Aimetis

Symphony SDK User Guide

Last Updated: 4/4/2018

Product Versions: 6.14

Contents

Overview 4
Quick Start 4
Requirements and Recommendations 5
Portability 5
Contact 5
Interoperability 6
Use Cases and Sample Code 8
- Farm Connectivity 8
- Video 8
- PTZ 8
- Events 8
- Farm Settings 8
- Recording 9
- Symphony Client Control 9
- Reports 9
- Security 9
- Video Analytics 9
- Timeline 9
- Navigation 10
- DIO Rules 10
- Hardware Devices 10
- Camera Usage 10
- Direct File Access (Persistence) 10

Symphony Managed Libraries used by the SDK 11
Symphony Media Streaming Libraries used by the SDK 12
Persistence 13
- Files 13
- Database 13

How to Add Streaming Video to a Project 14
- Requirements 14
- Procedure 14

How to Enable Camera Panel Switching in Symphony Client 14

How to Get a JPEG Image 15
Set Decoration Options 15
Get Live JPEG with Recording Set to Never 16
How to activate/deactivate a relay 17
How to enable/disable a camera tour using dbupdater 18
How to wrap OCX into a DLL and use in an MFC Application 18
How to Integrate 3rd Party Analytics into Symphony 23
Integrating a Sample into Symphony 24
Notes 25
Tools 26
AimetisIntelligenceStream-2.0.xml 27
AimetisIntelligenceStream-2.0.xsd 28
Disclaimers and Legal Information 31

Overview

Welcome to the Aimetis Symphony SDK. There are many exciting ways to integrate and extend Symhpony. 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?I=00198&L=en

The latest *Symphony SDK* is available here:
http://aimetis.com/Xnet/Downloads/Files.aspx?P=development%2fSDK

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/released.aspx
  - Once installed, use Symphony Client to add a camera.
    - From the main menu, select **Server**, and then **Configuration**. The **Server Configuration** dialog opens.
    - Click **New** to add a network camera.
  - Download and install the Symphony SDK: http://aimetis.com/Xnet/Downloads/Files.aspx?P=development%2fSDK
  - 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 2.0 or later. The recommended development IDE is Visual Studio 2005 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.

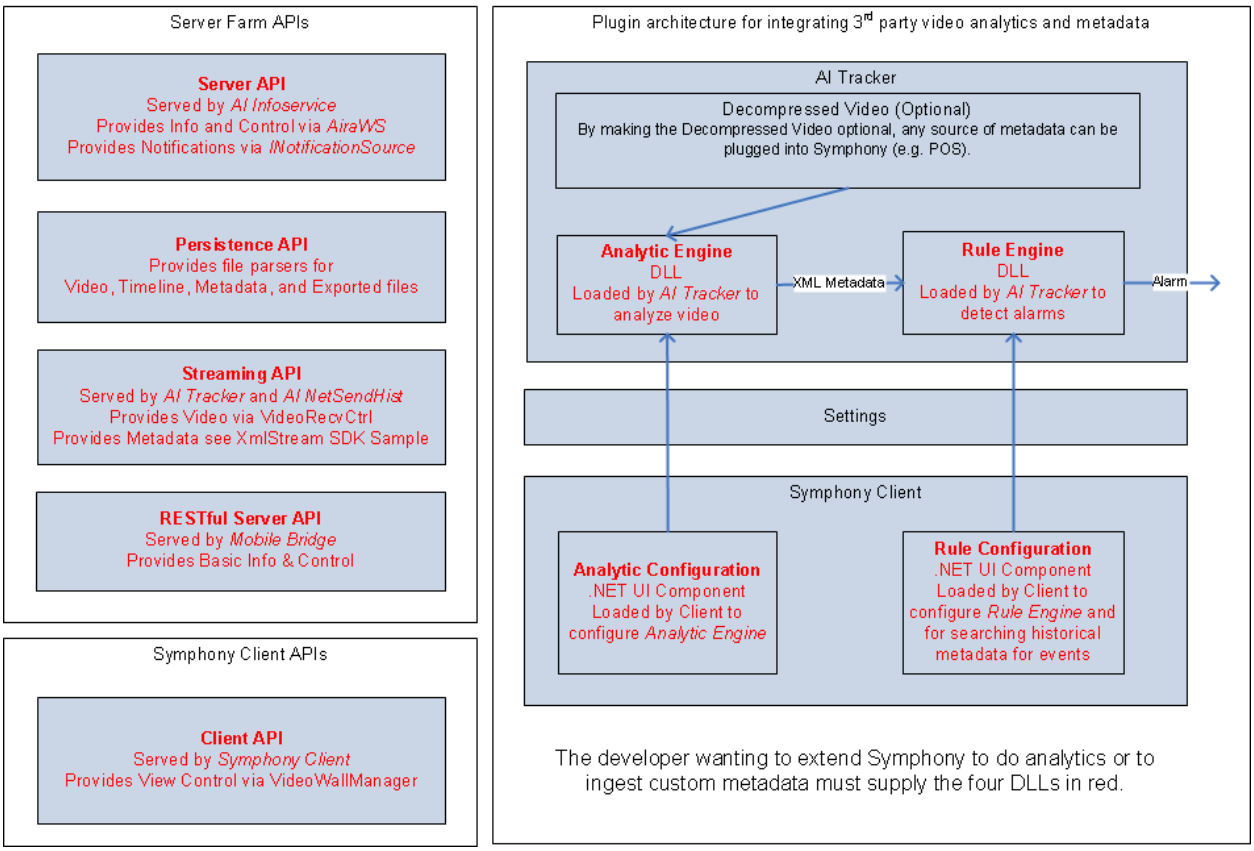
The portable Moble Bridge interface isn't as full featured as the Windows accessible AiraWS interface. If you need functionality in AiraWS from non-Windows platforms, then you must develop your own layer that provides this interface. Ported implementations of .NET will work (such as Mono), but the referenced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

Contact

Questions and comments can be sent to support@aimetis.com

Interoperability

Interoperability Overview



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, MobileBridge
Get a list of alarms	LiveStreamTest
Mark an alarm	LiveStreamTest
Listen for farm events	NotificationMonitorExample, NotificationMonitors, MobileBridge

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

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
Analyze video in Symphony from a 3rd party video system	AlgorithmIntegration
Load a symphony UI component in a 3rd party application	PluginHost
Incorporate your own analytics engine	AlgoSample
Incorporate your own rules engine	AlgoSample, TestRuleEngine
Incorporate your own analytics engine configuration	AlgoSampleConfiguration
Incorporate your own rules engine configuration	AlgoSampleConfiguration

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
  - Aimetis.Connectivity.dll
    - Library for connecting to servers. Usually used indirectly by samples.
  - BaseLibCS.dll
    - BaseLibCS.Communication.ServerConnectionManager for client server connectivity. Usually used indirectly by samples.
    - BaseLibCS.AiraWS.Signals for Web Service interface
    - BaseLibCS.CameraMessageStruct for structure definitions
    - BaseLibCS.Utils.MulticastMessage for constants
  - Core.dll (source part of SDK)
    - Aimetis.Symphony.SDK.SDKFarm for farm connectivity and event stream notifications
  - Farm.dll
    - DeviceModel.Client for client side device operations and information
    - DeviceModel.Server for server side device operations and information
  - Internationalization.dll
    - Aimetis.Symphony.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.
  - Aimetis.Symphony.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.

Run dbconfiguration.exe to view/change the current database connection string.

How to Add Streaming Video to a Project

Requirements

.NET framework 2.0 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 select **Server**, 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 decorations = new DecorationOptions(15, 0, true, 0);
// 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)
{
    // 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;
}
```

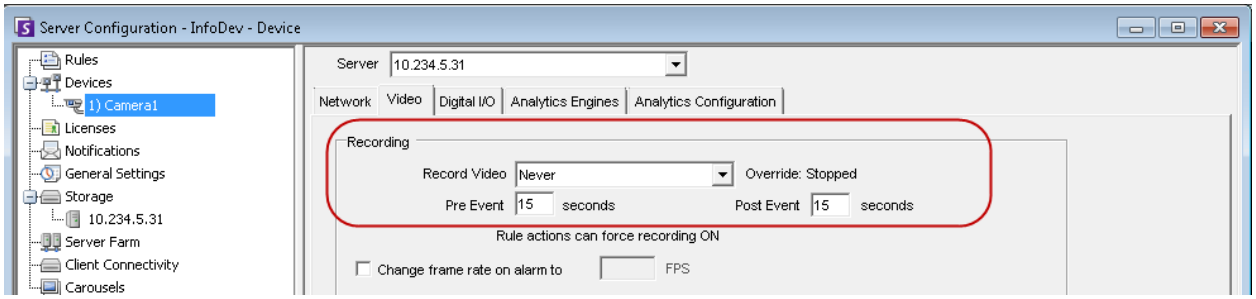
Get Live JPEG with Recording Set to Never

If the camera is set up to use an Analytic Engine, in which case it needs to decompress video, then ***recording is not required to get a live JPEG*.** Simply pass in a date-time to the **AiraWS** functions with a year less than 1990.

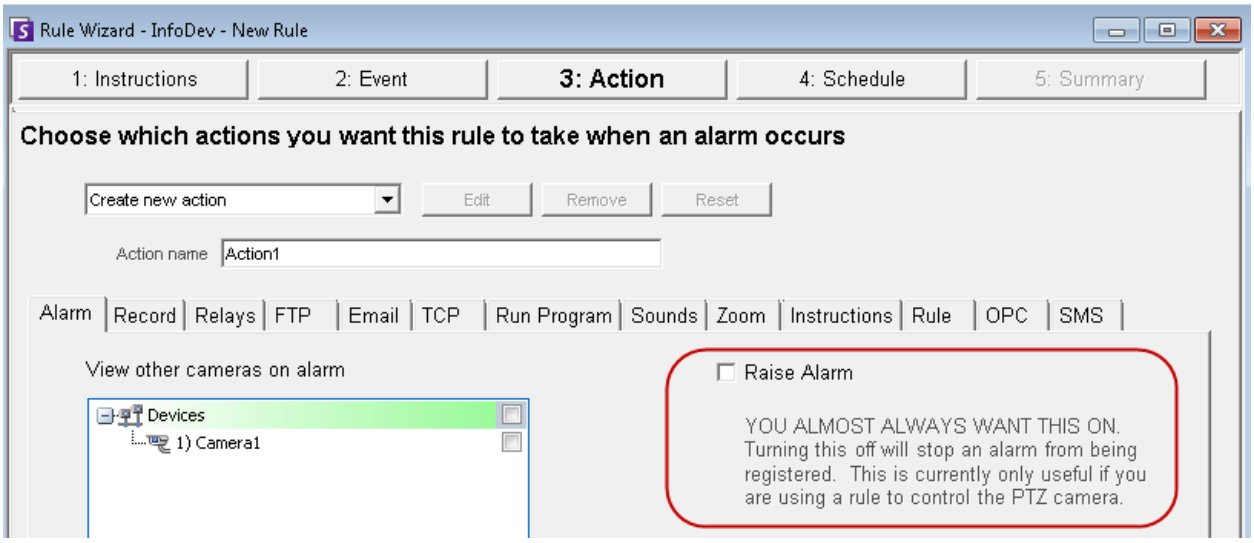
If Symphony supports edge storage for the camera (currently only AXIS) and edge storage is in use, it will also work. **NetSendHist** (the service that supplies historical JPEGs) will try to pull the JPEG directly from the camera.

The following approach could also work:

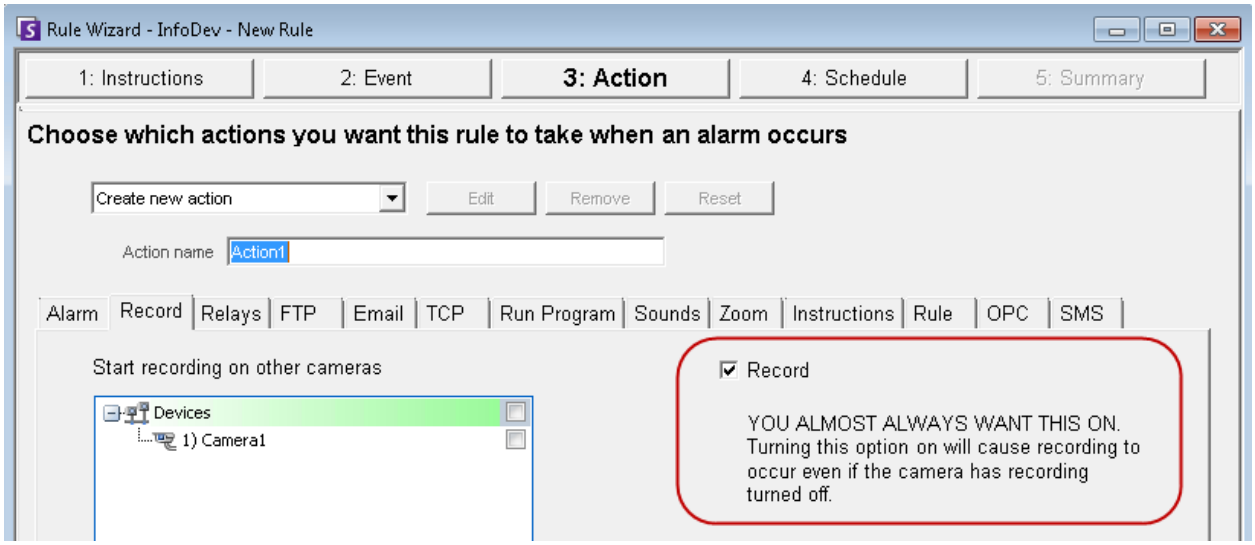
1. Under the **Video** tab for a device in the **Sever Configuration** dialog box, set the **Record Video** field to **Never**. **Pre Event** and **Post Event** seconds will default to 15 seconds.



2. In the **Rule Wizard**, add a rule.
3. In the **Alarm** tab, clear the **Raise Alarm** check box. You probably will not want these to register as alarms in the system (alarms will not appear in the timeline and will not appear in the alarm log).



4. In the **Record** tab, make sure the **Record** check box is selected.



5. When you want an image, cause the rule to break (you can use the **AddAlarmToCamera SDK** application). This will cause video to be stored.
6. Request a JPEG for the current time.

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().

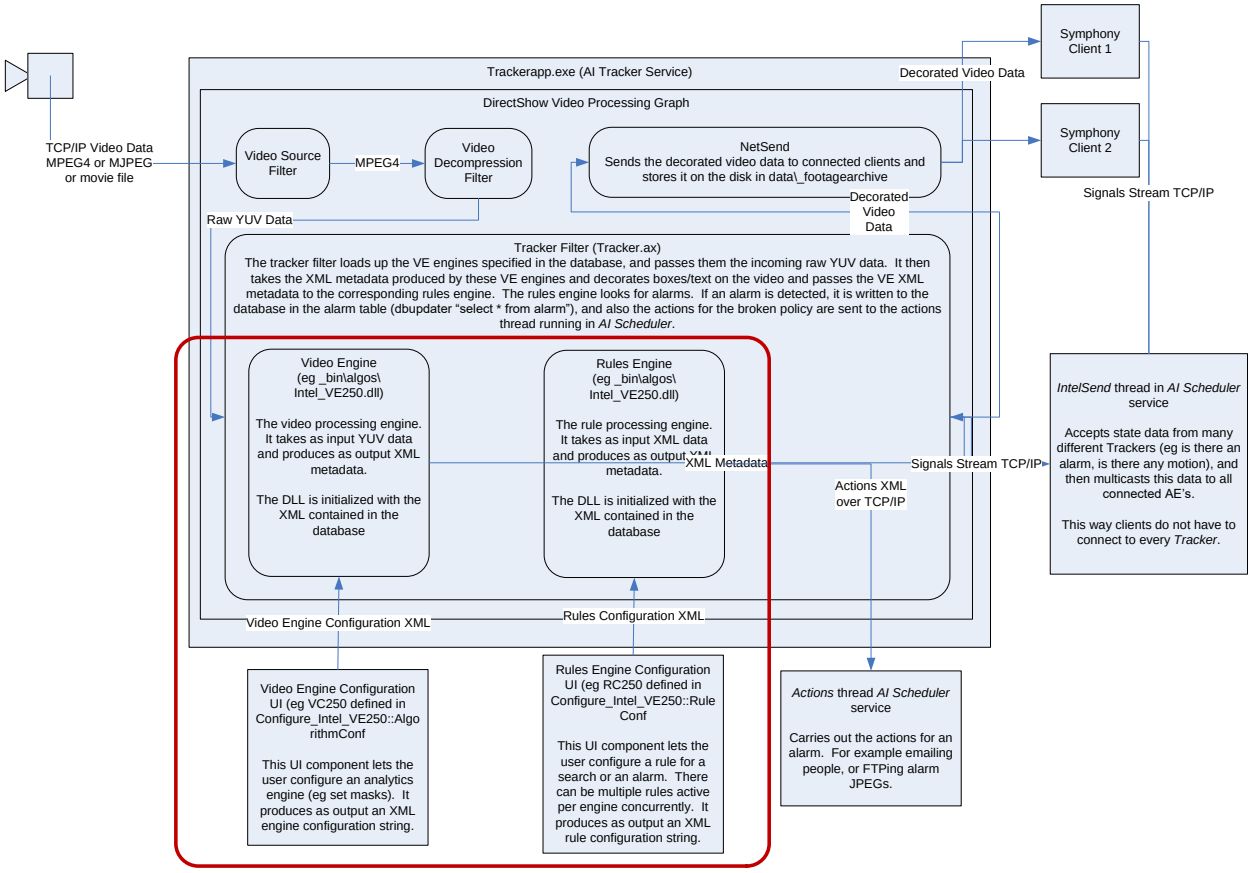
18. In Solution Explorer, right-click on the OCXEXE project -> Project Dependencies. Select OCXDLL as a dependency**.** Right-click on the OCXEXE project again and select Set as StartUp Project.

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

How to Integrate 3rd Party Analytics into Symphony

The ***bold/italic*** items below need to be provided by the technology integrator.

Every camera on a Symphony server has its own AI Tracker Service (Trackerapp.exe). The video from the camera is decompressed and passed into the *Tracker Filter* (Tracker.ax) which passes it to all analytics engines configured on that camera. The ***C/C++ analytic engine*** analyzes the video and produces XML metadata describing the scene that it passes to the corresponding ***C/C++ rule engine***. The rule engine processes the metadata and reports back to tracker whether or not an alarm occurred. If an alarm occurred, Tracker inserts the alarm into the Alarm table in the database and notifies other services/clients that are interested in the event.



A user configures the analytics and rule engines using corresponding ***C# analytic and rule configuration UI components***. These components are plugins for Symphony. They should be deployed into the Symphony Server's _bin\algos folder. Symphony Client pulls the UI plugins from the server when needed. Configurations are saved as XML in the database. The configuration strings are used to initialize the analytic and rule engines when the AI Tracker Service starts.

An ***XML analytics description file*** must be provided along with the other components to describe the capabilities of the *Analytic Engine* to Symphony. The following fields are of special note:

- DisplayName: the name of the analytic displayed in Symphony
- Description: the description of the analytic displayed in Symphony
- Display: flag to show/hide the analytic in Symphony
- SearchSupported: flag to enable/disable searching by the metadata created by the analytic.
- DLL: the filename of the dll of the analytic and rule engine
- UncompressedVideo: flag to enable/disable video decompressing for frame by frame analysis. This flag should be set to 0 when integrating metadata engine that does not require video for analysis.

Two projects in the SDK that provide a sample analytic that can be integrated into Symphony:

1. AlgoSample – the analytic and rule engines
2. AlgoSampleConfiguration – the UI used to configure the XML that is used to initialize the engines.

To add an analytics engine to Symphony that does not need decoded video, the developer simply should add/update one XML tag in algorithm XML: <UncompressedVideo>0</UncompressedVideo>. This is useful for integrating sources of metadata that do not depend on analyzed video (e.g. POS, ATM). The analytics configuration for this engine would probably include connectivity information for the metadata source. The rule engine/configuration can now parse the metadata within Symphony and add events through the Symphony infrastructure that are specific to the metadata type. Each metadata source must be associated with a camera (the alarms appear in that camera's timeline).

Integrating a Sample into Symphony

Integrate the sample into Symphony:

1. Download the latest Symphony SDK: http://aimetis.com/Xnet/Downloads/Files.aspx?P=development%2fSDK
2. On the Symphony server, shut down the services (killall 9).
3. In the SDK\bin\algos folder, copy these files to the **C:\Program Files\Aimetis\Symphony_bin\algos** directory on the Symphony server:
   (for 64-bit machines, copy to C:\Program Files(x86)\Aimetis\Symphony_bin\algos\)
   - AlgoSample.dll: ***analytics and rule engines***
   - Configure_AlgoSample.dll: ***analytics and rule configuration UI components***

- AlgoSample.xml: ***the XML analytics descripton file***
- The naming convention above should be followed when deploying an analytics engine. Replace the string "AlgoSample" in the filenames above with a short descriptive name for your engine.

Test the newly integrated sample:

4. On the Symphony server, start the services (killall 5).
5. In Symphony Client
    a. From the main menu, select **Server**, and then **Configuration**. The **Server Configuration** dialog opens.
    b. Select a camera, and click the **Analytics Engines** tab. The new analytic engine **AlgoSample** should be displayed in the list.
6. Test the analytic. AlgoSample counts the number of frames that have changed and displays either the count or the percentage as a decoration.
7. Create a rule and set the Activity Threshold as appropriate. (If percentage is displayed, enter a percentage. If the number of pixels is displayed, enter number of pixels). Alarms should be generated if the number/percentage displayed is above the threshold set.

Integrate your own analytics/rules engine/configuration into Symphony:

8. Integrate your analytic by modifying the AlgoSample and AlgoSampleConfiguration projects.
9. Change the AlgoSample.xml as needed. You will at least want to modify the <DisplayName> and <Description> tags.
10. Repeat steps 2-4 to integrate the updated analytic engine with Symphony.

Notes

- Symphony Client caches recently loaded analytics UI components. You can clear this cache manually by deleting **%appdata%\aimetis\sc_cache**.
- The **rule** and **analytic** configuration classes must inherit UserControl and implement IPlugin.
- To get the analytic to work with Symphony, you must compile the AlgoSample and AlgoSampleConfiguration projects in **Release** mode.
- If changing the analytic assembly names, the DLL for the analytics and rule configuration must start with Configure_ for Symphony to recognize it properly.
- The XML generated in AlgoSample.cpp is stored in the <data root>_signals2 folder. <data root> is set when installing Symphony.

- The Sections AimetisIntelligenceStream-2.0.xml and AimetisIntelligenceStream-2.0.xsd below provide sample metadata used in current Symphony analytics.
- To change the frames per second received by the analytic engine, add the AnalysisFPS tag to the analytic configuration XML. For example, <AnalysisFPS>15</AnalysisFPS>.

Tools

- To help develop the C# configuration UIs, you can use the PluginHost SDK application. It's a simple tool that allows you to load an analytic or rule configuration and easily test the UI and view the XML it generates. You may need to copy the application to the Symphony _bin folder.
- Instead of opening the _signals2 folder to look at the XML metadata, you may use the XmlStream SDK application to connect directly to the server and view the XML as it is generated.
- The AddAlarmToCamera SDK application can be used to manually generate alarms for a camera and policy/rule.

AimetisIntelligenceStream-2.0.xml

```
<?xml version="1.0"?>

<stream
  xmlns="http://www.aimetis.com"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="AimetisIntelligenceStream-2.0.xsd">

        <frame
                utc='11051078632640000'
                brightness='10'
                contrast='10'
                cordcut='1'
                cameraobstructed='1'>

                <!-- xml for creating a new object or tracking one already created -->
                <object id='124' confidence='95'>
                        <class>Car</class>
                        <rect x='0' y='0' width='192' height='475'/>
                        <speed>12.6</speed>
                </object>

                <!-- an object which came to rest in the video -->
                <!-- if the object was actively tracked before stopping, id will correspond to the tracked object -->
                <object id='122'>
                        <class>Car</class>
                        <rect x='45' y='102' width='150' height='50'/>
                        <speed>0</speed>
                        <state>Stationary</state>
                </object>

                <!-- xml for deleting an object which was tracked up until now -->
                <!-- "id" must have appeared at least once in an object tag -->
                <delete id='123'/>

                <!-- xml for flagging a policy break -->
                <!-- if the alarm was caused by an object moving or a stationary object -->
                <!-- the id will correspond with the offending object -->
                <!-- if the alarm was a cord-cut or camera-obstructed alarm, the id will be 0 -->
                <alarm id='123'>
                        <policyid>1</policyid>
                        <alarmonid>2</alarmonid>
                        <occur>0</occur>
                </alarm>
        </frame>
        <frame
                utc='11051078632641000'
                brightness='10'
                contrast='10'
                >
                <!-- xml for deleting an object which was tracked up until now -->
                <delete id='124'/>
```

```
                <!-- xml for deleting an object which was stationary up until now -->
                <delete id='122'/>
        </frame>
</stream>
```

AimetisIntelligenceStream-2.0.xsd

```
<?xml version="1.0"?>
<xs:schema      xmlns:xs="http://www.w3.org/2001/XMLSchema"
                targetNamespace="http://www.aimetis.com"
                xmlns="http://www.aimetis.com">


 <xs:simpleType name="classtype">
  <xs:restriction base="xs:string">
   <xs:enumeration value="Car"/>
   <xs:enumeration value="Person"/>
   <xs:enumeration value="Unknown"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="percentage">
  <xs:restriction base="xs:decimal">
   <xs:minInclusive value="0"/>
   <xs:maxInclusive value="100"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="statetype">
  <xs:restriction base="xs:string">
   <xs:enumeration value="Stationary"/>
   <xs:enumeration value="Leaving"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:simpleType name="colourvaluetype">
  <xs:restriction base="xs:string">
   <xs:pattern value="#[0-9a-fA-F]{6}"/>
  </xs:restriction>
 </xs:simpleType>

 <xs:element name="colour">
  <xs:complexType>
   <xs:simpleContent>
    <xs:extension base="xs:string">
     <xs:attribute type="colourvaluetype" name="value" use="required"/>
    </xs:extension>
   </xs:simpleContent>
  </xs:complexType>
 </xs:element>
```

```
<xs:element name="colourlist">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="colour" minOccurs="1" maxOccurs="16"/>
  </xs:sequence>
 </xs:complexType>
</xs:element>

<xs:element name="class">
 <xs:complexType>
  <xs:simpleContent>
   <xs:extension base="classtype">
    <xs:attribute name="confidence" type="percentage"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
</xs:element>

<xs:element name="rect">
 <xs:complexType>
  <xs:attribute name="x"              type="xs:integer"/>
  <xs:attribute name="y"              type="xs:integer"/>
  <xs:attribute name="width"          type="xs:integer"/>
  <xs:attribute name="height"         type="xs:integer"/>
 </xs:complexType>
</xs:element>

<xs:element name="object">
 <xs:complexType>
  <xs:sequence>
   <xs:element ref="class"/>
   <xs:element ref="rect"/>
   <xs:element name="speed"                  type="xs:decimal"/>
   <xs:element name="size"                           type="decimal"/>
   <xs:element name="state"                  type="statetype"/>
   <xs:element ref="colourlist"/>
  </xs:sequence>
  <xs:attribute name="id"                type="xs:integer" use="required"/>
  <xs:attribute name="confidence"  type="percentage" default="0"/>    <!-- Our confidence that this object will be
tracked as a foreground object 0 == definitely distracting motion, 100 == definitely foreground -->
  <xs:anyAttribute/>
 </xs:complexType>
</xs:element>

<xs:element name="alarm">
 <xs:complexType>
  <xs:attribute name="id"                type="xs:integer" use="required"/>
  <xs:attribute name="policyid"      type="xs:integer" use="required"/>
  <xs:attribute name="alarmonid"  type="xs:integer" use="required"/>
  <xs:attribute name="occur"                type="xs:unsignedLong"/>
  <xs:anyAttribute/>
 </xs:complexType>
</xs:element>
```

```
<xs:element name="delete">
 <xs:complexType>
  <xs:attribute name="id"                    type="xs:integer" use="required"/>
 </xs:complexType>
</xs:element>


<xs:element name="stream">
 <xs:complexType>
  <xs:sequence>
   <xs:element name="frame">
    <xs:complexType>
     <xs:sequence>
      <xs:element ref="object"                                  minOccurs="0" maxOccurs="65535"/>
      <xs:element ref="delete"                                  minOccurs="0" maxOccurs="65535"/>
      <xs:element ref="alarm"                                   minOccurs="0" maxOccurs="1"/>
     </xs:sequence>
     <xs:attribute name="utc"          type="xs:unsignedLong"/>
     <xs:attribute name="brightness"           type="xs:integer"/>
     <xs:attribute name="contrast"   type="xs:integer"/>
     <xs:attribute name="cordcut"    type="xs:boolean"           default="0"/>
     <xs:attribute name="cameraobstructed" type="xs:boolean"           default="0"/>
     <xs:anyAttribute/>
    </xs:complexType>
   </xs:element>
  </xs:sequence>
 </xs:complexType>
</xs:element>
</xs:schema>
```

Disclaimers and Legal Information

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

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.