

# Aimetis

## Symphony SDK User Guide

Last Updated: 4/12/2016

Product Versions: 7.0

## Contents

Overview .....	4
Quick Start.....	4
Requirements and Recommendations .....	4
Portability.....	5
Contact.....	5
Server API Overview .....	6
Use Cases and Sample Code .....	7
Farm Connectivity .....	7
Video .....	7
PTZ.....	7
Events.....	7
Farm Settings .....	7
Recording .....	8
Symphony Client Control .....	8
Reports.....	8
Security .....	8
Video Analytics.....	8
Timeline.....	8
Navigation .....	8
DIO Rules.....	9
Hardware Devices .....	9
Camera Usage .....	9
Direct File Access (Persistence) .....	9
Symphony Managed Libraries used by the SDK .....	10
Symphony Media Streaming Libraries used by the SDK.....	11
Persistence .....	12
Files .....	12
Database .....	12
How to Add Streaming Video to a Project .....	13
Requirements.....	13
Procedure.....	13
How to Enable Camera Panel Switching in Symphony Client.....	13

How to Get a JPEG Image .....	14
Set Decoration Options.....	14
How to activate/deactivate a relay.....	15
How to enable/disable a camera tour using dbupdater.....	16
How to wrap OCX into a DLL and use in an MFC Application.....	16
Disclaimers and Legal Information .....	21

## Overview

Welcome to the Aimetis Symphony SDK. There are many exciting ways to integrate and extend Symphony. Symphony Client uses these same methods ensuring quality and feature completeness. Sample applications are included for a wide variety of use cases. Reference documentation is provided in <SDK>/Documentation/readme.html.

The latest *Symphony SDK User Guide* is available here:

<http://www.aimetis.com/Library/Download.aspx?l=00198&L=en>

The latest *Symphony SDK* is available here:

<http://www.aimetis.com/Xnet/Downloads/VMSSoftware.aspx>

## Quick Start

- Running a pre-built SDK application
  - Create an Aimetis XNET account/Log In:
    - <http://aimetis.com/xnet/Login.aspx>
  - Download and install **Symphony Server** and **Client**:  
<http://aimetis.com/Xnet/Downloads/VMSSoftware.aspx>
  - Once installed, use a web browser to access Symphony Server and add a camera.
  - Download and install the Symphony SDK:  
<http://aimetis.com/Xnet/Downloads/VMSSoftware.aspx>
  - Start C:\AimetisSymphonySDK\bin\LiveStreamTest.exe to see live video from your Symphony Farm.
- Compiling an SDK application
  - Open up SDK.sln in Visual Studio.
  - Right click on *LiveStreamTest* in *Solution Explorer* and select *Set as Startup Project*.
  - Run it (press F5).

## Requirements and Recommendations

The managed interfaces must be used from executables targeting .NET 4.5.2 or later. The recommended development IDE is Visual Studio 2012 or later. The recommended programming language is C#. However, customers have reported Visual Basic working. The

developer should be familiar with Aimetis Symphony. It is not recommended that Symphony and the SDK be installed on the same machine.

## Portability

It is possible to integrate live video from Symphony directly from non-Windows platforms using gstreamer and the RESTful Mobile Bridge interface. Please contact us for details about gstreamer.

The portable Mobile Bridge interface isn't as full featured as the Windows accessible AirAWS interface. If you need functionality in AirAWS from non-Windows platforms, then you must develop your own layer that provides this interface. Ported implementations of .NET will work (such as Mono), but the referenced .NET libraries link to Windows style DLLs which require deeper platform equivalence.

## Contact

Questions and comments can be sent to [support@aimetis.com](mailto:support@aimetis.com)

# Server API Overview



## Use Cases and Sample Code

The projects in the SDK address the following use cases.

### Farm Connectivity

Use Case	SDK Project
Connect to a farm	NotificationMonitorExample, FarmTest

### Video

Use Case	SDK Project
Get a list of cameras	LiveStreamTest, FarmTest
Connect and render live video	LiveStreamTest
Connect and render historical video	LiveStreamTest
Get a historical JPG	AlarmHandlerExample
Export Video	ExportVideo

### PTZ

Use Case	SDK Project
Control PTZ on a video device	LiveStreamTest, FarmTest
Load guard tour locations	LiveStreamTest, FarmTest
Go to a specific guard tour location	LiveStreamTest, FarmTest

### Events

Use Case	SDK Project
Trigger an alarm	AddAlarmToCamera, FarmTest
Listen for alarms	AlarmHandlerExample
Get a list of alarms	LiveStreamTest
Mark an alarm	LiveStreamTest
Listen for farm events	NotificationMonitorExample, NotificationMonitors

### Farm Settings

Use Case	SDK Project
Retrieve farm settings	WebService
Execute a web service method	WebServiceConsole

## Recording

Use Case	SDK Project
Start/Stop recording	LiveStreamTest, FarmTest

## Symphony Client Control

Use Case	SDK Project
Switch cameras in Symphony Client through Server API	VideoWall, FarmTest
Switch cameras in Symphony Client through Client API	VideoWallClient
Create multiview in Symphony Client through Server API	CreateMultiViewPanel

## Reports

Use Case	SDK Project
Get a heatmap	GetDensityImageJpg

## Security

Use Case	SDK Project
Change the currently active security profile	SetActiveProfile
Add/Remove users	UserManagement

## Video Analytics

Use Case	SDK Project
Get live XML metadata from a running analytics engine	XMLStream

## Timeline

Use Case	SDK Project
Get timeline information	TimelineGetter

## Navigation

Use Case	SDK Project
Use navigation buttons to control video	FarmTest



## DIO Rules

Use Case	SDK Project
Import/Export DIO Rules from/to a CSV file	AddDIORulesFromCSV, ExportDIORulesToCSV

## Hardware Devices

Use Case	SDK Project
Get a list of all hardware devices	PSA
Add a new hardware device	PSA

## Camera Usage

Use Case	SDK Project
View camera use information	ClientBandwidth

## Direct File Access (Persistence)

Use Case	SDK Project
Show detailed footage file frame information	PrintFootageFile
Show detailed timeline file information	PrintSignalsFile
Show detailed metadata file information	PrintSignals2File
Show detailed export file information	PrintAiraFile

## Symphony Managed Libraries used by the SDK

- The following managed DLLs are used directly by SDK samples
  - [Seer.Connectivity.dll](#)
    - Library for connecting to servers. Usually used indirectly by samples.
  - [Seer.BaseLibCS.dll](#)
    - [Seer.BaseLibCS.Communication.ServerConnectionManager](#) for client server connectivity. Usually used indirectly by samples.
    - [Seer.BaseLibCS.AiraWS.Signals](#) for Web Service interface
    - [Seer.BaseLibCS.CameraMessageStruct](#) for structure definitions
    - [Seer.BaseLibCS.Utils.MulticastMessage](#) for constants
  - [Core.dll](#) (source part of SDK)
    - [Seer.SDK.SDKFarm](#) for farm connectivity and event stream notifications
  - [Seer.Farm.dll](#)
    - [Seer.Farm.DeviceModel.Client](#) for client side device operations and information
    - [Seer.Farm.DeviceModel.Server](#) for server side device operations and information
  - [Internationalization.dll](#)
    - [Seer.Internationalization.Translator](#) for language translation into all Symphony supported languages.
  - [NotificationMonitors.dll](#) (source part of SDK)
    - Interface to the event stream that comes from Symphony
- The following managed DLLs are dependencies of the above DLLs and likely do not need to be used by your code directly. However they may need to be added as references to your project. If they do need to be added, the compiler will fail with a message stating what reference needs to be added.
  - [Seer.Common.dll](#)
  - [DeviceModel.Dio.dll](#)
  - [DeviceModel.Security.dll](#)
  - [Farm.Security.dll](#)
  - [SecurityLib.dll](#)
- DLLs that enable access to video streaming
  - [AxVideoRecvCtrl.dll](#) wraps [VideoRecvCtrl.dll](#)
  - [BaseIDL.dll](#) is a set of COM interfaces that aid in streaming video

## Symphony Media Streaming Libraries used by the SDK

User may create video viewing panel to display live video and or pre-recorded video. Symphony defines the following interfaces to deal with video streaming:

- AxVideoRecvCtrl
  - This is ActiveX control which supports Dispatch Interface which can be used by simple applications such as VB or we-based.
- IVideoRecvCtrl
  - A COM interface exposed by AxVideoRecvCtrl. Supports richer API when combined with INetworkEndpoint and IHistoricalSeek. This interface is demonstrated by LiveStreamTest SDK sample and used by Symphony Client. This is the recommended interface to use.
- OCXDll.dll
  - A dynamic link library which allows user to dynamically create, use and destroy AxVideoRecvCtrls.
- OCXExe.exe
  - An executable which shows how to use OCXDll.dll. Useful if integrators need to support multiple NVRs and need a DLL with a common API for each NVR.

## Persistence

### Files

_FootageArchive (*.dat)	Video, Audio, Overlays (decorations)
_Signals (*.dat)	Timeline data
_Signals2 (*.dat)	Metadata (usually from Analytics engines)
Native exported/movie files (*.aira)	Video, Audio, Overlays

See Persistence use cases above for how to interpret these files.

### Database

Settings	Most of the settings (except users, groups, servers)
Servers	All servers in farm along with state
Alarms	All alarms
XnetData	License information
ObjectCounts	Count of objects per minute (analytics)
LineCounts	Count of line crossings per minute (analytics)

Dbupdater.exe can be used from the command line to run SQL against the database.

```
dbupdater "select top 10 * from settings order by sequenceid desc"
```

Shows the last 10 settings that changed.

```
dbupdater "select * from servers"
```

Shows all server addresses in the farm as well as their state.

```
dbupdater "select count(*) from alarms"
```

Shows the current count of all alarms.

All database times are UTC encoded as the number of seconds (or milliseconds) since 1970.

# How to Add Streaming Video to a Project

## Requirements

.NET framework 4.5.2 is required at run time. Please read the *Portability* section above.

## Procedure

1. Open the design view of the form where you want to add the control.
2. From the **Tools** menu, select **Add/Remove Toolbox Items...**
3. Click the **COM Components** tab.
4. Scroll down and select the **VideoRecvCtrl Control** check box.
5. Click **OK**. The Active X control should now appear in the toolbox, at the end of the **Windows Forms** section.
6. Drag the Active X control onto the design view where you want it.

# How to Enable Camera Panel Switching in Symphony Client

To switch what is displayed on a camera panel:

1. Use the FarmTest or VideoWall applications to switch panels through a server. Use VideoWallClient to switch panels in Symphony Client directly.
2. Ensure the user doing the switching has Video Wall Change Panel permissions.
3. Ensure the Symphony Client where the panel switch is to occur is registered as a Video Wall Client.
  - a. From the main menu in Symphony Client click on the settings icon, and then **Video Wall**. The **Video Wall Management** dialog opens.
  - b. Click the **Video Wall Client Configuration** tab and then click on **Register current Symphony Client**.
4. To get/set the Panel Name:
  - a. In Symphony Client right-click on a Camera Panel.
  - b. Select **Settings** and then click on the **Child** tab. If using the VideoWall project in command prompt and the panel name has a space in it, make sure to surround the name with double quotes.
5. In the FarmTest or VideoWall applications, for the IP address, make sure to use the IP address of the Symphony *server*. In VideoWallClient use the IP address of the Symphony *Client*.

## How to Get a JPEG Image

The recommended (simplest) approach is to use the `DeviceModel.Client.CCamera` `GetJPEGImage` function to get a JPEG.

This function calls one of the following **AiraWS** functions:

1. **GetJPEG** generates the JPEG and returns an URL to it. The other `GetJPEG*` functions return the bytes of the JPEG.
2. **GetJPEGImage** gets a JPEG bytes for a specified time and camera.
3. **GetJPEGImage3** extends `GetJPEGImage` by allowing you to send parameters for the decorations.
4. **GetJPEGImage4** extends `GetJPEGImage3` by allowing you to send parameters for the font used on the image.

## Set Decoration Options

To set the decoration options when getting a JPEG, you can do the following:

```
DecorationOptions options = new DecorationOptions(15, 0, true, 0, true);
// Pass decorations.EncodedDecorationOptions to GetJPEG* functions

// You will want to use the following DecorationOptions constructor
public DecorationOptions(int decorations, int dateFormat, bool
in24HourNotation, uint streamIndex, bool canViewPrivateVideo)
{
    // BITWISE OR together the following to select decoration
    // 1 = rectangles.
    // 2 = messages.
    // 4 = time.
    // 8 = paths.
    _rawDecoration = decorations;

    // SET to one of the following
    // 0 = yyyy/MM/dd
    // 1 = dd/MM/yyyy
    // 2 = MM/dd/yyyy
    _dateFormat = dateFormat;

    // True for 24 hour time format. False for 12 hour time format.
    _in24HourNotation = in24HourNotation;

    // 0 based stream number. The Symphony Client UI is 1 based.
    _streamIndex = streamIndex;
}
```

## How to activate/deactivate a relay

To activate/deactivate a relay, call the web service method:

```
public void PerformAction(string username, string password, string sActionReq)
```

To activate, sActionReq will have the format:

```
<action><ServerIP>10.234.8.30</ServerIP><camera><On Relay='1' DeviceID='2'/></camera></action>
```

To deactivate, sActionReq will have the format:

```
<action><ServerIP>10.234.8.30</ServerIP><camera><Off Relay='1' DeviceID='2'/></camera></action>
```

## How to enable/disable a camera tour using dbupdater

To enable/disable a camera tour, the following SQL can be used:

```
declare @xml xml, @cameraId int, @tourName varchar(250), @disabled int;
set @cameraId = 3;
set @tourName = 'Camera Tour 2';
set @disabled = 1;

select @xml = CAST(v as xml) from settings where Type = 'Camera' and Section = 'Camera' and K =
'cameraTour' and ID = @cameraId;

set @xml.modify('replace value of
(/TourGroup/cameraTour[@name=sql:variable("@tourName")]/disable/text())[1] with
sql:variable("@disabled")');

update Settings set V = CAST(@xml as nvarchar(max)) where Type = 'Camera' and Section = 'Camera' and
K = 'cameraTour' and ID = @cameraId;
```

If running a batch file on Symphony server, the command can be passed to dbupdater as follows:

```
dbupdater "declare @xml xml, @cameraId int, @tourName varchar(250), @disabled int; set @cameraId =
3; set @tourName = 'Camera Tour 2'; set @disabled = 1; select @xml = CAST(v as xml) from settings
where Type = 'Camera' and Section = 'Camera' and K = 'cameraTour' and ID = @cameraId; set
@xml.modify('replace value of
(/TourGroup/cameraTour[@name=sql:variable("@tourName")]/disable/text())[1] with
sql:variable("@disabled")'); update Settings set V = CAST(@xml as nvarchar(max)) where Type = 'Camera'
and Section = 'Camera' and K = 'cameraTour' and ID = @cameraId;"
```

## How to wrap OCX into a DLL and use in an MFC Application

Following the steps below will create a sample application that demonstrates how to dynamically create and destroy live video ActiveX controls contained in a DLL using an MFC EXE application. To dynamically add a control, the user will click the “Add” button. Clicking the “Start” button will connect to live video on localhost (127.0.0.1), camera #1, using the user “admin” with password “admin” on the most recently added control. Pressing the “Remove” button will remove the most recently added control.

To create the sample, follow these steps:

1. Create an MFC DLL project.

File->New->Project. Expand Visual C++ and select MFC. Select MFC DLL, type in OCXDLL for the name and click OK. You will be prompted with the MFC DLL Wizard. Click Finish.

2. Add a dialog to the DLL.



In the Resource View window, right-click on OCXDLL->Add->Resource. Select Dialog and click New. When the dialog appears, select and delete the OK and Cancel buttons.

3. Change some of the dialog's default properties.

Right-click on the dialog -> select properties. Change Border to None. Change No Fail Create to True. Change Style to Child.

4. Add a dialog class.

Right-click on the dialog -> Add class. Enter OCXDLLDialog as the class name and click Finish. Add `#include "resource.h"` in OCXDLLDialog.h.

5. Add the OCX control.

Select the dialog tab and then in the Toolbox window, right-click->Choose Toolbox Items. Click on COM Components and select VideoRecvCtrl Control. (The file is VideoR~2.OCX and should be in the aira\bin directory) Now drag the control onto the dialog and click and it should appear. Resize it to fill the space.

6. Add a variable to the OCX.

Right-click on the VideoRecvCtrl control -> Add Variable. Change Access to protected. Under Variable Name, enter `m_OCX` and click Finish.

7. Add Play, Stop functions in OCXDLLDialog.h and OCXDLLDialog.cpp.

In OCXDLLDialog.h, add the following as public declarations:

```
HRESULT Play(unsigned long IP, unsigned short port, LPCTSTR user, LPCTSTR pass, __int64 historicalTime);
HRESULT Stop(void);
```

In OCXDLLDialog.cpp, add the following methods:

```
HRESULT OCXDLLDialog::Play(unsigned long IP, unsigned short port, LPCTSTR user, LPCTSTR pass, __int64
historicalTime)
{
    HRESULT hr;

    if (FAILED(hr = m_OCX.Init()))
        return hr;
    if (FAILED(hr = m_OCX.Connect(IP, port, user, pass, historicalTime)))
        return hr;
    if (FAILED(hr = m_OCX.StartVideo()))
        return hr;
    return S_OK;
}

HRESULT OCXDLLDialog::Stop(void)
{
    HRESULT hr;
    if (FAILED(hr = m_OCX.StopVideo()))
```

```

        return hr;
    if (FAILED(hr = m_OCX.Destroy()))
        return hr;
    return S_OK;
}

```

8. In OCXDLL.cpp, add a call to [AfxEnableControlContainer\(\)](#) in InitInstance().

9. Create an interface for VideoCreate, VideoDestroy, VideoPlay and VideoStop functions. In Solution Explorer, right-click on Header Files -> Add -> New Item. Add a new header file called OCXDLLInterface and copy in the following code:

```

#pragma once

#ifdef OCXDLL_EXPORTS
#define OCXDLL_API __declspec(dllexport)
#else
#define OCXDLL_API __declspec(dllimport)
#endif

#include "OCXDLLDialog.h"

OCXDLL_API OCXDLLDialog*VideoCreate (CWnd *parent);
OCXDLL_API HRESULT    VideoDestroy(OCXDLLDialog *wnd);
OCXDLL_API HRESULT    VideoPlay (OCXDLLDialog *wnd, unsigned long IP, unsigned short port,
LPCTSTR user, LPCTSTR pass, __int64 historicalTime);
OCXDLL_API HRESULT    VideoStop (OCXDLLDialog *wnd);

```

In Solution Explorer, right-click on Source Files -> Add -> New Item. Add a new C++ file called OCXDLLInterface and copy in the following code:

```

#include "stdafx.h"
#include "OCXDLLInterface.h"
#include "OCXDLLDialog.h"

OCXDLL_API OCXDLLDialog*VideoCreate (CWnd *parent) {
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    OCXDLLDialog *wnd = new OCXDLLDialog();
    wnd->Create(OCXDLLDialog::IDD, parent);
    return wnd;
}

OCXDLL_API HRESULT    VideoDestroy(OCXDLLDialog *wnd) {
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    wnd->DestroyWindow();
    delete wnd;
    return S_OK;
}

```

```

OCXDLL_API HRESULT VideoPlay (OCXDLLDialog *wnd, unsigned long IP, unsigned short port,
LPCTSTR user, LPCTSTR pass, __int64 historicalTime) {
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    return wnd->Play(IP, port, user, pass, historicalTime);
}

OCXDLL_API HRESULT VideoStop (OCXDLLDialog *wnd) {
    AFX_MANAGE_STATE(AfxGetStaticModuleState());
    return wnd->Stop();
}

```

10. Add OCXDLL\_EXPORTS in the list of preprocessor definitions for the project. Right-click on the project->Properties. Expand C/C++, select Preprocessor. Add OCXDLL\_EXPORTS to the Preprocessor Definitions.

11. Add a Dialog based MFC Application project.

File->Add->New Project. Expand Visual C++ and select MFC. Select MFC Application, type in OCXEXE and click OK. You will be prompted with the MFC Application Wizard -> click Next. Change the Application type to Dialog based and click Finish.

12. In the Resource View window, expand OCXEXE->OCXEXE.rc->Dialog and double-click IDD\_OCXEXE\_DIALOG.

Delete the OK and cancel buttons and the "TODO: Place dialog controls here" text. Increase the width of the dialog window so two OCX controls can fit. Using the toolbox, drag on three new buttons where the OK button was. Change the following properties for the new buttons:

Caption: Add Video  
ID: IDB\_ADD

Caption: Start  
ID: IDB\_START

Caption: Remove Video  
ID: IDB\_REMOVE

13. Add a variable to the dialog.

Right-click on the dialog-> Add Variable. Click the Control variable checkbox. Under Variable Name, enter m\_btnAdd. Verify the Control ID is IDB\_ADD and click Finish.

14. Double-click on each of the new buttons and the methods OnBnClickedAdd, OnBnClickedRemove, and OnBnClickedStart should be added in OCXEXEDlg.cpp.

15. In OCXEXEDlg.h:

Add the required header files and std namespace:

```
#include <vector>
```

```
#include "..\OCXDLL\OCXDLLInterface.h"
using namespace std;
```

Add the following protected member:

```
vector<OCXDLLDialog*> m_windows;
```

16. In OCXEXEDlg.cpp:

Modify the OnBnClickedAdd, OnBnClickedRemove, and OnBnClickedStart methods as follows:

```
void COCXEXEDlg::OnBnClickedAdd()
{
    OCXDLLDialog *dlg = VideoCreate(NULL);

    dlg->SetParent(this);
    dlg->ShowWindow(SW_SHOW);
    dlg->MoveWindow((m_windows.size() & 1) * 250, (m_windows.size() & 2) * 50, 250, 100);
    m_windows.push_back(dlg);
}

void COCXEXEDlg::OnBnClickedRemove()
{
    if (m_windows.size() > 0) {
        OCXDLLDialog *dlg = m_windows.back();
        m_windows.pop_back();
        dlg->SetParent(NULL);
        VideoDestroy(dlg);
    }
}

void COCXEXEDlg::OnBnClickedStart()
{
    if (m_windows.size() > 0) {
        // This will connect to live video on localhost (127.0.0.1), camera #1, using user "admin"
        and password "admin"
        VideoPlay(m_windows.back(), 0x7F000001, 50010, _T("admin"), _T("A1crUF4="), 0);
    }
}
```

17. In OCXEXE.cpp, call `CoInitialize(NULL)` after `AfxEnableControlContainer()` in `InitInstance()`.

18. In Solution Explorer, right-click on the OCXEXE project -> Project Dependencies. Select OCXDLL as a dependency. Right-click on the OCXEXE project again and select Set as StartUp Project.

You may need to modify the camera details in file OCXEXEDlg.cpp in `OnBnClickedStart` to connect to the camera.

## Disclaimers and Legal Information

Copyright © 2016 Aimetis Inc. All rights reserved.

This material is for informational purposes only. AIMETIS MAKES NO WARRANTIES, EXPRESS, IMPLIED OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Aimetis Corp.

Aimetis may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Aimetis, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

Aimetis and Aimetis Symphony are either registered trademarks or trademarks of Aimetis Corp. in the United States and/or other countries.