February 13, 2014

IMINT
IMAGE INTELLIGENCE

Vidview Integration Considerations

Andreas Lifvendahl

Table of Contents

1 Introduction

Imint Vidview is a tool for integrating video viewing capabilities with Vidhance®[1] live video enhancements into other software applications, such as user panels and operator consoles. It handles video display, management, and enhancement functionality. Imint provides the Vidview SDK, written in C# and within the Microsoft .NET framework, to support integration. However, the host application does not have to align with this. This document provides an overview of the available means of integration, and some advice regarding software architecture design choices.

1. http://imint.se/technology/vidhance-0

2 Process isolation or same process?

Vidview can be running in the same process as the host application, or as an isolated process. A first evaluation of Vidview and the Vidview SDK is easier to do in one process, and the Vidview SDK for .NET Getting Started Guide[2] is based on this assumption. Keeping everything within one process reduces complexity, and facilitates debugging. On the other hand, running Vidview in an isolated process makes the host application more robust, as it can provide better recovery upon failures - in the event of a serious error, Vidview can be terminated by the host process and restarted without affecting other part of the host application.

Process isolation can be turned on in an existing single-process project by setting a flag. For more information, see the Vidview .NET SDK Getting Started Guide, Section 4.2[3].

2.1 Same process

This is the default and obvious route within the Vidview for .NET SDK, which uses Windows Forms. This requires the host application to run under .NET, or

2.2 Process isolation

If a process separation architecture is chosen, the next question is whether Vidview viewing windows should be implemented as a dockable window within the host application, or as a standalone window application. The dockable window method can provide a more seamless user interface, but can also create issues with user interaction in different windows in the application.

If the choice is made to not use dockable windows, the best supported way for the host application to communicate with the stand-alone Vidvew window is through a TCP/IP interface. This interface is described in more detail here[4].

The recommended approach to implement dockable windows is to develop using the Vidview SDK for .NET. Optionally, the Vidview SDK for ActiveX controls, which are currently under development and lack documentation, can be used.

3 Programming languages

Another important consideration relating to how the host/parent application is developed, or will be developed, is choice of development method, development tools and programming languages.

2. http://vidview.imint.se/sdk/net/Vidview .NET SDK Getting Started Guide.html
3. http://vidview.imint.se/sdk/net/Vidview .NET SDK Getting Started Guide.html#using.vidhance_process.separation
4. http://vidview.imint.se/sdk/tcp/Vidview TCP SDK Getting Started Guide.html#the.remote.interface

3.1 SDK for .NET

To use the Vidview SDK for .NET as is the following programming languages can be used:

- C#
- C++/CLI (derived from earlier "managed C++")
- Visual Basic .NET
- F#

The default supported project is Windows Forms, but Vidview can also run in a WPF project using a WindowsFormsHost (currently not documented).

3.2 SDK with ActiveX support

ActiveX support in the Vidview SDK is under development, and will provide means to interface with Vidview through ActiveX controls. This will enable integration opportunities with other software development frameworks and programming languages, such as:

- C (Win32 API)
- C++ (MFC)
- C++ QT (with connectors)
- Java
- Delphi

4 Windows, Linux, or OS X?

So far, Vidview has been available only for development on Windows. Work is ongoing to provide better support also for development on Linux, where .NET applications can run on the Mono platform. For early adopters, the Vidview application interface over TCP/IP as described above is a recommended pathway. Development on Mac OS X will also be possible in the future.

5 Video input

Vidview supports DirectShow input on Windows. On Linux, GStreamer is used instead of DirectShow. There are also a few Imint input modules for MJPEG streams, images, as well as a few hardware specific ones, such as Blackmagic Design's Decklink series of products. Because Vidview is highly modular, it's possible to make your own media modules to provide support for any kind of formats. For optimal performance, these modules should decode the input media and provide Vidview with YUV420 frames, but other pixel formats are also supported.

6 Summary overview chart

6.1 Process Isolation

	Vidview SDK for .NET	Vidview SDK for ActiveX	Vidview SDK over TCP
same process	yes	yes	not recommended
isolated process	yes	yes	yes

Table 1: Process isolation support

6.2 Operating System

	Vidview SDK for .NET	Vidview SDK for ActiveX	Vidview SDK over TCP
Windows XP 32-bit	yes	yes	yes
Windows 7 32-bit	yes	yes	yes
Windows 7 64-bit	yes	yes	yes
Windows 8	upon request	no	upon request
Linux (Ubuntu 12.10)	upon request	no	upon request
Mac OS X (Maverick)	upon request	no	upon request

Table 2: Operating systems support

6.3 Programming Languages

	Vidview SDK for .NET	Vidview SDK for ActiveX	Vidview SDK over TCP
C#	yes	not recommended	not recommended
VB.NET	upon request	not recommended	not recommended
F#	upon request	not recommended	not recommended
C++/CLI	yes	yes	not recommended
C++/MFC	no	yes	not recommended
C++/QT	no	yes	yes
Java	no	yes	yes
Delphi	no	yes	yes
LabView	no	no	yes

Table 3: Programming languages support