AvSelect

AvSelector is a tray tool which can be used to configure monitor and sound settings quickly using a context menu. It also has a command-line interface which can apply selections or host programs such as games.

# Features

It can change the following types of settings:

* Primary display device
* Display settings
  + Display Enabled or disabled
  + Display clone
  + Construct an extended desktop. It should be fully possible to build any extended desktop configuration (even for 3+ simultaneous displays) by specifying relative coordinates.
  + Resolution change
  + BitsPerPixel (e.g. 32 bit color), RefreshRate, ScanLineOrdering
* Default audio playback device (as seen in the Sound control panel)

In order to limit the displayed options only to those relevant to the user, and to permit a greater deal of flexibility and automation, the user supplies a file called “**config.xml**”, placed in the same directory as AvSelect.exe.

There is also a command-line interface which can be used with scripts or to launch a game under an alternative video/audio setup such that it is restored on game exit.

# Installing

There is currently no installer. Copy “AvSelect.exe” to your directory of choice and place a “config.xml” (described below) in the same directory.

If you Start->Run shell:startup, it will open an explorer window for startup shortcuts. You can place a shortcut to AvSelect.exe in order to automatically run on startup.

# Tutorial

#### PC Config

Need help figuring out what the “Friendly Names” of your devices are? Run AvSelect.exe (regardless of if config.xml exists or is correct yet); after launching, a new tray icon should appear in the taskbar. Right click this tray icon; in the menu and choose “About”. From the About window, click “**PC Config**”. This information is needed to build the XML such that the names/ids are recognized.

Consider copying out the text to a text editor for convenience. But note that the contents of PC config can change if you change settings or add a new device, so open PC config again when in doubt.

For advanced tweaking, use the normal control panel(s) to set up the display and audio state exactly like you’re targeting, then use AvSelect.exe’s PC config to view the settings as seen by the tool.

#### Menu Building Basics

AvSelect.exe expects to be launched from a directory that also contains a file named *config*.*xml.* Here is an example config.xml which provides a menu which allows the user to swap their primary display. Gray xml is *optional*.

<?xml version="1.0" encoding="utf-8"?>

<AvSelectorConfig>

<MenuItems>

<MenuItem Name="TV as primary monitor">

<Hotkey Char="T" ModAlt="True"/>

<TargetStates>

<State Type="PrimaryDisplay">

<Target FriendlyName="PanasonicTV0" />

</State>

</TargetStates>

</MenuItem>

<MenuItem Name="Default">

<Hotkey Char="X" ModAlt="True"/>

<TargetStates>

<State Type="PrimaryDisplay">

<Target FriendlyName="BenQ G2400W" />

</State>

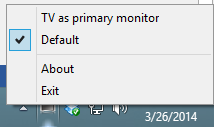
</TargetStates>

</MenuItem>

</MenuItems>

</AvSelectorConfig>

When the program is run, the tray icon will appear and when right-clicked, it produces this menu:



Notice that the menu item entry text corresponds to the xml <MenuItem Name="…">. When a menu item is clicked, the <State>(s) under its <TargetStates> will be applied. This very basic example will *not* actually enable/disable/reconfigure the monitor, it simply changes the primary display under the expectation that both devices are already enabled. More <TargetStates> are needed to utilize more complex behaviors.

The user may also associate a “**hotkey**” with each menu item, enabling selection via the keyboard. These are registered simply by calling the RegisterHotKey API and cannot collide with other system hotkeys.

Examples of hotkeys:

<Hotkey Char="1" ModCtrl="True" /> <!-- Ctrl+1 -->

<Hotkey Char="A" ModShift="True" ModAlt="True"/> <!-- Alt+Shift+A -->

<Hotkey VkHex="78" ModCtrl="True" /> <!-- Ctrl+F9 -->

See <http://msdn.microsoft.com/en-us/library/windows/desktop/dd375731(v=vs.85).aspx> for a list of Virtual-Key Codes that can be supplied (in hexadecimal) via with VkHex.

#### Multiple Target States

Multiple <TargetStates> can be provided under each <MenuItem>—when the MenuItem is clicked, the <State>s are then applied *sequentially*. The check box in the menu item will appear checked whenever *all* target states are already satisfied.

<MenuItem Name="TV as primary monitor">

<Hotkey Char="T" ModAlt="True"/>

<TargetStates>

<State Type="DisplaySettings">

<Target FriendlyName="PanasonicTV0" />

<Enabled Value="True" />

<Resolution Width="1920" Height="1080" BitsPerPixel="32" />

<LocationRelativeToTarget FriendlyName="BenQ G2400W" X="-1920" Y="0" />

</State>

<State Type="PrimaryDisplay">

<Target FriendlyName="PanasonicTV0" />

</State>

<State Type="DefaultAudioDevice">

<AudioDevice FriendlyName="\*(NVIDIA High Definition Audio)" />

</State>

</TargetStates>

</MenuItem>

The resulting text informs you of all relevant information specific to your setup. Each field provided is a usable attribute name for <Target/> or <AudioDevice/> to identify the device. You can use multiple if needed.

The FriendlyName attribute for <Target/> and <AudioDevice/> supports **wildcard matching**. This means you can use an \* or ? to match device names that fit a pattern. For example, <AudioDevice FriendlyName="**\***(NVIDIA High Definition Audio)" /> would match any audio devices which end with “(NVIDIA High Definition Audio)”. Note that you must carefully avoid ambiguity. This is advantageous when you want to switch to “whatever is connected to HDMI”, because attaching different HDMI devices results in different FriendlyNames but sharing a suffix that can be referenced. FriendlyName="\*(NVIDIA High Definition Audio)" /> actually ends up choosing whichever audio device is connected to the nVidia HDMI port. Again, consult PC Config to see all connected devices to decide how you plan to identify them.

# Reference

### Misc Options

<AvSelectorCo**nfig DoubleClickTray="[MenuItem Name]" RestoreOnTrayExit="True" SetOnExit="[M**enuItem Name]">

DoubleClickTray: Supply a menu item name (defined in a <MenuItem>), and this menu option will automatically be chosen when the tray icon is double clicked.

RestoreOnTrayExit: If set to “True”, then the state of the monitors and default audio devices is saved at launch and when the program exits, the original settings are restored.

SetOnExit: Supply a menu item name (defined in a <MenuItem>), and this menu option will automatically be chosen when the program exits, due to the user closing it, or due to Shutdown.

## Audio Settings

The only configurable audio option is choosing the default output device.

<State Type="DefaultAudioDevice">

<WaitUntilDisplayEnabledComplete Value="True" DelayMs="8000" />

<PlayTestSound Value="True" />

<AudioDevice FriendlyName="\*(NVIDIA High Definition Audio)" />

</State>

This selects the default audio device by its FriendlyName.

However, a common problem associated with HDMI audio is a need to wait for display enablement to complete (if we’re configuring audio and video at the same time). Such audio devices often do not appear until several seconds *after* the display has been enabled. WaitUntilDisplayEnabledComplete will asynchronously delay the attempt to reference the new audio device for a specified amount of time. Note that the State xml for enabling the display should be prior to the DefaultAudioDevice state, because States are applied sequentially. If the time is long enough, the new device should be present and it will work. Watch the Sound control panel while enabling the HDMI device to learn the timing of your system.

## DisplaySettings

### Targetting a Display

Target <Target FriendlyName="…" /> is usually sufficient, but it is possible to refer to a monitor using other attributes such as AdapterLuid or Id. Use Pc Config while the device is connected to find these values.

### Disable Monitor

The following is sufficient to disable a monitor under normal circumstances.

<State Type="DisplaySettings">

<Target FriendlyName="PanasonicTV0" />

<Enabled Value="False" />

</State>

(Note: In an extended desktop config of 3+ monitors, more work is required to disable the “middle monitor” because any “gap” needs to be avoided. Other monitor positions will require adjustment; see below.)

### Extended Desktop

Extended desktop coordinates need to be specified when enabling a monitor, as this information is not cached. So <Enabled Value="True" /> is likely to be insufficient—more info should be included. Both the position and resolution must be possible to resolve, unless <CloneTarget> is supplied. This snippet enables this TV, with an “extended” desktop that extends to the left of the primary display.

<State Type="DisplaySettings">

<Target FriendlyName="PanasonicTV0" />

<Enabled Value="True" />

<Resolution Width="1920" Height="1080" BitsPerPixel="32" />

<LocationRelativeToTarget FriendlyName="BenQ G2400W" X="-1920" Y="0" />

</State>

For <LocationRelativeToTarget>, the X and Y coordinates control the **top-left coordinate of the monitor’s desired display rectangle.** This is within a “virtual space” of the desired virtual extended display. Coordinates are allowed to be negative; if X is negative, it will appear to the left of the primary monitor, or for the Y case, negative will result in appearing above the primary monitor. Since the **display rectangle edges must be connected**, the display resolution (which is the rectangle size) must also be considered. For example, consider two 1080p monitors. The primary X,Y is 0,0, and to place a secondary to right, it’s coordinates would be 1080,0. Or to the left, -1080,0.

You should normally use <LocationRelativeToTarget> on a second display to position itself relative to the primary monitor (which is at 0, 0)

### Cloning

<CloneTarget FriendlyName="BenQ G2400W" /> would enable PanasonicTV0 as a clone of BenQ G2400W. It will share its resolution and position with the existing display, so these attributes are no longer mandatory. However, they can still be provided. Note that *both* displays must support the resolution and bits per pixel or the change will fail.

### Other

There are further options that are rarely needed.

<State Type="DisplaySettings">

…

<RefreshRate="59973/1000" />

<ScanLineOrdering="Progressive" />

</State>

If the right thing isn’t happening automatically, then probably another control panel should be used to choose the values, then use PC config to help read out the state so it can be duplicated under AvSelect.exe.

## Error Handling

<State Type="DisplaySettings" Optional="True" ContinueOnError="True">

<Target FriendlyName="PanasonicTV0" />

<Enabled Value="False" />

</State>

The Optional attribute indicates that if the target cannot be found, it is not an error (and the <State> is skipped). It also causes the checkbox not to require that <State> at all in order for the checkbox to appear.

The ContinueOnError attribute causes subsequent <State>s to be processed, even if something goes wrong on when applying the current <State>.

## Command-Line interface

Important: **Order** currently **matters** on command-line arguments due to parsing laziness. Multiple commands can be specified as long as they are in the correct order as listed in the list in this document. For example, -log must always come first, unless it is omitted.

AvSelect -log

launch with logging. Log.txt will appear in the working directory)

AvSelect -nomessageboxes

Suppresses popup error message boxes

AvSelect -set [menu item name]

Reference a menu item from the config.xml and apply it. Do not launch the Tray icon.

AvSelect -setUntilExit [menu item name]

Reference a menu item from the config.xml and apply it, but then restore it back to the way it started as AvSelect.exe ends.

AvSelect -runworkingdir

Only affects the below -run switch. Working directory for the launched process.

AvSelect -run [Create Process Command]

This option can be combined with the above commands. In “run” mode, we will not launch the tray. Instead, we will ‘host’ an executable, such as a game and await its exit. By combining this command with -setUntilExit, it provides a way to run a process such that [menu item name]’s settings are applied only while that process is running. When the hosted process terminates, AvSelect.exe will terminate and the monitor/sound configuration will revert back to before the change.

Example command to run a game:

AvSelect -setuntilexit "TV as primary monitor" -run "C:\Program Files (x86)\Steam\SteamApps\common\South Park - The Stick of Truth\South Park - The Stick of Truth.exe"

Hint: Create a shortcut for a game with a command similar to the above example in order to “run that game on the TV”.

# Limitations

* Restore functionality (setUntilExit, restoreOnExit) will likely fail if the user connects/disconnects a display, as the display rig differs from when the state was captured.

# Interaction with Steam Big Picture Mode

Steam big picture mode has an option to target any monitor *that is currently* *enabled* when configuring steam or launching big-picture mode. In order to run Steam Big Picture Mode on the TV, the TV simply needs to be enabled as a display device but not necessarily made “primary”. Then, launching into big-picture mode will make the TV primary. So the AvSelect side probably just needs to enable/disable the TV, probably as extended desktop.