October 26, 2012

Vidview SDK TCP Getting Started Guide

Simon Mika

Table of Contents

1 Introduction

Vidview Window is a video viewer application that can use Vidhance video enhancements. Although Vidview Window is a standalone application, it is indented to be controlled from another application over TCP/IP. Vidview Window will play video either available on disk, from capture devices, or received over the network.

2 Prerequisites

2.1 System requirements

Vidview requires Windows 7, .NET 4.0 and a GPU from NVidia, ATI/AMD or an Intel HD Graphics 3000 or better.

2.2 Installation

The installation process is described in the Vidview SDK Demo Installation Guide.html[1].

2.3 Requesting a License

In order to use Vidhance video enhancements and/or extra video codecs you need to request a license for the computer you run on. The SDK Demo Application installer does this for you when you enter your key, as does the Demo Application itself upon launch should a license be missing. If you wish to request a new license, you can use the license tool (the padlock) in the Demo Application.

Alternatively, you can start the Demo Application with the following commandline: `Imint.Vidview.Window.exe -s "license.request 01234ABCDabcd"` where `01234ABCDabcd` is replaced with your license key.

All of the above will require a working internet connection.

2.4 Telnet Client

During the integration process you will need a telnet client. If you have no other preferences we recommend using Putty[2], but the one bundled with Windows will also suffice.

3 Getting Started

1. Start Vidview Window with the argument `Imint.Vidview.Window.exe -r telnet://:23`, either from the command line, or by modifying the start menu shortcut.
2. Use your telnet client to connect to localhost, port 23.
3. When connected you will see a prompt: `# >`.
4. Type: `media.open file:///c:/users/user+name/documents/video.avi` and press enter. (Path will depend on operating system and user name.)
5. Type: `media.play` and press enter.

1. http://vidview.imint.se/installation/Vidview SDK Demo Installation Guide.html
2. http://www.chiark.greenend.org.uk/%7Esgtatham/putty/download.html

6. The video should now play.

4 The Remote Interface

4.1 Introduction

The remote interface comes in two flavors:

TCP
: A raw interface suitable for usage by software.

Telnet
: A more user friendly interface that supports editing (backspace, delete, left and right arrow), history of previous commands (up and down arrow), auto completion (tab) and help (tab pressed twice)

4.2 Activating

The server is activated by passing `-r [protocol]://[address]:[port]` as an argument at the command line when starting Vidview.

`protocol`
: The protocol the server should speak, either `tcp` or `telnet`.

`address`
: The IP address the server should bind to. Omit to bind to all interfaces.

`port`
: The port on which the server should listen to connections.

Examples:

- `Imint.Vidview.Window.exe -r telnet://:23`
- `Imint.Vidview.Window.exe -r tcp://localhost:1337`

The number of concurrent connections is unlimited. Every connection will get its own replies and notifications.
Vidview can listen to multiple ports by adding several arguments: `Imint.Vidview.Window.exe -r telnet://:23 -r tcp://localhost:1337`. This way your application may use the raw tcp connection, and you may use the more convenient telnet interface during development.

4.3 Usage

4.3.1 Paths

The remote interface is structured using three classes of items:

properties
Properties are used to contain a single value.

methods
Methods are used make something happen.

objects
Objects are used to group items together. They may contain properties, methods and other objects.

Objects, properties and methods are qualified by paths. A path starting with a single dot is always an absolute path. Paths not starting with a dot are relative to the current working object. Changing the current working object in the object hierarchy is done by sending the path to the desired object. The path to the current working object is indicated in the prompt. Sending a `..` the current working object is changed one step higher up the hierarchy. Starting a path with `.` will make it absolute. Sending a line with only `.` will change the current working object to the topmost one (i.e. the root object).

4.3.2 Response Types

All responses begin on a new line. The first letter of the line qualifies the response type. The second letter is always a whitespace. The four response types are:

prompt
The prompt response is followed by the current working object path and ends with `>`. Example: `# media>`.

value
The value response is the result of getting or setting a property. Example: `$ media.position 00:00:00`

notification
The notification response is generated when a property, for which notifications are activated, is changed. Example: `% media.position 00:00:00`

error
The error response signals that the operation failed. Example: `! media.open`

4.3.3 Getting a Property

To get the value of a property send its path without any additional argument. The answer will be in the form of a value response.
Example command: `media.position`, response: `media.position 00:00:00`

4.3.4 Setting a Property

To set the value of a property send its path with the value as argument. It will return a value response.
Example command: `media.position 00:00:00`, response:`$ media.position 00:00:00`.

4.3.5 Activating Notifications for a Property

To get notifications of changes to a property just type the full name of the property at the prompt followed by a space and the string `notify` and press enter.
Example command:
`media.position notify`,
response:
`$ media.position 00:00:00`
and on changes a notification like this will arrive:
`! media.position 00:00:01`

4.3.6 Invoking Methods

Methods may require one or more arguments. Arguments are separated from each other using a space character. When a value contains space characters they must be prepended with backspace, or the value must be surrounded by double quotes.
Example:
`media.play`,
`media.open file://c:/video.avi`.

4.3.7 The Telnet Interface

Editing on the command line can be done by using backspace, delete, left and right arrow keys. Auto completion is performed using the tab button. If more than one auto completion option is available, a help text is displayed.

4.3.8 Reference Manual

To get a complete reference of all available properties, methods, and objects, invoke the `help` method at the root level. This will generate a reference manual and open it in your browser.

4.3.9 Settings

Some properties you might want to be set every time you start Vidview. In that case, put them in a text file named `settings.conf` in the Vidview installation folder. In case you want to use different settings for different instances of Vidview you may want to specify a settings file by adding something like `-c settings1.conf` to the command line.

4.3.10 The Raw TCP Interface

When using the raw interface the server does not echo any characters back. Data is encoded using UTF-8[3]. A single linefeed (short name: `LF`, C escape sequence: `\n`, hexadecimal: `0x0a`, decimal: `10`) is used as the newline character.

5 Window Handling

If you want to run Vidview Window as a viewer that appears as a part of your application's window, you need to add `-s window.visible false` to your commandline. We recommend that you then dock the window and resize it using the Windows API (`SetParent` and `MoveWindow` functions in `user32.dll`). It is also possible to simply move the window into the right position like this:

`window.position 10, 20`
: Move the upper left corner of the window to screen coordinates 10, 20 in pixels.

`window.size 1000, 800`
: Change the viewer's width to 1000 pixels and its height to 800 pixels.

`window.border none`
: Remove borders from the viewer window.

`window.topmost true`
: Make viewer window stay on top.

`window.visible true`
: Make viewer window visible.

Whenever your application loses focus it should send `window.topmost false`, and when it regains focus, it should send `window.topmost true`. When your application window moves, it should update the viewer's position, and when the size changes, it should update the viewer's size.

If your application has multiple windows and of them is a Vidview window, you may want to not shut down Vidview when the user clicks the close icon. Instead, you may want to hide Vidview and continue running it in the background so that, the next time the user wants to view the video, no additional startup time is needed. This can be accomplished by using the following initialization sequence:

3. http://en.wikipedia.org/wiki/UTF-8

`window.visible notify`
: Turn on notifications on window visibility in case you want to change the status of your user interfaces depending on whether the Vidview window is visible.

`window.title "Video 1"`
: Change the title of the Vidview Window window to "Video 1"

`window.closeaction hide`
: Make the window hide instead of closing down the application.

`window.visible true`
: Make the viewer window visible.

When you intend to close down your application you need to send:

`window.visible false`
: Hide the viewer window.

`close`
: Close the viewer window.