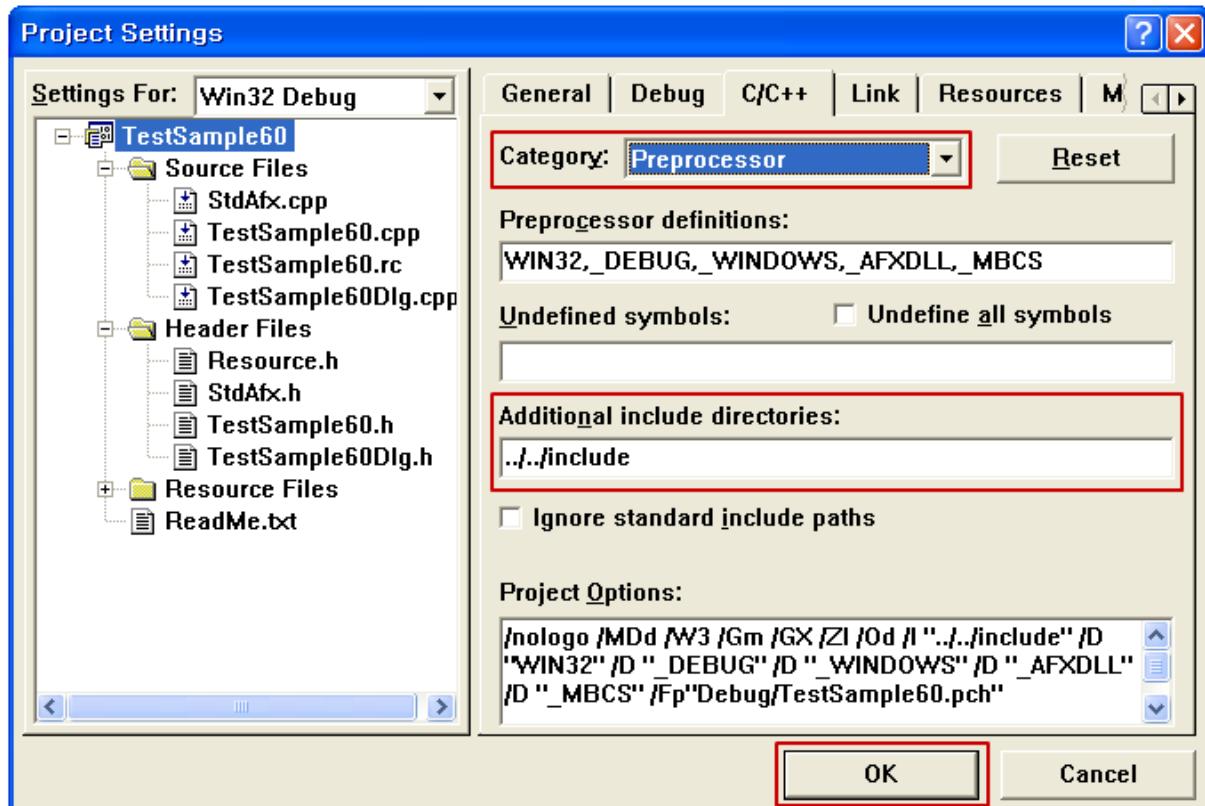
절차

Step 1. [Project] > [Settings...] 메뉴를 선택합니다.



Step 2. [C/C++] 탭에서,

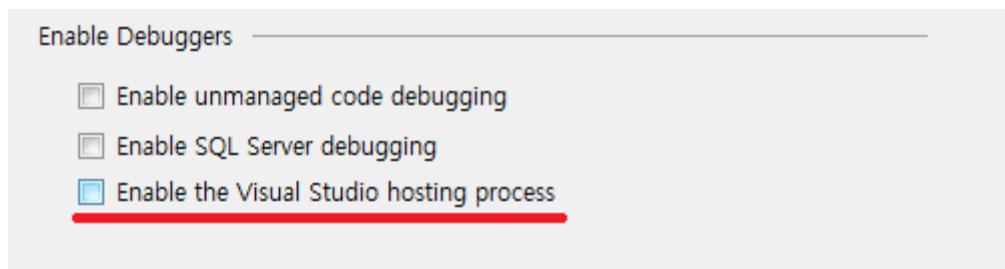
- A. [Category]는 "Preprocessor"를 선택합니다.
- B. [Additional include directories] 필드에 헤더 파일의 폴더 경로를 지정합니다.
- C. [OK]를 클릭합니다.



XP 이상 OS에서 C#, Visual Basic Debug 문제 해결 방법

절차

[properties] – [Debug] 탭에서, 만약 하단의 그림과 같이 Enable the Visual Studio hosting process 부분이 체크가 되어있다면 해제.



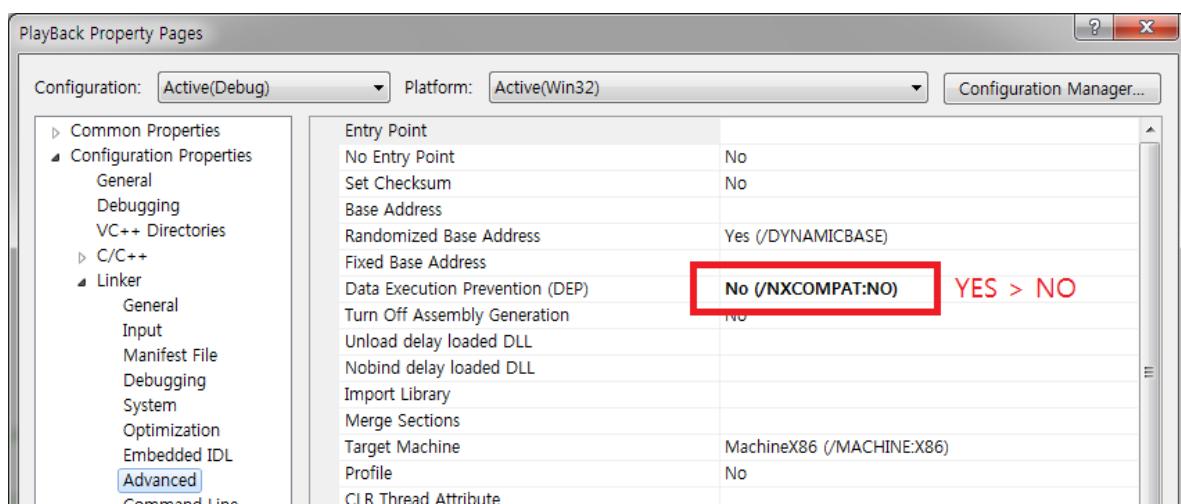
DEP 문제 해결 방법

DEP(데이터 실행 방지)는 프로그램을 모니터링하여 프로그램이 시스템 메모리를 안전하게 사용하게 함으로써 바이러스 및 다른 보안 위험으로부터 컴퓨터가 손상되는 것을 방지해 줄 수 있는 Windows의 보안 기능입니다. DEP 기능을 설정으로 인하여 SDK 프로그램이 올바르게 실행되지 않을 수 있습니다. 이 경우에는 다음과 같이 개발환경에서 직접 DEP를 해제할 수 있습니다.

C++ 프로젝트 설정

절차

- Step 1. [Solution Explorer]에서 프로젝트를 선택한 후, 마우스 오른쪽 버튼을 클릭하여 [Properties] 메뉴를 선택합니다. (37page 그림 참고.)
- Step 2. [Configuration Properties] > [Linker] > [Advanced] 메뉴를 선택합니다.
- Step 3. [Advanced] 메뉴에서 Data Execution Prevention(DEP) 항목이 YES로 되어 있다면, No로 변경합니다.



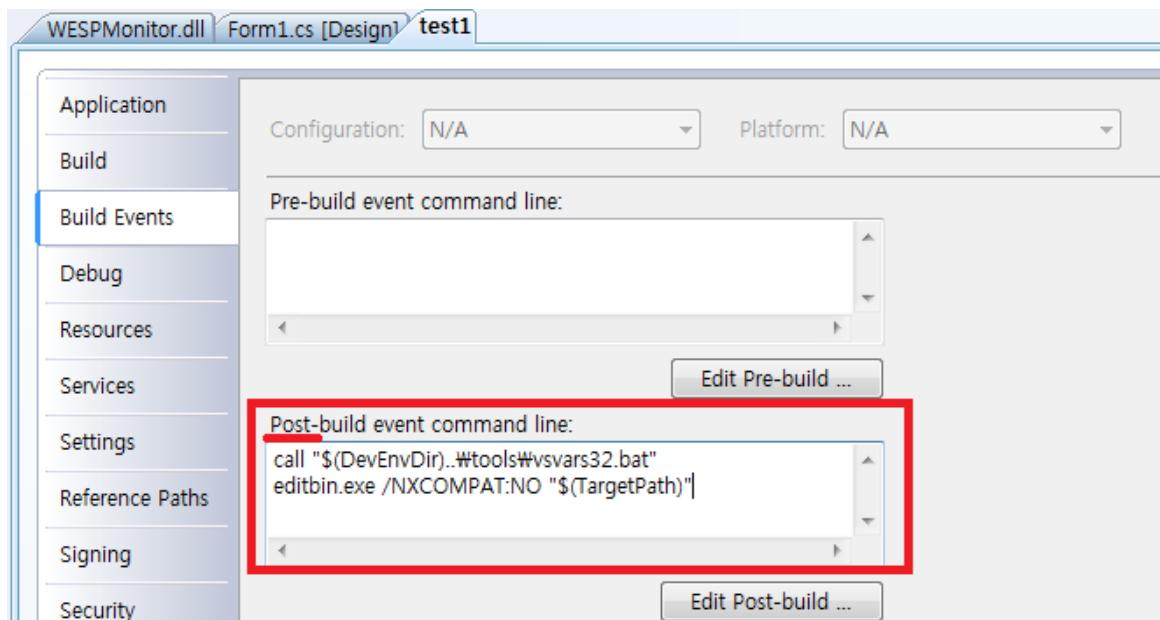
C# 프로젝트 설정

절차

- Step 1. [Solution Explorer]에서 프로젝트를 선택한 후, 마우스 오른쪽 버튼을 클릭하여 [Properties] 메뉴를 선택합니다.

Step 2. [Build Events] 탭을 선택한 후, 아래와 같이 명령어를 입력합니다.

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

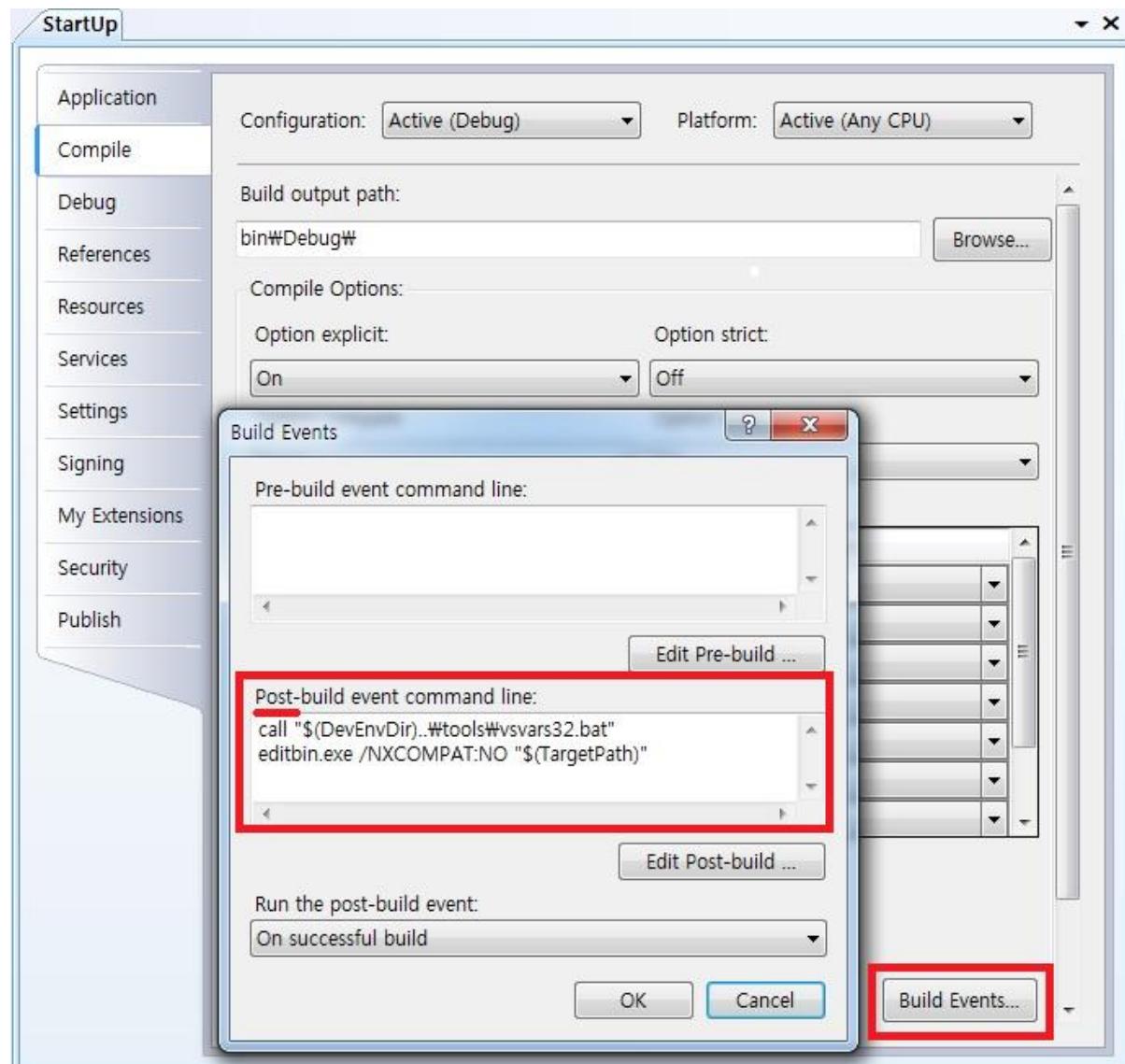


Visual Basic 프로젝트 설정

절차

- Step 1. [Solution Explorer]에서 프로젝트를 선택한 후, 마우스 오른쪽 버튼을 클릭하여 [Properties] 메뉴를 선택합니다.
- Step 2. [Compile] 탭을 선택한 후, [Build Events] 버튼을 클릭하여, 아래와 같이 명령어를 입력합니다.

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



CHAPTER 3

샘플 프로그램 소개

이 장에서는 XNS ActiveX SDK가 제공하는 샘플 프로그램을 소개합니다. 샘플 프로그램은 XNS ActiveX 라이브러리를 사용하여 사용자가 구현할 수 있는 프로그램의 예시를 보이기 위해 제공합니다.

Contents

위치

샘플 프로그램 목록

위치

샘플 프로그램은 {\$SDK path}\sample_code 폴더에서 확인하실 수 있습니다.

샘플 프로그램 목록

표 2 샘플 프로그램 목록

Sample Program	Description
Start Up	ActiveX XNS 객체를 삽입하는 방법을 설명하기 위한 예제
Single Live	1 채널 라이브 영상을 화면에 출력하는 방법을 설명하기 위한 예제
SingleLiveStream	1 채널 라이브 영상을 화면에 출력하는 방법을 설명하기 위한 예제
Multiple Connect	여러 장비의 라이브 영상을 화면에 출력하는 방법을 설명하기 위한 예제
Playback	녹화된 영상을 재생하는 방법을 설명하기 위한 예제
PTZ	카메라의 팬/틸트/줌을 제어하는 방법을 설명하기 위한 예제
Alarm & Event	이벤트 알람을 제어하는 방법을 설명하기 위한 예제
Local Recording	라이브 영상을 로컬 PC에 녹화하는 방법을 설명하기 위한 예제
Video Raw Data	Video Raw Data를 가지고 오는 방법을 설명하기 위한 예제
Backup	장비에 녹화된 데이터를 로컬 PC에 백업하는 방법을 설명하기 위한 예제
Listen & Talk	애플리케이션과 장비 간에 오디오 데이터를 송수신하는 방법을 설명하기 위한 예제
SdkTest	XNS ActiveX SDK의 전반적인 기능을 확인하고 테스트해 볼 수 있는 예제 프로그램

CHAPTER 4

Start Up 샘플 프로그램

Start Up 샘플 프로그램은 ActiveX XNS 객체를 삽입하는 방법을 설명하기 위한 예제입니다.

Contents

[샘플 프로그램 소개](#)

[API 호출 순서](#)

[구현 방법 상세 설명](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

Start Up 샘플 프로그램은 XnsSdkDevice 컨트롤과 XnsSdkWindow 컨트롤이 삽입된 디자인으로 그 박스를 실행합니다. XnsSdkDevice 컨트롤은 네트워크 장비 제어를 위해 획득하며, XnsSdkWindow 컨트롤은 사용자 화면에 영상을 출력하기 위해 획득합니다.

샘플 프로그램 사용법은 다음과 같습니다.

Step 1. 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.

Step 2. XnsSdkWindow 컨트롤이 삽입된 디자인로그 박스가 출력되는지 확인합니다.

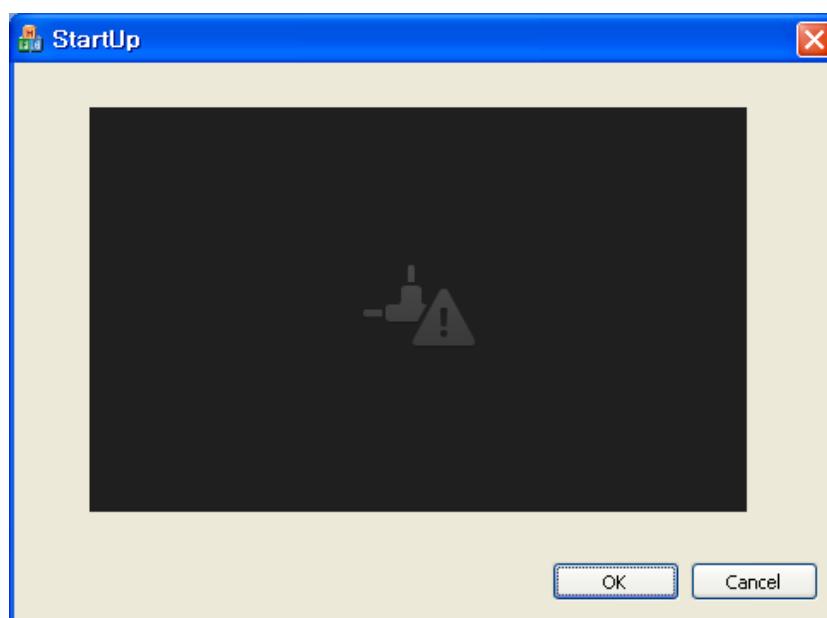


그림 2 Start Up 샘플 프로그램 실행 화면

Step 3. 프로그램을 종료합니다.

API 호출 순서

ActiveX XNS 컨트롤을 삽입하기 위해서 호출하는 API는 각 컨트롤의 초기화 함수입니다. 다음의 순서와 같이 XnsSdkDevice 컨트롤을 먼저 초기화한 후에 XnsSdkWindow 컨트롤을 초기화합니다.

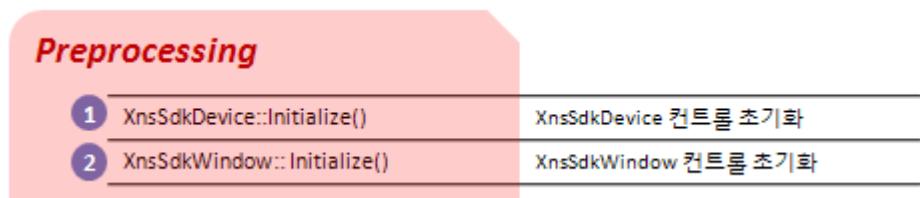


그림 3 Start Up 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 StartUp 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 삽입

[C++] ActiveX 컨트롤 삽입

절차

Visual Studio 2008에서 디자인로그를 생성한 후, 해당 디자인로그에 ActiveX 컨트롤을 삽입하고 변수를 추가합니다.

- Step 1. 디자인로그를 생성합니다.
- Step 2. 생성한 디자인로그에서 오른 클릭을 하면 선택창이 나타나며, 여기서 [Insert ActiveX Control] 버튼을 선택합니다.

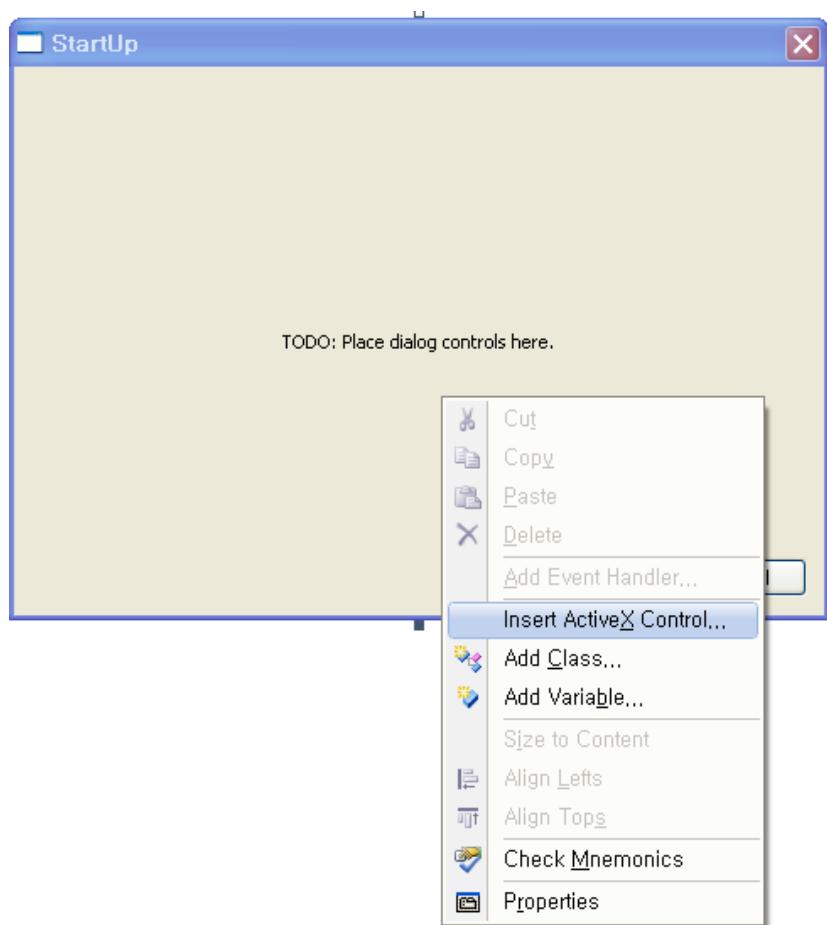
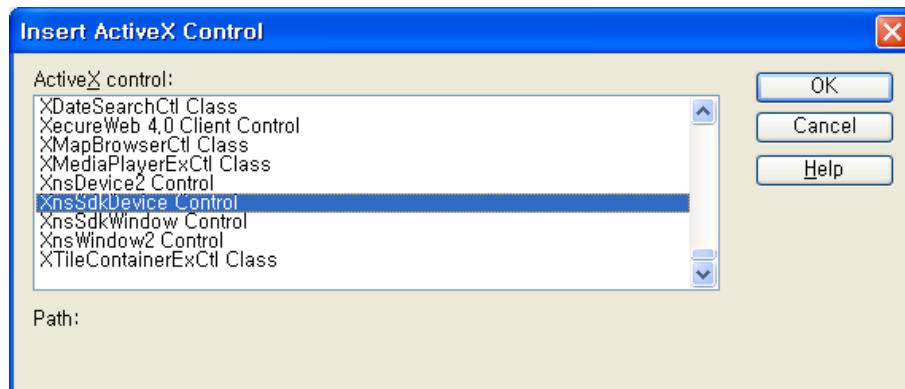


그림 4 ActiveX 컨트롤 삽입 절차(1)

Step 3. 추가할 ActiveX를 선택할 수 있는 팝업창이 나타나면 'XnsSdkDevice Control'과 'XnsSdkWindow Control'을 선택합니다.



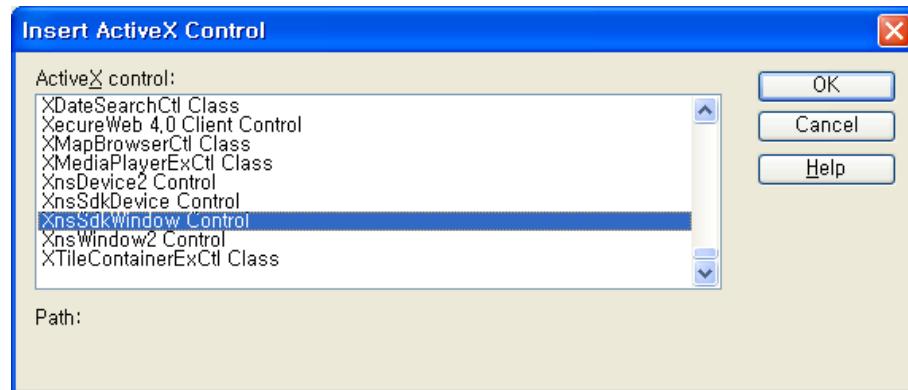


그림 5 ActiveX 컨트롤 삽입 절차(2)

Step 4. 추가가 완료되면 해당 디자인로그에 컨트롤이 배치됩니다.

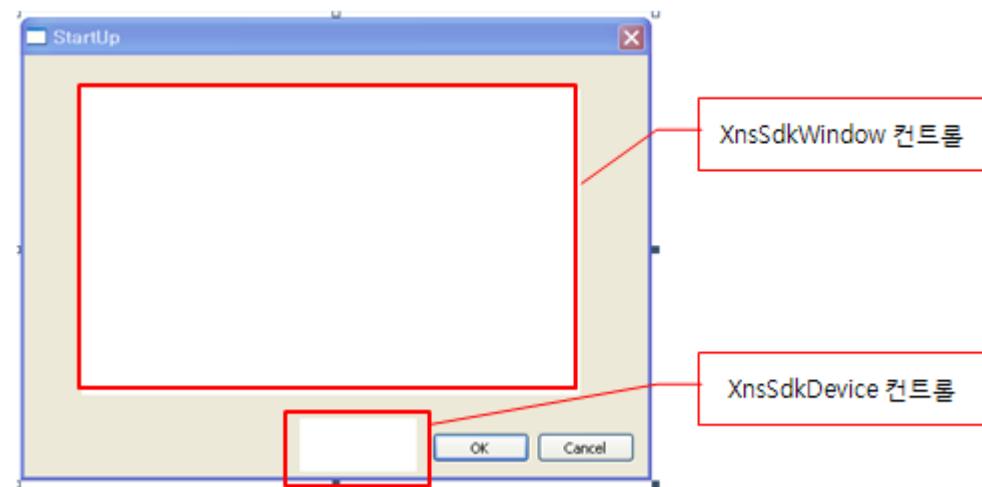


그림 6 ActiveX 컨트롤 삽입 결과

Step 5. XnsSdkDevice 컨트롤 변수를 추가합니다. 디자인로그에 배치된 컨트롤을 선택한 후 마우스 오른쪽 버튼을 누릅니다. 선택창에서 [Add Variable]을 선택합니다.

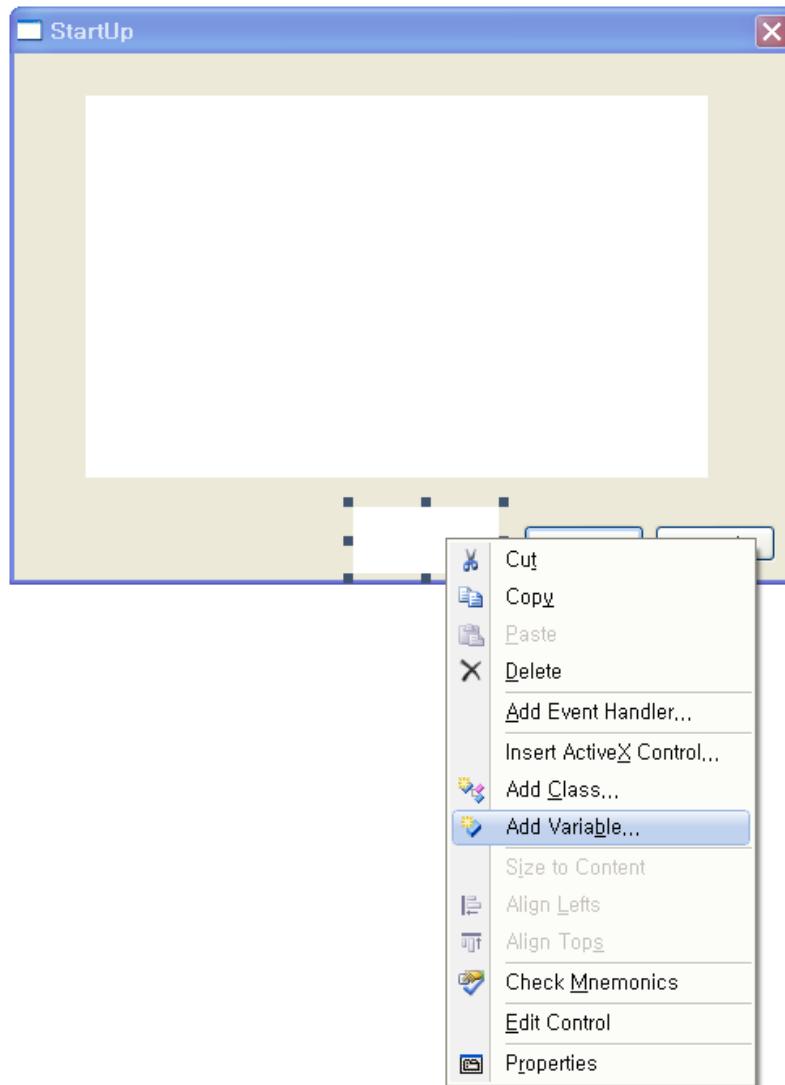


그림 7 컨트롤 변수 추가 절차(1)

Step 6. [Add Member Variable Wizard] 팝업창에서 각 필드를 채워 넣습니다.

- Access: 추가하려는 멤버 변수의 접근자를 선택합니다.
- Variable type: 멤버 변수의 타입을 선택합니다. XnsSdkDevice 변수일 경우 CXnssdkdevicectrl을 선택합니다.
- Control variable: 체크 박스를 선택합니다.
- Control ID: IDC_XNSSDKDEVICECTRL을 선택합니다.
- Control type: OCX를 선택합니다.
- Category: control을 선택합니다.
- .h file: xnssdkdevicectrl.h를 선택합니다. (원하는 이름으로 변경 가능)
- .cpp file: xnssdkdevicectrl.cpp를 선택합니다. (원하는 이름으로 변경 가능)

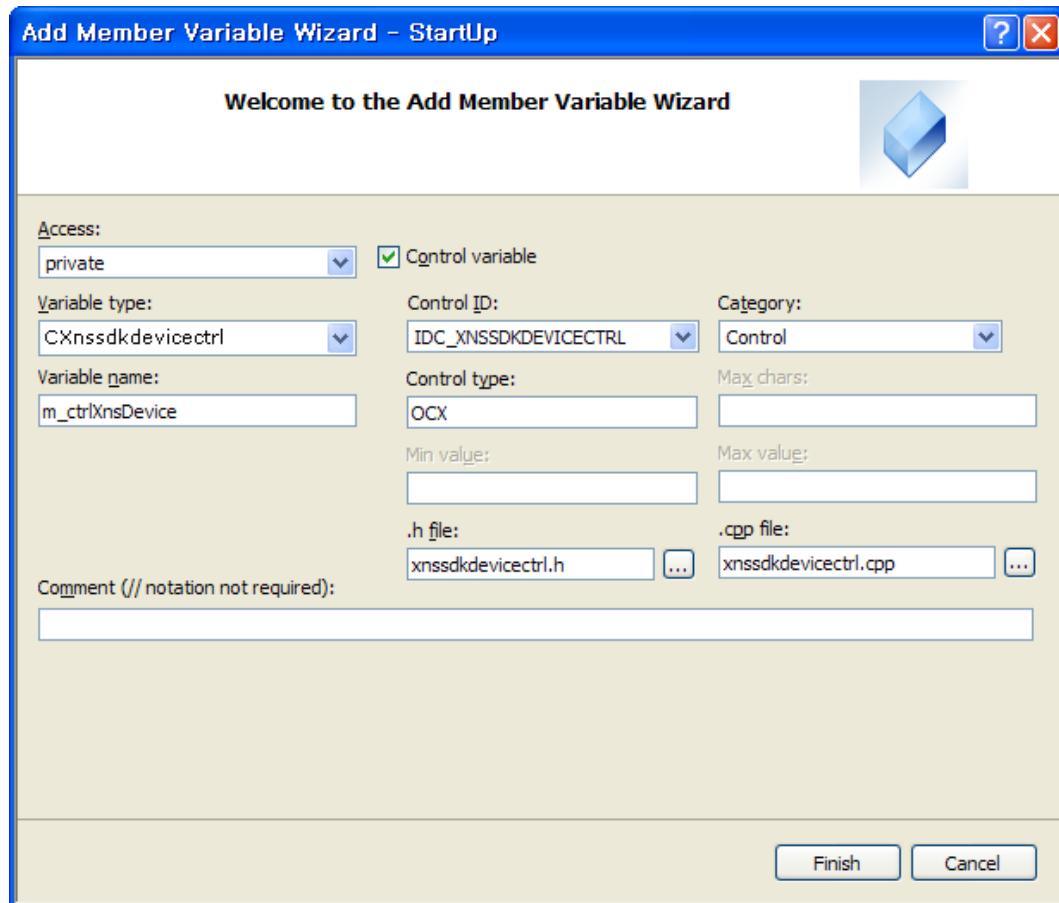


그림 8 컨트롤 변수 추가 절차(2)

참고

Visual Studio 6.0에서 컨트롤 변수를 추가하려면 [ClassWizard]를 선택합니다. [MFC ClassWizard] 화면에서 컨트롤ID를 선택한 다음, 클래스 정보를 입력합니다. 그리고 나서 해당 클래스의 멤버 변수를 추가합니다.



Step 7. CDialog 클래스에 컨트롤 변수가 정상적으로 추가되었는지 확인합니다.

```
// StartUpDlg.h
class CStartUpDlg : public CDialog
{
// Construction
```

```

public:
    CStartUpDlg(CWnd* pParent = NULL); // standard constructor
    //종략

private:
    CXnssdkdevicectrl m_ctrlXnsDevice; //XnsSdkDevice 컨트롤 변수 삽입
    CXnssdkwindowctrl m_ctrlXnsWindow; //XnsSdkWindow 컨트롤 변수 삽입
};

```

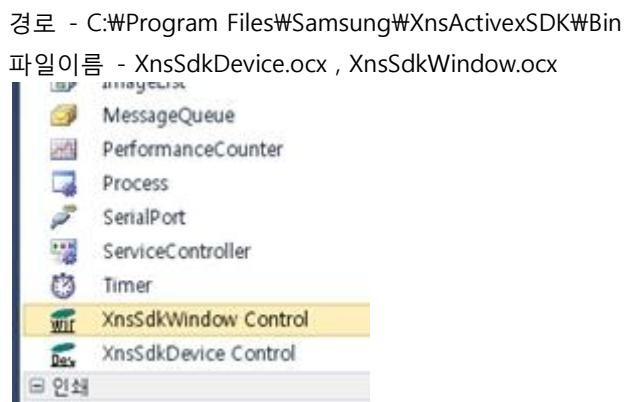
[C# Winform, Visual Basic] ActiveX 컨트롤 삽입

절차

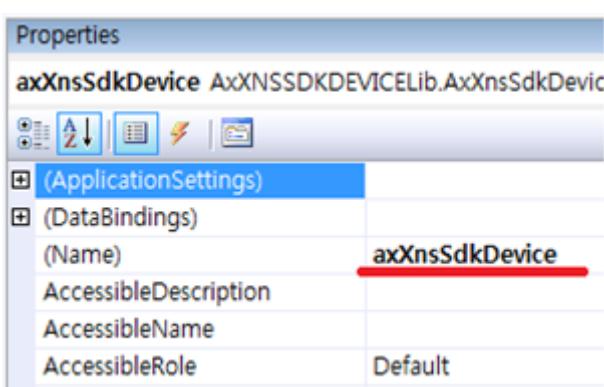
Visual Studio 2008에서 Windows Forms를 생성한 후, 해당 Windows Forms에 ActiveX 컨트롤을 삽입합니다.

Step 1. Windows Forms를 생성합니다.

Step 2. [view] – [ToolBox] 선택한 후 [ToolBox]에다가 OCX 파일을 Drag & Drop하여 추가합니다



Step 3. ActiveX 컨트롤의 Name이 XnsSdkWindow Control 변수명으로 사용됩니다. (기존 C#, Visual Basic 컨트롤들과 유사하게 사용하면 됩니다)

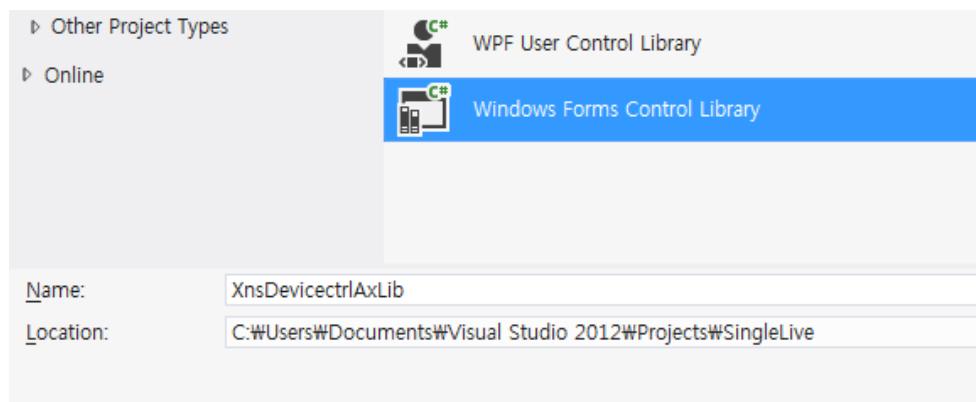


[C# WPF] ActiveX 컨트롤 삽입

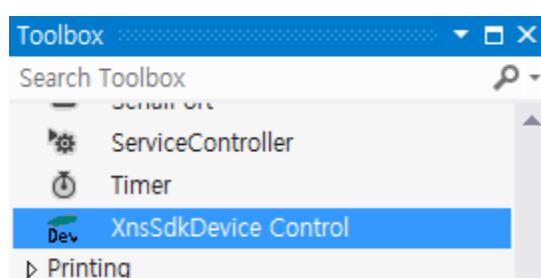
절차

Visual Studio 2012에서 WPF application을 생성한 후, ActiveX 컨트롤을 만들어 WPF 페이지에 호스팅합니다.

- Step 1. WPF application을 생성합니다. (SingleLive)
- Step 2. Window Forms 컨트롤 라이브러리 프로젝트를 솔루션에 추가하고 프로젝트의 이름을 XnsDevicectrlAxLib으로 지정합니다.



- Step 3. XnsDevicectrlAxLib 프로젝트에서 XnsSdkDevice.ocx를 참조 추가합니다.
경로 - C:\Program Files\Samsung\XnsActivexSDK\Bin
- Step 4. [view] – [ToolBox] 선택한 후 마우스 오른쪽 단추를 클릭한 다음 [Choose Toolbox Items]를 선택합니다. [COM Components] 탭을 클릭하고 XnsSdkDevice Control을 선택한 후 확인을 클릭합니다. XnsSdkDevice Control이 [ToolBox]에 추가됩니다.



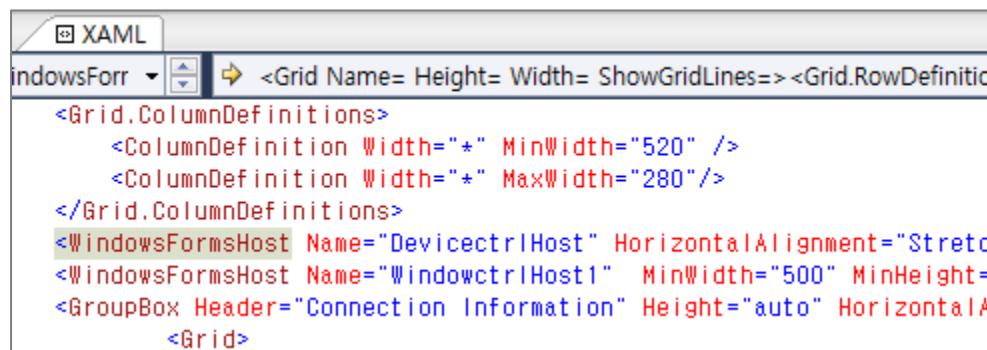
- Step 5. XnsDevicectrlAxLib 컨트롤 라이브러리 프로젝트에서 UserControl1.cs 파일명을 XnsDevicectrl.cs로 변경합니다. 모든 참조의 이름을 바꾸라는 메시지가 나타나면 예를 클릭합니다.
- Step 6. Windows Form 디자이너에서 XnsDevicectrl.cs를 엽니다. [Toolbox]에서 XnsSdkDevice Control을 XnsDevicectrl 디자인 화면에 추가하고, [Properties] – [Dock] 속성 값을 Fill로 설정합니다. XnsDevicectrlAxLib 컨트롤 라이브러리를 빌드합니다.
- Step 7. SingleLive 프로젝트에 생성된 ActiveX 상호 운용성 어셈블리에 대한 참조를 추가합니다.

경로 – XnsDevicectrlAxLib 프로젝트의 Debug 폴더

파일명 – AxInterop.XNSSDKDEVICELib.dll

- Step 8.** Singlelive 프로젝트에 WindowsFormsIntegration.dll과 System.Windows.forms.dll을 참조 추가합니다.
- Step 9.** MainWindow.xaml과 MainWindow.xaml.cs에 아래와 같은 코드를 작성합니다. 이 코드는 WindowsFormsHost 컨트롤의 인스턴스를 만들고 AxXnsSdkDevice 컨트롤의 인스턴스를 자식으로 추가합니다.

[MainWindow.xaml]



[MainWindow.xaml.cs]

```
AxXNSSDKDEVICELib.AxXnsSdkDevice AxXnsSdkDevice
    = new AxXNSSDKDEVICELib.AxXnsSdkDevice();
DevicectrlHost.Child = AxXnsSdkDevice;
int nError = AxXnsSdkDevice.Initialize();
m_DebugLog.WLOGD("Initialize():: ", nError, AxXnsSdkDevice.GetErrorString());
```

헤더 파일 추가

StartUpDlg.cpp 파일에 XNS ActiveX SDK 라이브러리를 사용하는데 필요한 헤더 파일을 추가합니다.

```
// StartUpDlg.cpp
#include "XnsCommon.h"
#include "XnsDeviceInterface.h"
```

컨트롤 초기화

XnsSdkDevice::initialize() 함수와 XnsSdkWindow::initialize()를 호출하여 각 컨트롤을 초기화합니다. XnsActiveX 라이브러리를 정상적으로 사용하기 위해서는 config.xml, device.xml, xns.xml 파일이 필요하고, xns.xml 파일에 DLL 파일의 경로가 명시되어 있어야 합니다. DLL 파일의 경로는

512 바이트를 넘을 수 없습니다. XnsActiveX 라이브러리는 기본적으로 {\$SDK path}\Config 경로에 설치된 XnsSDKDevice.ocx 파일을 사용해서 xns.xml 파일을 찾습니다.

CStartUpDlg 클래스의 OnInitDialog() 함수 안에서 initialize() 함수를 호출합니다.

```
// StartUpDlg.cpp
long nRet = m_ctrlXnsDevice.Initialize();
nRet = m_ctrlXnsWindow.Initialize(NULL, NULL);
```

See also

Samsung XNS ActiveX API Reference

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.

현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 5

Single Live 샘플 프로그램

Single Live 샘플 프로그램은 XNS ActiveX SDK를 사용하여 1채널 라이브 영상을 화면에 출력하는 방법을 설명하기 위한 예제입니다.

Contents

[샘플 프로그램 소개](#)

[API 호출 순서](#)

[구현 방법 상세 설명](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

Single Live 샘플 프로그램은 특정 네트워크 장비로부터 1채널 라이브 영상을 수신하여 화면에 출력하는 예제입니다. 이 작업을 위해 먼저 XnsSdkDevice 컨트롤과 XnsSdkWindow 컨트롤을 생성합니다. 컨트롤 생성 후에는 네트워크 장비간 연결을 시도하고, 연결 성공 시 라이브 영상 스트림을 받아 화면에 출력합니다. 라이브 영상 수신이 종료되면 장비와 연결을 끊고 리소스를 해제합니다.

샘플 프로그램 사용법은 다음과 같습니다.

- Step 1.** 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.
- Step 2.** 장비 모델, IP address, VNP 포트, ID/Password 등의 장비 연결 정보를 설정한 후 [Connect] 버튼을 누릅니다.

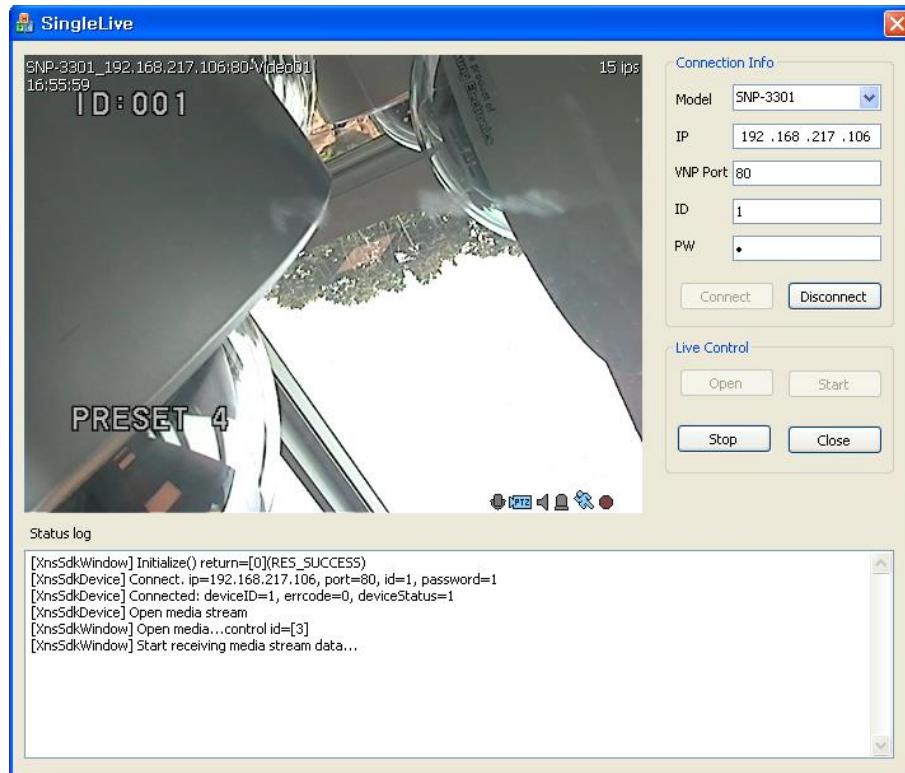


그림 9 Single Live 샘플 프로그램 실행 화면

- Step 3.** 연결이 정상적으로 이루어졌는지 확인합니다.
- Step 4.** [Open] 버튼을 클릭하여 라이브 영상 수신을 시작합니다.
- Step 5.** [Start] 버튼을 클릭하여 라이브 영상을 화면에 출력합니다. 정상적으로 출력되는지 확인합니다.

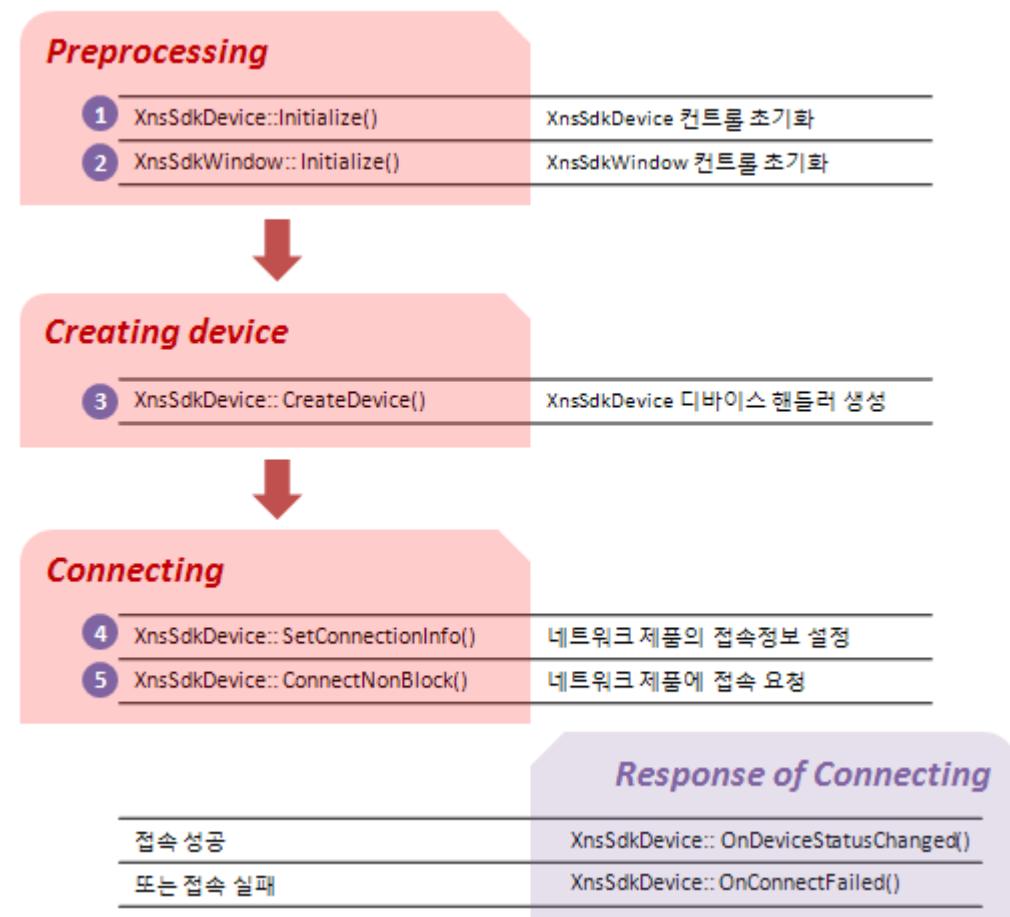
Step 6. [Stop] 버튼을 클릭하여 라이브 영상 출력이 중지되는지 확인합니다.

Step 7. [Close] 버튼을 클릭하여 라이브 영상 수신을 종료합니다.

Step 8. 프로그램을 종료합니다.

API 호출 순서

이 샘플 프로그램은 XNS ActiveX SDK가 제공하는 라이브러리를 사용하여 1채널 라이브 영상 출력 기능을 구현하였습니다. 샘플 프로그램의 API 호출 순서는 다음과 같습니다.



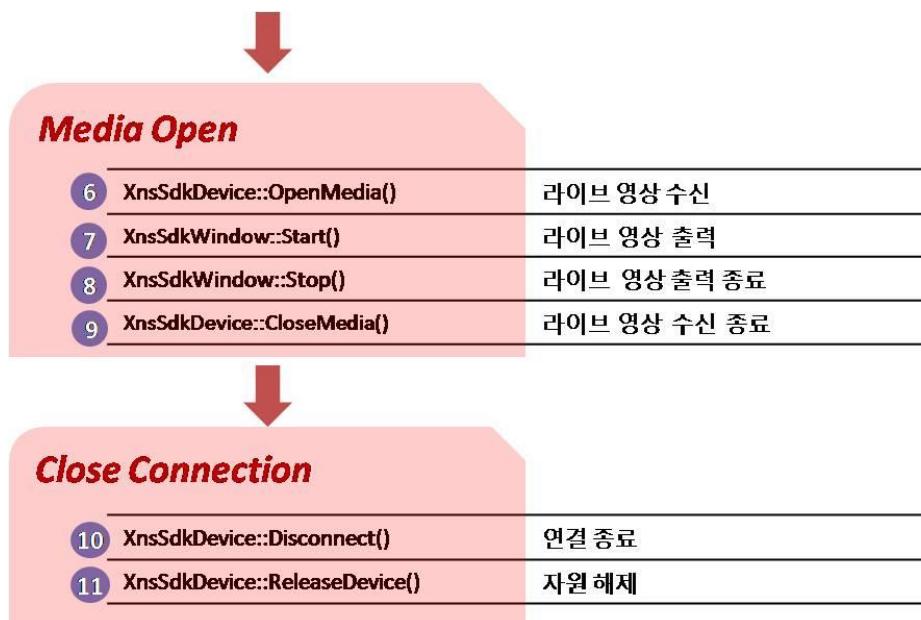


그림 10 Single Live 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 Single Live 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 생성

참고

ActiveX 컨트롤 삽입 및 초기화 방법은 본 문서의 CHAPTER 4에 기술된 Start Up 샘플 프로그램의 '구현 방법 상세 설명' 절을 참조합니다.

디바이스 핸들러 생성

장비 제어를 위한 디바이스 핸들러를 생성합니다. 디바이스 핸들러를 생성하면, 장비의 정보를 저장하기 위한 메모리 공간이 할당되고, 디바이스 핸들(device handle)이 반환됩니다. 반환되는 값이 0보다 큰 정수일 경우에 유효합니다.

디바이스 핸들러는 CSingleLiveDlg 클래스의 OnInitDialog() 함수 안에서 CreateDevice()를 호출

함으로써 생성됩니다. OnInitDialog() 함수는 프로그램 시작 시 자동으로 호출되는 함수입니다. CreateDevice() 함수는 XnsSdkDevice 컨트롤에 존재하는 함수입니다.

```
// SingleLiveDlg.cpp
m_hDevice = m_ctrlXnsDevice.CreateDevice(1);
```

디바이스 연결 설정

장비에 접속하기 전에 XnsSdkDevice 컨트롤의 SetConnectionInfo() 함수를 호출하여 디바이스 연결 정보를 설정해야 합니다. 두 번째 파라미터인 szVendorName에는 "Samsung"이라는 고정 값을 입력하고, 포트 번호 파라미터에는 VNP 포트 번호를 지정합니다.

```
// SingleLiveDlg.cpp
m_ctrlXnsDevice.SetConnectionInfo(
    m_hDevice,           // [in] Device handle
    _T("Samsung"),       // [in] Fixed as 'Samsung'
    m_strModelName,      // [in] Name of model to connect to.
    XADDRESS_IP,         // [in] Address type
    strIpAddress,        // [in] actual address according to address type.
    m_nPort,             // [in] Port number
    0,                  // [in] Port number for web access
    m_strId,             // [in] Login ID
    m_strPasswd          // [in] Password
);
```

이후 장비에 접속하기 위해서 XnsSdkDevice 컨트롤의 ConnectNonBlock() 함수를 호출합니다. 이 함수는 non-blocking 방식으로 동작하기 때문에 접속이 완료되지 않아도 바로 리턴됩니다.

```
// SingleLiveDlg.cpp
long nRet = m_ctrlXnsDevice.ConnectNonBlock(
    m_hDevice,           // [in] Device handle
    TRUE,                // [in] Flag to decide where to forcibly log in or not.
    TRUE                 // [in] If this value is 1, try to connect again.
);
if(nRet != ERR_SUCCESS)
{
    WLOGD(_T("ConnectNonBlock() fail: errno=[%d](%s)\\n"),
        nRet, m_ctrlXnsDevice.GetErrorString(nRet));
}
```

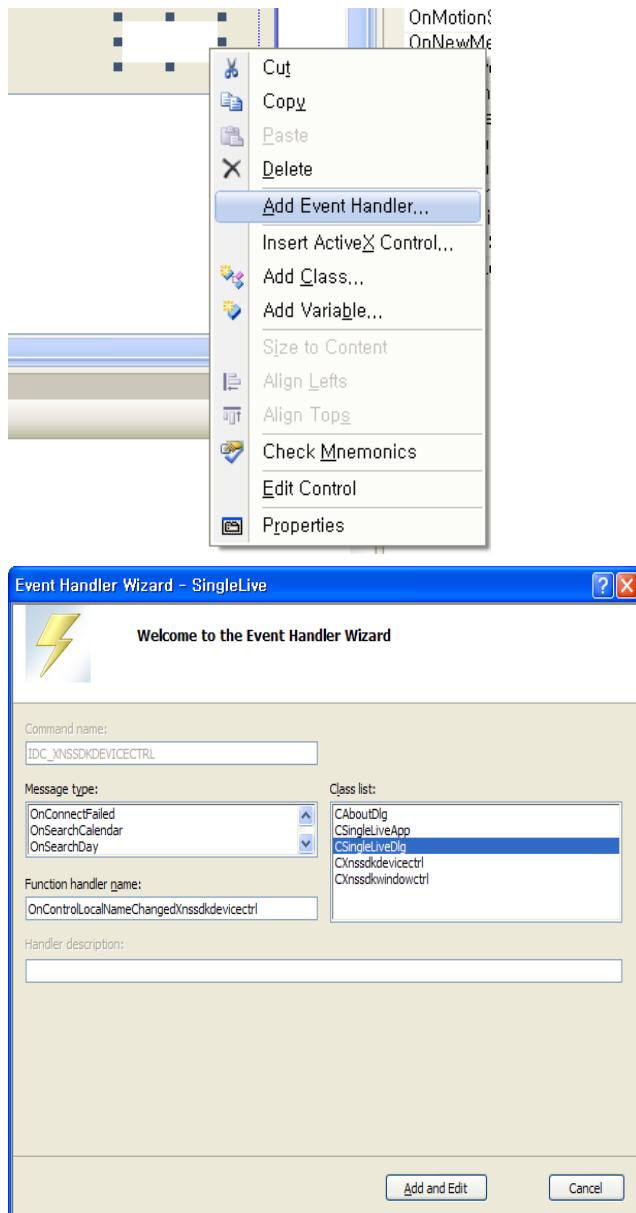
접속 결과는 이벤트를 통해 알 수 있으며, 접속 성공 시에는 OnDeviceStatusChanged 이벤트가 발생하고, 접속 실패 시에는 OnConnectFailed 이벤트가 발생합니다.

이벤트 핸들러 구현

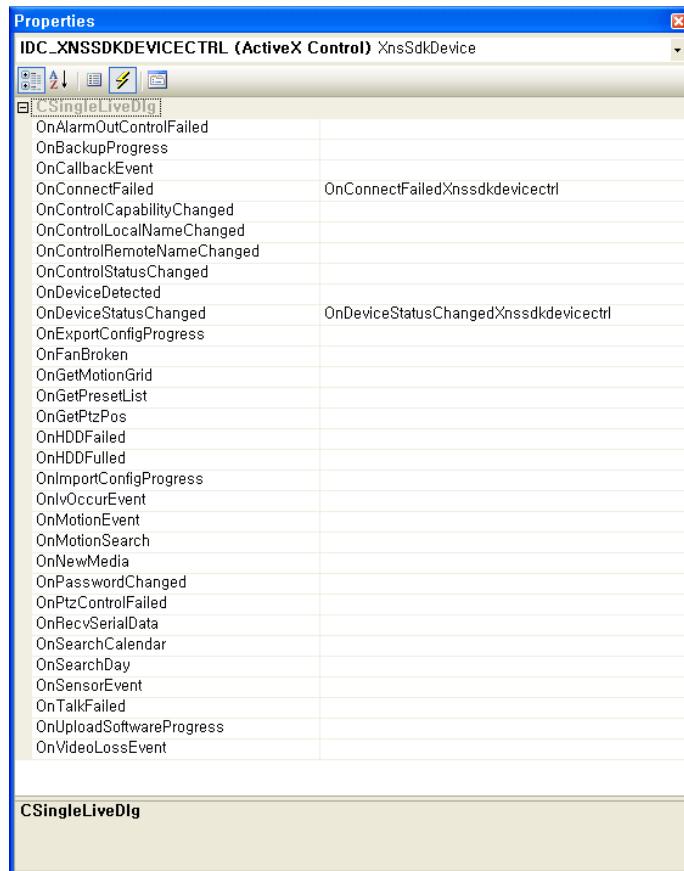
[C++] 이벤트 핸들러 추가 및 구현

장비 접속 결과를 처리하기 위해서 해당 이벤트를 선언하고 이벤트 핸들러를 구현해야 합니다.

리소스 편집 화면에서 XnsSdkDevice 컨트롤을 선택한 후 이벤트 핸들러를 추가할 수 있습니다.



또는 XnsSdkDevice 컨트롤의 Properties 창에서 Control Events 버튼을 선택하면 원하는 이벤트들을 삽입할 수 있습니다.



CSingleLiveDlg 클래스에 OnDeviceStatusChangedXnssdkdevicectrl()와 OnConnectFailedXnssdkdevicectrl() 이벤트 핸들러를 작성합니다.

```
// SingleLiveDlg.cpp
void CSingleLiveDlg::OnDeviceStatusChangedXnssdkdevicectrl(long nDeviceID, long nErrorCode,
long nDeviceStatus, long nHddCondition)
{
    if (nErrorCode != ERR_SUCCESS || nDeviceStatus != 1)
    {
        return;
    }

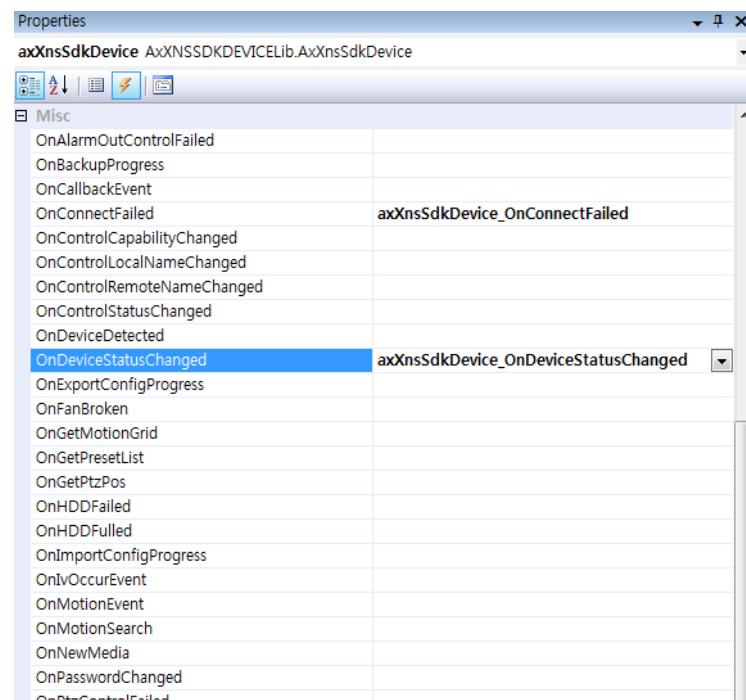
    WLOGD(_T("Connected: deviceID=%d, errcode=%d, deviceStatus=%d\n"),
nDeviceID, nErrorCode, nDeviceStatus);
    setBtnStatus(SL_STATUS_CONNECTED);
}
```

```
// SingleLiveDlg.cpp
void CSingleLiveDlg::OnConnectFailedXnssdkdevicectrl(long nDeviceID, long nErrorCode)
```

```
{
    WLOGD(_T("Disconnected: deviceID=%d, errcode=[%d](%S)\n"),
          nDeviceID, nErrorCode, m_ctrlXnsDevice.GetErrorString(nErrorCode));
    setBtnStatus(SL_STATUS_DISCONNECTED);
}
```

[C# Winform, Visual Basic] 이벤트 핸들러

Windows Forms에서 XnsSdkDevice 컨트롤을 선택한 후 Properties 창에서 Events 버튼을 선택하여 원하는 이벤트를 삽입할 수 있습니다. C++에서와 같이 Add Event Handler는 지원하지 않습니다.

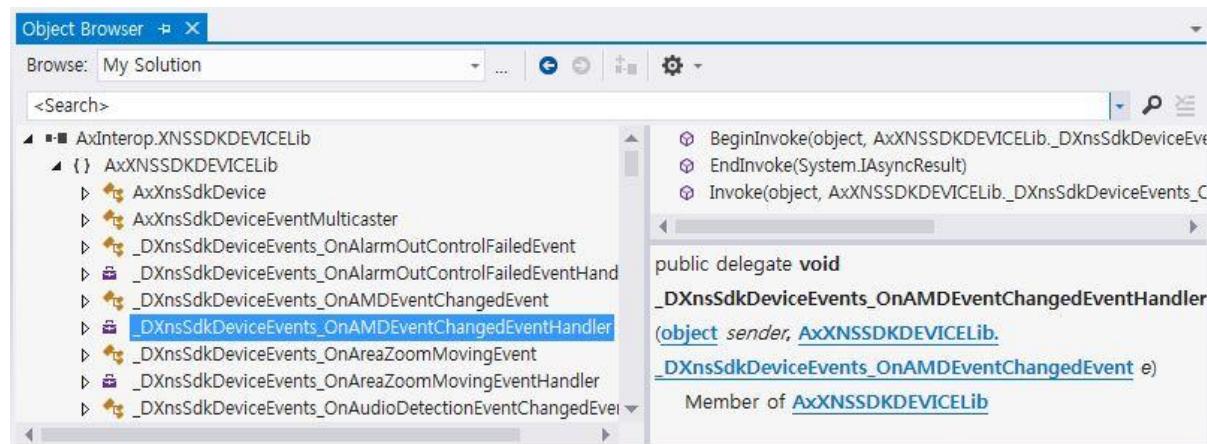


이벤트 핸들러를 추가할 경우, 아래와 같은 이벤트 선언문과 이벤트 맵 코드가 자동으로 소스에 추가됩니다. 개발자는 추가된 이벤트 핸들러 함수 내에서 애플리케이션 기능을 추가하면 됩니다.

[C# WPF] 이벤트 핸들러

사용자정의 delegate 함수를 오버라이딩 하여 원하는 이벤트핸들러를 삽입할 수 있습니다.

[Object Browser – 사용자정의 delegate 함수]



[이벤트 핸들러 작성]

```

MainWindow.xaml.cs* X
SingleLive.MainWindow
Main Window_Loaded(object sender, RoutedEventArgs e)
private void AxXnsSdkDevice_OnDeviceStatusChanged
(object sender, AxXNSSDKDEVICELib._DXnsSdkDeviceEvents_OnDeviceStatusChangedEvent e)
{
    m_DebugLog.WLOGD("OnDeviceStatusChanged() EVENT:: device_id=" + e.nDeviceID +
        ", status=" + e.nDeviceStatus + ", error=" + e.nErrorCode +
        "[" + AxXnsSdkDevice.GetErrorString(e.nErrorCode) + "]");
    if ((e.nErrorCode == (int)XErrorCode.ERR_SUCCESS) && (e.nDeviceStatus == 1))
    {
        setBtnStatus(SingleLive_ButtonStatus.SL_STATUS_CONNECTED);
        m_DebugLog.WLOGD("Connected...");
    }
}
  
```

미디어 오픈

디바이스와 PC간 데이터 송수신을 위한 미디어 소스 핸들을 획득한 후에, 이 핸들을 윈도우 컨트롤에 등록합니다.

미디어 소스 핸들을 획득하려면 XnsSdkDevice 컨트롤의 OpenMedia() 함수를 호출하여 미디어 스트림 수신을 시작합니다. OpenMedia() 함수의 파라미터(out-parameter)로 미디어 소스의 핸들(m_hMediaSource)을 얻을 수 있습니다.

XnsSdkWindow 컨트롤의 Start() 함수를 호출하여 미디어 소스 핸들을 윈도우 컨트롤 객체에

등록합니다. XnsSdkWindow 컨트롤은 해당 핸들의 미디어 스트림을 수신하여 디코딩한 후 화면에 출력합니다.

```
// SingleLiveDlg.cpp
if (m_ctrlXnsSdkDevice.GetControlType(m_hDevice, 1) & XCTL_DVR) // Device Type :DVR
{
    // XCTL_CAMERA : Camera channel of DVR.
    for(int i=0; i<m_ctrlXnsSdkDevice.GetControlCount(m_hDevice, XCTL_CAMERA); i++)
    {
        // Displays video in real time.
        if (m_ctrlXnsSdkDevice.GetControlCapability(m_hDevice, i+2, XCTL_CAP_LIVE) )
        {
            // Start getting media streams from the device.
            m_ctrlXnsSdkDevice.OpenMedia(m_hDevice,
                i+2,
                MEDIA_TYPE_LIVE,
                0,
                0,
                &m_hMediaSource);

            m_ctrlXnsSdkWindow.Start(m_hMediaSource);
            m_bPlay = TRUE;
            return;
        }
    }
}
```

버튼 이벤트 핸들러 작성

CSingleLiveDlg 클래스에 OnBnClickedButtonOpen 이벤트 핸들러를 작성합니다.

```
// SingleLiveDlg.cpp
void CSingleLiveDlg::OnBnClickedButtonClose()
{
    if( !m_hMediaSource )
    {
        WLOGD(_T("May be device is not opened\n"));
        return;
    }
```

```
if( m_bIsMediaPlay)
{
    // stop
    OnBnClickedButtonStop();
}

m_ctrlXnsDevice.CloseMedia(m_hDevice, m_hMediaSource);
WLOGW(_T("Close media...#\n"));
m_hMediaSource = 0;
m_nControlId = 0;
setBtnStatus(SL_STATUS_CLOSE);
}
```

See also

[Start Up 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.

현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 6

Multi channel 샘플 프로그램

Multi channel 샘플 프로그램은 XNS ActiveX SDK를 사용하여 다채널 라이브 영상을 화면에 출력하는 방법을 설명하기 위한 예제입니다.

Contents

[샘플 프로그램 소개](#)

[API 호출 순서](#)

[구현 방법 상세 설명](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

Multi Channel 샘플 프로그램은 특정 네트워크 장비로부터 최대 16개 채널의 라이브 영상을 수신하여 화면에 출력하는 예제입니다. 이 작업을 위해 먼저 XnsSdkDevice 컨트롤과 16개의 XnsSdkWindow 컨트롤을 생성합니다. 컨트롤 생성 후에는 네트워크 장비간 연결을 시도하고, 연결 성공 시 라이브 영상 스트림을 받아 화면에 출력합니다. 라이브 영상 수신이 종료되면 장비와 연결을 끊고 리소스를 해제합니다.

샘플 프로그램 사용법은 다음과 같습니다.

- Step 1.** 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.
- Step 2.** 장비 모델, IP address, VNP 포트, ID/Password 등의 장비 연결 정보를 설정한 후 [Connect] 버튼을 누릅니다.

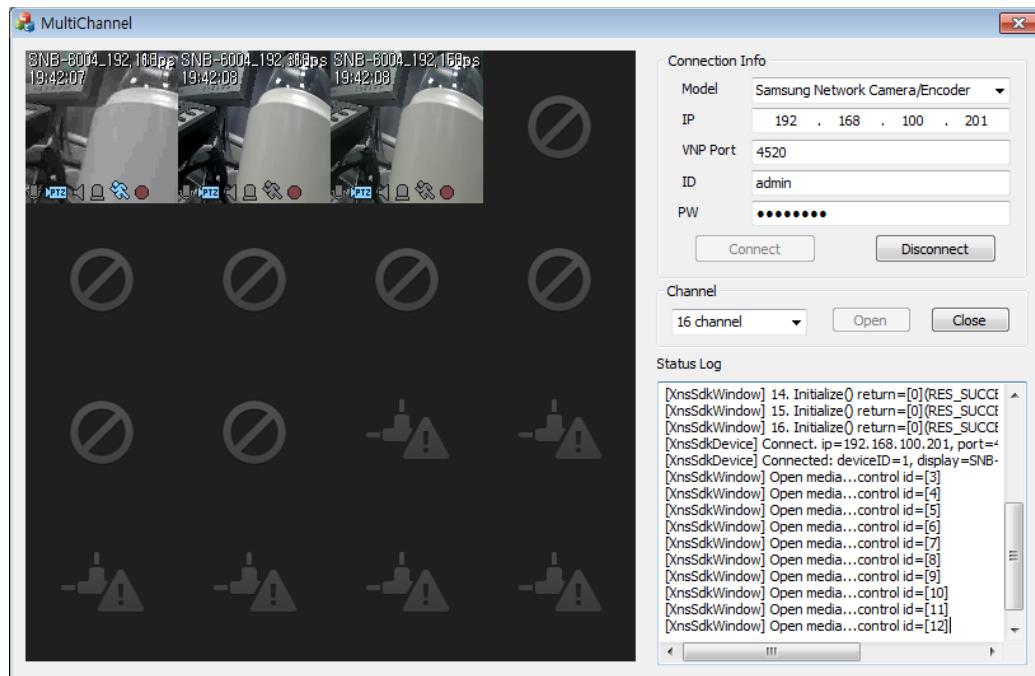
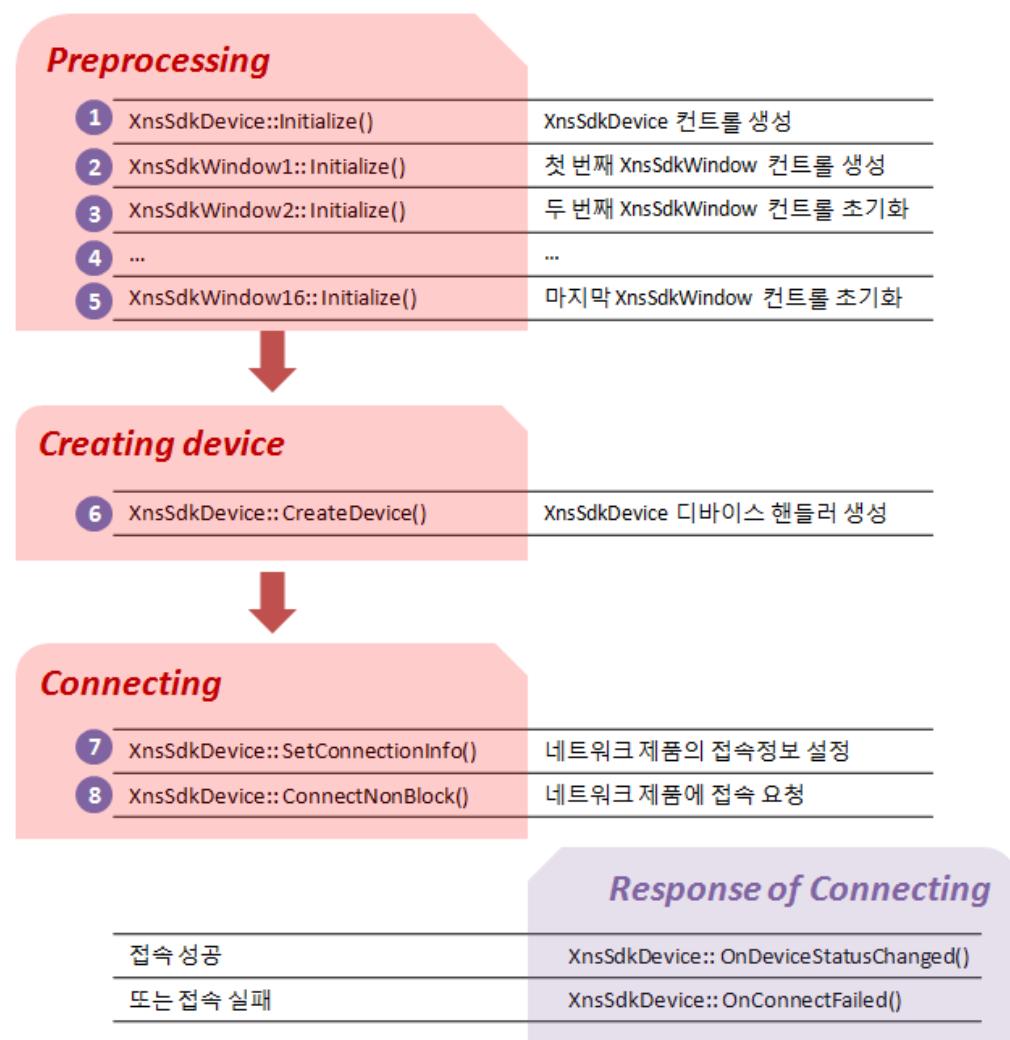


그림 11 Multi Channel 샘플 프로그램 실행 화면

- Step 3.** 연결이 정상적으로 이루어졌는지 확인합니다.
- Step 4.** 하단의 확장 화면에서 원하는 [channel]을 선택합니다.
- Step 5.** [Open] 버튼을 클릭하여 라이브 영상 수신을 시작합니다. 정상적으로 출력되는지 확인합니다.
- Step 6.** [Close] 버튼을 클릭하여 라이브 영상 수신을 종료합니다.
- Step 7.** 프로그램을 종료합니다.

API 호출 순서

이 샘플 프로그램이 사용하는 API는 Single Live 샘플 프로그램에서 호출하는 API와 동일합니다. 다만, XnsSdkWindow 컨트롤의 초기화 함수만 채널 개수만큼 더 호출했다는 점에서 차이가 있습니다.



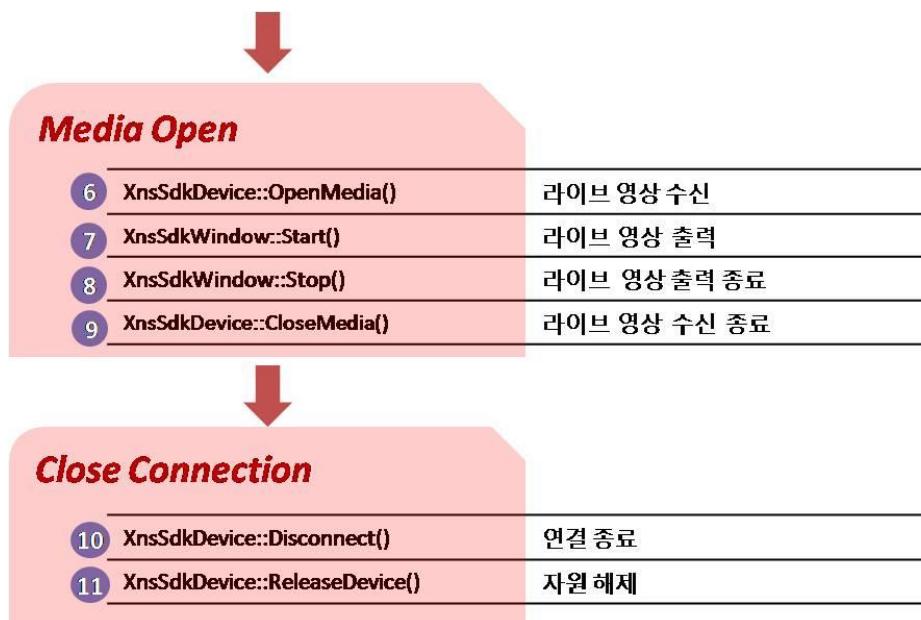


그림 12 Multi Channel 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 Single Live 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 생성

참고

ActiveX 컨트롤 삽입 및 초기화 방법은 본 문서의 CHAPTER 4에 기술된 Start Up 샘플 프로그램의 '구현 방법 상세 설명' 절을 참조합니다.

디바이스 핸들러 생성

참고

디바이스 핸들러 생성 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '디바이스 핸들러 생성' 절을 참조합니다.

디바이스 연결 설정

참고

디바이스 연결 설정 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '디바이스 연결 설정' 절을 참조합니다.

이벤트 핸들러 구현

참고

이벤트 핸들러 구현 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '이벤트 핸들러 구현' 절을 참조합니다.

미디어 오픈

디바이스와 PC간 데이터 송수신을 위한 미디어 소스 핸들을 획득한 후에, 이 핸들을 윈도우 컨트롤에 등록합니다.

미디어 소스 핸들을 획득하려면 XnsSdkDevice 컨트롤의 OpenMedia() 함수를 호출하여 미디어 스트림 수신을 시작합니다. OpenMedia() 함수의 파라미터(out-parameter)로 미디어 소스의 핸들(m_hMediaSource)을 얻을 수 있습니다.

XnsSdkWindow 컨트롤의 Start() 함수를 호출하여 미디어 소스 핸들을 윈도우 컨트롤 객체에 등록합니다. XnsSdkWindow 컨트롤은 해당 핸들의 미디어 스트림을 수신하여 디코딩한 후 화면에 출력합니다.

```
// MultiChannel.cpp
if (m_ctrlXnsSdkDevice.GetControlType(m_hDevice, 1) & XCTL_DVR) // Device Type :DVR
{
    // XCTL_CAMERA : Camera channel of DVR.
    for(int i=0; i<m_ctrlXnsSdkDevice.GetControlCount(m_hDevice, XCTL_CAMERA); i++)
    {
        // Displays video in real time.
        if (m_ctrlXnsSdkDevice.GetControlCapability(m_hDevice, i+2, XCTL_CAP_LIVE) )
        {
            // Start getting media streams from the device.
            m_ctrlXnsSdkDevice.OpenMedia(m_hDevice,
                i+2,
                MEDIA_TYPE_LIVE,
                0,
                0,
```

```

        &m_hMediaSource[i]);  
  

        m_ptrXnsSdkWindow[i]->Start(m_hMediaSource[i]);  

    }  

}  

}

```

버튼 이벤트 핸들러 작성

CMultiChannelDlg 클래스에 OnBnClickedButtonClose 이벤트 핸들러를 작성합니다.

```

// MultiChannel.cpp
void CMultiChannelDlg::OnBnClickedButtonClose()
{
    for( int i = 0; i < 16; i++ )
    {
        if( !m_hMediaSource )
        {
            return;
        }
        else
        {
            m_ptrXnsSdkWindow[i]->Stop();
            m_ctrlXnsDevice.CloseMedia(m_hDevice, m_hMediaSource[i]);
            m_hMediaSource[i] = 0;
        }
    }
}

```

See also

[Start Up 샘플 프로그램](#)

[SingleLive 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.
현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 7

SingleLive Stream 샘플 프로그램

SingleLive Stream 샘플 프로그램은 XNS ActiveX SDK를 사용하여 1채널 라이브 영상을 화면에 출력하는 방법을 설명하기 위한 예제입니다. OpenMedia() 대신 OpenStream()을 사용합니다.

Contents

[샘플 프로그램 소개](#)

[API 호출 순서](#)

[구현 방법 상세 설명](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

Single Live 샘플 프로그램은 특정 네트워크 장비로부터 1채널 라이브 영상을 수신하여 화면에 출력하는 예제입니다. 이 작업을 위해 먼저 XnsSdkDevice 컨트롤과 XnsSdkWindow 컨트롤을 생성합니다. 컨트롤 생성 후에는 네트워크 장비간 연결을 시도하고, 연결 성공 시 라이브 영상 스트림을 받아 화면에 출력합니다. 라이브 영상 수신이 종료되면 장비와 연결을 끊고 리소스를 해제합니다.

샘플 프로그램 사용법은 다음과 같습니다.

- Step 1.** 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.
- Step 2.** 장비 모델, IP address, VNP 포트, ID/Password 등의 장비 연결 정보를 설정한 후 [Connect] 버튼을 누릅니다.

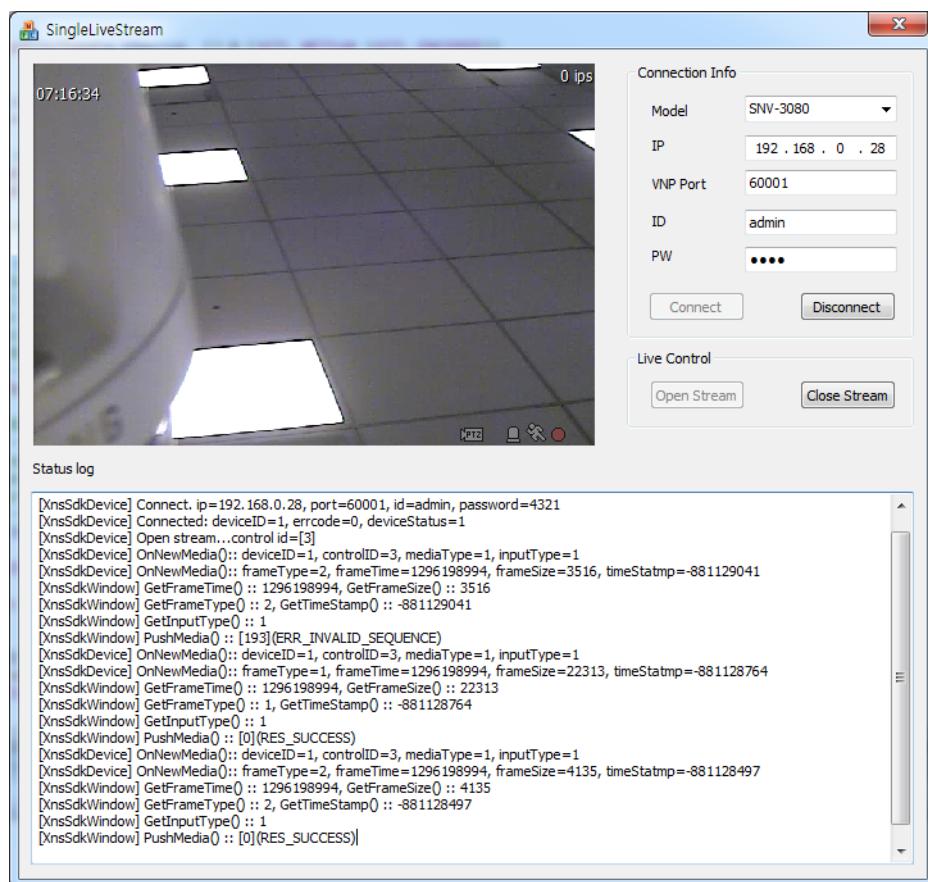


그림 13 SingleLiveStream 샘플 프로그램 실행 화면

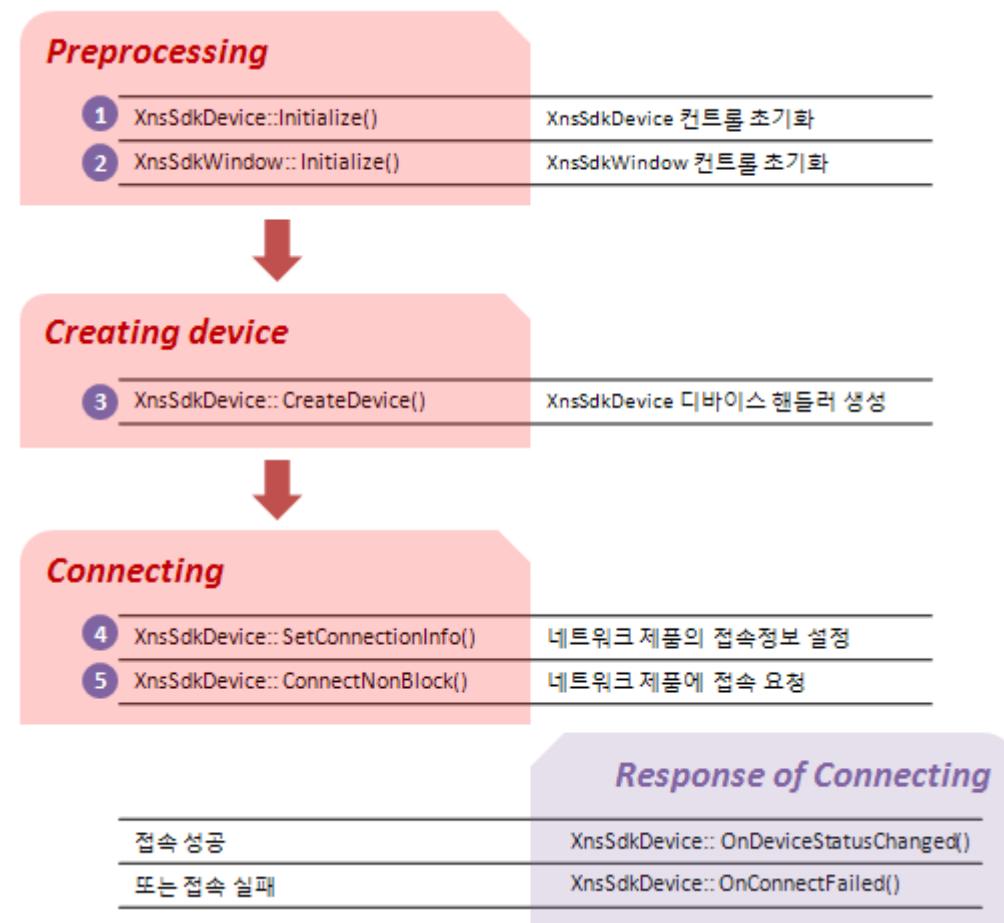
- Step 3.** 연결이 정상적으로 이루어졌는지 확인합니다.
- Step 4.** [Open Stream] 버튼을 클릭하여 라이브 영상을 화면에 출력합니다. 정상적으로 출력되는지 확인합니다.

Step 5. [Close Stream] 버튼을 클릭하여 라이브 영상 수신을 종료합니다.

Step 6. 프로그램을 종료합니다.

API 호출 순서

이 샘플 프로그램은 XNS ActiveX SDK가 제공하는 라이브러리를 사용하여 1채널 라이브 영상 출력 기능을 구현하였습니다. 샘플 프로그램의 API 호출 순서는 다음과 같습니다.



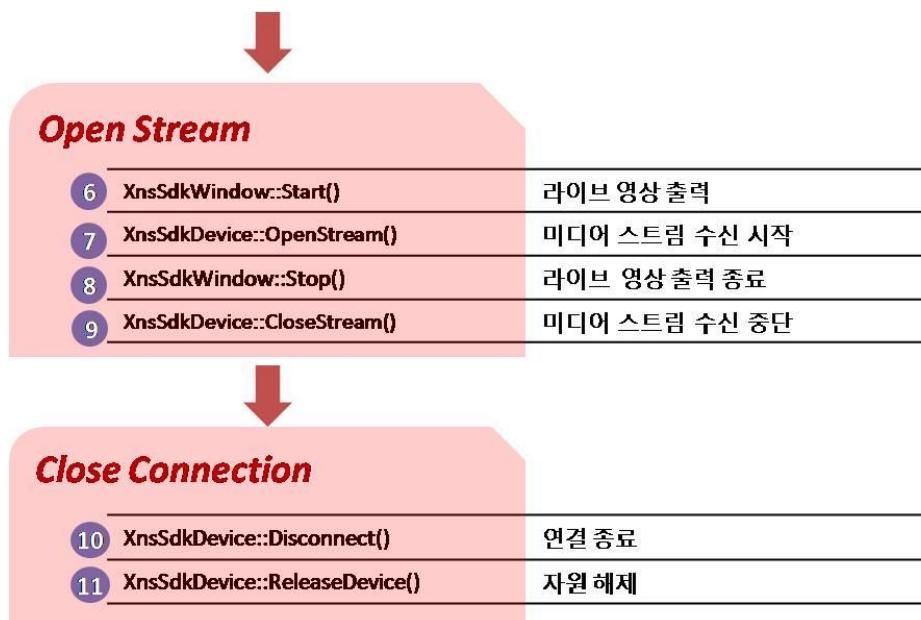


그림 14 SingleLiveStream 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 SingleLive Stream 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 생성

참고

ActiveX 컨트롤 삽입 및 초기화 방법은 본 문서의 CHAPTER 4에 기술된 Start Up 샘플 프로그램의 '구현 방법 상세 설명' 절을 참조합니다.

디바이스 핸들러 생성

참고

디바이스 핸들러 생성 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '디바이스 핸들러 생성' 절을 참조합니다.

디바이스 연결 설정

참고

디바이스 연결 설정 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '디바이스 연결 설정' 절을 참조합니다.

이벤트 핸들러 구현

참고

이벤트 핸들러 구현 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '이벤트 핸들러 구현' 절을 참조합니다.

이벤트 핸들러 작성

CSingleLiveStreamDlg 클래스에 OnDeviceStatusChangedXnssdkdevicectrl()와 OnConnectFailedXnssdkdevicectrl() 이벤트 핸들러를 작성하는 부분은 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 이벤트 핸들러와 같습니다.

추가적으로 OnNewMediaXnssdkdevicectrl() 이벤트 핸들러를 작성합니다.

```
// SingleLiveStreamDlg.cpp
void CSingleLiveStreamDlg::OnDeviceStatusChangedXnssdkdevicectrl(long nDeviceID, long
nControlID, long hMediaData, long nInputType, long nFrameType, long nFrameTime, long
nFrameSize, long nTimestamp)
{
    // Transfers stream data to XnsSdkWindow so that the media stream will be displayed
    // on the window after decoded.
    m_ctrlXnsWindow.PushMedia(hMediaData);
}
```

스트림 오픈

XnsSdkDevice 컨트롤의 OpenStream() 함수를 호출하여 미디어 스트림 수신을 시작합니다. 수신된 미디어 스트림은 OnNewMedia 이벤트로 전달됩니다. 미디어 스트림을 재생하기 위해서는 OnNewMedia 이벤트에서 XnsSdkWindow::PushMedia() 함수를 호출하면 됩니다. XnsSdkWindow::PushMedia() 함수는 사전에 XnsSdkWindow::Start()가 호출되어야 하며 파라미터는 0을 입력하면 됩니다.

```
// SingleLiveStreamDlg.cpp
if (m_ctrlXnsSdkDevice.GetControlType(m_hDevice, 1) & XCTL_DVR) // Device Type :DVR
{
```

```
// XCTL_CAMERA : Camera channel of DVR.  
for(int i=0; i<m_ctrlXnsSdkDevice.GetControlCount(m_hDevice, XCTL_CAMERA); i++)  
{  
    // Displays video in real time.  
    if (m_ctrlXnsSdkDevice.GetControlCapability(m_hDevice, i+2, XCTL_CAP_LIVE) )  
    {  
        m_ctrlXnsSdkWindow.Start( 0 );  
        m_ctrlXnsSdkDevice.OpenStream(m_hDevice,  
                                       i+2,  
                                       MEDIA_TYPE_LIVE,  
                                       0,  
                                       0 );  
    }  
}  
}
```

See also

[Start Up 샘플 프로그램](#)

[Single Live 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.

현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 8

Multiple Connect 샘플 프로그램

Multiple Connect 샘플 프로그램은 XNS ActiveX SDK를 사용하여 한 대 이상의 장비로부터 수신한 라이브 영상을 화면에 출력하는 방법을 설명하기 위한 예제입니다. 이 예제에서는 4개의 장비와 연결하여 4개의 라이브 영상을 화면에 출력하는 기능을 구현하였습니다.

Contents

[샘플 프로그램 소개](#)

[API 호출 순서](#)

[구현 방법 상세 설명](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

Multiple Connect 샘플 프로그램은 최대 4개의 네트워크 장비로부터 라이브 영상을 수신하여 화면에 출력하는 예제입니다. 이 작업을 위해 먼저 XnsSdkDevice 컨트롤 한 개를 생성하고, 장비 개수만큼 XnsSdkWindow 컨트롤을 생성합니다. 컨트롤 생성 후에는 네트워크 장비에 연결을 시도하고 성공 시 장비로부터 미디어 스트림을 수신합니다.

샘플 프로그램 사용법은 다음과 같습니다.

- Step 1.** 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.
- Step 2.** 장비 모델, IP address, VNP 포트, ID/Password 등의 장비 연결 정보를 설정한 후 [ADD] 버튼을 누릅니다. 이렇게 하면 디바이스 정보가 디바이스 목록(List Control)에 저장됩니다.

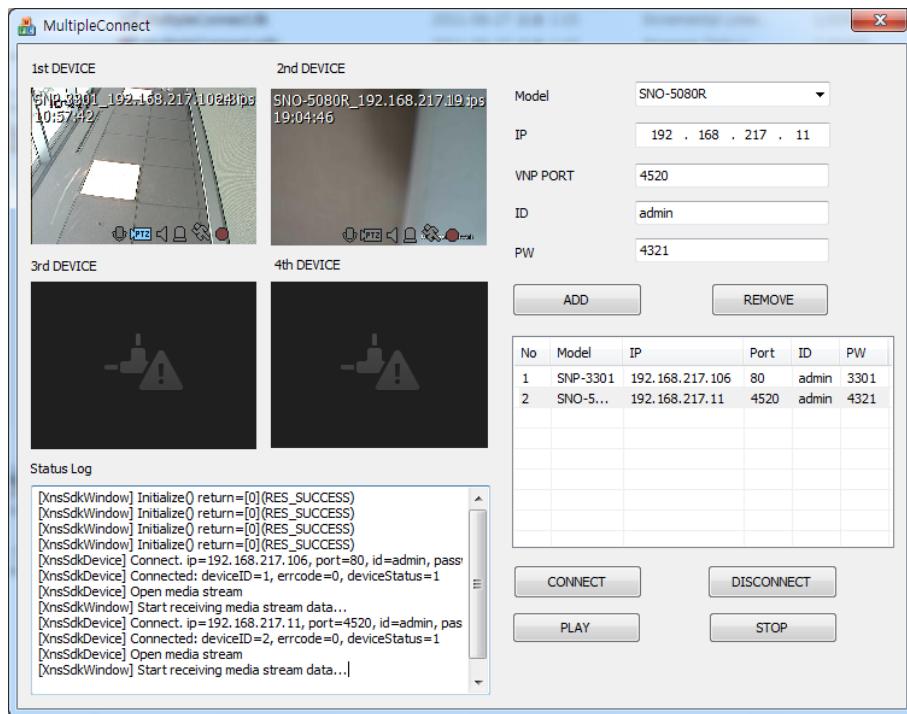


그림 15 Multiple Connect 샘플 프로그램 실행 화면

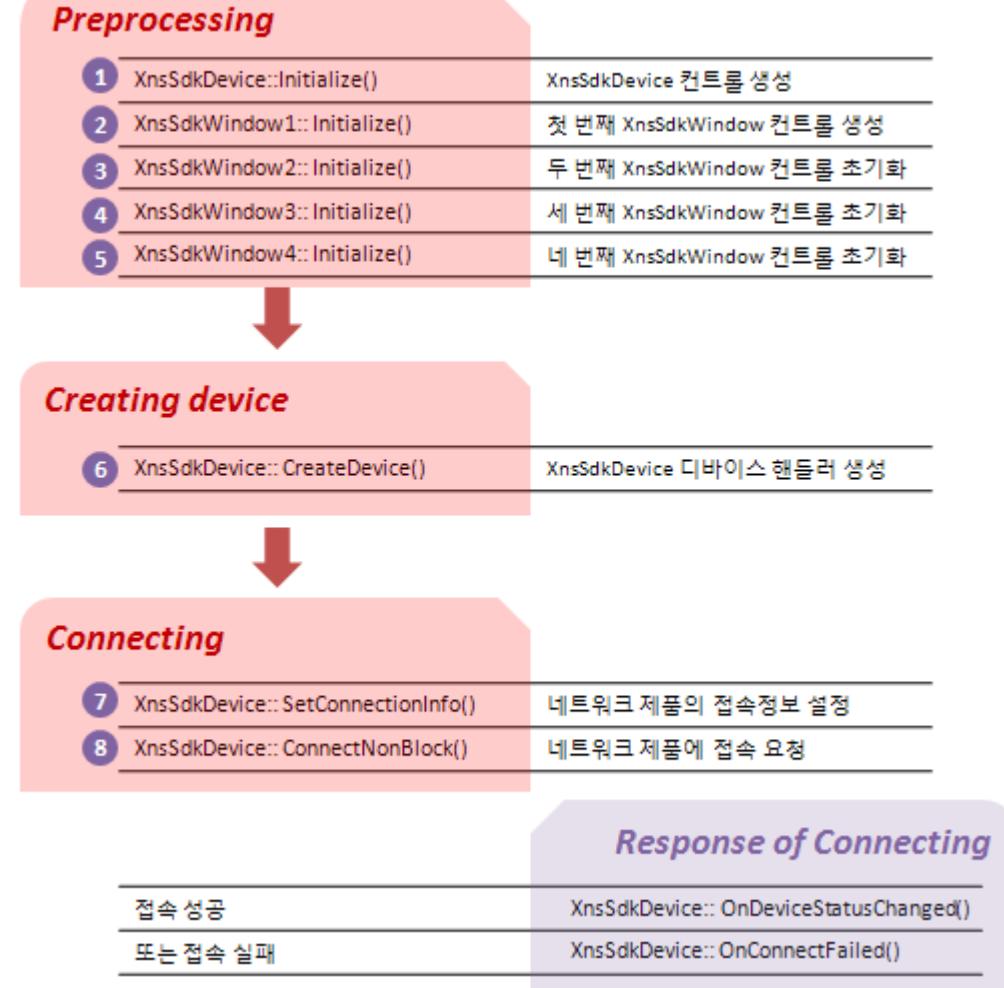
- Step 3.** 디바이스 목록에서 연결하려는 장비를 선택한 후 [CONNECT] 버튼을 클릭합니다. 장비와 정상적으로 연결되는지 확인합니다.
- Step 4.** [PLAY] 버튼을 클릭하여 라이브 영상을 화면에 출력합니다. 정상적으로 출력되는지 확인합니다.
- Step 5.** [STOP] 버튼을 클릭하여 라이브 영상 출력이 중지되는지 확인합니다.
- Step 6.** 디바이스 목록에 추가한 모든 장비에 대해서 3)~5)번 절차를 반복합니다.

Step 7. 디바이스 목록에서 특정 장비를 선택한 후 [REMOVE] 버튼을 클릭합니다.

Step 8. 프로그램을 종료합니다.

API 호출 순서

이 샘플 프로그램이 사용하는 API는 Single Live 샘플 프로그램에서 호출하는 API와 동일합니다. 다만, XnsSdkWindow 컨트롤의 초기화 함수만 채널 개수만큼 더 호출했다는 점에서 차이가 있습니다.



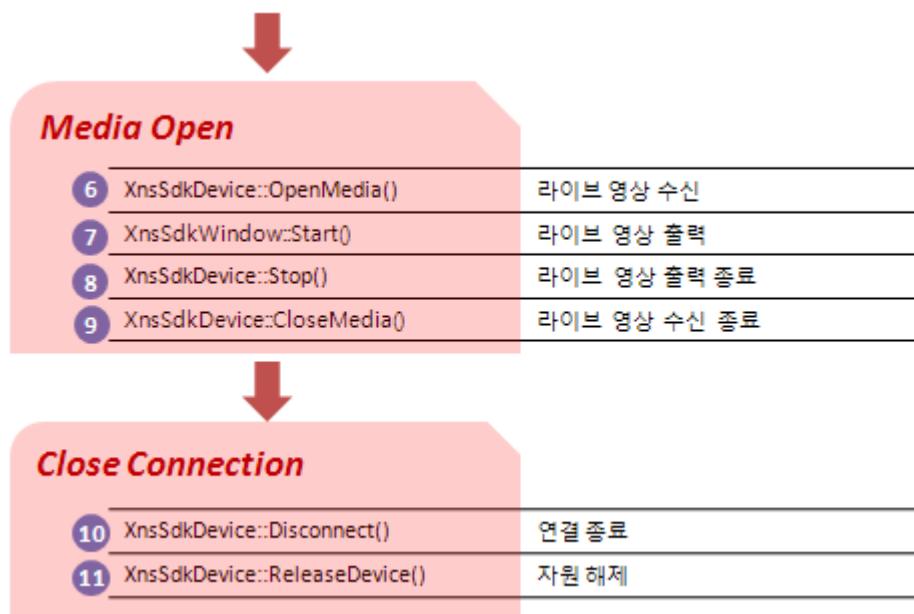


그림 16 Multiple Connect 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 Multiple Connect 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 삽입 및 초기화

참고

ActiveX 컨트롤 삽입 및 초기화 방법은 본 문서의 CHAPTER 4에 기술된 Start Up 샘플 프로그램의 ‘구현 방법 상세 설명’ 절을 참조합니다.

DEVICEINFO 구조체 및 배열 선언

이 샘플 프로그램은 여러 개의 장비와 연결하기 때문에 다수 개 장비에 대한 정보를 저장할 수 있도록 배열을 사용합니다. DEVICEINFO 구조체를 선언하고, CMultipleConnectDlg 클래스에 DEVICEINFO 타입의 DevInfo[] 배열을 선언합니다.

```
// MultipleConnectDlg.h
typedef struct DEVICEINFO
```

```

{
    CString strModel;           // Model name
    CString strIpAddress;       // IP Address
    int nPort;                  // Port Number
    CString strId;              // User ID
    CString strPasswd;          // Password
    long hDevice;               // Device Handle
    long hMediaSource;          // Media stream ID
    bool bIsMediaPlay;
} DEVICEINFO;

class CMultipleConnectDlg : public CDialog
{
private:
    DEVICEINFO DevInfo[4];      // Device information
}

```

List Control 구현

다이얼로그에서 디바이스 연결 정보를 입력한 후 [ADD] 버튼을 누르면, 해당 정보가 디바이스 목록(List Control)에 추가되도록 프로그램을 작성합니다.

먼저 List Control을 생성하기 위해 CMultipleConnectDlg 클래스에 m_ctrlListDevice 객체를 생성하고, 이 객체를 다이얼로그 객체와 연결합니다.

```

// MultipleConnectDlg.cpp
void CMultipleConnectDlg::DoDataExchange(CDataExchange* pDX)
{
    DDX_Control(pDX, IDC_LIST_DEVICE, m_ctrlListDevice);
}

```

[ADD] 버튼이 눌렸을 때, List Control로부터 디바이스 연결 정보를 가져와서 DevInfo[]에 저장하도록 구현합니다.

```

// MultipleConnectDlg.cpp
// nDeviceIndex is device number
DevInfo[nDeviceIndex-1].strModel = m_ctrlListDevice.GetItemText(nDeviceIndex-1, 1);
DevInfo[nDeviceIndex-1].strIpAddress = m_ctrlListDevice.GetItemText(nDeviceIndex-1, 2);
DevInfo[nDeviceIndex-1].nPort = _ttoi(m_ctrlListDevice.GetItemText(nDeviceIndex-1, 3));
DevInfo[nDeviceIndex-1].strId = m_ctrlListDevice.GetItemText(nDeviceIndex-1, 4);

```

```
DevInfo[nDeviceIndex-1].strPasswd = m_ctrlListDevice.GetItemText(nDeviceIndex-1, 5);
```

사용자가 List Control에서 특정 장비를 선택했을 때, 현재 선택된 장비가 어떤 것인지를 알 수 있도록 디바이스 위치 정보를 처리합니다.

```
// MultipleConnectDlg.cpp
void CMultipleConnectDlg::OnNMClickListDevice(NMHDR *pNMHDR, LRESULT *pResult)
{
    *pResult = 0;

    NM_LISTVIEW* pNMListView = (NM_LISTVIEW*) pNMHDR;
    int idx = pNMListView->iItem;
    m_nSelItem = _toi(m_ctrlListDevice.GetItemText(idx, 6))-1;
}
```

디바이스 핸들러 생성

디바이스 개수만큼 디바이스 핸들러를 생성합니다. 디바이스 핸들러를 생성하면, 장비의 정보를 저장하기 위한 메모리 공간이 할당되고, 디바이스 핸들(device handle)이 반환됩니다. 반환되는 값이 0보다 큰 정수일 경우에 유효합니다.

디바이스 핸들러는 CMultipleConnectDlg 클래스의 OnBnClickedButtonAdd() 이벤트 핸들러 안에서 디바이스별 CreateDevice()를 호출함으로써 생성됩니다. CreateDevice() 함수는 XnsSdkDevice 컨트롤에 존재하는 함수입니다.

```
// MultipleConnectDlg.cpp
m_hDevice = m_ctrlXnsDevice.CreateDevice(n_DeviceIndex);
```

디바이스 연결 설정

CMultipleConnectDlg 클래스의 OnBnClickedButtonConnect() 이벤트 핸들러 함수 안에서 SetConnectionInfo() 함수를 호출하여 디바이스 연결 정보를 설정합니다. 두 번째 파라미터인 szVendorName에는 "Samsung"이라는 고정 값을 입력하고, 포트 번호 파라미터에는 VNP 포트 번호를 지정합니다.

```
// MultipleConnectDlg.cpp
m_ctrlXnsDevice.SetConnectionInfo(
    DevInfo[m_nSelItem].hDevice,           // [in] Device handle
    _T("Samsung"),                      // [in] Fixed as 'Samsung'
    DevInfo[m_nSelItem].strModel,         // [in] Name of model to connect to.
```

```

    XADDRESS_IP,           // [in] Address type
    DevInfo[m_nSelItem].strIpAddress, // [in] actual address according to address
    type.
    DevInfo[m_nSelItem].nPort,      // [in] Port number
    0,                           // [in] Port number for web access
    DevInfo[m_nSelItem].m_strId,   // [in] Login ID
    DevInfo[m_nSelItem].m_strPasswd // [in] Password
);

```

이후 장비에 접속하기 위해서 XnsSdkDevice 컨트롤의 ConnectNonBlock() 함수를 호출합니다. 이 함수는 non-blocking 방식으로 동작하기 때문에 접속이 완료되지 않아도 바로 리턴됩니다.

```

// MultipleConnectDlg.cpp
long nRet = m_ctrlXnsDevice.ConnectNonBlock(
    DevInfo[m_nSelItem].hDevice, // [in] Device handle
    TRUE,                      // [in] Flag to decide where to forcibly log in or not.
    TRUE                       // [in] If this value is 1, try to connect again.
);
if(nRet != ERR_SUCCESS)
{
    WLOGD(_T("ConnectNonBlock() fail: errno=[%d](%s)\n"),
        nRet, m_ctrlXnsDevice.GetErrorString(nRet));
}

```

접속 결과는 이벤트를 통해 알 수 있으며, 접속 성공 시에는 OnDeviceStatusChanged 이벤트가 발생하고, 접속 실패 시에는 OnConnectFailed 이벤트가 발생합니다.

참고

이벤트를 등록하고 이벤트 핸들러를 구현하는 방법은 본 문서 CHAPTER 5에서 ‘이벤트 핸들러 구현’ 절을 참조합니다.

미디어 오픈

디바이스와 PC간 데이터 송수신을 위한 미디어 소스 핸들을 획득한 후에, 이 핸들을 윈도우 컨트롤에 등록합니다.

미디어 소스 핸들을 획득하려면 XnsSdkDevice 컨트롤의 OpenMedia() 함수를 호출하여 미디어 스트림 수신을 시작합니다. OpenMedia() 함수의 파라미터(out-parameter)로 미디어 소스의 핸들(m_hMediaSource)을 얻을 수 있습니다.

XnsSdkWindow 컨트롤의 Start() 함수를 호출하여 미디어 소스 핸들을 윈도우 컨트롤 객체에

등록합니다. XnsSdkWindow 컨트롤은 해당 핸들러의 미디어 스트림을 수신하여 디코딩한 후 화면에 출력합니다.

OnDeviceStatusChanged() 이벤트 핸들러에서 OpenMedia() 함수를 호출합니다. 디바이스 개수 만큼 미디어 소스 핸들러를 획득해야 합니다.

```
// MultipleConnectDlg.cpp
void CSingleLiveDlg::OnDeviceStatusChangedXnssdkdevicectrl(long nDeviceID, long nErrorCode,
long nDeviceStatus, long nHddCondition)
{
if (m_ctrlXnsDevice.GetControlType(DevInfo[m_nSelItem].hDevice, 1) & XCTL_DVR)
{
for(int i=0; i<m_ctrlXnsDevice.GetControlCount(DevInfo[m_nSelItem].hDevice,
XCTL_CAMERA); i++)
{
if (m_ctrlXnsDevice.GetControlCapability(DevInfo[m_nSelItem].hDevice, i+2,
XCTL_CAP_LIVE) )
{
m_ctrlXnsDevice.OpenMedia(
DevInfo[m_nSelItem].hDevice,
i+2,
MEDIA_TYPE_LIVE,
0,
0,
& DevInfo[m_nSelItem].hMediaSource
);
return;
}
}
else if (m_ctrlXnsDevice.GetControlType(DevInfo[m_nSelItem].hDevice, 1) & (XCTL_NETCAM
|XCTL_ENCODER))
{
for(int i=0; i<m_ctrlXnsDevice.GetControlCount(DevInfo[m_nSelItem].hDevice,
XCTL_VIDEO); i++)
{
if (m_ctrlXnsDevice.GetControlCapability(DevInfo[m_nSelItem].hDevice, i+3,
XCTL_CAP_LIVE) && m_ctrlXnsDevice.GetControlStatus(DevInfo[m_nSelItem].hDevice, i+3, 1))
{
m_ctrlXnsDevice.OpenMedia(DevInfo[m_nSelItem].hDevice, i+3,
MEDIA_TYPE_LIVE, 0, 0, & DevInfo[m_nSelItem].hMediaSource);
}
}
}
}
```

```

        return;
    }
}
}
```

획득한 미디어 소스 핸들러를 디바이스에 해당하는 XnsSdkWindow 컨트롤러에 등록해야 합니다. OnBnClickedButtonPlay() 이벤트 핸들러에서 XnsSdkWindow 컨트롤의 Start() 함수를 호출합니다.

```
// MultipleConnectDlg.cpp
void CMultipleConnectDlg::OnBnClickedButtonPlay()
{
    setCtrlXnsWindow(m_nSelItem+1);
    m_ptrXnsWindow->Start(DevInfo[m_nSelItem].hMediaSource);

    WLOGW(_T("Start receiving media stream data...\\n"));
    DevInfo[m_nSelItem].bIsMediaPlay = TRUE;
}
```

See also

[Start Up 샘플 프로그램](#)

[Single Live 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.

현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 9

Playback 샘플 프로그램

Playback 샘플 프로그램은 XNS ActiveX SDK를 사용하여 녹화된 영상을 재생하는 방법을 설명하기 위한 예제입니다.

Contents

[샘플 프로그램 소개](#)

[API 호출 순서](#)

[구현 방법 상세 설명](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

Playback 샘플 프로그램은 네트워크 디바이스에 녹화된 영상을 로컬PC에 출력하는 예제입니다. 먼저 XnsSdkDevice 컨트롤과 XnsSdkWindow 컨트롤을 하나씩 생성한 후에 네트워크 장비에 연결합니다. 그 다음에 녹화 영상이 기록된 날짜를 검색하고, 녹화 파일 리스트에서 재생을 원하는 영상을 선택합니다. 이렇게 하면 네트워크 디바이스로부터 재생(Playback) 영상을 수신하여 출력하게 됩니다.

샘플 프로그램 사용법은 다음과 같습니다.

- Step 1.** 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.
- Step 2.** 장비 모델, IP address, VNP 포트, ID/Password 등의 장비 연결 정보를 설정한 후 [Connect] 버튼을 누릅니다. 연결이 정상적으로 이루어지는지 확인합니다.

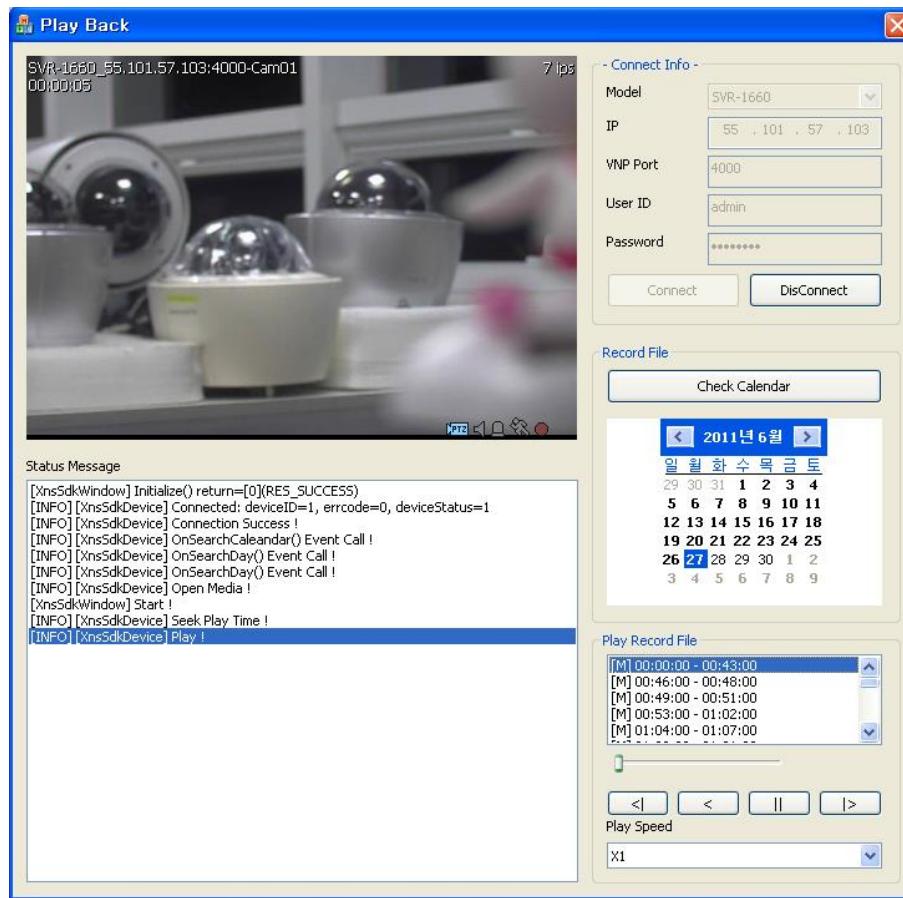


그림 17 Playback 샘플 프로그램 실행 화면

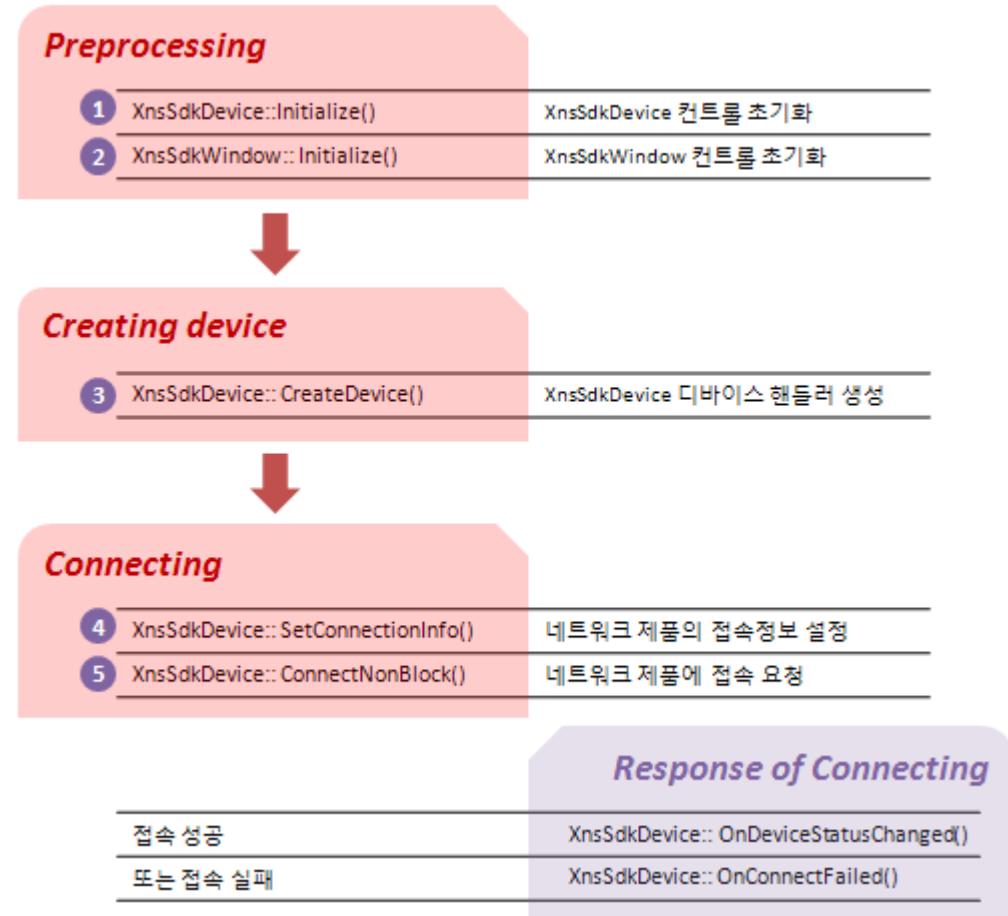
- Step 3.** [Check Calendar] 버튼을 눌러 Calendar Control에 녹화 영상이 기록된 날짜가 표기되는지 확인합니다.
- Step 4.** Calendar Control에서 날짜를 선택합니다.

Step 5. [Play Record File] 패널에서 Record List가 정상적으로 나타나는지 확인합니다. Record List에는 해당 일에 녹화 영상이 저장된 시간이 출력됩니다.

Step 6. 프로그램을 종료합니다.

API 호출 순서

이 샘플 프로그램은 XNS ActiveX SDK가 제공하는 라이브러리를 사용하여 녹화 영상 재생 기능을 구현하였습니다. 샘플 프로그램의 API 호출 순서는 다음과 같습니다.



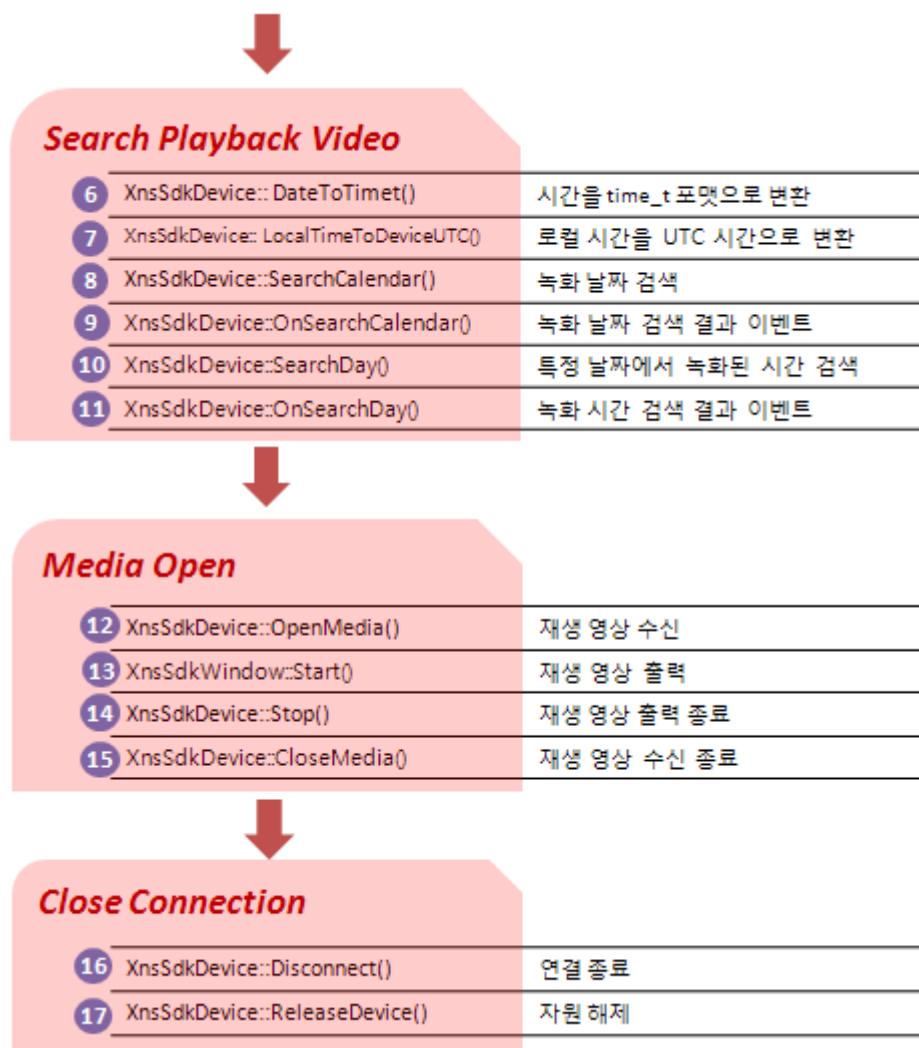


그림 18 Playback 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 Playback 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 삽입 및 초기화

참고

ActiveX 컨트롤 삽입 및 초기화 방법은 본 문서의 CHAPTER 4에 기술된 Start Up 샘플 프로그램의 '구현 방법 상세 설명' 절을 참조합니다.

디바이스 핸들러 생성

참고

디바이스 핸들러 생성 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '디바이스 핸들러 생성' 절을 참조합니다.

디바이스 연결 설정

참고

디바이스 연결 설정 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '디바이스 연결 설정' 절을 참조합니다.

Calendar 검색

다이얼로그 Calendar Control에서 특정 날짜를 선택하면 네트워크 카메라에 녹화 영상을 요청하여 수신하도록 프로그램을 작성합니다.

- Step 1.** Calendar Control에서 선택된 날짜의 시간 포맷을 변경합니다. time_t 포맷으로 변환한 후, 로컬 타임을 UTC 타임으로 다시 변환합니다.

```
// PlayBackDlg.cpp
void CPlayBackDlg::OnBnClickedButtonCheckCalendar2()
{
    SYSTEMTIME curtime;
    m_ctrlMonthCalendar.GetCurSel(&curtime); // Get Current Time in Calendar

    long nYear = curtime.wYear;
    long nMonth = curtime.wMonth;

    long nTargetMonth = m_ctrlXnsSdkDevice.DateToTimet(
        nYear,
        nMonth,
```

```

1,
0,
0,
0);

long nUTCtime = 0;
nUTCtime = m_ctrlXnsSdkDevice.LocalTimeToDeviceUTC(
    m_hDevice,
    nTargetMonth)
//하략

```

Step 2. 그런 다음 디바이스에 녹화된 미디어를 재생하기 위한 권한을 획득합니다.

```
// PlayBackDlg.cpp
long ret = m_ctrlXnsSdkDevice.AcquireMediaAuthority(m_hDevice);
```

Step 3. SearchCalendar() 함수를 호출하여, 선택한 날짜에 저장된 녹화 영상 시간을 가져옵니다.

```
// PlayBackDlg.cpp
void CPlayBackDlg::OnSearchCalendarXnssdkdevicectrl1(long nDeviceID, long nControlID, long
nRecDay)
{
    long nBitFlag = 0x01;

    SYSTEMTIME timeFrom;
    SYSTEMTIME timeUntil;
    int nCount = m_ctrlMonthCalendar.GetMonthRange(&timeFrom, &timeUntil,
GMR_DAYSTATE);
    LPMONTHDAYSTATE pDayState;
    pDayState = new MONTHDAYSTATE[nCount];
    memset(pDayState, 0, sizeof(MONTHDAYSTATE) * nCount);
    int nIndex = (timeFrom.wDay == 1) ? 0 : 1;

    for (int i=0; i<31; i++)
    {
        if (nRecDay & nBitFlag)
        {
            strRec = " 1 ";
        }
    }
}
```

```

        nBitFlag <<=1;
    }

VERIFY(m_ctrlMonthCalendar.SetDayState(nCount, pDayState));
delete[] pDayState;
}

```

녹화 영상 불러오기

Record List에서 특정 시간을 선택하면 녹화 데이터를 가져와서 재생하도록 구현합니다.

- Step 1.** Record List에서 선택된 시간의 포맷을 변경합니다. 녹화 시간 시간과 녹화 종료 시간을 time_t 포맷으로 변환한 후, 로컬 타임을 UTC 타임으로 다시 변환합니다.

```

// PlayBackDlg.cpp
void CPlayBackDlg::OnMcnSelectMonthcalendar1(NMHDR *pNMHDR, LRESULT *pResult)
{
    NMSELCHANGE      *pSelChange = (NMSELCHANGE*)pNMHDR;
    m_seltime = pSelChange->stSelStart;
    m_nYear = m_seltime.wYear;
    m_nMonth = m_seltime.wMonth;
    m_nDay = m_seltime.wDay;

    long tStart = m_ctrlXnsSdkDevice.DateToTimet(
        (long)m_nYear,
        (long)m_nMonth,
        (long)m_nDay,
        0,
        0,
        0);

    long tEnd = m_ctrlXnsSdkDevice.DateToTimet(
        (long)m_nYear,
        (long)m_nMonth,
        (long)m_nDay,
        23,
        59,
        59);
}

```

```

long tUTCStart = 0;
tUTCStart = m_ctrlXnsSdkDevice.LocalTimeToDeviceUTC(
    m_hDevice,
    tStart);

long tUTCEnd = 0;
tUTCEnd = m_ctrlXnsSdkDevice.LocalTimeToDeviceUTC(
    m_hDevice,
    tEnd);

```

- Step 2.** SearchDay() 함수를 호출하여 녹화된 데이터 정보를 가져옵니다. 함수의 입력 파라미터로 UTC 타임으로 변환한 녹화 시작 시간과 녹화 종료 시간을 입력합니다. 이 함수의 결과는 OnSearchDay 이벤트를 통해 수신합니다.

```

// PlayBackDlg.cpp
long ret = m_ctrlXnsSdkDevice.SearchDay(
    m_hDevice,
    m_nControlId,
    tUTCStart,
    tUTCEnd,
    REC_TYPE_ALL);

```

- Step 3.** OnSearchDay 이벤트를 처리할 OnSearchDayXnssdkdevicectrl1() 핸들러를 구현합니다. OnSearchDay 이벤트가 발생하면 타임라인 핸들을 파라미터로 수신합니다.

타임라인에 녹화 데이터 블록이 여러 개 존재할 경우, 타임라인 개수만큼 순환을 돌면서 녹화 데이터 블록의 정보를 가져옵니다. 가져온 정보는 로컬 시간 포맷으로 변경한 후, 내부 버퍼에 저장합니다.

```

// PlayBackDlg.cpp
void CPlayBackDlg::OnSearchDayXnssdkdevicectrl1(long nDeviceID, long nControlID, long
hTimeline)
{
    long nRecType
    long nRecId
    long nTotalCount = 0
    long nCamCount;
    CString strDaylist;

    // Record Data Time

```

```

long Year, Month, Day;
long nTimelineCount = 0;

if((nCamCount = m_ctrlXnsSdkDevice.GetCameraCount(hTimeline)) == 0){
    ERROR_BOX(_T("[Error!] Data Empty !"));
    return ;
}

m_ctrlRecDataList.ResetContent();

for (int i = 0; i < nCamCount; i++)
{
    nTimelineCount = m_ctrlXnsSdkDevice.GetTimelineCount(hTimeline, i);
    for(int j=0; j<nTimelineCount; j++)
    {
        nTotalCount++;
        m_ctrlXnsSdkDevice.GetTimeline(hTimeline, i, j, XTIME_UTC,
&m_stratTime, &m_endTime, &nRecType, &nRecId);

        long tLocalStart = 0;
        tLocalStart = m_ctrlXnsSdkDevice.UTCToDeviceLocalTime(
            m_hDevice,
            m_stratTime);
        long tLocalEnd = 0;
        tLocalEnd = m_ctrlXnsSdkDevice.UTCToDeviceLocalTime(
            m_hDevice,
            m_endTime);

        m_ctrlXnsSdkDevice.TimetToDate(
            tLocalStart,
            &Year,
            &Month,
            &Day,
            &m_startHour,
            &m_startMin,
            &m_startSec);

        m_ctrlXnsSdkDevice.TimetToDate(

```

```

        tLocalEnd,
        &Year,
        &Month,
        &Day,
        &m_endHour,
        &m_endMin,
        &m_endSec);

    // Record Types messages
    switch(nRecType)
    {
        // 각 이벤트에 따른 결과화면 출력.
    }
    m_ctrlRecDataList.AddString(strDaylist);
}

}

// m_pRecDataInfo Initialization.
if (m_pRecDataInfo)
{
    delete m_pRecDataInfo;
    m_pRecDataInfo = NULL;
}
m_pRecDataInfo = new sRecDataInfo[nTotalCount];
nTotalCount = 0;

// Time data input.
for (int i=0; i<nCamCount; i++)
{
    nTimelineCount = m_ctrlXnsSdkDevice.GetTimelineCount(hTimeline, i);
    for(int j=0; j<nTimelineCount; j++)
    {
        m_ctrlXnsSdkDevice.GetTimeline(
            hTimeline,
            i
            j,
            XTIME_UTC,
            &m_stratTime,
            &m_endTime,

```

```

        &nRecType,
        &nRecId);

    m_pRecDataInfo[nTotalCount].tStartTime = m_stratTime;
    m_pRecDataInfo[nTotalCount].tEndTime = m_endTime;
    m_pRecDataInfo[nTotalCount].nRecType = nRecType;
    nTotalCount++;
}

}

}

```

미디어 플레이

Record List를 더블 클릭하면, 해당 영상을 화면에 출력하도록 구현합니다. CPlayBackDlg 클래스 OnLbnDbclkListRecdatal 이벤트 핸들러에서 OpenMedia() 함수를 호출하여 미디어 소스 핸들을 획득하고, 획득한 핸들을 XnsSdkWindow 컨트롤에 등록합니다.

```

// PlayBackDlg.cpp

m_ctrlXnsSdkDevice.AcquireMediaAuthority(m_hDevice);

if (m_ctrlXnsSdkDevice.GetControlCapability(m_hDevice, 1, XCTL_CAP_FSPEED1))
{
    for(int i=0; i<m_ctrlXnsSdkDevice.GetControlCount(m_hDevice, XCTL_CAMERA); i++)
    {
        if (m_ctrlXnsSdkDevice.GetControlCapability(m_hDevice, i+2,
XCTL_CAP_PLAYBACK))
        {
            if (!m_hMediaSource)
            {
                m_ctrlXnsSdkDevice.OpenMedia(
                    m_hDevice,
                    i+2,
                    MEDIA_TYPE_PLAYBACK,
                    tStart,
                    tEnd,
                    &m_hMediaSource);

                m_ctrlXnsSdkWindow.Start(m_hMediaSource);
            }
            if (!m_ctrlXnsSdkDevice.IsPlaying(m_hDevice))

```

```
{  
    m_ctrlXnsSdkDevice.Seek(  
        m_hDevice,  
        i+2,  
        tStart);  
  
    m_ctrlXnsSdkDevice.Play(  
        m_hDevice,  
        i +2,  
        1.0);  
    m_bPlay = TRUE;  
}  
m_nControlId = i+2;  
return;  
}  
}  
}
```

See also

[Start Up 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.

현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 10

PTZ 샘플 프로그램

PTZ 샘플 프로그램은 XNS ActiveX SDK를 사용하여 카메라의 팬/틸트/줌을 제어하는 방법을 설명하기 위한 예제입니다.

Contents

[샘플 프로그램 소개](#)

[API 호출 순서](#)

[구현 방법 상세 설명](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

PTZ 샘플 프로그램은 카메라의 동작을 제어하는 예제입니다. 이 예제에는 OSD 메뉴 제어, 스캔 설정 및 수행, preset 설정 및 수행, Area Zoom 수행, Zoom 1X 수행 및 PTZ 제어 기능이 포함되어 있습니다.

다른 예제와 마찬가지로 먼저 XnsSdkDevice 컨트롤과 XnsSdkWindow 컨트롤을 하나씩 생성한 후에 네트워크 장비에 연결합니다.

샘플 프로그램 사용법은 다음과 같습니다.

- Step 1.** 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.
- Step 2.** 장비 모델, IP address, VNP 포트, ID/Password 등의 장비 연결 정보를 설정한 후 [Connect] 버튼을 눌러 화면에 1채널 라이브 영상이 정상적으로 출력되는지 확인합니다.

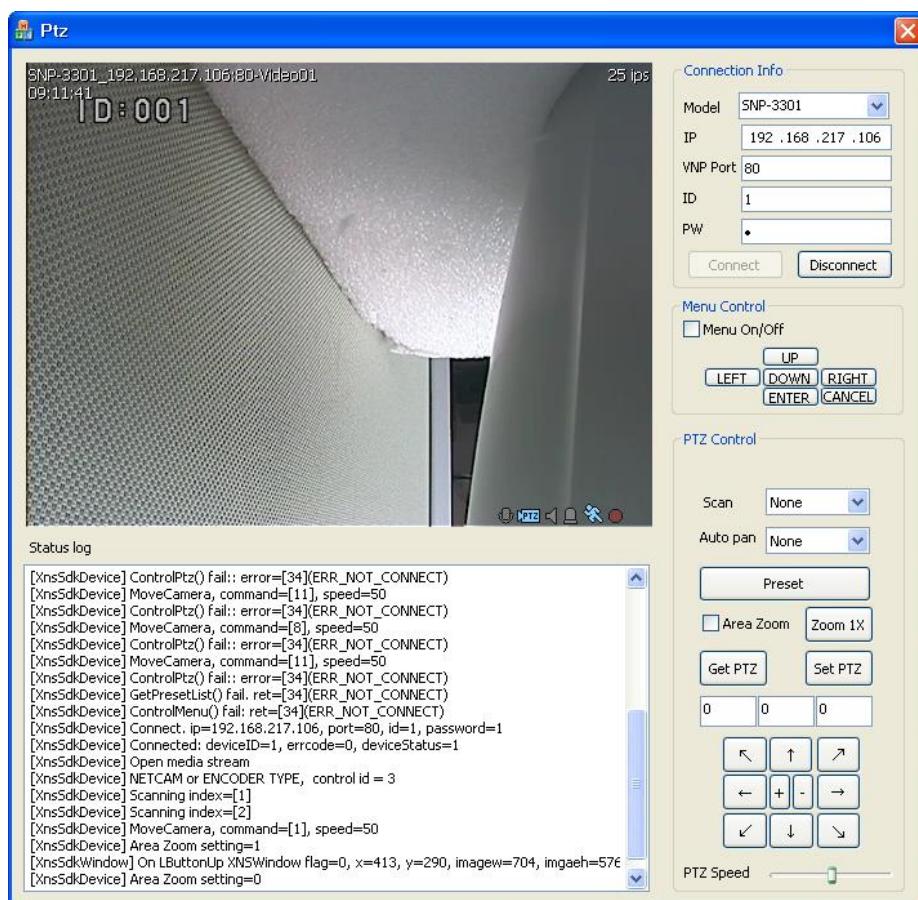


그림 19 PTZ 샘플 프로그램 실행 화면

- Step 3.** [Menu Control] 패널에서 [Menu On/Off] 체크박스를 선택하면 화면에 OSD 메뉴가 나타납니다. UP, LEFT, DOWN, RIGHT, ENTER, CANCEL 버튼을 사용하여 OSD 메뉴 간을 이동하고 메뉴를 선택합니다.

- Step 4.** [PTZ Control] 패널의 Scan 선택박스에서 Scan[1]~Scan[6] 중 하나를 선택하여 정상적으로 스캐닝 되는지 확인합니다. Scan[1]~Scan[6]은 카메라에 기 설정된 스캔리스트를 뜻합니다.
- Step 5.** [PTZ Control] 패널의 Auto Pan 선택박스에서 Auto pan[1]~Auto pan[4] 중 하나를 선택하여 Auto Pan 기능이 정상적으로 수행되는지 확인합니다. Auto pan[1]~Auto pan[4]는 카메라에 기 설정되어 있어야 합니다.
- Step 6.** [PTZ Control] 패널의 [Area Zoom] 체크박스를 선택한 후, 영상 화면 위에서 마우스를 클릭하거나 드래그 함으로써 Area Zoom이 정상 동작하는지 확인합니다.
- Step 7.** [PTZ Control] 패널에서 [Preset] 버튼을 클릭하면 다음과 같이 Presetダイ얼로그가 실행됩니다. 여기서 특정 preset을 더블 클릭하여 설정된 preset으로 카메라가 이동하는지 확인합니다. 이ダイ얼로그에서 preset 저장 및 삭제가 정상적으로 수행되는지도 확인합니다.

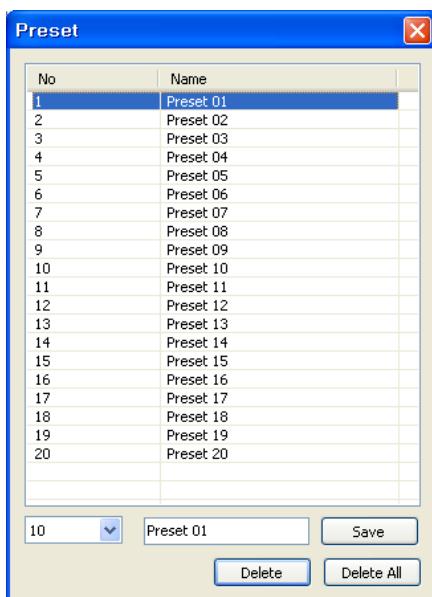
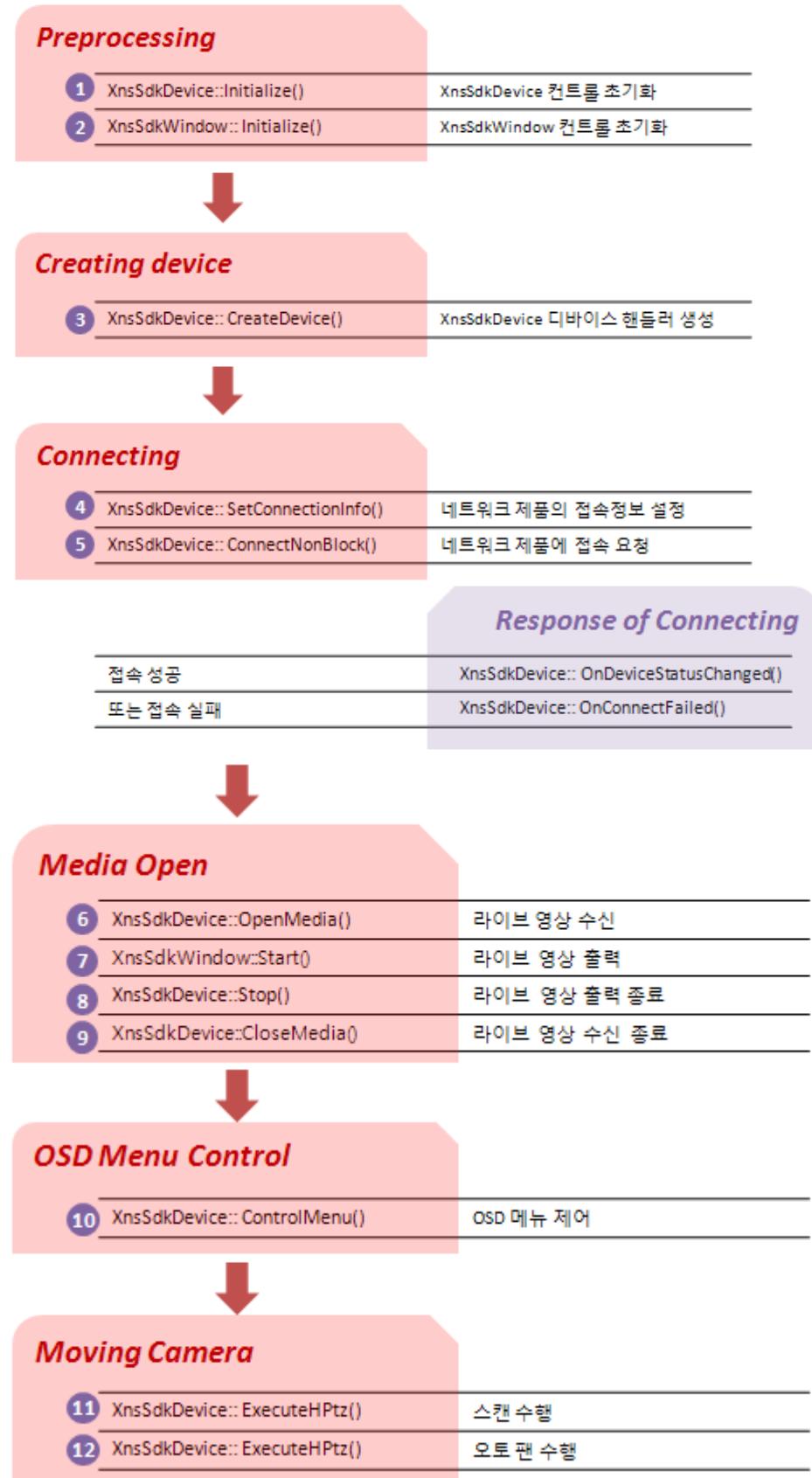


그림 20 Presetダイアル로그

- Step 8.** [PTZ Control] 패널에서 [Get PTZ] 버튼을 클릭하여 현재 카메라의 절대 좌표를 제대로 불러오는지 확인합니다.
- Step 9.** [PTZ Control] 패널에 있는 방향키를 이용하거나 절대 좌표 값을 입력하여 PTZ 위치를 맞춘 후 [Set PTZ] 버튼을 클릭합니다.
- Step 10.** [PTZ Speed] 스피드버튼을 이동하여 PTZ 속도 제어 기능이 정상적인지 확인합니다.
- Step 11.** 프로그램을 종료합니다.

API 호출 순서

이 샘플 프로그램은 XNS ActiveX SDK가 제공하는 라이브러리를 사용하여 디바이스의 동작을 제어하는 기능을 구현하였습니다. 샘플 프로그램의 API 호출 순서는 다음과 같습니다.



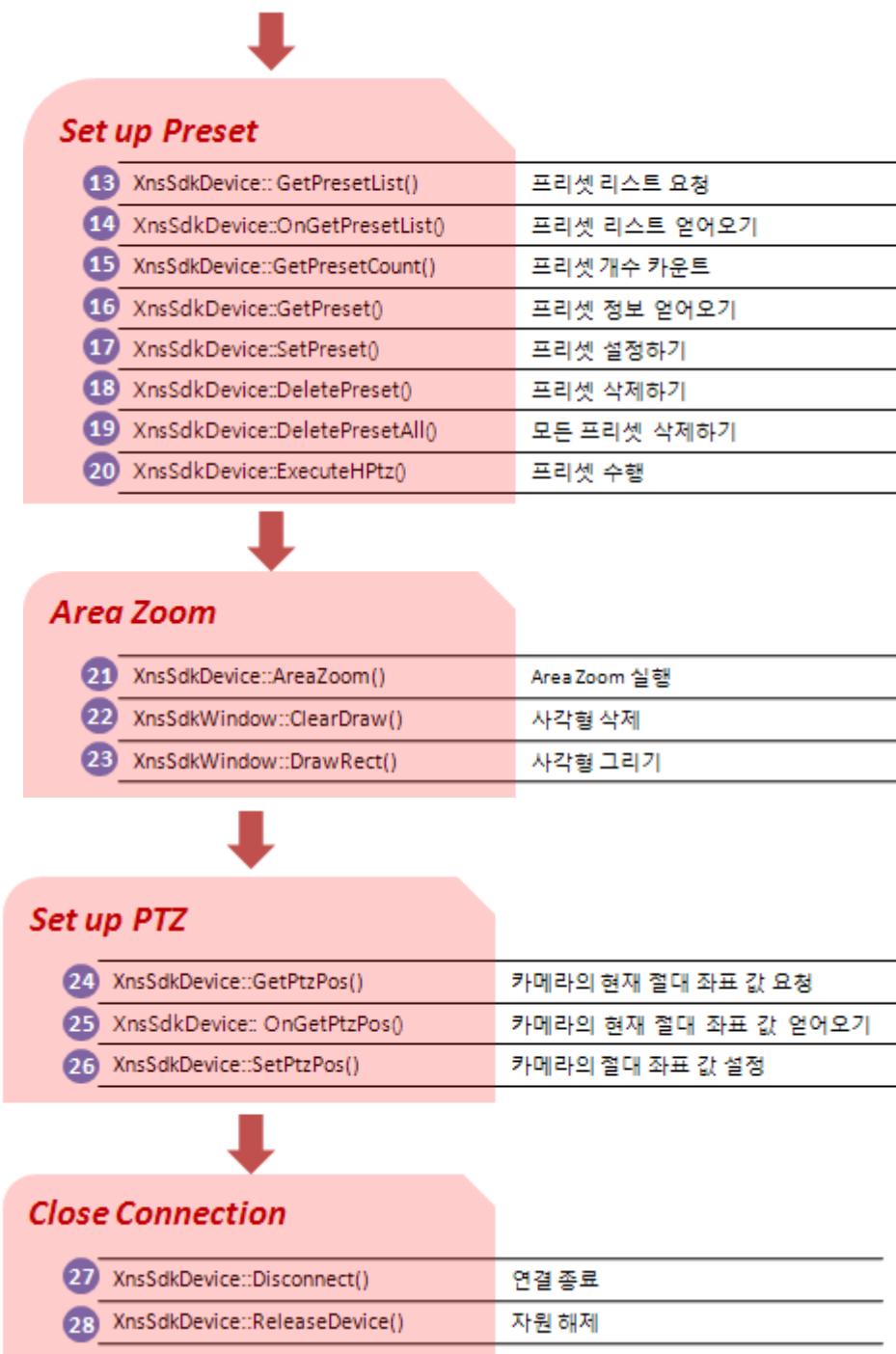


그림 21 PTZ 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 PTZ 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 삽입 및 초기화

참고

ActiveX 컨트롤 삽입 및 초기화 방법은 본 문서의 CHAPTER 4에 기술된 Start Up 샘플 프로그램의 '구현 방법 상세 설명' 절을 참조합니다.

디바이스 핸들러 생성

참고

디바이스 핸들러 생성 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '디바이스 핸들러 생성' 절을 참조합니다.

디바이스 연결 설정

참고

디바이스 연결 설정 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '디바이스 연결 설정' 절을 참조합니다.

OSD 메뉴 제어

OSD 메뉴 제어에는 윈도우 상에 OSD 메뉴를 출력하거나 출력하지 않도록 제어하는 것과 OSD 메뉴 간 이동 및 입력을 처리하는 것이 포함됩니다. 다이얼로그에서 [Menu On/Off] 메뉴를 선택하거나 방향키를 이용하여 메뉴 간 이동을 했을 때 이를 반영하도록 프로그램을 작성합니다.

Step 1. CPtzDlg 클래스의 OSD 메뉴 버튼 이벤트 핸들러 내에서 ControlMenu() 함수를 호출합니다. ControlMenu() 함수는 미디어 스트림을 수신하고 있는 동안에만 유효합니다.

```
// PtzDlg.cpp
int nRet = m_ctrlXnsDevice.ControlMenu(
    m_hDevice,
    m_nControlId,
    (m_bMenuControl?XMENU_ON:XMENU_OFF)
);
if (nRet != ERR_SUCCESS)
{
```

```
WLOGD(_T("ControlMenu() fail: ret=[%d](%s)\n"),
       nRet, m_ctrlXnsDevice.GetErrorString(nRet));
}
```

ControlMenu() 함수의 입력 파라미터로 디바이스 핸들과 컨트롤ID, Command ID를 입력합니다. Command ID는 다음 표와 같습니다.

표 3 Menu Control Command

Command	Value	Description
XMENU_UP	1	Menu up
XMENU_DOWN	2	Menu down
XMENU_LEFT	3	Menu left or prev menu
XMENU_RIGHT	4	Menu right or next menu
XMENU_ENTER	5	Menu enter, select
XMENU_ON	6	Menu on
XMENU_OFF	7	Menu off
XMENU_CANCEL	8	Cancel. (Changes are not saved.)

카메라 동작 제어

카메라의 동작 제어에는 프리셋(preset), 오토판(autopan), 스캔(scan), 패턴(pattern) 기능 등이 포함됩니다. 다이얼로그에서 [Scan] 또는 [Auto Pan] 콤보박스를 선택하거나 [Preset] 명령을 수행했을 때 이를 처리하도록 프로그램을 작성합니다.

- Step 1. CPtzDlg 클래스의 카메라 제어 버튼 이벤트 핸들러 내에서 ExecuteHPtz() 함수를 호출합니다. 파라미터로 디바이스 핸들과 컨트롤ID, 제어 명령, 카메라 기능 인덱스를 입력합니다. 제어 명령은 '표 4 Camera Control Command'를 참조합니다.

```
// PtzDlg.cpp
long nRet = m_ctrlXnsDevice.ExecuteHPtz(
    m_hDevice,
    m_nControlId,
    XHPTZ_SCAN,
    nIndex
);
```

표 4 Camera Control Command

Command	Value	Description
XHPTZ_PRESET	1	Preset
XHPTZ_AUTOPAN	2	Autopan(swing)
XHPTZ_SCAN	3	Scan
XHPTZ_PATTERN	4	Pattern

Preset List 이벤트

GetPresetList() 함수를 호출하여 프리셋 리스트를 요청한 경우, 실제 응답은 OnGetPresetList 이벤트를 통해 수신합니다. 이때 프리셋 리스트의 핸들은 OnGetPresetList 이벤트 핸들러의 파라미터로 전달됩니다.

Step 1. OnGetPresetList 이벤트 핸들러를 작성합니다.

```
// PtzDlg.cpp
void CPtzDlg::OnGetPresetListXnssdkdevicectrl(long nDeviceID, long nControlID, long
hPresetListList)
{
    // TODO: Add your message handler code here
    long nNumber;
    long nPresetCount;
    CString strPresetName;
    CMap< int, int, CString, CString > cmPresetList;

    nPresetCount = m_ctrlXnsDevice.GetPresetCount(hPresetListList);

    for( int i=0 ; i<nPresetCount ; i++ )
    {
        // Returns the preset number and name corresponding to the
        // given index in the preset list of XnsSdkDevice.
        strPresetName = m_ctrlXnsDevice.GetPreset(
            hPresetListList,// [in] Handle of the preset list.
            i,              // [in] Preset index to get the preset information
            &nNumber       // [out] Preset index specified in XnsSdkDevice.
        );
        cmPresetList.SetAt(nNumber, strPresetName);
    }
}
```

```
PresetDlg dlg(&m_ctrlXnsDevice, m_hDevice, m_nControlId, &cmPresetList);
INT_PTR nResult = dlg.DoModal();
}
```

Preset List 추가 및 삭제

Preset 다이얼로그에서 카메라 위치를 설정한 후 [Save] 버튼을 누르면, Preset List에 설정 값을 저장하도록 구현합니다. 반면 Preset 다이얼로그에서 [Delete] 또는 [Delete All] 버튼을 누르면, Preset List에서 특정 preset을 삭제하거나 모두 삭제하도록 구현합니다.

- Step 1. Preset List에 preset을 추가하려면, PresetDlg 클래스의 OnBnClickedSave() 내에서 m_pCtrlXnsDevice 객체의 SetPreset() 함수를 호출합니다.

```
// PresetDlg.cpp
long nRet = m_pCtrlXnsDevice->SetPreset(
    m_hDevice,           // [in] Device handle.
    m_nControlId,        // [in] Control ID.
    nIndex,              // [in] Preset number to save (start with 1).
    m_strPresetName     // [in] Preset name. The camera will save the name
                        // including the preset position (English only).
);
```

- Step 2. Preset List에서 특정 preset을 삭제하려면, PresetDlg 클래스의 OnBnClickedDelete() 내에서 m_pCtrlXnsDevice 객체의 DeletePreset() 함수를 호출합니다.

```
// PresetDlg.cpp
long nRet = m_pCtrlXnsDevice->DeletePreset(
    m_hDevice,           // [in] Device handle.
    m_nControlId,        // [in] Control ID.
    nIndex               // [in] Preset number to save (start with 0).
);
```

마우스 이벤트 처리

다이얼로그 영상 윈도우 상에 마우스 입력을 처리하도록 XnsSdkWindow 컨트롤의 이벤트 처리 기능을 구현합니다. 위에서 등록한 이벤트에 대한 이벤트 핸들러를 작성합니다.

```
// PtzDlg.cpp
void CPtzDlg::OnLButtonUpXnssdkwindowctrl(long nFlags, long nX, long nY)
{
```

```
m_bIsMouseDown = false;  
...  
long nImageWidth = m_ctrlXnsWindow.GetImageWidth();  
long nImageHeight = m_ctrlXnsWindow.GetImageHeight();  
  
CRect rt;  
m_ctrlXnsWindow.GetClientRect(rt);  
long nWindowWidth = rt.Width();  
long nWindowHeight = rt.Height();  
  
m_nEndX = nX;  
m_nEndY = nY;  
}
```

AreaZoom 설정하기

CPtzDlg 클래스에서 AreaZoom() 함수를 호출하여 카메라의 AreaZoom 기능을 설정합니다. 이 함수는 애플리케이션이 카메라로부터 미디어 스트림을 수신하고 있는 동안에만 유효합니다.

```
// PtzDlg.cpp  
long nRet = m_ctrlXnsDevice.AreaZoom(  
    m_hDevice,           // Device handle.  
    m_nControlId,       // Control ID.  
    m_nStartX,          // X-axis start position  
    m_nStartY,          // Y-axis start position  
    m_nEndX,            // X-axis end position  
    m_nEndY,            // Y-axis end position  
    nWindowWidth,        // Window width  
    nWindowHeight,       // Window height  
    nImageWidth,         // Image width  
    nImageHeight);      // Image height
```

PTZ 절대 좌표 값 얻어오기

GetPtzPos() 함수를 호출하여 PTZ 절대 좌표 값을 요청한 경우, 실제 응답은 OnGetPtzPos 이벤트를 통해 수신합니다. 이때 PTZ 절대 좌표 값은 OnGetPtzPos 이벤트 핸들러의 파라미터로 전달됩니다.

PTZ 절대 좌표 값 요청하기

CPtzDlg 클래스에서 GetPtzPos() 함수를 호출하여 PTZ 절대 좌표 값을 요청합니다.

```
long nRet = m_ctrlXnsDevice.GetPtzPos(m_hDevice, m_nControlId);
```

PTZ 절대 좌표 값 읽어오기

이벤트 맵에 OnGetPtzPos 이벤트를 선언하고, OnGetPtzPos 이벤트 핸들러를 작성합니다.

```
void CPtzDlg::OnGetPtzPosXnssdkdevicectrl(
    long nDeviceID,
    long nControlID,
    long nErrorCode,
    long nPan,
    long nTilt,
    long nZoom
)
{
    // TODO: Add your message handler code here
    WLOGD(_T("Get Ptz Position device_id=%d, pan=%d, tilt=%d, zoom=%d, ret=%d\n"),
        nDeviceID, nPan, nTilt, nZoom, nErrorCode);

    m_nPan = nPan;
    m_nTilt = nTilt;
    m_nZoom = nZoom;
    CString strTmp;
    strTmp.Format(_T("%d"), nPan);
    GetDlgItem(IDC_EDIT_PAN)->SetWindowText(strTmp);
    strTmp.Format(_T("%d"), nTilt);
    GetDlgItem(IDC_EDIT_TILT)->SetWindowText(strTmp);
    strTmp.Format(_T("%d"), nZoom);
    GetDlgItem(IDC_EDIT_ZOOM)->SetWindowText(strTmp);
```

PTZ 제어하기

ControlPtz() 함수를 호출하여 PTZ 제어 명령을 카메라로 전송합니다. ControlPtz() 함수의 파라미터로 디바이스 핸들, 컨트롤ID, PTZ 커맨드, PTZ 속도 관련 속성을 입력합니다. PTZ 커맨드는 '표 5 PTZ Command'를 참조합니다.

```
if (bIsCap)
```

```
{
    ret = m_ctrlXnsDevice.ControlPtz(
        m_hDevice,           // [in] Device handle.
        m_nControlId,       // [in] Control ID.
        nCommand,            // [in] PTZ command
        m_nPtzSpeed         // [in] If nPtzCommand is a speed-related command,
                            // this value indicates the PTZ operation speed.
                            // (1~100) This value is valid only if the camera
                            // supports the XCTL_CAP_PTZ_SPEED capability.
    );
}
```

표 5 PTZ Command

Command	Value	Description
XPTZ_UP	1	Tilt up
XPTZ_DOWN	2	Tilt down
XPTZ_LEFT	3	Pan left
XPTZ_RIGHT	4	Pan right
XPTZ_UPLEFT	5	Tilt up and pan left
XPTZ_UPRIGHT	6	Tilt up and pan right
XPTZ_DOWNLEFT	7	Tilt down and pan left
XPTZ_DOWNRIGHT	8	Tilt down and pan right
XPTZ_ZOOMIN	9	Zoom in
XPTZ_ZOOMOUT	10	Zoom out
XPTZ_STOP	11	Stop the PTZ moving
XPTZ_FOCUS_NEAR	12	Focus near
XPTZ_FOCUS_FAR	13	Focus far
XPTZ_FOCUS_STOP	14	Stop focus moving
XPTZ_IRIS_OPEN	15	Open iris
XPTZ_IRIS_CLOSE	16	Close iris

See also

[Single Live 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.
현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 11

Alarm&Event 샘플 프로그램

Alarm&Event 샘플 프로그램은 XNS ActiveX SDK를 사용하여 이벤트 알람을 제어하는 방법을 설명하기 위한 예제입니다.

Contents

- 샘플 프로그램 소개
- API 호출 순서
- 구현 방법 상세 설명
- See Also
- FAQ

샘플 프로그램 소개

Alarm&Event 샘플 프로그램은 네트워크 디바이스에 발생한 이벤트를 사용자 화면에 출력하고 처리하는 예제입니다. 먼저 XnsSdkDevice 컨트롤과 XnsSdkWindow 컨트롤을 하나씩 생성한 후에 네트워크 장비에 연결하고 미디어를 오픈합니다. 사전에 등록한 이벤트가 발생하면 화면에 이벤트 로그를 출력합니다. 샘플 프로그램 사용법은 다음과 같습니다.

- Step 1. 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.
- Step 2. 장비 모델, IP address, VNP 포트, ID/Password 등의 장비 연결 정보를 설정한 후 [Connect] 버튼을 누릅니다. 연결이 정상적으로 이루어지는지 확인합니다.
- Step 3. [Play] 버튼을 눌러 영상을 수신합니다.
- Step 4. 움직임 감지 이벤트(Motion Detection), 영상 손실 이벤트(Video Loss) 등의 이벤트가 발생하면, 다이얼로그 하단에 이벤트 로그가 출력되는지 확인합니다.

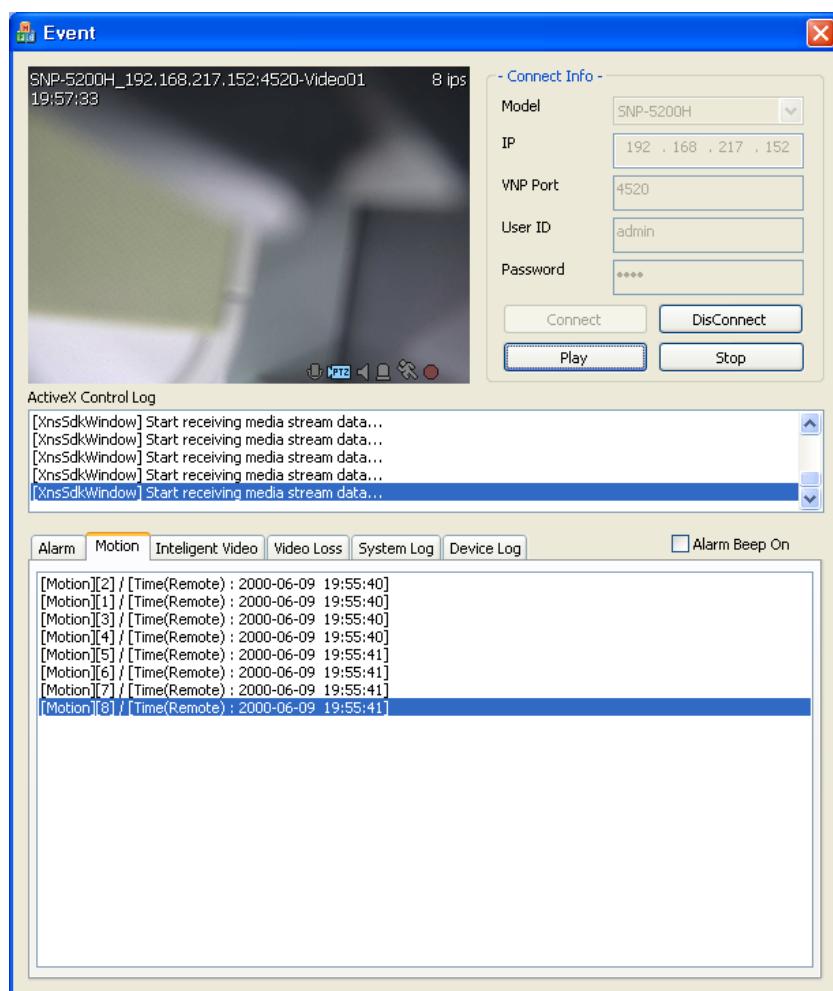


그림 22 Alarm&Event 샘플 프로그램 실행 화면

Step 5. 프로그램을 종료합니다.

API 호출 순서

이 샘플 프로그램은 XNS ActiveX SDK가 제공하는 라이브러리를 사용하여 녹화 영상 재생 기능을 구현하였습니다. 샘플 프로그램의 API 호출 순서는 다음과 같습니다.

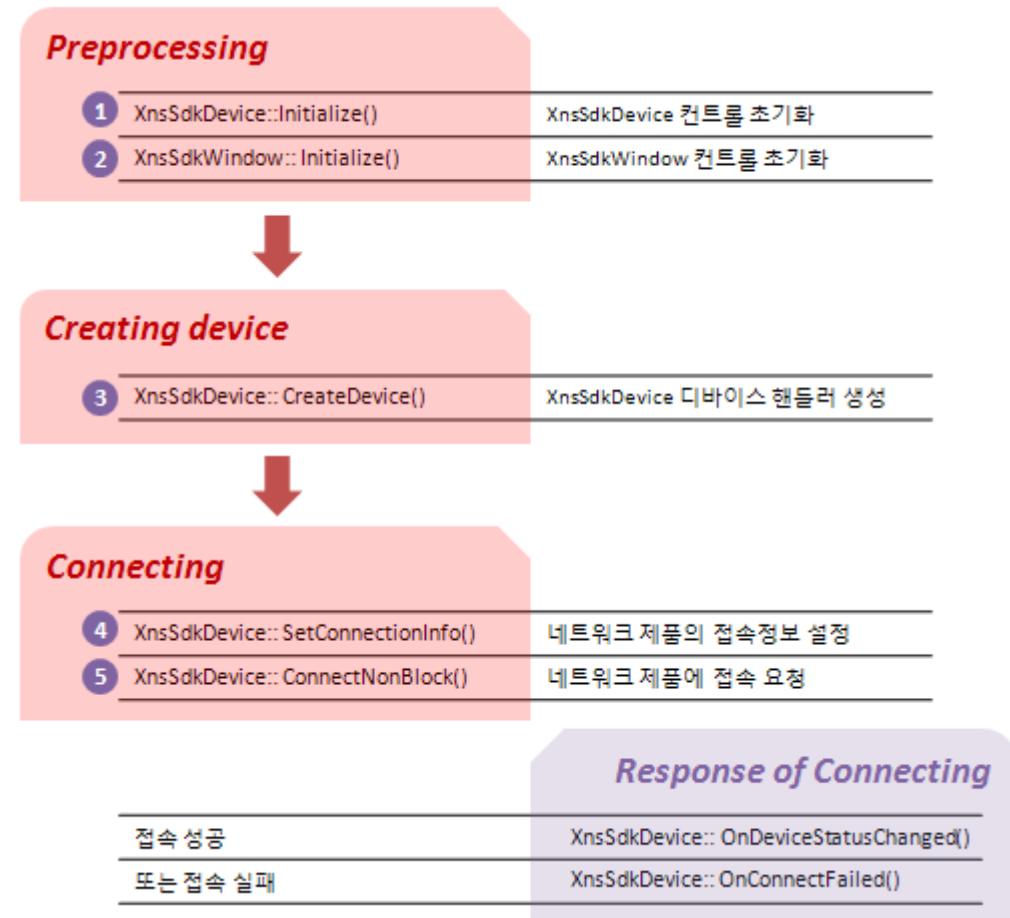




그림 23 Alarm&Event 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 Alarm&Event 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 삽입 및 초기화

참고

ActiveX 컨트롤 삽입 및 초기화 방법은 본 문서의 CHAPTER 4에 기술된 Start Up 샘플 프로그램의 ‘구현 방법 상세 설명’ 절을 참조합니다.

디바이스 핸들러 생성

참고

디바이스 핸들러 생성 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 ‘디바이스 핸들러 생성’ 절을 참조합니다.

디바이스 연결 설정

참고

디바이스 연결 설정 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 ‘디바이스 연결 설정’ 절을 참조합니다.

미디어 플레이

참고

미디어 플레이 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 ‘미디어 오픈’ 절을 참조합니다.

이벤트 등록

프로그램에서 관리할 이벤트를 사전에 등록합니다. 다음은 XNS ActiveX 라이브러리에서 제공하는 이벤트의 목록입니다.

표 6 이벤트 목록

Event	Description
OnControlLocalNameChanged	로컬 PC에서 control module의 이름이 변경되면 ActiveX object에서 OnControlLocalNameChanged 이벤트를 발생시킵니다. 또한 이 이벤트는 애플리케이션이 ChangeControlLocalName()을 이용하여 control module의 이름을 변경했을 때도 발생합니다.

OnControlRemoteNameChanged	control module 의 이름이 변경되었을 때 발생합니다
OnControlStatusChaged	이 이벤트는 장비의 연결 상태나 control module 의 기능(capability)이 변경되었을 때 발생합니다.
OnPasswordChanged	DVR 의 비밀번호가 변경되었을 때 발생합니다.
OnSensorEvent	장비에서 sensor-in(즉, alarm-in) 이벤트가 발생했음을 알려줍니다.
SetAlarm	Alarm-out 장비(digital-out device 라고도 함)를 켜고 끄는 함수입니다. 이 함수를 사용하려면 control module 이 XCTL_CAP_ALARM_ONOFF capability 를 지원해야 합니다.
OnAlarmOutControlFailed	Alarm-out 제어를 실패했을 때 발생하는 이벤트입니다.
OnMotionEvent	장비에서 모션 이벤트가 발생했음을 알려줍니다.
OnIVOccurEvent	장비에서 IV 이벤트가 발생했음을 알려줍니다.
OnVideoLossEvent	장비에서 영상 손실(video loss) 이벤트가 발생했음을 알려줍니다.

이벤트 등록 방법

Visual Studio 2008(또는 2010)에서 이벤트를 등록하는 방법은 다음과 같습니다.

Step 1. Device Control을 선택한 후, [속성] 창에서 이벤트를 선택합니다.

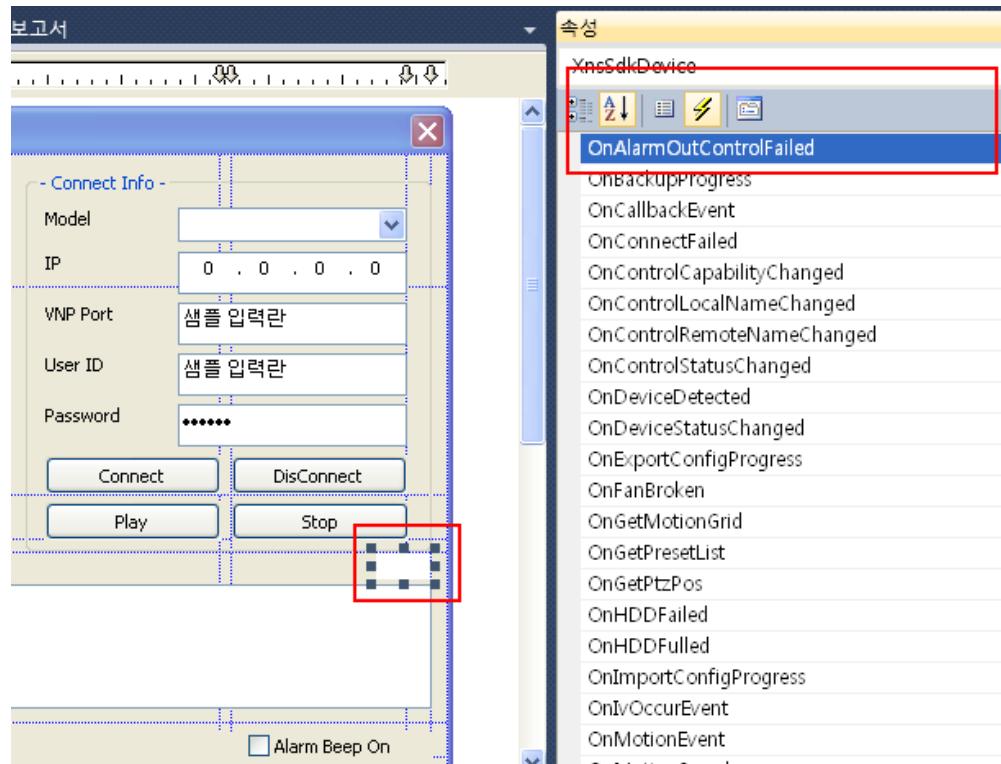


그림 24 이벤트 등록 방법

Step 2. CEventDlg 클래스에서 위에서 등록한 이벤트에 대한 이벤트 핸들러를 작성합니다.

```
// EventDlg.cpp
void CEventDlg::OnSensorEventXnssdkdevicectrl1(long nDeviceID, long nControlID, long
nSensorNumber, long tEventUTC)
{
    CString str;
    long Year, Mon, Day;
    long Hour, Min, Sec;
    long tLocal = 0;
    tLocal = m_ctrlXnsSdkDevice.UTCToDeviceLocalTime(
        m_hDevice,
        tEventUTC);
    m_ctrlXnsSdkDevice.TimetToDate(
        tLocal,
        &Year,
        &Mon,
        &Day,
        &Hour,
        &Min,
        &Sec);
    str.Format(_T("[Alarm][%d] / [Time(Remote) : %d-%02d-%02d %02d:%02d:%02d]"),
    ++m_nIndex_Sensor, Year, Mon, Day, Hour, Min, Sec);
    m_ctrlSensorLog.AddString(str);
    m_ctrlSensorLog.SetCurSel(m_ctrlSensorLog.GetCount()-1);
}
```

참고

모션 이벤트, 지능형 비디오 이벤트, 영상 손실 이벤트도 위와 동일한 방법으로 구현합니다.

Step 3. 등록한 이벤트가 control module 이름 변경에 관한 이벤트일 경우에는, 다음과 같이 이벤트에 따른 로그 처리를 하도록 프로그램을 작성합니다.

```
// EventDlg.cpp
void CEventDlg::OnControlLocalNameChangedXnssdkdevicectrl1(long nDeviceID, long
nControlID)
{
    // 이벤트에 따른 로그처리
    CString str;
    str.Format(_T("[Device Log][%d] /
```

```
[The control module is renamed in the local PC."), ++m_nIndex_Device);
m_ctrlDeviceLog.AddString(str);
m_ctrlDeviceLog.SetCurSel(m_ctrlDeviceLog.GetCount()-1);
}
```

알람 출력 설정

다이얼로그에서 [Alarm Beep On] 체크박스를 선택했을 때, Alarm-out을 켜도록 구현합니다. 반대로 체크를 해제하면, Alarm-out을 끄도록 구현합니다.

`SetAlarm()` 함수를 호출하기 전에 전체 디바이스의 개수와 `XCTL_CAP_ALARM_ONOFF`를 지원하는 디바이스의 개수, Alarm-out을 설정한 디바이스의 개수를 카운팅하여 Alarm Control ID를 얻어옵니다.

```
// EventDlg.cpp
void CEventDlg::OnBnClickedCheckSetAlarm()
{
    m_bAlarm = !m_bAlarm;

    // 전체 디바이스 개수를 파악하고, Alarm Control ID를 얻어오는 작업
    if (m_bAlarm == TRUE)
    {
        // Device Count
        for(int i=0;
            i<m_ctrlXnsSdkDevice.GetControlCount(m_hDevice, XCTL_DVR);
            i++)
        {
            m_nDeviceCount++;
        }

        // Camera Count
        for(int i=0;
            i<m_ctrlXnsSdkDevice.GetControlCount(m_hDevice, XCTL_CAMERA);
            i++)
        {
            m_nDeviceCount++;
        }

        // Alarm Count
        for(int i=0;
```

```
i<m_ctrlXnsSdkDevice.GetControlCount(m_hDevice, XCTL_ALARM);
i++)
{
    m_nDeviceCount++;
}

// Alarm Beep Count
for(int i=0;
    i<m_ctrlXnsSdkDevice.GetControlCount(m_hDevice, XCTL_ALARM_BEEP);
    i++)
{
    m_nDeviceCount++;
    m_ctrlXnsSdkDevice.SetAlarm(
        m_hDevice,
        m_nDeviceCount,
        ALARM_ON);
}
else
{
    m_ctrlXnsSdkDevice.SetAlarm(
        m_hDevice,
        m_nDeviceCount,
        ALARM_OFF);
}
}
```

See also

[Start Up 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.

현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 12

Local Recording 샘플 프로그램

Local Recording 샘플 프로그램은 XNS ActiveX SDK를 사용하여 라이브 영상을 로컬PC에 녹화하는 방법을 설명하기 위한 예제입니다.

Contents

[샘플 프로그램 소개](#)

[API 호출 순서](#)

[구현 방법 상세 설명](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

Local Recording 샘플 프로그램은 특정 네트워크 장비로부터 라이브 영상을 수신하여 로컬 PC에 녹화하는 예제입니다. 네트워크 장비에 연결하려면 우선 장비를 제어할 수 있는 XnsSdkDevice 컨트롤을 생성해야 하고, 사용자 UI를 제공하기 위한 XnsSdkWindow 컨트롤을 생성해야 합니다. 그리고 녹화 윈도우 컨트롤을 위해서 XnsSdkWindow_Recording 컨트롤을 생성합니다. 이 세 개의 컨트롤을 획득한 후에는 네트워크 장비에 연결한 후, 네트워크 장비로부터 영상이 출력되면 녹화를 시작합니다.

녹화 파일의 포맷에는 REC와 SEC가 있으며, REC 타입일 경우 애플리케이션에서 파일을 재생할 수 있습니다. SEC 타입일 경우에는 로컬 PC에 설치된 자체 플레이어를 이용하여 재생합니다.

샘플 프로그램 사용법은 다음과 같습니다.

Step 1. 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.

Step 2. 장비 연결 정보를 설정한 후 [Connect] 버튼을 누릅니다.

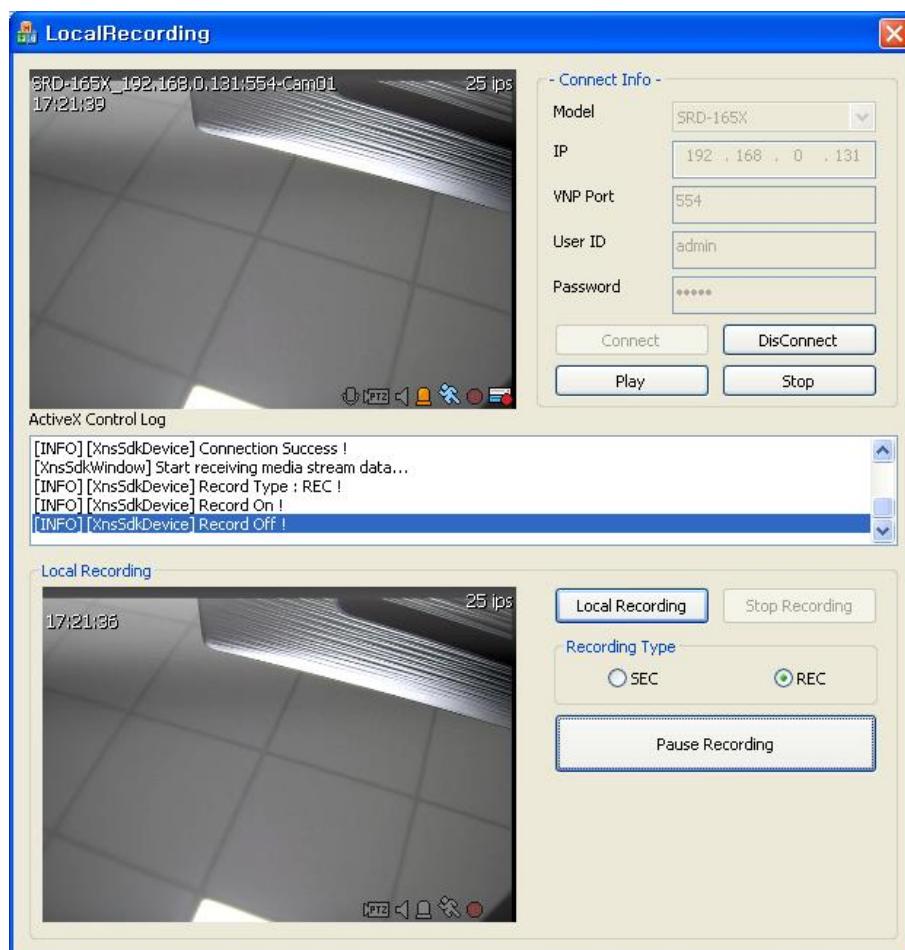


그림 25 Local Recording 샘플 프로그램 실행 화면

- Step 3. 연결이 정상적으로 이루어졌는지 확인합니다.
- Step 4. [Play] 버튼을 클릭하여 화면에 라이브 영상이 출력되는지 확인합니다.
- Step 5. 하단의 확장 화면에서 [Recording Type]을 선택한 후, [Local Recording] 버튼을 클릭합니다. 여기까지 하면 녹화가 시작됩니다.
- Step 6. [Stop Recording] 버튼을 클릭하여 녹화를 정지합니다.
- Step 7. [Play Recording] 버튼을 클릭하여 녹화된 영상을 재생합니다.

API 호출 순서

이 샘플 프로그램은 XNS ActiveX SDK가 제공하는 라이브러리를 사용하여 로컬 녹화 기능을 구현하였습니다. 샘플 프로그램의 API 호출 순서는 다음과 같습니다.

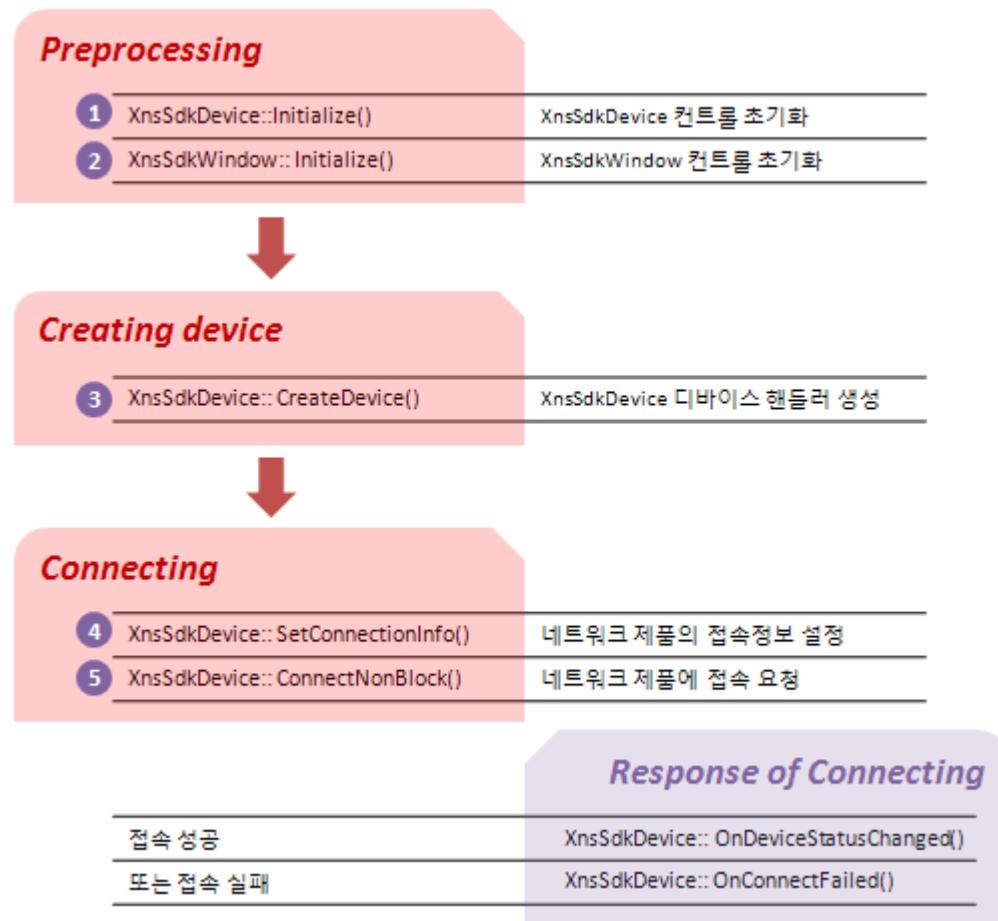




그림 26 Local Recording 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 Local Recording 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 생성

CLocalRecordingDlg 클래스에 ActiveX 컨트롤을 선언합니다.

```
// LocalRecordingDlg.cpp
DDX_Control(pDX, IDC_XNSSDKDEVICECTRL, m_ctrlXnsSdkDevice);
DDX_Control(pDX, IDC_XNSSDKWINDOWCTRL, m_ctrlXnsSdkWindow);
DDX_Control(pDX, IDC_XNSSDKWINDOWCTRL_RECORDING, m_ctrlXnsSdkWindow_Recording);
```

샘플 프로그램을 실행하면 자동으로 실행되는 초기화 함수인 OnInitXnsSdk()에서 ActiveX 컨트롤을 초기화합니다.

```
// LocalRecordingDlg.cpp
long nRet = m_ctrlXnsSdkDevice.Initialize();
nRet = m_ctrlXnsSdkWindow.Initialize(NULL, NULL);
nRet = m_ctrlXnsSdkWindow_Recording.Initialize(NULL, NULL);
```

디바이스 핸들러 생성

장비 제어를 위한 디바이스 핸들러를 생성합니다. 디바이스 핸들러를 생성하면, 장비의 정보를 저장하기 위한 메모리 공간이 할당되고, 디바이스 핸들(device handle)이 반환됩니다. 반환되는 값이 0보다 큰 정수일 경우에 유효합니다.

디바이스 핸들러는 CLocalRecordingDlg 클래스의 OnInitXnsSdk() 함수 안에서 CreateDevice()를 호출함으로써 생성됩니다. CreateDevice() 함수는 XnsSdkDevice 컨트롤에 존재하는 함수입니다.

```
// LocalRecordingDlg.cpp
m_hDevice = m_ctrlXnsDevice.CreateDevice(1);
```

디바이스 연결 설정

장비에 접속하기 전에 XnsSdkDevice 컨트롤의 SetConnectionInfo() 함수를 호출하여 디바이스 연결 정보를 설정해야 합니다. 두 번째 파라미터인 szVendorName에는 "Samsung"이라는 고정

값을 입력하고, 포트 번호 파라미터에는 VNP 포트 번호를 지정합니다.

```
// LocalRecordingDlg.cpp
m_ctrlXnsDevice.SetConnectionInfo(
    m_hDevice,
    _T("Samsung"),
    m_strmodelName,
    XADDRESS_IP,
    strIpAddress,
    m_nPort,
    0,
    m_strId,
    m_strPasswd);
```

이후 장비에 접속하기 위해서 XnsSdkDevice 컨트롤의 ConnectNonBlock() 함수를 호출합니다. 이 함수는 non-blocking 방식으로 동작하기 때문에 접속이 완료되지 않아도 바로 리턴됩니다.

```
// LocalRecordingDlg.cpp
long nRet = m_ctrlXnsDevice.ConnectNonBlock(
    m_hDevice,
    TRUE,
    TRUE);
if(nRet != ERR_SUCCESS)
{
    WLOGD(_T("ConnectNonBlock() fail: errno=[%d](%s)\n"),
        nRet, m_ctrlXnsDevice.GetErrorString(nRet));
}
```

접속 결과는 이벤트를 통해 알 수 있으며, 접속 성공 시에는 OnDeviceStatusChanged 이벤트가 발생하고, 접속 실패 시에는 OnConnectFailed 이벤트가 발생합니다.

접속 결과를 처리하기 위해서 해당 이벤트를 선언하고 이벤트 핸들러를 구현해야 합니다.

이벤트 핸들러 작성

CLocalRecordingDlg 클래스에 OnDeviceStatusChangedXnssdkdevicectrl()와 OnConnectFailedXnssdkdevicectrl() 이벤트 핸들러를 작성합니다.

```
// LocalRecordingDlg.cpp
void CLocalRecordingDlg::OnDeviceStatusChangedXnssdkdevicectrl(long nDeviceID, long
nErrorCode, long nDeviceStatus, long nHddCondition)
{
if((nErrorCode != ERR_SUCCESS) || (m_bDestroy == TRUE))
```

```

    return;

    WLOGD(_T("Connected: deviceID=%d, errcode=%d, deviceStatus=%d\n"),
          nDeviceID, nErrorCode, nDeviceStatus);

    CString str;

    // Connect Success.
    if ((nErrorCode == ERR_SUCCESS) && (nDeviceStatus == TRUE))
    {
        m_bCheckConnect = TRUE;
        if ((m_bCheckConnect == TRUE) && (m_bDestroy == FALSE))
        {
            OnEnableControlWindow(m_bCheckConnect);
            GetDlgItem(IDC_BUTTON_START_RECORDING)->EnableWindow(FALSE);
            GetDlgItem(IDC_BUTTON_STOP_RECORDING)->EnableWindow(FALSE);
            GetDlgItem(IDC_BUTTON_PLAY_RECORDING)->EnableWindow(FALSE);
            WLOGD(_T("Connection Success !"));
        }
    }
    // DisConnect Success.
    else if ((nErrorCode == ERR_SUCCESS) && (nDeviceStatus == FALSE))
    {
        WLOGD(_T("Connected: deviceID=%d, errcode=%d, deviceStatus=%d\n"),
              nDeviceID, nErrorCode, nDeviceStatus);

        m_bCheckConnect = FALSE;
        OnEnableControlWindow(m_bCheckConnect);
        GetDlgItem(IDC_BUTTON_START_RECORDING)->EnableWindow(FALSE);
        GetDlgItem(IDC_BUTTON_STOP_RECORDING)->EnableWindow(FALSE);
        GetDlgItem(IDC_BUTTON_PLAY_RECORDING)->EnableWindow(FALSE);
        WLOGD(_T("DisConnection Success !"));
    }
}

```

```

// LocalRecordingDlg.cpp
void CLocalRecordingDlg::OnConnectFailedXnssdkdevicectrl(long nDeviceID, long nErrorCode)
{

```

```

WLOGD(_T("Disconnected: deviceID=%d, errcode=[%d](%S)\n"),
nDeviceID, nErrorCode, m_ctrlXnsSdkDevice.GetErrorString(nErrorCode));
}

```

미디어 오픈

네트워크 디바이스로부터 라이브 영상을 받아서 화면에 출력하는 기능으로, 로컬 레코딩 수행 전에 장비로부터 미디어 스트림을 가져오기 위해 수행합니다.

미디어를 재생하려면 먼저 장비의 상태와 카메라 개수를 확인하고, 라이브 영상인지 여부를 확인합니다. 그리고 나서 XnsSdkDevice 컨트롤의 OpenMedia() 함수를 호출하여 미디어를 오픈합니다. 미디어를 오픈하면 OpenMedia() 함수의 파라미터(out-parameter)로 미디어 소스의 핸들(m_hMediaSource)을 얻을 수 있습니다.

XnsSdkWindow 컨트롤의 Start() 함수를 호출하여 미디어 소스 핸들을 윈도우 컨트롤 객체에 등록합니다. 이후 XnsSdkWindow 컨트롤은 미디어 스트림을 수신하여 디코딩 후 재생합니다.

```

// LocalRecordingDlg.cpp
if (m_ctrlXnsSdkDevice.GetControlType(m_hDevice, 1) & XCTL_DVR) // Device Type :DVR
{
    // XCTL_CAMERA : Camera channel of DVR.
    for(int i=0; i<m_ctrlXnsSdkDevice.GetControlCount(m_hDevice, XCTL_CAMERA); i++)
    {
        // Displays video in real time.
        if (m_ctrlXnsSdkDevice.GetControlCapability(m_hDevice, i+2, XCTL_CAP_LIVE) )
        {
            // Start getting media streams from the device.
            m_ctrlXnsSdkDevice.OpenMedia(m_hDevice,
                i+2,
                MEDIA_TYPE_LIVE,
                0,
                0,
                &m_hMediaSource);

            m_ctrlXnsSdkWindow.Start(m_hMediaSource);
            m_bPlay = TRUE;
            return;
        }
    }
}

```

로컬 레코딩 시작

[Local Recording] 버튼이 눌리면 녹화를 시작하도록, CLocalRecordingDlg 클래스의 OnBnClickedButtonStartRecording 이벤트 핸들러에서 StartLocalRecording() 함수를 호출합니다. StartLocalRecording() 함수의 인자로 파일 이름을 지정해야 하는데, 파일 이름에 파일 타입이 포함되어 있으면 정상 수행이 되지 않으므로 이점을 주의합니다. 함수 수행에 성공할 경우 ERR_SUCCESS를 반환합니다.

```
// LocalRecordingDlg.cpp
if (m_ctrlXnsSdkDevice.GetControlType(m_hDevice, 1) & XCTL_DVR)
{
    // XCTL_CAMERA : Camera channel of DVR.
    for(int i=0;
        i<m_ctrlXnsSdkDevice.GetControlCount(m_hDevice, XCTL_CAMERA); i++)
    {
        // Displays video in real time.
        if (m_ctrlXnsSdkDevice.GetControlCapability(m_hDevice, i+2, XCTL_CAP_LIVE) )
        {
            strFilePath.Format(_T("c:\WWLiveVideo"));
            if (m_bSecType == TRUE)
            {
                nErrCode = m_ctrlXnsSdkDevice.StartLocalRecording(
                    m_hDevice,
                    i+2,
                    strFilePath,
                    SEC_WRITER);
            }
            else
            {
                nErrCode = m_ctrlXnsSdkDevice.StartLocalRecording(
                    m_hDevice,
                    i+2,
                    strFilePath,
                    REC_WRITER);
            }
            WLOGD(_T("Record On !"));
            return;
        }
    }
}
```

```
}
```

로컬 레코딩 종료

로컬 레코딩을 중지하려면 XnsSdkDevice 컨트롤의 StopLocalRecording() 함수를 호출합니다. 함수의 두 번째 인자로 컨트롤ID를 입력해야 합니다. DVR의 경우 컨트롤ID가 2번부터 시작하고 그외 장비의 경우에는 3번부터 시작합니다.

```
// LocalRecordingDlg.cpp
m_ctrlXnsDevice.StopBackup(m_hDevice, m_nControlId);
```

로컬 레코딩 파일 재생

로컬 파일의 형식이 REC일 경우에만 애플리케이션에서 재생할 수 있습니다. 샘플 프로그램에서는 [Play Recording] 버튼이 눌렸을 때 CreateFileReader() 함수를 호출하여 파일 재생기를 생성하도록 구현했습니다. 이 기능을 구현하려면 [Play Recording] 버튼을 생성하고, 버튼에 대한 이벤트 핸들러를 작성해야 합니다.

CLocalRecordingDlg 클래스에 OnBnClickedButtonPlayRecording 이벤트 핸들러를 작성합니다. 먼저 CreateFileReader() 함수의 입력 파라미터로 파일 이름을 입력하며, 파일 재생기의 핸들을 반환합니다. 그리고 나서 GetMediaSource() 함수를 호출하여 파일 재생기 핸들로부터 미디어 소스의 핸들을 얻어옵니다. 그런 다음 XnsSdkWindow_Recording 컨트롤의 Start() 함수를 호출함으로써 미디어 소스의 핸들을 재생 윈도우(XnsSdkWindow) 컨트롤에 등록합니다. 여기까지 하면 로컬 레코딩 파일의 미디어 스트림을 수신하게 됩니다.

이제 로컬 레코딩 파일을 재생하려면 XnsSdkDevice 컨트롤의 PlayReader() 함수를 호출합니다. 만약 파일 재생을 일시적으로 중지하려는 경우에는 XnsSdkDevice 컨트롤의 PauseReader() 함수를 호출하여 파일을 일시 정지시킬 수 있습니다.

```
// LocalRecordingDlg.cpp
CString strFilePath;
strFilePath.Format(_T("c:\WWLiveVideo"));
int hTimeline;
long m_hFileReader = m_ctrlXnsSdkDevice.CreateFileReader(strFilePath);
m_hMediaSource = m_ctrlXnsSdkDevice.GetMediaSource(m_hFileReader);
if (m_bPlayRecord == FALSE)
{
    m_ctrlXnsSdkWindow_Recording.Start(m_hMediaSource);
    long nErrorCode = m_ctrlXnsSdkDevice.PlayReader( m_hFileReader, SPEED_1);
    m_bPlayRecord = TRUE;
    GetDlgItem(IDC_BUTTON_PLAY_RECORDING)->SetWindowText(_T("Pause Recording"));
```

```
    }  
else  
{  
    long nErrCode = m_ctrlXnsSdkDevice.PauseReader(m_hFileReader);  
    m_bPlayRecord = FALSE;  
    GetDlgItem(IDC_BUTTON_PLAY_RECORDING)->SetWindowText(_T("Play Recording"));  
}
```

주의

로컬 레코딩 파일 Read 시, 파일 타입(확장자)은 제외하고 경로와 이름만 반영하여 읽어야 실행 오류가 없습니다.

See also

[Start Up 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.

현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 13

VideoRawData 샘플 프로그램

VideoRawData 샘플 프로그램은 XNS ActiveX SDK를 사용하여 Video Raw Data를 가지고 오는 방법을 설명하기 위한 예제입니다.

Contents

[샘플 프로그램 소개](#)

[API 호출 순서](#)

[구현 방법 상세 설명](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

VideoRawData 샘플 프로그램은 네트워크 디바이스에서 미디어 스트림을 받을 때마다 콜백 함수를 호출함으로써 Video Raw Data를 얻어오는 예제입니다. 다른 예제와 마찬가지로 XnsSdkDevice 컨트롤과 XnsSdkWindow 컨트롤을 하나씩 생성한 후에 네트워크 장비에 연결하고 미디어를 오픈합니다. 미디어 오픈을 통해 획득한 미디어 소스 핸들을 윈도우 컨트롤에 등록한 후에, 다시 이 핸들을 콜백함수에 등록합니다. 이로써 미디어 스트림을 받을 때마다 콜백 함수가 호출되도록 애플리케이션을 구현합니다.

샘플 프로그램 사용법은 다음과 같습니다.

- Step 1. 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.
- Step 2. 장비 모델, IP address, VNP 포트, ID/Password 등의 장비 연결 정보를 설정한 후 [Connect] 버튼을 누릅니다. 연결이 정상적으로 이루어지는지 확인합니다.

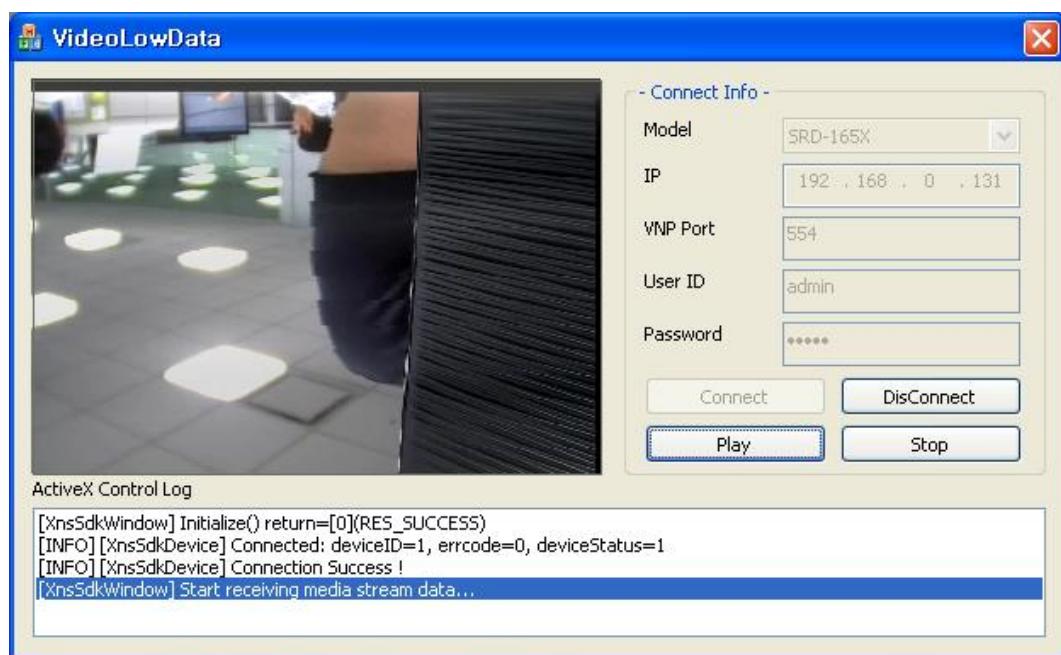
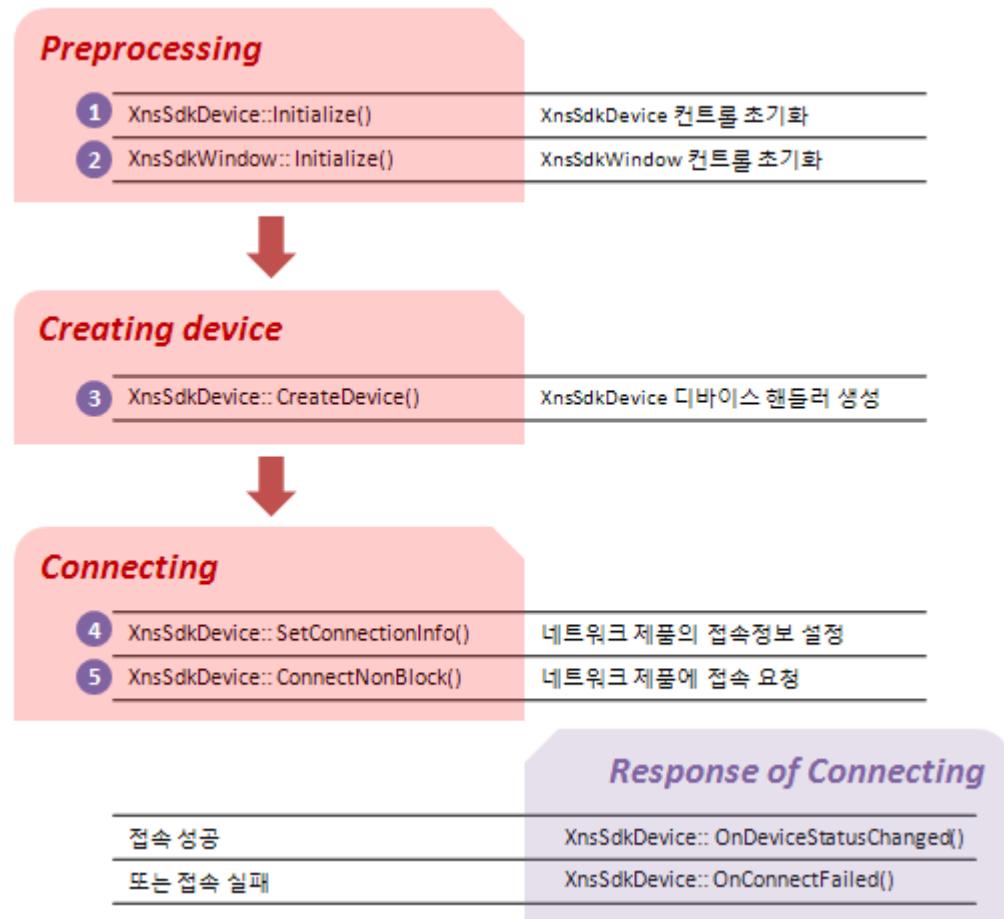


그림 27 VideoRawData 샘플 프로그램 실행 화면

- Step 3. [Play] 버튼을 눌러 라이브 영상이 출력되는지 확인합니다.
- Step 4. [Stop] 버튼을 눌러 영상 출력이 중지되는지 확인합니다.
- Step 5. 프로그램을 종료합니다.

API 호출 순서

이 샘플 프로그램은 XNS ActiveX SDK가 제공하는 라이브러리를 사용하여 Video Raw Data를 가져오는 기능을 구현하였습니다. 샘플 프로그램의 API 호출 순서는 다음과 같습니다.



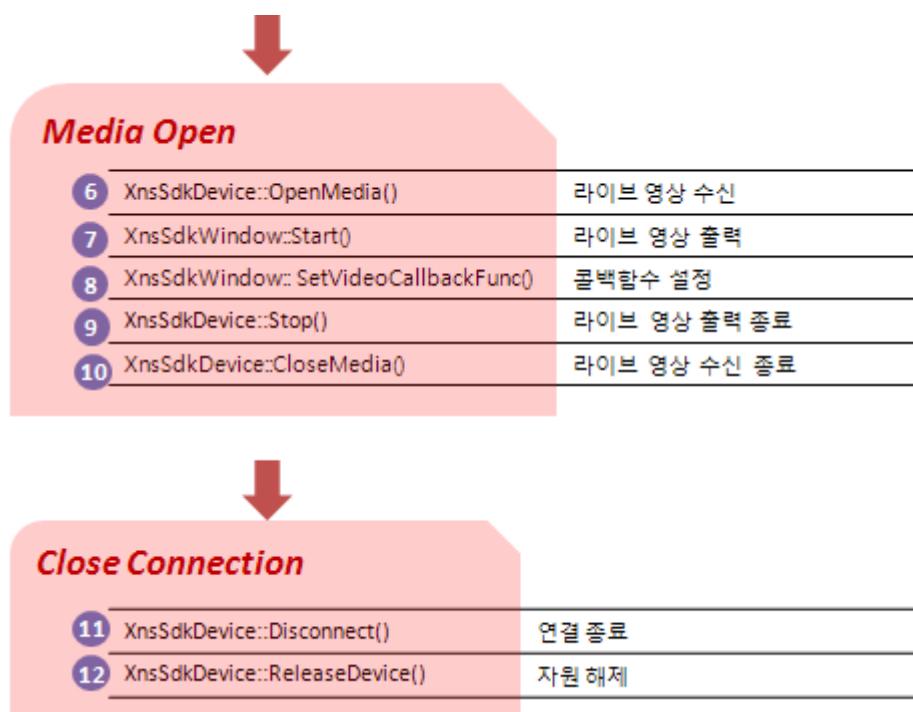


그림 28 VideoRawData 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 VideoRawData 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 삽입 및 초기화

참고

ActiveX 컨트롤 삽입 및 초기화 방법은 본 문서의 CHAPTER 4에 기술된 Start Up 샘플 프로그램의 ‘구현 방법 상세 설명’ 절을 참조합니다.

디바이스 핸들러 생성

참고

디바이스 핸들러 생성 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 ‘디바이스 핸들러 생성’ 절을 참조합니다.

디바이스 연결 설정

참고

디바이스 연결 설정 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '디바이스 연결 설정' 절을 참조합니다.

미디어 플레이

Play 함수 부분 구현

디바이스와 PC간 데이터 송수신을 위한 미디어 소스 핸들을 획득한 후에, 이 핸들을 윈도우 컨트롤에 등록합니다.

미디어 소스 핸들을 획득하려면 XnsSdkDevice 컨트롤의 OpenMedia() 함수를 호출하여 미디어 스트림 수신을 시작합니다. OpenMedia() 함수의 파라미터(out-parameter)로 미디어 소스의 핸들(m_hMediaSource)을 얻을 수 있습니다.

XnsSdkWindow 컨트롤의 Start() 함수를 호출하여 미디어 소스 핸들을 윈도우 컨트롤 객체에 등록합니다. XnsSdkWindow 컨트롤은 해당 핸들의 미디어 스트림을 수신하여 디코딩한 후 화면에 출력합니다.

Video Raw Data를 가지고 오려면 Start()를 호출한 후에 SetVideoCallbackFunc() 함수를 호출하여 이미지 스트림이 올 때마다 이벤트를 받아 처리합니다. 이때 함수의 원형을 함수 포인터로 해서 인자 값으로 넘겨 주어도 되며, 함수 포인터 변수를 이용하여 인자 값을 넘겨주어도 됩니다. 단, 중요한 것은 콜백함수로 호출할 함수는 클래스 내 메소드가 되어서는 안되며, friend로 선언해 주어야 합니다.

```
// VideoRawDataDlg.cpp
if (m_ctrlXnsSdkDevice.GetControlType(m_hDevice, 1) & XCTL_DVR) // Device Type : DVR
{
    // XCTL_CAMERA : Camera channel of DVR.
    for(int i=0; i<m_ctrlXnsSdkDevice.GetControlCount(m_hDevice, XCTL_CAMERA); i++)
    {
        // Displays video in real time.
        if (m_ctrlXnsSdkDevice.GetControlCapability(m_hDevice, i+2, XCTL_CAP_LIVE) )
        {
            // Start getting media streams from the device.
            m_ctrlXnsSdkDevice.OpenMedia(m_hDevice,
                i+2,
                MEDIA_TYPE_LIVE,
                0,
```

```

        0,
        &m_hMediaSource);

m_ctrlXnsSdkWindow.Start(m_hMediaSource);

m_ctrlXnsSdkWindow.SetVideoCallbackFunc(
    m_hMediaSource,
    (long)m_pCallbackFunc);

m_bPlay = TRUE;
return;
}

}

}

```

콜백 함수 호출 함수

콜백 함수에서 호출하는 ReceiveDecodingVideoFrame() 함수는 CVideoRawDataDlg 클래스 내 메소드가 아니며, 이미지 데이터가 전송될 때마다 호출되는 함수입니다. 이 함수는 CVideoRawDataDlg 클래스의 OnVideoFrame() 이벤트를 호출하며, OnVideoFrame() 함수는 실제로 이미지를 복사합니다. OnVideoFrame() 함수는 OnPaint 이벤트를 호출하여 복사한 이미지를 새로 그리도록 합니다. 이때 이미지 복사는 RGB 타입으로 이루어 집니다.

```

// VideoRawDataDlg.cpp
void ReceiveDecodingVideoFrame(long hMediaSource, long nWidth, long nHeight, long
nFrameSize, unsigned char* pVideo)
{
    ((CVideoRawDataDlg*)AfxGetApp()->m_pMainWnd)->
OnVideoFrame(hMediaSource, nWidth, nHeight, nFrameSize, pVideo);
}

void CVideoRawDataDlg::OnVideoFrame(long hMediaSource, long nWidth, long nHeight, long
nFrameSize, unsigned char* pVideo)
{
    m_nWidth = nWidth;
    m_nHeight = nHeight;

    ZeroMemory( &m_bmiHeader, sizeof(BITMAPINFOHEADER) );
    m_bmiHeader.biWidth           = m_nWidth;
    m_bmiHeader.biHeight          = m_nHeight*(-1);
}

```

```

m_bmiHeader.biSize          = sizeof(BITMAPINFOHEADER);
m_bmiHeader.biPlanes        = 1;
m_bmiHeader.biBitCount      = 24;
m_bmiHeader.biCompression   = BI_RGB;
m_bmiHeader.biClrImportant  = 0;
m_bmiHeader.biSizeImage     = m_nWidth * m_nHeight *
(m_bmiHeader.biBitCount / 8);

for(int i = 0 ; i < nHeight*nWidth; i++)
{
    memcpy(&m_pVideoBuf[i*3],&pVideo[i*4],3);
}
Invalidate(FALSE);
}

```

See also

[Start Up 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.
현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 14

Backup 샘플 프로그램

Backup 샘플 프로그램은 XNS ActiveX SDK를 사용하여 장비에 녹화된 데이터를 로컬 PC에 백업하는 방법을 설명하기 위한 예제입니다.

Contents

[샘플 프로그램 소개](#)

[API 호출 순서](#)

[구현 방법 상세 설명](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

Backup 샘플 프로그램은 특정 네트워크 장비에 녹화된 스트리밍을 로컬 PC에 백업하는 예제입니다. 네트워크 장비에 연결하려면 우선 장비를 제어할 수 있는 XnsSdkDevice 컨트롤을 획득해야 하고, 사용자 UI를 제공하기 위한 XnsSdkWindow 컨트롤을 획득해야 합니다. 두 개의 컨트롤을 획득한 후에는 디바이스 핸들러를 생성하고 장비간 연결을 시도하며, 연결 성공 시에 스트리밍 데이터 백업을 실행합니다.

백업 파일의 포맷에는 REC와 SEC가 있으며, REC 타입일 경우 애플리케이션에서 파일을 재생할 수 있습니다. SEC 타입일 경우에는 로컬 PC에 설치된 자체 플레이어를 이용하여 재생합니다.

샘플 프로그램 사용법은 다음과 같습니다.

- Step 1. 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.
- Step 2. 장비 연결 정보를 설정한 후 [CONNECT] 버튼을 누릅니다.
- Step 3. 연결이 정상적으로 실행되는지 확인합니다.
- Step 4. [Backup] 버튼을 클릭하여 팝업된ダイ얼로그에서 백업 시간 정보를 입력합니다.

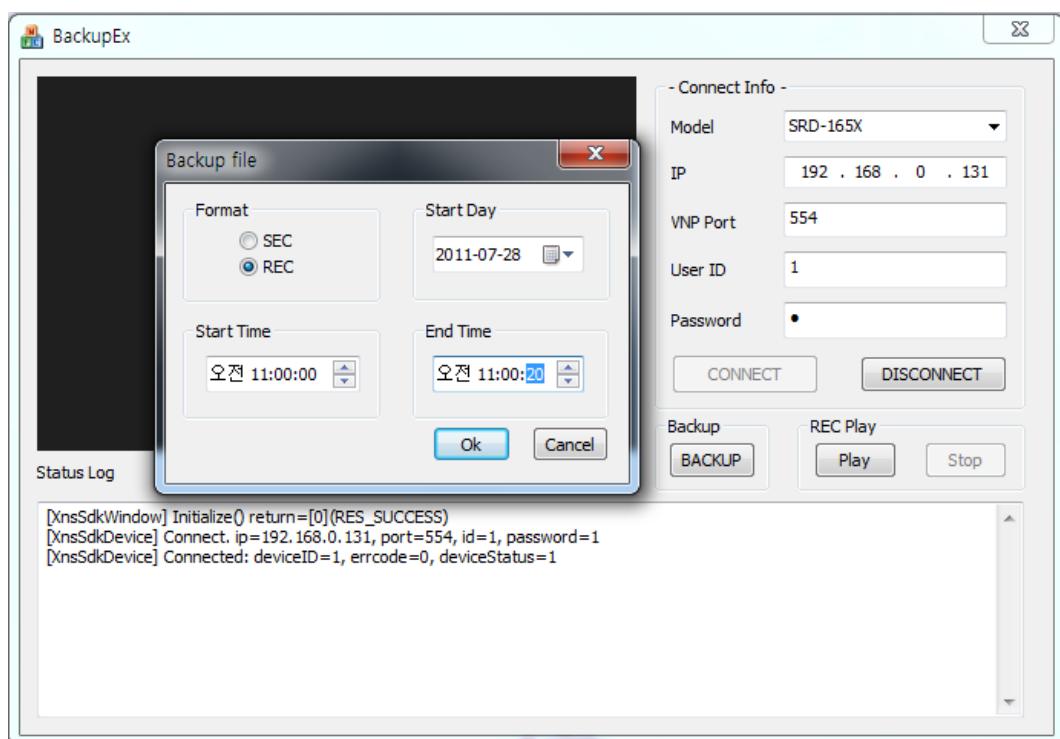


그림 29 Backup 샘플 프로그램 실행 화면(1)

- Step 5. 백업 파일을 재생하려면 메인 화면에서 [REC Playback] 버튼을 클릭합니다. 백업 파일 타입이 REC인 경우, 메인 화면에 백업 파일이 재생됩니다. (SEC 파일 타입일 경우, 자체 플레이어로 재생해야 합니다)

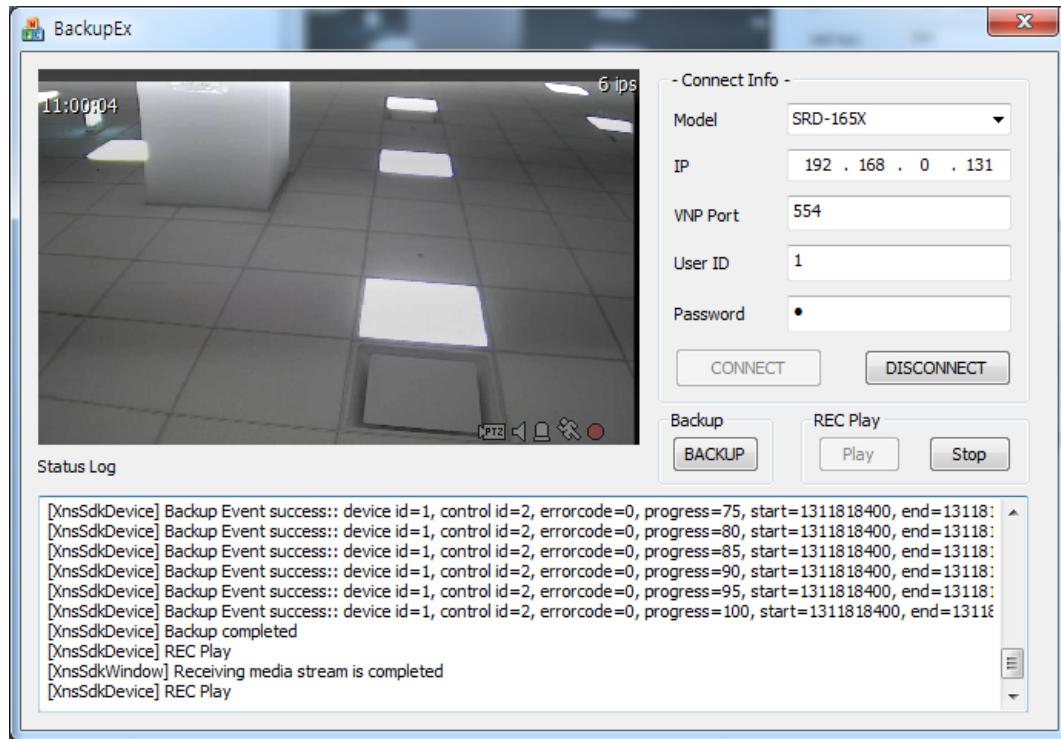


그림 30 Backup 샘플 프로그램 실행 화면(2)

Step 6. 백업 파일 재생을 마치려면 [Pause REC] 버튼을 클릭합니다.

Step 7. 백업을 마치려면 [DISCONNECT] 버튼을 눌러 장비와의 연결을 해제합니다.

API 호출 순서

이 샘플 프로그램은 XNS ActiveX SDK가 제공하는 라이브러리를 사용하여 백업 기능을 구현하였습니다. 샘플 프로그램의 API 호출 순서는 다음과 같습니다.

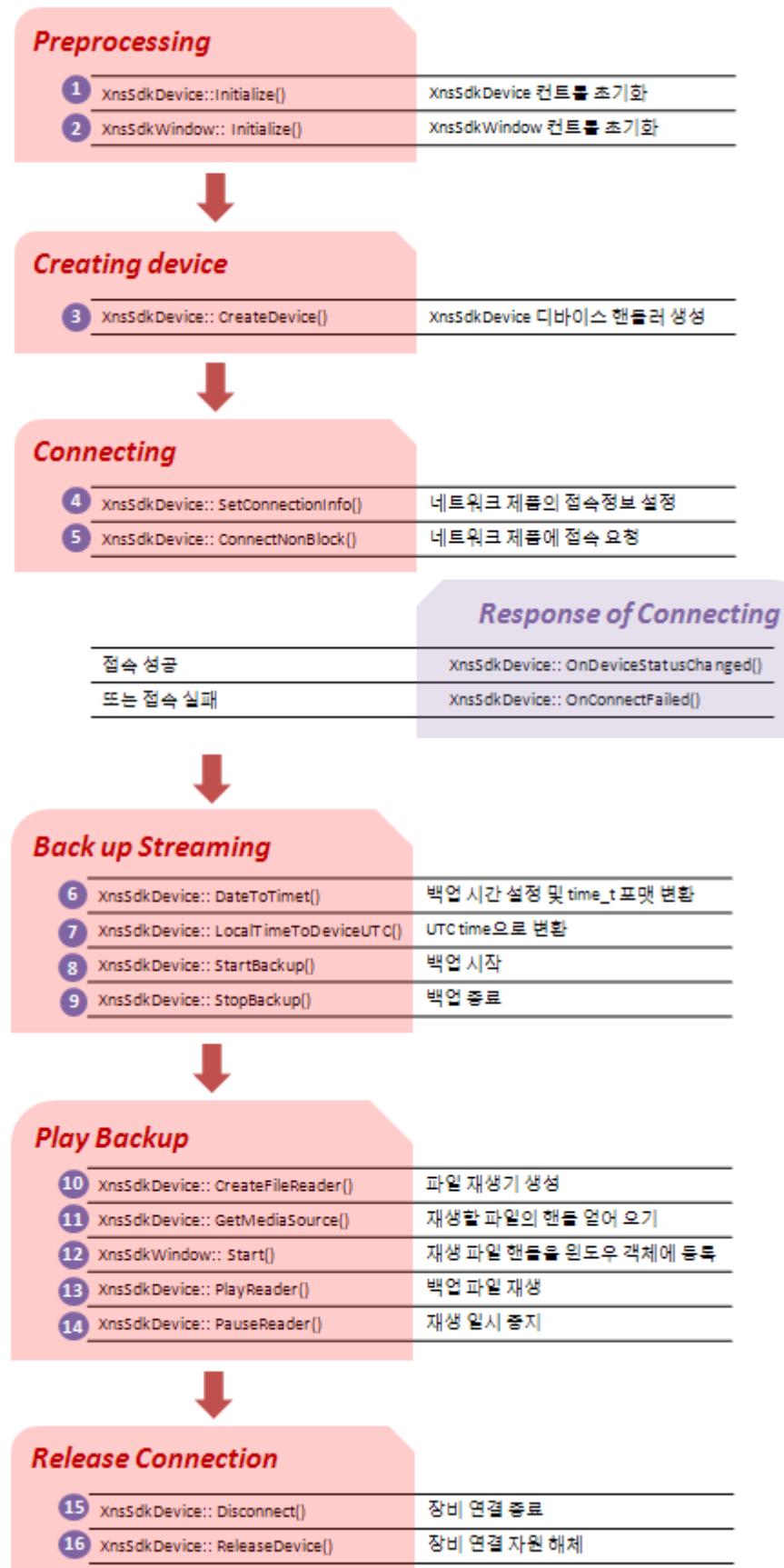


그림 31 Backup 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 생성

XnsSdkDevice::initialize() 함수를 호출하여 XnsSdkDevice 컨트롤을 초기화합니다. XnsActiveX 라이브러리를 정상적으로 사용하기 위해서는 config.xml, device.xml, xns.xml 파일이 필요하고, xns.xml 파일에 DLL 파일의 경로가 명시되어 있어야 합니다. DLL 파일의 경로는 512 바이트를 넘을 수 없습니다. XnsActiveX 라이브러리는 기본적으로 {\$SDK path}\Config 경로에 설치된 XnsSDKDevice.ocx 파일을 사용해서 xns.xml 파일을 찾습니다.

CBackupExDlg 클래스의 OnInitDialog() 안에서 XnsSdkDevice::initialize() 함수를 호출합니다.

```
// BackupExDlg.cpp
long nRet = m_ctrlXnsDevice.Initialize();
```

디바이스 핸들러 생성

장비 제어를 위한 디바이스 핸들러를 생성합니다. 디바이스 핸들러를 생성하면, 장비의 정보를 저장하기 위한 메모리 공간이 할당되고, 디바이스 핸들(device handle)이 반환됩니다. 반환되는 값이 0보다 큰 정수일 경우에 유효합니다.

디바이스 핸들러는 CBackupExDlg 클래스의 OnInitDialog() 함수 안에서 CreateDevice()를 호출함으로써 생성됩니다. CreateDevice() 함수는 XnsSdkDevice 컨트롤에 존재하는 함수입니다.

```
// BackupExDlg.cpp
m_hDevice = m_ctrlXnsDevice.CreateDevice(1);
```

백업 시간 설정

백업 시작 시간과 종료 시간을 설정합니다.

BackupSearchDlg 클래스의 멤버로 백업 시작시간과 백업 종료시간이 선언되어 있습니다. 백업에 대한 사용자 입력이 완료되면 호출되는 OnBnClickedButtonBackupOk 이벤트 핸들러에서 입력 시간을 time_t 포맷으로 변환합니다.

```
// BackupSearchDlg.cpp
m_nStartTime = m_ctrlXnsDevice->DateToTimet(
```

```

        m_nYear,
        m_nMonth,
        m_nDay,
        m_nStartHour,
        m_nStartMin,
        m_nStartSec);

m_nEndTime = m_ctrlXnsDevice->DateToTimet(m_nYear, m_nMonth, m_nDay, m_nEndHour,
m_nEndMin, m_nEndSec);

```

그런 다음 CBackupExDlg 클래스의 OnBnClickedButtonBackup 이벤트 핸들러에서 time_t 포맷의 백업 시간을 UTC Time 포맷으로 변환합니다. UTC Time으로 변환하는 이유는 Local PC와 장비의 시간을 동기화 해서 백업 시작 시간과 종료 시간을 정확히 맞추기 위해서입니다.

```

// BackupExDlg.cpp
long tStart = m_ctrlXnsDevice.LocalTimeToDeviceUTC(
    m_hDevice,
    m_nStartTime);
long tEnd = m_ctrlXnsDevice.LocalTimeToDeviceUTC(m_hDevice, m_nEndTime);

```

UTC Time

UTC(Coordinated Universal Time) 설명 보충할 것

백업 시작

[Backup] 팝업창에서 [OK] 버튼이 눌리면 백업을 시작하도록, CBackupExDlg 클래스의 OnBnClickedButtonBackup 이벤트 핸들러에서 StartBackup() 함수를 호출합니다. 함수의 인자로 디바이스 핸들, 컨트롤ID, 백업 파일 이름, 백업 파일 포맷, 백업 시작 시간, 백업 종료 시간을 입력합니다.

장비가 DVR일 경우 컨트롤ID는 2번부터 시작하므로 i+2의 값을 입력합니다. 파일 이름 지정 시에는 파일 타입이 포함되어 있으면 정상적으로 수행되지 않습니다. 예를 들어, "c:\WWtest"와 같이 파일 경로만 지정해 주어야 합니다. 파일 포맷은 SEC일 경우 '1'을 입력하고, REC일 경우 '2'를 입력합니다. 백업 시작 시간과 백업 종료 시간은 UTC time 포맷으로 지정해야 합니다.

함수 수행에 성공할 경우 ERR_SUCCESS를 반환하며, 백업이 진행되는 동안 진척도가 5% 증가 할 때마다 주기적으로 OnBackupProgress() 이벤트가 발생합니다.

```

// BackupExDlg.cpp
if (m_ctrlXnsDevice.GetControlCapability(m_hDevice, 1, XCTL_CAP_FSPEED1))
{

```

```

for(int i=0; i<m_ctrlXnsDevice.GetControlCount(m_hDevice, XCTL_CAMERA); i++)
{
    if (m_ctrlXnsDevice.GetControlCapability(m_hDevice, i+2, XCTL_CAP_PLAYBACK))
    {
        m_nControlId = i+2;
        break;
    }
}
CString strFilePath = _T("c:\WWWBackupVideo");
long errCode = m_ctrlXnsDevice.StartBackup(
    m_hDevice,
    m_nControlId,
    strFilePath,
    m_nFileType,
    tStart,
    tEnd);

if(errCode != ERR_SUCCESS)
{
    ErrorMessage(_T("StartBackup() fail: error=%d(%s)\n"), errCode,
                m_ctrlXnsDevice.GetErrorString(errCode));
    return;
}

```

백업 종료

백업을 중지하려면 StopBackup() 함수를 호출합니다. 입력 파라미터로 디바이스 핸들과 컨트롤 ID를 입력합니다. 정상적으로 중지될 경우 ERR_SUCCESS를 반환합니다.

```
// BackupExDlg.cpp
m_ctrlXnsDevice.StopBackup(m_hDevice, m_nControlId);
```

백업 파일 재생

백업 파일의 형식이 REC일 경우에만 애플리케이션에서 재생할 수 있습니다. 샘플 프로그램에서는 [Play] 버튼이 눌렸을 때 CreateFileReader() 함수를 호출하여 파일 재생기를 생성하도록 구현했습니다. 이 기능을 구현하려면 [REC Playback] 버튼을 생성하고, 버튼에 대한 이벤트 핸

들러를 작성합니다.

버튼 이벤트 핸들러 생성

CBackupExDlg 클래스에 OnBnClickedButtonReplayback 이벤트 핸들러를 작성합니다.

먼저 CreateFileReader() 함수의 입력 파라미터로 파일 이름을 입력하며, 파일 재생기의 핸들을 반환합니다. 그리고 나서 GetMediaSource() 함수를 호출하여 파일 재생기 핸들로부터 미디어 소스의 핸들을 얻어옵니다. 그런 다음 XnsSdkWindow 컨트롤의 Start() 함수를 호출함으로써 미디어 소스의 핸들을 윈도우(XnsSdkWindow) 컨트롤에 등록합니다. 여기까지 하면 장비로부터 미디어 데이터를 수신하게(백업하게) 됩니다.

백업 파일을 재생하려면 XnsSdkDevice 컨트롤의 PlayReader() 함수를 호출합니다. 만약 파일 재생을 일시적으로 중지하려는 경우에는 XnsSdkDevice 컨트롤의 PauseReader() 함수를 호출하여 파일을 일시 정지시킬 수 있습니다.

```
// BackupExDlg.cpp
if(m_ctrlXnsWindow.IsMedia() == false)
{
    m_hFileReader = m_ctrlXnsDevice.CreateFileReader(m_strFile);
    if(!m_hFileReader)
    {
        ErrorMessage(_T("No file to play\n"));
        return;
    }
    m_hMediaSource = m_ctrlXnsDevice.GetMediaSource(m_hFileReader);
    m_ctrlXnsWindow.Start(m_hMediaSource);
}
Long nErrorCode = m_ctrlXnsDevice.PlayReader(
    m_hFileReader,           // [in] Handle of the file to play
    SPEED_1);                // [in] Play speed
```

주의

파일 이름에는 확장자가 포함될 수 없으며, _1 또는 _2와 같은 suffix는 사용할 수 없습니다. 파일 저장 시에 1시간 단위로 파일을 분할하여 저장하는데 이때 파일 이름 뒤에 '_숫자' 형식의 suffix를 사용하기 때문입니다.

참고

백업 파일의 형식이 SEC일 경우에는 애플리케이션에서 재생할 수 없고 자체 플레이어를 이용해야 합니다.

See also

[Single Live 샘플 프로그램](#)

[Playback 샘플 프로그램](#)

[Local Recording 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.

현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 15

Listen&Talk 샘플 프로그램

Listen&Talk 샘플 프로그램은 XNS ActiveX SDK를 사용하여 애플리케이션과 디바이스 간에 오디오 데이터를 송수신하는 방법을 설명하기 위한 예제입니다.

Contents

[샘플 프로그램 소개](#)

[API 호출 순서](#)

[구현 방법 상세 설명](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

Listen&Talk 샘플 프로그램은 애플리케이션과 네트워크 디바이스 간 오디오 스트림 송수신 기능을 구현한 예제입니다. 다른 예제와 마찬가지로 XnsSdkDevice 컨트롤과 XnsSdkWindow 컨트롤을 하나씩 생성한 후에 네트워크 장비에 연결하고 미디어를 오픈합니다. 미디어를 오픈한 후 미디어 음성 수신 설정 및 미디어 음성 송출 설정을 합니다.

샘플 프로그램 사용법은 다음과 같습니다.

- Step 1.** 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.
- Step 2.** 장비 모델, IP address, VNP 포트, ID/Password 등의 장비 연결 정보를 설정한 후 [Connect] 버튼을 누릅니다. 연결이 정상적으로 이루어지는지 확인합니다.

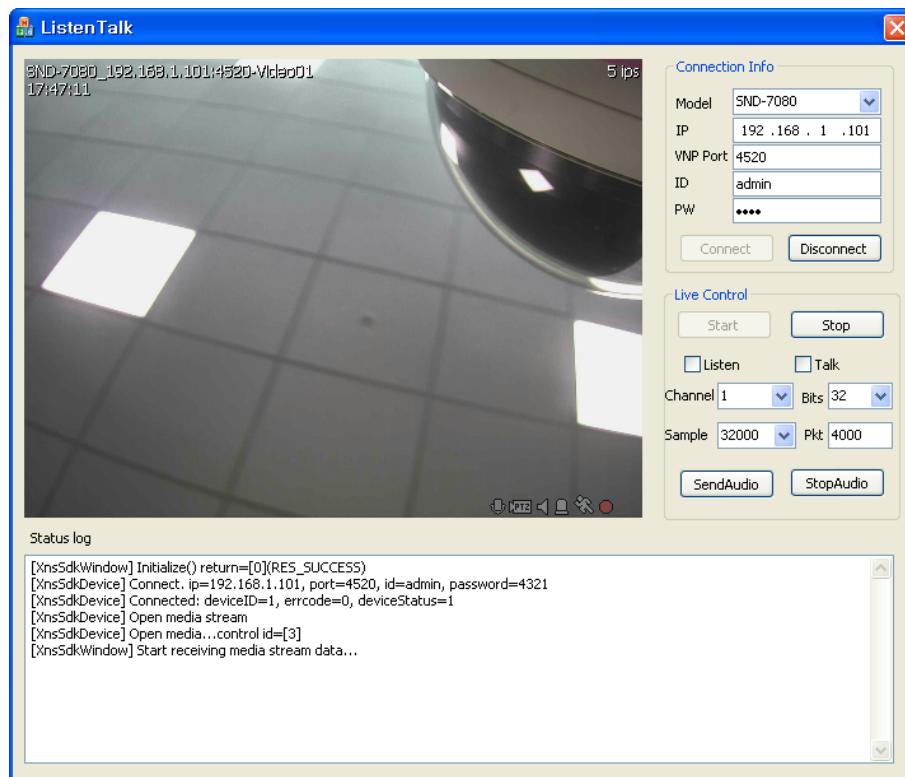


그림 32 Listen&Talk 샘플 프로그램 실행 화면

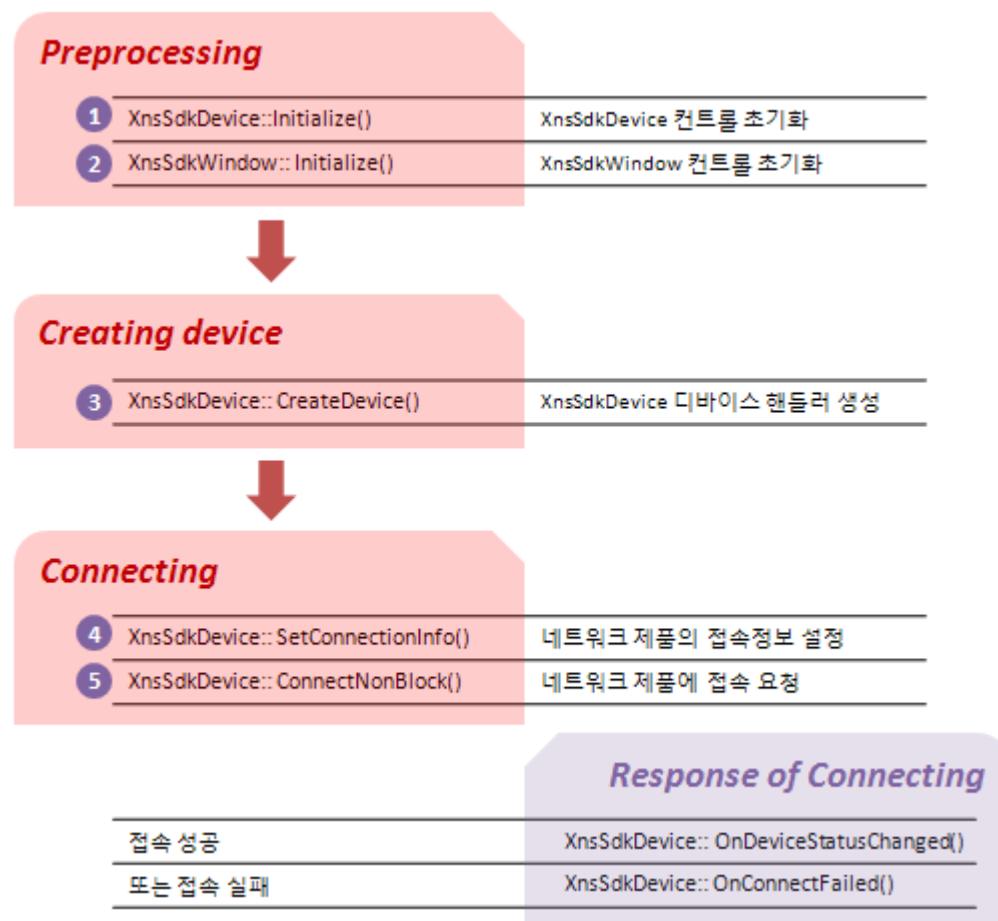
- Step 3.** [Start] 버튼을 눌러 라이브 영상이 출력되는지 확인합니다.
- Step 4.** [Listen] 체크박스를 선택하여, 오디오 수신이 정상적으로 이루어지는지 확인합니다. 이때 디바이스는 Audio-in을 통해서 오디오 입력을 받고 있는 상태여야 합니다.
- Step 5.** [Talk] 체크박스를 선택합니다.
- Step 6.** Talk 기능 사용 시 마이크로 입력된 음성은 PCM 인코딩을 거친 후 디바이스에 전송됩니다.

이때 필요한 Channel, Bits per sample, Sample per second, 데이터 패킷 사이즈를 설정합니다.

- Step 7. [SendAudio] 버튼을 클릭하여 오디오 송출이 정상적으로 이루어지는지 확인합니다. 이때 사용자 단말기는 마이크를 통해 음성을 입력 받아야 하고, 원격 디바이스 장치는 Audio-out을 통해 음성 출력을 하고 있는 상태여야 합니다.
- Step 8. [Stop]을 실행하여 영상 출력이 중지되는지 확인합니다.
- Step 9. 프로그램을 종료합니다.

API 호출 순서

이 샘플 프로그램은 XNS ActiveX SDK가 제공하는 라이브러리를 사용하여 Video Raw Data를 가져오는 기능을 구현하였습니다. 샘플 프로그램의 API 호출 순서는 다음과 같습니다.



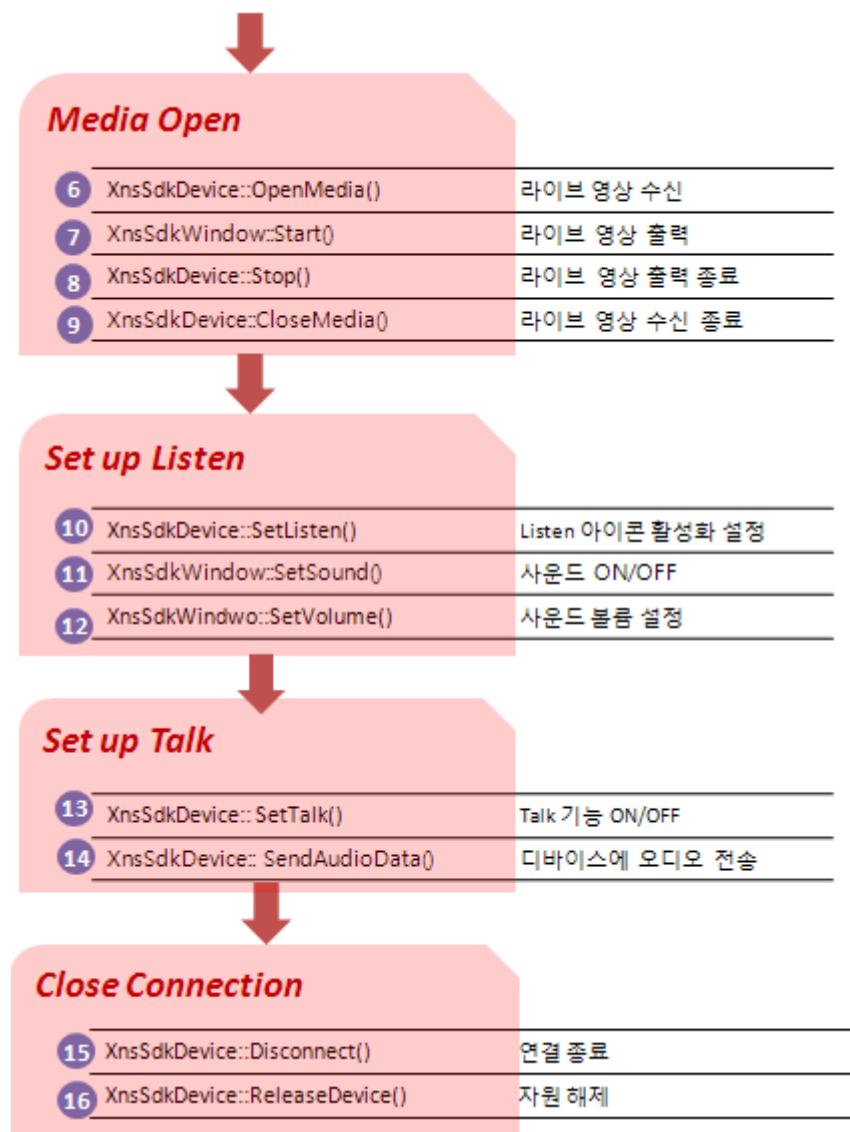


그림 33 Listen&Talk 샘플 프로그램 API 호출 순서

구현 방법 상세 설명

이 절에서는 Listen&Talk 샘플 프로그램 구현 방법을 상세히 설명합니다.

ActiveX 컨트롤 삽입 및 초기화

참고

ActiveX 컨트롤 삽입 및 초기화 방법은 본 문서의 CHAPTER 4에 기술된 Start Up 샘플 프로그램의 '구현 방법 상세 설명' 절을 참조합니다.

디바이스 핸들러 생성

참고

디바이스 핸들러 생성 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '디바이스 핸들러 생성' 절을 참조합니다.

디바이스 연결 설정

참고

디바이스 연결 설정 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '디바이스 연결 설정' 절을 참조합니다.

미디어 플레이

참고

디바이스 연결 설정 방법은 본 문서의 CHAPTER 5에 기술된 Single Live 샘플 프로그램의 '미디어 오픈디바이스 연결 설정' 절을 참조합니다.

음성 수신 설정

다이얼로그에서 [Listen] 체크박스를 선택 또는 선택 해제할 때 음성 수신 설정을 변경하도록 프로그램을 작성합니다. 이 기능을 사용하려면 디바이스가 XCTL_CAP_LISTEN capability를 지원해야 합니다.

CListenTalkDlg 클래스의 OnBnClickedCheckListen()에서 SetListen() 함수를 호출합니다. 함수의 파라미터로 디바이스 핸들, 컨트롤ID, 음성 데이터의 종류 등을 입력합니다. 음성 데이터의 종류는 라이브 음성일 경우 XMEDIA_PLAY를 입력하고, 녹화 음성일 경우에는 XMEDIA_RECORD를 입력합니다.

```
// ListenTalkDlg.cpp
long nRet = m_ctrlXnsDevice.SetListen(
    m_hDevice,
    m_nControlId,
    XMEDIA_PLAY,
```

```
m_bListen
);
```

수신 음성 재생 설정

음성 수신을 설정한 후에는, 원도우 컨트롤에서 수신한 음성을 재생 출력할지 여부를 설정합니다. CListenTalkDlg 클래스의 OnBnClickedCheckListen()에서 SetSound() 함수를 호출합니다.

```
// ListenTalkDlg.cpp
m_ctrlXnsWindow.SetSound(m_bListen);
```

음성 전송 설정

다이얼로그에서 [Talk] 체크박스를 선택 또는 선택 해제할 때 음성 송신 설정을 변경하도록 프로그램을 작성합니다. Talk 기능이 ON일 경우, 원격 디바이스는 음성 데이터를 수신하기 위한 준비 작업을 수행합니다. 이 기능은 라이브 스트리밍 모드에서만 지원하며, 제어 디바이스가 XCTL_CAP_TALK capability를 지원할 때만 사용할 수 있습니다.

CListenTalkDlg 클래스의 OnBnClickedCheckTalk()에서 SetTalk() 함수를 호출합니다. 파라미터로 디바이스 핸들, 컨트롤ID 등을 입력합니다.

```
// ListenTalkDlg.cpp
long nRet = m_ctrlXnsDevice.SetTalk(
    m_hDevice,
    m_nControlId,
    m_bTalk
);
```

음성 입력

마이크를 통해 입력된 음성은 PCM 방식으로 적절하게 인코딩 되어야 합니다. PCM 인코딩은 SDK 내부에서 다시 각 디바이스에 맞는 음성 코덱으로 변환됩니다. 디바이스가 내부에서 사용하는 음성 코덱은 다를 수 있습니다. 다음은 오디오 코덱에 대한 정보입니다.

Cod ec	IMA ADPC M	G726 504X	G726 SNR	G72 3 SVR	G711 IPCAM	G711U HISILICO N	G723	DVI4 HISILIC ON	Othe r
PCM (Bps)	16	16	16	16	16	16	16	16	8
PCM (Sps)	8000	8000	8000	8000	8000	8000	16000	16000	8000

Send Byte s	2048	4096	4096	6400	4000	3200	6400	3200	2048
Mod el	SNC- 570, SND- 560/4 60V, SNP- 370/3 350,	SNC- 550, SNS- 100/400 , SNP- 100A/3 300A	SNR- 6400/ 3200		IP CAM			SRD- 1640/84 0/440/4 42	

(Bps: Bits per sample, Sps: Sample per second)

음성입력 및 PCM 인코딩은 Winmm 라이브러리나 DirectX의 API를 사용하여 구현할 수 있으며, 예제 소스는 Winmm.lib를 사용하여 구현하였습니다. 다음은 입력장치로부터 음성을 입력받아 처리한 Winmm API의 사용 예입니다.

음성 데이터 입력 초기화

```
void CListenTalkDlg::OnBnClickedButtonSendAudio()
{
    memset((char*)&wave_format_in, 0, sizeof(wave_format_in));
    wave_format_in.wFormatTag          = WAVE_FORMAT_PCM;
    wave_format_in.nChannels          = CHANNEL_NUM;
    wave_format_in.nSamplesPerSec     = SAMPLE_PER_SEC;
    wave_format_in.wBitsPerSample     = BIT_PER_SAMPLE;
    wave_format_in.nBlockAlign        = BLOCK_ALIGN;
    wave_format_in.nAvgBytesPerSec   = AVG_BYTES;
    wave_format_in.cbSize            = 0;

    terminate_thread_in=false;
    curr_in = 0;
    WaveInEvent = CreateEvent(NULL, FALSE, FALSE, NULL);
    AfxBeginThread(WaveInThreadProc, this, THREAD_PRIORITY_NORMAL , 0, 0, NULL);

    int result = waveInOpen( &wavein_handle, WAVE_MAPPER,
                           (WAVEFORMATEX *) &wave_format_in,
                           (DWORD) WaveInEvent,
                           0, CALLBACK_EVENT);
```

```

for(int i=0; i<BUFF_NUM; i++)
{
    WaveHeaderIn[i].lpData = abuffer[i];
    WaveHeaderIn[i].dwBufferLength = NOTIFY_SIZE;
    waveInPrepareHeader( wavein_handle, &WaveHeaderIn[i], sizeof(WAVEHDR) );
    waveInAddBuffer( wavein_handle, &WaveHeaderIn[i], sizeof(WAVEHDR) );
}
waveInStart(wavein_handle);
}

```

음성 데이터 처리 쓰레드

```

UINT WaveInThreadProc(void *param)
{
    CListenTalkDlg *parent = (CListenTalkDlg *)param;
    WaitForSingleObject(WaveInEvent, INFINITE);
    tick_count = 0;
    while(1)
    {
        if((WaitForSingleObject(WaveInEvent, INFINITE) == WAIT_OBJECT_0) &&
        (!terminate_thread_in) )
        {
            int nRet = parent->m_ctrlXnsDevice.SendAudioData(
                parent->m_hDevice,
                (long)((char
*)abuffer[curr_in]),
                NOTIFY_SIZE, tick_count );
            tick_count += TIME_OFFSET;
            curr_in++;
            if(curr_in >= BUFF_NUM)
            {
                curr_in = 0;
            }
            waveInUnprepareHeader (wavein_handle, &WaveHeaderIn[curr_in],
sizeof(WAVEHDR));
            waveInPrepareHeader( wavein_handle, &WaveHeaderIn[curr_in],
sizeof(WAVEHDR) );
            waveInAddBuffer( wavein_handle, &WaveHeaderIn[curr_in],
sizeof(WAVEHDR) );
            continue;
        }
    }
}

```

```

    }
    break;
}

return 0;
}

```

참고

Winmm.lib의 Waveform Audio API 사용방법은 <http://msdn.microsoft.com/en-us/library/ms713504> 페이지를 참고 바랍니다.

음성 데이터 송출

`SendAudioData()` 함수를 호출하여 버퍼에 입력된 음성 데이터를 디바이스로 전송하는 기능을 구현합니다. 한 번에 전송할 수 있는 음성 데이터의 크기는 디바이스 모델별로 차이가 있을 수 있습니다. `SendAudioData()` 함수의 마지막 파라미터인 `time_stamp`은 현재 시간과 직전에 전송했던 시간간의 차이 값이며, 단위는 msec입니다. 예를 들어, 최초 전송 이후 20msec 경과 후에 두 번째 음성 데이터를 보낸다면, 이때 `time_stamp` 값은 200이 됩니다.

See also

[Single Live 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.

현재까지는 접수된 문의가 없으므로 비워둡니다.

CHAPTER 16

SdkTest 샘플 프로그램

SdkTest 샘플 프로그램은 XNS ActiveX SDK의 전반적인 기능을 확인하고 테스트해 볼 수 있는 프로그램입니다.

Contents

[샘플 프로그램 소개](#)

[See Also](#)

[FAQ](#)

샘플 프로그램 소개

SdkTest 샘플 프로그램은 XNS ActiveX SDK의 전반적인 기능을 테스트 해 볼 수 있는 프로그램 예제입니다. 이 샘플을 사용하면 XnsSdkDevice 컨트롤과 XnsSdkWindow 컨트롤에서 지원하는 주요 기능을 테스트할 수 있습니다. 샘플 프로그램 사용법은 다음과 같습니다.

Step 1. 샘플 프로그램 빌드 후 정상적으로 실행되는지 확인합니다.

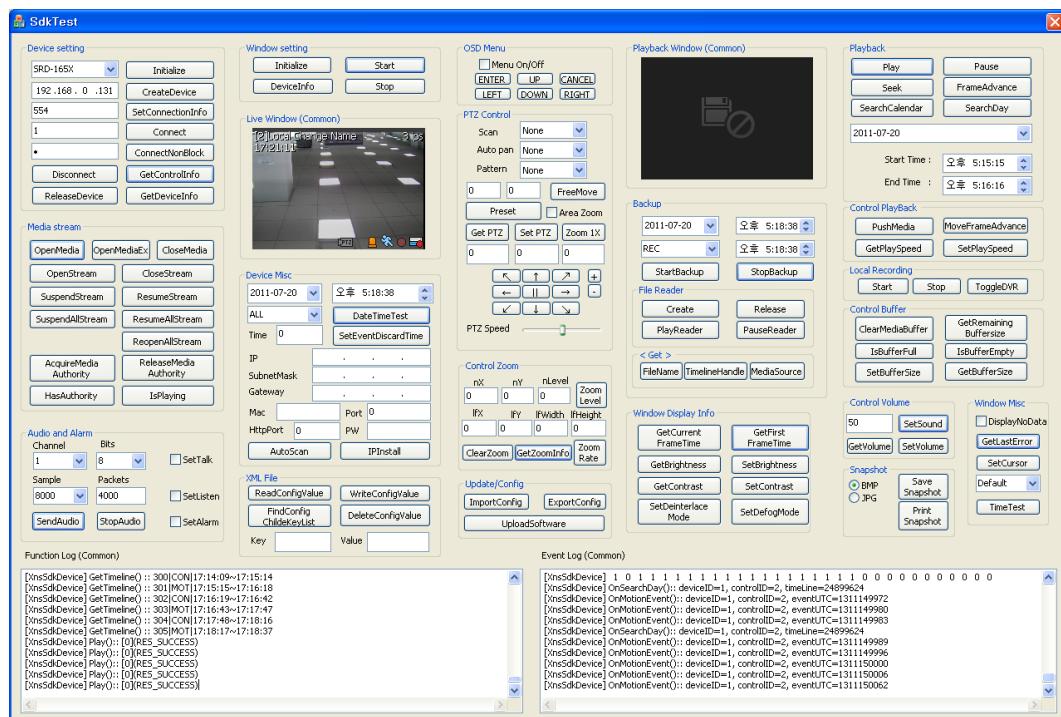


그림 34 SdkTest 샘플 프로그램 실행 화면

Step 2. 초기화 및 디바이스 연결이 정상적으로 수행되는지 테스트합니다.



그림 35 초기화 및 디바이스 연결 함수 테스트 화면

- 위 화면의 Device Setting 패널에서 디바이스 접속 정보를 설정합니다.
- [Initialize] 버튼을 눌러 XnsSdkDevice 컨트롤을 초기화합니다.
- [CreateDevice] 버튼을 눌러 디바이스 핸들을 생성합니다.
- [SetConnectionInfo] 버튼을 눌러 디바이스 연결 정보를 조회합니다.
- [Connect] 또는 [ConnectNonBlock] 버튼을 눌러 디바이스 연결을 시도합니다.
- 연결 성공 시에 [GetControlInfo] 버튼을 누르면 컨트롤ID를 가져올 수 있습니다.
- 위 화면의 Window Setting 패널에서 [Initialize] 버튼을 눌러 XnsSdkWindow 컨트롤러를 초기화합니다.

Step 3. Function Log 창에서 SDK API 함수의 수행 내역을 확인합니다.

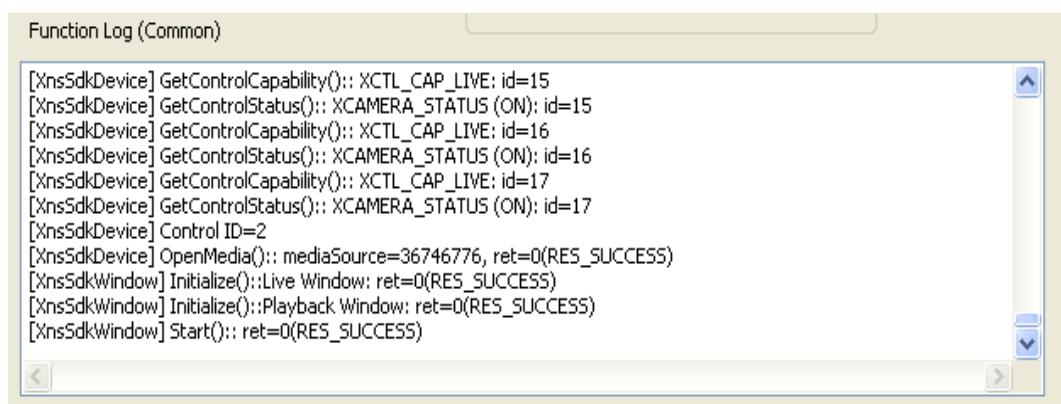


그림 36 Function Log 확인

Step 4. Event Log 창에서 XnsSdkDevice 컨트롤과 XnsSdkWindow 컨트롤에 발생하는 이벤트를 실시간 조회합니다.

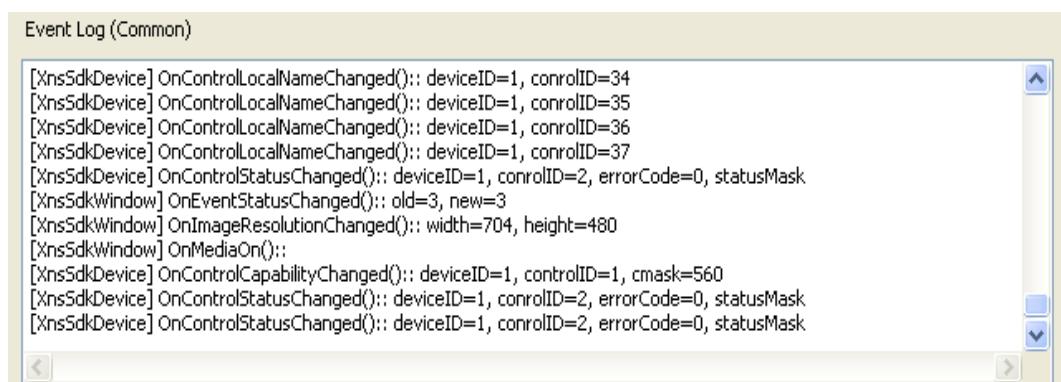


그림 37 Event Log 확인

Step 5. 미디어 오픈, 미디어 플레이 등 XnsSdkDevice 컨트롤에서 제공하는 미디어 스트림 제어 명령이 정상적으로 수행되는지 테스트합니다.



그림 38 미디어 스트림 제어 함수 테스트 화면

- [AcquireMediaAuthority] 버튼을 눌러 녹화된 미디어를 재생하기 위한 권한을 획득합니다. 라이브 스트림에 대해서는 본 기능을 사용하지 않습니다. 이 기능을 사용하기 위해서는 Control module이 XCTL_CAP_MEDIA_AUTHORITY capability를 지원해야 합니다.
- [ReleaseMediaAuthority] 버튼을 눌러 위에서 획득한 권한을 해제합니다. 미디어 재생 후에는 반드시 권한 해제를 수행하도록 합니다.
- [OpenMedia] 버튼을 눌러 장비로부터 미디어 스트림을 가져오기 시작합니다. 미디어 스트림ID는 함수 내 레퍼런스로 넘겨 받습니다. 수신된 미디어 스트림은 XnsSdkWindow 컨트롤로 전달됩니다.
- [CloseMedia] 버튼을 눌러 OpenMedia 또는 OpenMediaEx로 실행한 미디어 스트림 전송을 중단합니다.
- [OpenStream] 버튼을 눌러 미디어 스트림을 수신합니다. OpenMedia와 다른 점은 미디어 스트림이 OnNewMedia 이벤트를 통해 전달된다는 점이며, 함수 리턴 값으로 미디어 스트림ID를 넘겨줍니다.
- [CloseStream] 버튼을 눌러 미디어 스트림 수신을 중단합니다.
- [SuspendStream] 버튼을 눌러 미디어 스트림 데이터 전송을 일시적으로 중단합니다.
- [SuspendAllStream] 버튼을 눌러 모든 미디어 스트림 데이터 전송을 일시 중단합니다.
- [ResumeStream] 버튼을 눌러 일시 정지된 미디어 스트림 전송을 재개합니다.
- [ResumeAllStream] 버튼을 눌러 일시 정지된 모든 미디어 스트림의 전송을 재개합니다.
- [IsPlaying] 버튼을 눌러 라이브 영상이 아닌 Playback 또는 Backup 미디어 스트림을 재생하고 있는지 확인합니다. 즉 XnsSdkDevice::Play() 함수를 호출하여 재생 중인 미디어 스트림이 있는지를 확인합니다. XnsSdkDevice::Play() 함수 수행에 성공하면 1을 리턴하고, 실패하거나 XnsSdkDevice::Pause() 함수 수행에 성공하면 0을 리턴합니다.
- [HasAuthority] 버튼을 눌러 미디어를 재생하기 위한 권한이 있는지 확인합니다. 재생 권한이 있다면 1을 리턴합니다.

Step 6. 라이브 영상을 제어 테스트를 통해 라이브 영상 출력 및 디바이스 정보와 관련된 함수들이

정상적으로 수행되는지 확인합니다.



그림 39 라이브 영상 제어 함수 테스트 화면

- 위 화면에서 [Start] 버튼을 눌러 미디어 소스 핸들로부터 스트림 데이터를 수신합니다. 이 작업을 하려면 OpenMedia 또는 OpenMediaEx 함수 호출이 선행되어야 합니다.
- [Stop] 버튼을 눌러 스트림 데이터 수신을 중단합니다.
- [DeviceInfo] 버튼을 눌러 디바이스 정보를 조회합니다. 조회 결과는 Function Log 패널에 출력됩니다.

참고

DeviceInfo 버튼을 누르면 다음의 함수들이 호출됩니다:

GetDeviceID(), GetControlID(), GetImageWidth(), GetImageHeight(), IsMedia(), GetMediaType()
함수에 대한 상세한 설명은 'Samsung XNS ActiveX API Reference' 문서를 참조합니다.

Step 7. 오디오 및 알람 제어 기능이 정상적으로 수행되는지 테스트합니다.

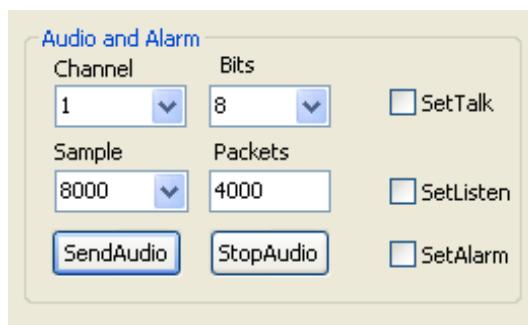


그림 40 오디오 및 알람 제어 함수 테스트 화면

- [SetTalk] 체크박스를 클릭하여 음성 전송 여부를 설정합니다. Talk 기능을 사용하려면 control module이 XCTL_CAP_TALK capability를 지원해야 합니다.
- [SetListen] 체크박스를 클릭하여 음성 수신 여부를 설정합니다. Listen 기능을 사용하려면 control module이 XCTL_CAP_LISTEN capability를 지원해야 합니다. 음성 수신을 ON으로

설정할 경우에는 XnsSdkWindow::SetSound()를 실행해야 합니다.

- [SetAlarm] 체크박스를 클릭하여 알람 수신 여부를 설정합니다. 알람 기능을 사용하려면 control module이 XCTL_CAP_ALARM_ONOFF capability를 지원해야 합니다.
- [SendAudio] 버튼을 클릭하여 SendAudioData() 함수가 정상 수행되는지 확인합니다. 이 때 Channel, Bits per Second, Sample per second, Send packet byte값은 디바이스의 지원 음성코덱에 따라 다르게 설정해야 합니다.
- [SendAudio] 버튼을 클릭하여 음성 전송을 종료합니다.

Step 8. AutoScan, IPInstall, 시간변환 함수 등 XnsSdkDevice 컨트롤의 기타 제어 함수들을 테스트합니다.



그림 41 기타 XnsSdkDevice 제어 함수 테스트 화면

- [DateTimeTest] 버튼을 클릭하면 XnsSdkDevice::DateToTimet, XnsSdkDevice::TimetToDate, XnsSdkDevice::LocalTimeToDeviceUTC, XnsSdkDevice::UTCToDeviceLocalTime 함수를 호출합니다. 호출 결과는 Function Log에 출력됩니다.
- [SetEventDiscardTime] 버튼을 클릭하여 콤보박스에서 선택된 이벤트 종류와 Edit Control에 입력한 시간으로 이벤트가 삭제되는 시간을 설정합니다.
- [AutoScan] 버튼을 클릭하여 네트워크에 연결된 모든 디바이스를 10초간 검색합니다. 검색된 장비로부터 각각의 응답이 올 때마다 OnDeviceDetected 이벤트가 발생하므로, 이 이벤트가 Event Log에 출력되는지 확인합니다.
- [IPInstall] 버튼을 클릭하여 연결된 네트워크 디바이스에 대한 네트워크 정보를 입력 받은 정보로 재설정합니다.

Step 9. SetCursor, GetLastError 등 XnsSdkWindow의 기타 함수들을 테스트합니다.

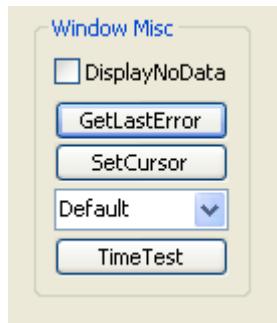


그림 42 기타 XnsSdkWindow 제어 함수 테스트 화면

- [DisplayNoData] 체크박스를 선택하면, 녹화된 영상이 없는 경우 화면에 "No Data"라고 출력합니다. 이 기능은 Playback 영상일 때만 사용합니다.
- [GetLastError] 버튼을 클릭하면 윈도우 컨트롤에서 가장 최근에 발생한 에러코드를 반환합니다.
- [SetCursor] 버튼을 클릭하면 콤보박스에서 선택한 타입으로 커서 모양을 설정합니다.
- [TimeTest] 버튼을 클릭하면 XnsSdkDevice::DateToTimet, XnsSdkDevice::TimetToDate 함수를 호출합니다.

Step 10. XnsSdkDevice 컨트롤의 PTZ, OSD 제어 기능을 테스트합니다.



그림 43 PTZ, OSD 제어 기능 테스트 화면

- 위 화면의 OSD Menu 패널에서 [Menu On/Off] 체크박스와 방향 버튼을 사용하여 OSD 메뉴를 제어합니다. 이 기능을 사용하려면 control module이 XCTL_CAP_CAM_MENU capability를 지원해야 합니다.

- 위 화면의 PTZ Control 패널에서 스캔, 오토팬, 패턴, 프리셋 기능을 수행합니다. 스캔, 오토팬, 패턴 기능을 사용하려면 control module이 XCTL_PTZ_FUNCTION capability를 지원해야 하고, 프리셋 기능을 사용하려면 control module이 XCTL_PTZ_PRESET capability를 지원해야 합니다.
- PTZ 절대 좌표 값을 입력하거나 방향키를 사용하여 카메라의 PTZ를 제어합니다. 이 기능을 사용하려면 control module이 각각 XCTL_CAP_PTZ_PAN, XCTL_CAP_PTZ_TILT, XCTL_CAP_PTZ_ZOOM capability를 지원해야 합니다. 이외 Zoom 1X 기능과 PTZ speed 기능, FreeeMove 기능을 테스트합니다. FreeMove 기능을 사용하려면 control module이 XCTL_CAP_PTZ_FREE_MOVE를 지원해야 하고, PTZ speed 기능을 사용하려면 XCTL_CAP_PTZ_SPEED를 지원해야 합니다.

Step 11. XnsSdkWindow 컨트롤의 디지털 줌 제어기능을 테스트합니다.

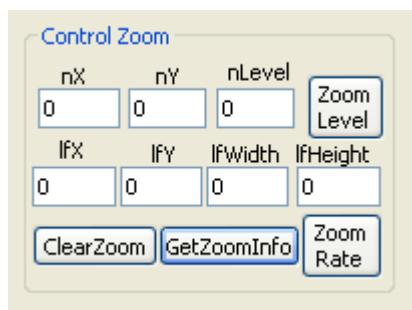


그림 44 디지털 줌 제어 테스트 화면

Step 12. XnsSdkDevice의 영상 백업 기능을 테스트합니다.

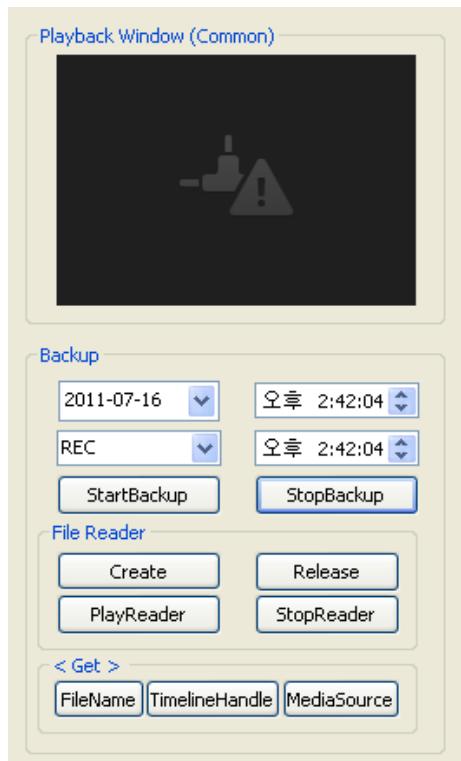


그림 45 영상 백업 기능 테스트 화면

- 위 화면의 Backup 패널에서 녹화 시작 시간과 종료 시간을 입력한 후 [StartBackup] 버튼을 클릭합니다. 장비에 녹화된 스트리밍 데이터가 입력 받은 시간에 맞게 존재해야 하며, 백업에 성공하면 'C:\W'에 'SdkTest' 파일이 생성됩니다.
- 위 화면에서 Create > MediaSource > PlayReader 버튼을 차례대로 눌러 백업 영상을 재생합니다.
- [Create] 버튼을 눌러 파일 재생기를 생성한 후에 Get 패널에 있는 세 개의 버튼을 눌러 미디어 파일 정보를 가져옵니다.

Step 13. XnsSdkWindow 컨트롤의 영상 출력 제어 기능을 확인합니다.

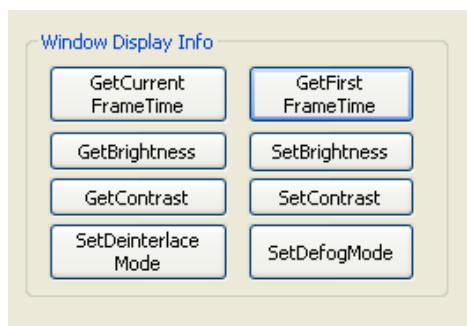


그림 46 영상 출력 테스트 화면

- [GetCurrentFrameTime] 버튼과 [GetFirstFrameTime] 버튼을 눌러 현재 프레임의 시간과 프레임이 처음 시작했을 때의 시간을 정상적으로 반환하는지 확인합니다.
- [GetBrightness], [SetBrightness], [GetContrast], [SetContrast] 버튼을 눌러 현재 출력되는 영상의 밝기 및 채도를 설정하거나 설정 값을 조회합니다. 이 설정으로 인해 장비 내에서 출력되는 영상 자체가 변경되지는 않습니다.
- [SetDefoMode] 버튼을 눌러 영상에서 안개를 제거합니다. 이 기능은 1채널에서만 지원하는 기능입니다.

Step 14. XnsSdkDevice의 Playback 기능을 테스트합니다.

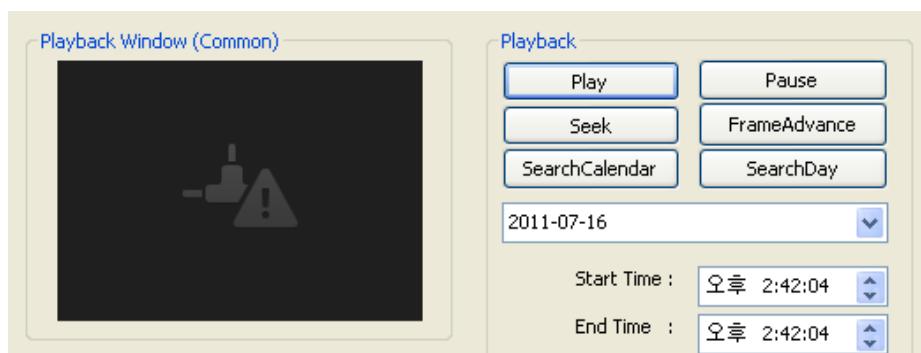


그림 47 Playback 기능 테스트 화면

- [SearchCalendar] 버튼을 눌러 디바이스에 기록된 영상이 있는지 검색합니다. 영상이 있을 경우 해당 날짜에 '1'로 표기되며, 없을 경우에는 '0'으로 표기됩니다. 이 기능은 디바이스에 접속된 상태에서만 사용할 수 있습니다.
- [SearchDay] 버튼을 눌러 특정 날짜에 기록된 영상이 있는지 검색합니다. 시간 구간 사이

에 기록된 데이터들의 시간이 로그박스에 표시가 됩니다. 이 기능은 디바이스에 접속된 상태에서만 사용할 수 있습니다.

- 특정 시간 구간의 영상만 재생하려면, [SearchCalendar] > [SearchDay] > 시간구간 조정 > [Seek] > [Play]를 순차적으로 실행합니다.

Step 15. XnsSdkWindow의 미디어 스트림 제어 기능을 테스트합니다.



그림 48 Playback 제어 테스트 화면

- [PushMedia] 버튼을 눌러 미디어 스트림을 화면에 출력합니다. 이 기능은 OpenStream을 호출하여 미디어 스트림을 오픈한 후에 수행할 수 있습니다.
- [MoveFrameAdvance] 버튼을 눌러 다음 프레임을 재생합니다. 이 기능은 PlayBack 모드로 스트림을 오픈했을 경우만 유효하며, 정지된 영상에서 동작합니다.

Step 16. XnsSdkDevice의 로컬 레코딩 기능을 테스트합니다.

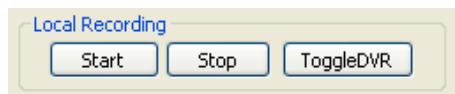


그림 49 로컬 레코딩 기능 테스트 화면

- [Start] 버튼을 눌러 로컬 레코딩을 시작합니다. 녹화파일은 C 드라이브에 LiveVideo 라는 파일 이름으로 생성되며, 확장자는 REC입니다. 레코딩을 시작하면 라이브 영상 화면에 빨간색으로 녹화 표시가 나타나게 됩니다.

Step 17. XnsSdkWindow의 영상 버퍼 제어기능을 테스트합니다.

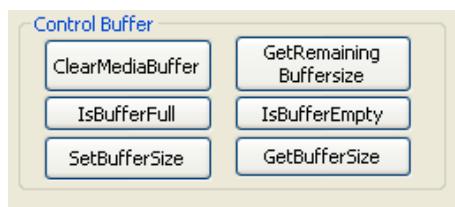


그림 50 영상 버퍼 제어 기능 테스트 화면

- 버퍼를 비우거나, 버퍼 상태를 확인하거나, 버퍼 크기를 설정합니다. 영상 버퍼 제어 기능은 라이브 영상이 재생되어 있을 경우에만 실행 가능합니다.

Step 18. XnsSdkWindow의 음성 제어기능을 테스트합니다.

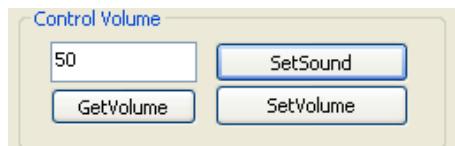


그림 51 음성 제어 기능 테스트 화면

- [SetSound] 버튼을 눌러 사운드 ON/OFF를 설정합니다. 이 기능을 사용하려면 Listen 기능이 ON으로 설정되어 있어야 합니다.
- [SetVolume] 버튼을 눌러 사운드 볼륨을 조절합니다. Edit Control에 0~100 사이의 정수를 입력하여 볼륨 값을 설정합니다. 0보다 작은 값은 0으로, 100보다 큰 값은 100으로 변환합니다.
- [GetVolume] 버튼을 눌러 현재 설정된 사운드 볼륨을 Function Log에 출력합니다.

Step 19. XnsSdkWindow의 Snapshot 기능을 테스트합니다.

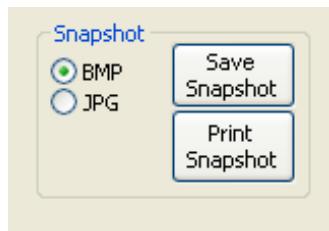


그림 52 스냅샷 기능 테스트 화면

- [Save Snapshot] 버튼을 눌러 현재 이미지의 스냅샷을 로컬에 저장합니다. 저장 이미지의 확장자는 BMP 또는 JPG 중에 선택합니다. 이미지 저장에 성공하면 C:\ 경로에 해당 파일 포맷 확장자를 가진 snapshot 파일이 생성됩니다.
- [Print Snapshot] 버튼을 눌러 현재 이미지의 스냅샷을 인쇄합니다. 이때 프린터 종류를 지정할 수 있습니다.

Step 20. 프로그램을 종료합니다.

See also

[본 문서에 설명된 전체 샘플 프로그램](#)

[Samsung XNS ActiveX API Reference](#)

FAQ

이 절은 샘플 프로그램에 대한 문의 중 자주 들어오는 문의와 답변을 정리하는 항목입니다.

현재까지는 접수된 문의가 없으므로 비워둡니다.

APPENDIX A

UTC Time

이장에서는 UTC Time 에 대해 설명합니다.

UTC(Coordinated Universal Time) 협정시계시는 고정밀도 원자 시간 표준으로 컴퓨터 서버나 온라인 서비스 또는 여러 기관들은 UTC 시간을 통해 표준 시간대에 대해 중립적인 시간을 제공합니다. 전 세계의 표준 시간대는 아래와 같이 UTC에서의 양 또는 음의 오프셋으로 표시됩니다.

Time Zone	Locations
UTC -12:00	Baker Island, Howland Island (both uninhabited)
UTC -11:00	Samoa, American Samoa
UTC -10:00	Hawaii, Papeete
UTC -09:30	Marquesas Islands
UTC -09:00	Anchorage, Fairbanks, Juneau
UTC -08:00	Vancouver, Washington (state), Portland, Las Vegas, California, Baja California
UTC -07:00	Alberta, Colorado, Arizona, Chihuahua, Sonora
UTC -06:00	Chicago, Costa Rica, Dallas, El Salvador, Guatemala, Honduras, Houston, Manitoba, Mexico City, Nicaragua, Saskatchewan
UTC -05:00	Ottawa, Toronto, Montreal, Boston, New York, North Carolina, Washington D.C., Georgia, Miami, Cuba, Jamaica, Haiti, Panama, Colombia, Continental Ecuador, Peru
UTC -04:30	Venezuela
UTC -04:00	Nova Scotia, Dominican Republic, Puerto Rico, Trinidad and Tobago, Amazonas, Bolivia, Continental Chile, Paraguay, San Luis Province
UTC -03:30	Newfoundland
UTC -03:00	Rio de Janeiro, São Paulo, Argentina (except San Luis Province), Uruguay, Nuuk
UTC -02:00	Fernando de Noronha, South Georgia and the South Sandwich Islands
UTC -01:00	Azores, Cape Verde
UTC	Iceland, Faroe Islands, United Kingdom, Ireland, Continental Portugal, Madeira, Morocco, Senegal, Ghana, Côte d'Ivoire

<u>UTC+01:00</u>	Albania, Slovenia, Macedonia, Norway, Sweden, Denmark, Germany, the Netherlands, Belgium, Metropolitan France, Switzerland, Austria, Poland, Czech Republic, Slovakia, Hungary, Continental Spain, Italy, Croatia, Serbia, Kosovo, Bosnia and Herzegovina, Tunisia, Algeria, Nigeria, Cameroon, Angola, Kinshasa
<u>UTC+02:00</u>	Finland, Lithuania, Latvia, Estonia, Belarus, Ukraine, Romania, Bulgaria, Greece, Turkey, Cyprus, Syria, Lebanon, Jordan, Palestine, Israel, Egypt, Libya, Mozambique, Malawi, Zambia, Zimbabwe, South Africa
<u>UTC+03:00</u>	Samara, Iraq, Saudi Arabia, Yemen, Sudan, Ethiopia, Somalia, Kenya, Uganda, Tanzania, Madagascar
<u>UTC+03:30</u>	<u>Iran</u>
<u>UTC+04:00</u>	Georgia, Armenia, Azerbaijan, United Arab Emirates, Oman, Seychelles, Mauritius, Moscow, Saint Petersburg
<u>UTC+04:30</u>	<u>Afghanistan</u>
<u>UTC+05:00</u>	Sverdlovsk, Uzbekistan, Pakistan, Maldives, Kazakhstan
<u>UTC+05:30</u>	India, Sri Lanka
<u>UTC+05:45</u>	<u>Nepal</u>
<u>UTC+06:00</u>	Novosibirsk, Almaty, Bangladesh
<u>UTC+06:30</u>	Myanmar, Cocos Islands
<u>UTC+07:00</u>	Krasnoyarsk, Thailand, Vietnam, Jakarta
<u>UTC+08:00</u>	Irkutsk, Ulan Bator, China, Taiwan, Hong Kong, Philippines, Malaysia, Singapore, Western Australia
<u>UTC+09:00</u>	Zabaykalsky, Japan, North Korea, South Korea, East Timor
<u>UTC+09:30</u>	Northern Territory, South Australia
<u>UTC+10:00</u>	Victoria, Tasmania, Queensland, New South Wales, Primorsky
<u>UTC+10:30</u>	<u>Lord Howe Island</u>
<u>UTC+11:00</u>	Kamchatka, Solomon Islands, New Caledonia
<u>UTC+11:30</u>	<u>Norfolk Island</u>
<u>UTC+12:00</u>	Fiji, New Zealand
<u>UTC+12:45</u>	<u>Chatham Islands</u>
<u>UTC+13:00</u>	<u>Tonga</u>
<u>UTC+14:00</u>	<u>Line Islands</u>

약어

D

DEP

Data Execution Prevention

DLL

Dynamic Linking Library

DVR

Digital Video Recorder

M

MFC

Microsoft Foundation Class

O

OCX

Object Linking and Embedding (OLE) Custom Control

P

PTZ

Pan/Tilt/Zoom

S

SDK

Software Development Kit